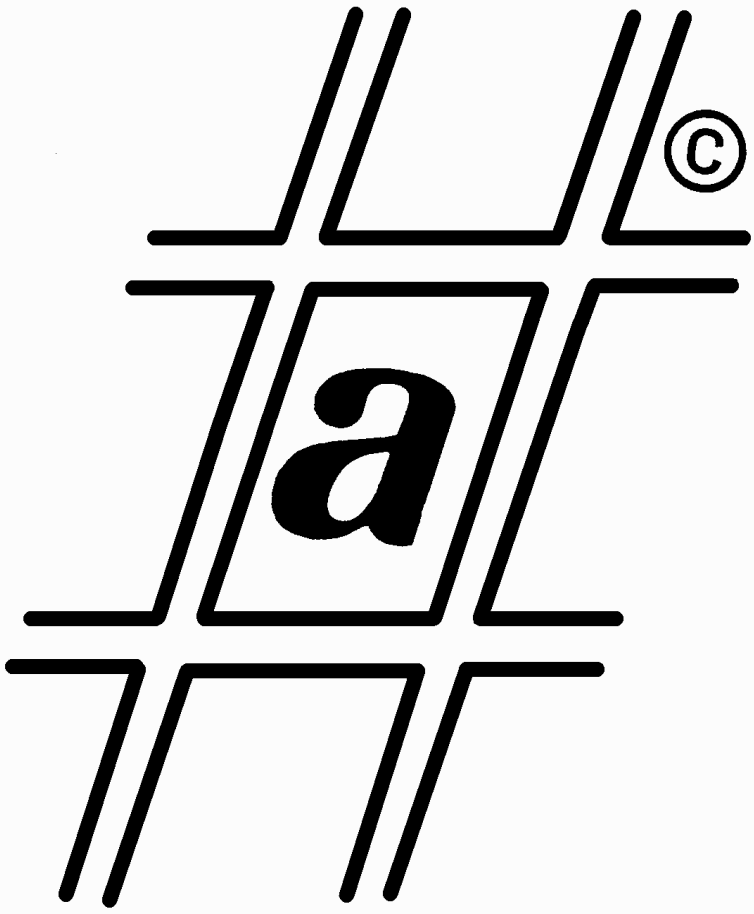
allegro-C

34



Systemhandbuch

allegro-C

Systemhandbuch

Version 34

Bernhard Eversberg

2014

Universitätsbibliothek Braunschweig

Bisher erchienene Ausgaben

Version 8.5, Okt. 1988
Version 9.7, Nov. 1989
Version 10.1, Febr. 1990
Version 10.3, Nov. 1990
Version 11.2, Sept. 1991
korr. Nachdruck März 1992
Version 12.2, Nov. 1992
korr. Nachdruck März 1993
Version 13, Dez. 1993
Version 14, Dez. 1994
Version 15.0, Dez. 1996
Version 15.2, Dez. 1997
Version 20, Mai 2000
Version 22, Sept. 2002
Version 26, Jan. 2006
Version 30, 2010 (pdf)
Version 32, 2012 (pdf)

Einschränkung der Gewährleistung

Es wird keine Gewähr für die Richtigkeit des Inhaltes dieses Handbuchs übernommen. Da sich Fehler, trotz aller Bemühungen, nie vollständig vermeiden lassen, werden Hinweise und Verbesserungsvorschläge jederzeit dankbar entgegengenommen.

Die Texte sind in herkömmlicher Orthographie abgefaßt.

Dieses Handbuch ist e-publiziert in
`http://www.digibib.tu-braunschweig.de/`

Vorwort

Die erste gebundene Ausgabe des ***allegro-C***-Systemhandbuchs erschien mit der Version 9.7 im November 1989. Lange Zeit wurde fast jedes Jahr ein neues Handbuch herausgebracht. Im Juni 2006 erschien zur Version 26.6 die letzte Druckausgabe des Handbuchs. In der Folge wurde der Text mehrfach aktualisiert und das Handbuch dann als PDF kostenlos bereitgestellt. In jede Ausgabe sind Anregungen von Anwendern eingeflossen, es wurden Fehler und Unstimmigkeiten korrigiert, viele Formulierungen verbessert und natürlich alle Neuerungen eingearbeitet. Die V32 war die erste, die komplett auch unter Windows‘7 in der 64bit-Variante laufen konnte. Aktuell ist im Mai 2014 die V34.2, auf die sich diese Ausgabe bezieht.

Die **Windows-Programme** werden im **Kapitel 2** behandelt. Dazu gibt es ferner, das Handbuch beträchtlich erweiternd, ein umfangreiches System von Hilfetexten, die man direkt im Programm abrufen und bei Bedarf selektiv ausdrucken kann. An vielen Stellen findet man im Handbuch Hinweise, wie man den Online-Hilfetext erhält. Die Kap. 1 und 3 für das DOS-System werden sicher nur noch selten gebraucht, aber das Programm hat noch immer seine Anhänger. Die Windows-Programme bieten viele neue Funktionen und Eigenschaften, dennoch können beide Oberflächen gleichzeitig auf derselben Datenbank arbeiten, denn es besteht volle Kompatibilität in den Datenstrukturen. Langjährige Anwender brauchen deshalb auf die altgewohnten DOS-Programme noch nicht zu verzichten, können daneben aber auch die neuen nutzen.

allegro-C war ursprünglich ohne den Genuß öffentlicher oder kommerzieller Förderung zustandegekommen, wurde aber seit Oktober 1990 vom Niedersächsischen Ministerium für Wissenschaft und Kultur mit Projektmitteln unterstützt und damit auf eine breitere Basis gestellt. So entstanden auch Programme bzw. Pakete für Monographienerwerbung, Ausleihe und Sacherschließung, mit jeweils eigenen Dokumentationen. Dieses Handbuch stellt "nur" das Kernsystem vor, mit dessen Hilfe Katalogisierung und Benutzerkatalog realisiert werden können. 1995 wurde ein neuer Programmkern in C++ geschaffen, eine sog. "Klassenbibliothek", und darauf fußend auch ein "Datenbankserver", der sog. ***avanti***-Server. Damit war die Grundlage für "Client/Server"-Anwendungen gelegt, die für Netzwerkumgebungen (Internet und Intranet) von Bedeutung sind. (Mehr dazu unter `http://www.allegro-c.de/doku/avanti`.) Die seit Ende 2012 damit entwickelte Web-Anwendung **a35** ist nun auf dem Wege, zu einem wirklich plattformunabhängigen System namens ***allegro-B*** zu werden, das beim Endnutzer nur noch einen Browser voraussetzt – daher das ***-B***. (Das ***-C*** rührt dagegen von der Programmiersprache C her.) Insbesondere wird mit a35 die Nutzung von Unicode entscheidend erleichtert.

Dieses Handbuch soll umfassende Auskunft in systematischer Darstellung bieten. Es kann deshalb nicht zugleich Lehrbuch sein, deshalb wurde der Titel "Systemhandbuch" gewählt. Ein Lehrbuch, das durch Übungen Fertigkeiten vermittelt und mit viel Beispielmaterial zu einem vertieften Verständnis hinführt, war lange Zeit ein Desiderat, ebenso eine Kurzanleitung für Endbenutzer, die nur in einer fertigen Anwendungsumgebung arbeiten. Für beides ist das Systemhandbuch jedoch eine notwendige Grundlage. Um Einsteigern entgegenzukommen, ist das Kapitel 0 und in jedem Kapitel der Abschnitt 0 als Konzeptüberblick und Einleitung angelegt und weitgehend (hoffentlich) allgemeinverständlich formuliert. Eine Anzahl Fortbildungen ergaben außerdem, unter Beteiligung namhafter ExpertInnen, eine Sammlung von ergänzenden Texten und Lektionen, die auf der Website `allegro-c.de` abrufbereit liegen und mit vielen Tips ein aktives Aneignen der wichtigen Fertigkeiten und Kenntnisse fördern. Diese Ausgabe soll die als letzte umfassende Darstellung mit Einbezug der alten DOS-Programme sein. Seit langem haben die Online-Dokumentationen für den Anwender viel größeres Gewicht.

allegro ist, wie jedes offene Datenbanksystem, in dem Sinne ein "Halbfertigprodukt", daß man es für konkrete Anwendungen konfigurieren und parametrieren muß. Das Systemhandbuch soll die gebotenen Möglichkeiten dokumentieren und so die Grundlage bieten für die Gestaltung unterschiedlichster Anwendungsumgebungen. In der Praxis wurden damit viele Projekte möglich, für die ein reines Bibliothekssystem sich nicht geeignet hätte. Mitgelieferte Standard-Parameterdateien erleichtern in vielen Fällen einen sofortigen Beginn, stellen aber im Prinzip erweiterungs- und wandlungsfähige Beispiele dar. Die mitgelieferte Demo-Datenbank leistet dabei ebenfalls Hilfestellung. Sie enthält Daten aus dem Online-Katalog der UB-Braunschweig, vom Pica-Format gewandelt in das *allegro*-Standardformat. Die vollständige OPAC-Datenbank mit knapp 1.5 Mio Sätzen ist auf der DVD enthalten, die zuletzt im Nov. 2013 erschien und den Lizensenten zugesandt wurde.

Trotz vieler Bemühungen könnte ***allegro*** kaum jemals ein "ganz einfaches" System werden, mit dem jeder intuitiv zurechtkäme. Zwar kann man mit den wichtigen Dingen unmittelbar beginnen und kommt mit den jetzt vorhandenen Grundlagen und Hilfen schon sehr weit voran. Eine Einarbeitungszeit von 4 bis 8 Wochen kann aber nötig sein für anspruchsvollere Nutzer, die eigene Strukturen, Parameter und Prozeduren brauchen, insbesondere wenn mit wenig allgemeiner Computerkenntnis angefangen wird. Das liegt einerseits an der Flexibilität des Systems, andererseits aber an der immanenten Komplexität der bibliothekarischen Aufgaben, die von Einsteigern in aller Regel weit unterschätzt wird.

Daß eine über viele Jahre hinweg kontinuierliche Arbeit an diesem System innerhalb einer Universitätsbibliothek möglich war - doch zum Nutzen von Bibliotheken aller Sparten - ist nicht zuletzt deren langjährigem Leiter, Prof. Dr. Dietmar Brandes, zu danken. Es erforderte vielfältige Initiativen und beträchtlichen Einsatz, die Rahmenbedingungen für diese Arbeit zu schaffen, zu verbessern und aufrechtzuerhalten. Im Jahre 2012 wurden schließlich die wesentlichen Teile der Quellprogramme im Sinne des "OpenSource"-Konzepts offengelegt, alle notwendigen Komponenten auch unter Linux/Solaris lauffähig gemacht und damit die Zukunftssicherheit noch besser befestigt. Mit zwei verschiedenen Web-Oberflächen, PHPAC und a35, beide ebenfalls quelloffen, sind mittlerweile weit über 100 Datenbanken am Netz, nicht eingerechnet die der Öffentlichen Bibliotheken, die vor allem von der Büchereizentrale Niedersachsen (Lüneburg) mit einer eigenen Version betreut werden.

Nach aller bisherigen Erfahrung werden aus den Bemühungen von Anwendern wertvolle Anregungen für die weitere Verbesserung und Ausgestaltung des Systems hervorgehen. Eine weit verteilte Basis von versierten Anwendern nicht nur in Deutschland kann aber heute und künftig die meisten Probleme unabhängig von "Braunschweig" lösen. Diese Errungenschaft ist vielleicht das wertvollste Ergebnis der Entwicklung.

B.E.

Leitfaden zur Einarbeitung

(Hier geht es z.T. auch noch um das DOS-System. Mehr zu den Windows-Programmen → Kap. 2)

1. Installieren Sie das "Gesamtpaket", und zwar als Administrator. Beachten Sie die Hinweise im **Abschnitt 0.10**. Wenn es dabei Unklarheiten gibt, wenden Sie sich an einen PC-Kenner.
2. Die Windows-Programme können alles, was man bei der normalen Arbeit braucht. ***a99*** und ***alcarta*** ersetzen die DOS-Programme (Kap.0.4) PRESTO und APAC, teilweise auch UPD. Es werden noch 5 Programme des DOS-Systems für bestimmte Aufgaben gebraucht: IMPORT, INDEX, QRIX, ASORT, SRCH (allesamt auf 32bit umgestellt, unter Win'7/64 lauffähig), ferner ist UPD ersetzt durch das Konsolprogramm `acon` mit `update.job`). Die Kommandofenster werden automatisch geöffnet und geschlossen - man braucht sich damit nicht mehr auszukennen, es wird nur manchmal ein Tastendruck oder eine einfache Eingabe verlangt.
3. Sammeln Sie erste Eindrücke mit der **Beispieldatenbank**. Man startet sie mit dem automatisch angelegten Icon. Dann hilft **Kapitel 2**. Machen Sie sich vertraut mit den Such- und Zugriffsverfahren, und schauen Sie sich das große Gesamtmenü namens **Füllhorn** an. Bei einigen der Untermenüs, z.B. Export, Admin, Reorganisieren, gibt es ausführlichere, erklärende Versionen, wenn oben rechts HILFE..HILFE..HILFE steht. Anklicken!
   Wenn **Fehlermeldungen** auftreten, schlagen Sie im Anhang C nach, was zu tun ist!
4. Überfliegen Sie kurz den **Abschnitt 0.9**. Dabei kommt es noch gar nicht auf umfassendes Verständnis an, nehmen Sie nur die **Anpassungsmöglichkeiten** zunächst einmal zusammenfassend zur Kenntnis. Außerdem steht dort eine Liste mit schnellen Antworten auf häufige Fragen. Vielleicht entdecken Sie dabei schon Punkte, die Ihnen wichtig sind. Behalten Sie das zunächst im Hinterkopf, Sie gewinnen damit leichter den Überblick.
5. Nehmen Sie sich etwas Zeit für das **Kapitel 0**. Grundlegend wichtig sind die Abschnitte 0.0 bis 0.4. Hier geht es vor allem um ein gutes Verständnis der **"zentralen Begriffe"** in Abschnitt 0.0; bei den anderen Abschnitten genügt zunächst ein Gesamtüberblick. (Kehren Sie später immer hierher zurück, wenn Sie Rat suchen.) Nehmen Sie die Funktionsdiagramme in **Anhang C** hinzu und versuchen Sie, die in den **Abschnitten 0.3 und 0.4** beschriebenen Dateitypen und Programme darin wiederzufinden. Werfen Sie zum Vergleich auch einen Blick in das Datenverzeichnis DEMO2 Ihrer Installation, wo sich die Bestandteile der Beispieldatenbank befinden. Wenn Sie mit den **DOS-Programmen** arbeiten wollen (bis Win'7/32 möglich): Abschnitt **0.11** erklärt die Benutzung des ***CockPit***, das Ihnen dabei helfen soll. Im Handbuch sind viele Hinweise auf ***CockPit***-Menüs zu finden: eine Abkürzung wie **`µo k`** verweist auf das **Menü** "**O**ptionen", Unterfunktion **k** = **K**onfiguration.
6. Steigen Sie nun wieder in die Beispieldatenbank ein und vergleichen Sie die Datensätze mit dem **Kategorienschema** im **Anhang B**. Sie sehen die interne Form der Datensätze (mit Kategorienummern), wenn Sie einmal F5 drücken. (Erneutes F5 schaltet zurück auf die normale Anzeigeform. Versuchen Sie auch F7.) Der Anhang B verdient ein genaueres Studium; als Nichtbibliothekar erfahren Sie dort auch einiges über das Problemfeld "Katalogisierung".
7. Versuchen Sie nun, immer noch in der Beispieldatenbank, die Funktionen [Form#] (= Formular) und [Neusatz] (= **Input**, Eingabe eines neuen Datensatzes), wobei dann F1 Hilfe bringt.
8. Wenn Sie sehen, daß für Ihre Zwecke Änderungen am Kategorienschema, an der Bildschirmanzeige, an der Druckausgabe oder an der Indexgestaltung nötig sind, dann brauchen Sie einen **Text-Editor** - ein Programm, mit dem man "ASCII-Dateien" (= normale Textdateien) bearbeiten kann. In solchen Dateien (z.B. A.CFG für die Eingabeformulare, D-1.APR für die Titelanzeige, CAT.API für die Register) stehen nämlich die Angaben, die Sie ändern müssen. Diese Dinge gehen nicht über Menüs!
   Wenn Ihnen unklar ist, um was es hier geht, lesen Sie nochmals **0.0** und/oder fragen Sie einen PC-Kenner, ob er Ihnen den Gebrauch eines solchen Editor-Programms vermitteln kann. Das ***CockPit*** ruft einen Editor namens X auf, der sich sehr gut für diese Aufgaben eignet. X wird im **Anhang D** beschrieben. Auf der ersten Seite dieses Anhangs steht alles, was Sie brauchen - zwei Minuten, die sich lohnen! X läßt sich jedoch durch andere (z.B. Wordpad von Microsoft) ersetzen (→Kap.0.11.6), für Windows wird WinVi empfohlen. Sie lassen sich allerdings auf recht komplexe Dinge ein: die Kapitel 10 (Export) und evtl. 11 (Import) sowie der Anhang A (Konfiguration) beschreiben zwar alles, aber schon an deren Umfang erkennen Sie, daß hier echte Arbeit wartet. Es wäre sinnlos, das verschleiern zu wollen - es ist der Preis der Flexibilität. Dringend wird das Studium der mitgelieferten Parameterdateien empfohlen, die ausführlich kommentiert sind und für viele Zwecke als Beispiele und Grundlage dienen können.
   Sie wollen ein ganz neues Projekt beginnen und müssen dafür eine eigene Struktur entwerfen? Kap. 1.1 beschreibt die möglichen Wege.
9. Schritt für Schritt und Thema für Thema können Sie sich immer weiter einarbeiten mit den **Fortbildungstexten**. Diese erhalten Sie, wenn Sie eingeben: janas fb.htm [oder: http://www.allegro-c.de/fb/]
10. **Kapitel 12** zeigt, wie man die ***allegro***-Programme in größere **Prozesse** einbauen kann, in sogenannte "Batchdateien" (= "Stapelprogramme", Dateityp .BAT). Besonders die Programme SRCH und IMPORT lassen sich als Hilfsprogramme in andere "Umgebungen" einbeziehen. Dieses Verfahren hat allerdings viel an Bedeutung verloren: die Zukunft liegt im FLEX-System. Damit kann man Vorgänge aller Art automatisieren. Die Dokumentation kommt, wenn man `h flex` eingibt.

Tasten werden in diesem Handbuch mit spitzen Klammern bezeichnet, **Buttons** zum Draufklicken dagegen mit eckigen: <Entf> meint also die Taste auf der Tastatur und [Entf.] meint einen Button mit dieser Aufschrift.

Ein Hinweis wie "*Tip für a99*: geben Sie **`h handbuch`** ein" bezieht sich immer auf das Schreibfeld des Programms a99. Auf der ersten Abbildung in Kap. 2 (auf S. 67) sehen Sie, was damit gemeint ist.

Inhalt

Leitfaden zur Einarbeitung ii

0 Das *allegro*-Konzept 1

0.0 Zehn Begriffe, die jeder kennen sollte 1
0.1 Allgemeines über Dateien 5
0.2 Mehr über Datenspeicherung und Datenstruktur 7
0.3 Dateitypen 9
0.4 Programme 17
0.5 Auswahl-Listen (in DOS-Programmen) 23
0.6 Menütexte, Sprache 24
0.7 Datensicherung: Konzept und Methoden 24
0.8 Was gehört alles zu einer Datenbank? 26
0.9 Häufige Fragen. Änderungen : Was? Wo? Wie? 28
0.10 Installation 30
0.11 *CockPit* : Das DOS-Steuerprogramm 32
1 µf : Das Menü "Funktionen" 32
2 µr : Das Menü "Routinen" 34
3 µm : Das Menü "Makros" 35
4 µd : Das Menü "Dateien" 37
5 µo : Das Menü "Optionen" 39
6 Die Vorgabendatei CP.OPT 40
Nachwort zu Kapitel 0 42

1 Schnellzugriff (DOS-Programm PRESTO) **.....43**

1.0 Aufgabenstellung und Konzept 43
1.1 Anlegen einer neuen Datenbank 44
1 Der schnelle Standard-Einstieg 45
2 PRONTO : Menügeführter Datenbankentwurf 45
3 Neue Datenbank "zu Fuß" anlegen (ohne Menü) 46
1.2 Datenbank-Generierung aus vorhandenen Daten 47
1.3 Benutzung einer Datenbank 48
1 Einstieg per *CockPit* oder von Hand 48
2 Aufruf des Programms in Stapeldateien 50
1.4 Der Register-Bildschirm 51
1 Browsing, Blätterfunktionen 51
2 Trunkierung, Endemarken 52
3 Direkte Indexeingriffe 55
4 Logische Kombination, Ergebnismengen 56
5 Kurztitelanzeigen 57
6 Restriktionen benutzen 58
1.5 Der Anzeige-Bildschirm 59
1 Zugriff, Anzeige, Bearbeitung 59
2 Zugriffsfunktionen 60
3 Blätterfunktionen 61
4 Bearbeitungsfunktionen 62
5 Ausgabe- und Anzeigefunktionen 64
6 Hilfsfunktionen und Sprache 64
7 Funktionstasten : Zusammenfassung 65
1.6 Hinweise zu Normdateien 66

2 *a99 / alcarta* - Die Windows-Programme 67
2.1 Die Grundfunktionen, Menüs und Buttons 67
2.2 Die Komfort-Funktionen (Das "Füllhorn") 76
2.3 FLEX : Die Skript- und Makrosprache 82
2.4 Fremddaten mit *a99* : Nur ein Knopfdruck 89
2.5 Einrichtung des Windows-Systems 89
2.6 Database Publishing mit *alcarta* 92
2.7 *a99* als Server 92
2.8 *Weitere Themen:* JanaS, Unicode, XML, SQL, a30, a35 92

3 Datenerfassung (im DOS-System) **.....93**
3.0 Allgemeines 93
3.1 Erfassen und Bearbeiten: Überblick, Wichtige Befehle . 94
3.2 Abfrageliste (Dateneingabe) 96
3.3 Zeichen mit Sonderbedeutung 97
3.4 Besondere Tastenfunktionen 98
3.5 Editor-Befehle von a bis z 99

4 Volltext-Suche und Selektion 119
4.1 Programmablauf 120
4.2 Aufruf des Programms in Batchdateien 123
4.3 Beispiele für Suchbegriffe 124

5 Fremddaten konvertieren (Import) 125
5.1 Programmablauf 128
5.2 Aufruf des Programms in Batchdateien 130

6 Export 131
6.1 Sortierte Listen produzieren : Konzept 132
6.2 Und so produziert man eine sortierte Liste 134
6.3 QUEX : Listenproduktion ganz einfach 136
6.4 QUANT für Statistiken 139
6.5 EXPEX : Exportieren lernen mit MARC
6.6 Listen mit Registern 142

7 Indexieren 143
7.0 Datenbank-Generierung : Aufgabe und Konzept 143
7.1 Datenbank-Generierung : Ablauf 144
7.2 Datenbank erneuern, ganz oder teilweise 147
7.3 "Entlüften" von Datenbankdateien 147
7.4 Datenbank als Ganzes erneuern 148
7.5 QRIX - Werkzeug für Datenbankingenieure 149
7.6 Einzelnes Register erneuern 151
7.7 Neues Register hinzufügen 152
7.8 Ein Register ganz löschen 152
7.9 Registerauszüge 152

8 Sortieren 153

9 Datenbank-Management / Update 155
9.1 Daten einspeisen, Ablauf 155
9.2 LOG-Datei auswerten/umwandeln 158
9.3 Sätze löschen mit UPD 158
9.4 Datenbank ganz neu aufbauen mit UPD 159
9.5 Datenbanken aktualisieren mit dem *avanti*-Server 159
9.6 Besondere Möglichkeiten beim Updating 160

10 Export 161
10.0 Zum Konzept 161
1 Teilbereiche des Exportverfahrens 162
2 Inhalt einer Parameterdatei 164
10.1 Wie macht man eine Parameterdatei? 166
10.2 Die Exportsprache 167
0 Zwischenteile 169
1 Aufnahme-Parameter 174
2 Formular- und Seiten-Parameter 180
3 Zeilen-Parameter 181
4 Drucker- und Codierungs-Parameter 181
5 Kategorie-Parameter 187
6 Die Kategorieliste - das Kernprogramm 188
1 Anweisungen 189
2 Sonderkategorien 191
3 Manipulationsbefehle 194
4 Steuerbefehle 204
5 Sprünge, Unterprogramme 205
6 Rechenbefehle 209
7 Satzverknüpfungen 210
8 Normdaten-Verknüpfungen 215
9 Satzübergreifende Suche 220
7 Flips : Querverbindungen 225
8 Programmierte Eingabeprüfung 228
9 Restriktionen 231
10.3 Testen und Optimieren 235
1 Direkte Änderung von Parametern 235
2 Mehrere Parameterdateien gleichzeitig 235
3 Export-Parameterdatei visuell testen 235
4 Export- und Index-Parameter optimieren 237
10.4 Beispiele: Prototypen 239

11 Import-Parameter 249
11.0 Zum Konzept 249
1 Struktur von *allegro*-Dateien 250
2 Inhalt einer Import-Parameterdatei 251
11.1 Erstellung einer Import-Parameterdatei 252
11.2 Die Import-Sprache 253
1 Basis-Parameter 253
2 Globale Konvertierungsbefehle 258
3 Die Kategorieliste - das Kernprogramm 260
11.3 Beispiel einer Umwandlung mit *allegro* 273

12 Programm-Aufrufe (Batch bzw. Shellscript) **....279**
12.0 Konzept 279
12.1 Liste der Optionen 281
12.2 Beispiele für Aufrufe 287

Anhänge

A Konfiguration 289
A.0 Vorbemerkungen 289
A.1 Konfigurationsbefehle 290
A.2 Abfrageliste 296
A.3 Farben (DOS-System) 298
A.4 Beispiel: Standard-Konfiguration `$a.cfg` 300
A.5 Umstieg von älteren Versionen auf V13 oder neuere . 306

B *allegro*-Standard-Datenschema 307
B.0 Vorbemerkungen 307
B.1 Kurzübersicht zum "konsolidierten" Format 308
Katalogisieren - Was ist das? (Terminologie) 310
B.2 Normdatensätze = Stammsätze 314
B.3 Geschäftsgangsdaten (Erwerbung, Ausleihe) 320

C Diagramme – Fehler - Literatur 325
C.1 Funktionsdiagramme 325
C.2 **Fehlermeldungen** 331
C.3 Literaturangaben 335

D Texteditor X 337

E Zeichensatz (DOS und Windows) 342

INDEX 347

0 Das *allegro*-Konzept

0.0 Zehn Begriffe, die jeder kennen sollte

Tip für a99: Geben Sie **`h handbuch`** ein

Wer ein **Datenbanksystem** nutzt, interessiert sich in der Regel erheblich mehr für die eigenen **Daten** als für das **System**. Von **Programmen** und ihren internen Funktionen oder gar von **Datenstrukturen** möchte nicht jeder viel wissen müssen. Weil Computer vorerst die Umgangssprache weder sprechen noch verstehen, ist ein Minimum an DV-Kenntnissen wichtig, auch um zu einer realistischen Einschätzung der Möglichkeiten und Grenzen des Computereinsatzes zu gelangen. Soweit es um ***allegro*** geht, soll dieses Handbuch dabei helfen, obwohl es nicht zugleich ein Lehrbuch sein kann: das Kapitel 0 und in den übrigen Kapiteln jeweils der Abschnitt 0 sollen die Begriffe und Konzepte erklären und die Einarbeitung erleichtern.

> Wenn Sie gerade erst mit ***allegro*** anfangen wollen:
> Der **Leitfaden zur Einarbeitung** (hinter dem Vorwort) weist den Weg.
> Im Windows-Programm: Hilfemenü **?** wählen!

Jedes Rechnersystem speichert Daten in **Dateien**. Die Vorgänge des Speicherns und Lesens von Daten sowie jede Art des Umgangs mit den Daten besorgen **Programme**.

Jedes Programm folgt dem bekannten "E-V-A-Schema":

Eingabe (Daten) ➔ **Verarbeitung** (Programm) ➔ **Ausgabe** (Daten)

wobei allerdings mehrere Ein- und Ausgabevorgänge in dauerndem Wechselspiel ablaufen können. Bei den heutigen objektorientierten Programmen mit graphischen Oberflächen ist diese grundlegende Tatsache von außen nicht mehr leicht zu erkennen, bleibt aber gültig: stets wird irgendetwas eingegeben, in welcher Weise auch immer, ein Programm(system) führt damit interne Vorgänge durch und gibt etwas wieder aus, in ein Fenster und/oder in eine Datei, auf einem Drucker oder ins Netz:

Eingabedaten kommen entweder

a) über die Tastatur (vom Anwender); es kann sich um "richtige" Daten, aber auch um Funktionstasten-Betätigungen, Mausklicks oder eingetippte Befehle handeln,
b) oder werden aus Dateien (evtl. über das Netz) eingelesen. Dabei kommt es immer sehr auf deren Struktur an (siehe unten).

Ausgabedaten als Ergebnis einer Verarbeitung

a) erscheinen in einem Fenster auf dem Bildschirm oder auf einem Drucker
b) oder werden wieder gespeichert, d.h. in Dateien geschrieben oder per Netz einem anderen System übermittelt.

Im Web erfolgt die Verarbeitung an mehr als einer Stelle; typischerweise teilen sich ein "Client" und ein "Server" die Arbeit, ohne daß der Endnutzer etwas davon merkt.
Als Ausgabe können zwar auch Katalogzettel herauskommen, doch ist dies mehr ein Nebenprodukt. Eingegeben und gespeichert werden natürlich keine Katalogkarten, sondern **Datensätze**. Meistens entspricht ein Datensatz aber einer **Titelaufnahme** im alten Sinne. Deswegen steht in diesem Buch manchmal "Aufnahme" statt "Datensatz". Aus gespeicherten Datensätzen kann das System sehr vieles machen: Karten, Listen, Bildschirmanzeigen, Auswertungen (auch mit Berechnungen), alphabetische Register, Tabellen. Alles das ist nicht fest eingebaut, sondern kann variiert, angepaßt und ausgebaut werden. Um eigene Wünsche zu verwirklichen, muß man freilich dem System geeignete Vorschriften machen. Solche Vorschriften sind **FLEX-Skripte** (Kap.2.3) oder **Parameterdateien** (Kap.10 und 11). Sie bestehen oft aus sehr vielen einzelnen Befehlen. Viele fertige Parameterdateien für Listen, Tabellen usw. werden mitgeliefert. Ist man damit nicht zufrieden, muß man Änderungen machen oder ganz neue Vorschriften erarbeiten und die Befehle dafür zusammenstellen. Die Befehle werden, wie gesagt, nicht in natürlicher Umgangssprache formuliert. Man muß jeweils eine besondere Sprache dazu benutzen, die sog. **Exportsprache** (Kap.10). Will man Fremddaten in ***allegro***-Daten umwandeln, kommt noch eine Sprache hinzu, die **Importsprache** (Kap.11). Der Wortschatz dieser Sprachen ist viel kleiner, die Grammatik viel einfacher als bei natürlichen Sprachen. Dennoch gehört das Schreiben von Export- und Importvorschriften zu den schwierigsten Aufgaben im ***allegro***-System, denn man muß dabei sehr präzise und in ungewohnt kleinen Schritten denken. Viele solche Dateien werden aber schon fertig mitgeliefert und können bei Bedarf auch angepasst werden.

Dieses einführende Kapitel ist nicht nur zum Lesen, sondern auch zum Nachschlagen gedacht. Ein alphabetischer Überblick zu den wichtigsten Eingreifmöglichkeiten und eine **Liste häufiger Fragen** befinden sich im Abschnitt 0.9.

Die wichtigsten Begriffe

Zehn Grundbegriffe sind für jeden wichtig, der zu einem eigenständigen Umgang mit ***allegro*** finden will. Aber auch "Nur-AnwenderInnen" werden sich im System besser zurechtfinden, wenn sie sich mit diesen Grundbegriffen vertraut machen. Versuchen Sie bei Ihrer weiteren Beschäftigung mit ***allegro***, sich immer an diesem Gerüst zu orientieren. (Siehe auch Diagramm 2 in Anhang C.) Die Grundbegriffe bleiben auch für die neuen Windows-Programme gültig, denn die internen Strukturen des Systems haben sich bewährt und bleiben bestehen.

Was bedeutet **µ**? [Nur beim DOS-System wichtig]
Am rechten Rand sehen Sie jeweils, wie man aus dem DOS-***CockPit*** an die betreffenden Dateien oder Programmteile herankommt. **Das Zeichen 'µ' steht für "Menü"**, der Buchstabe dahinter ist der Anfangsbuchstabe des Menüs. Also heißt z.B. **µo k** soviel wie "Menü *Optionen*, Unterpunkt k". Erreichbar ist dieser Menüpunkt mit der Tastenfolge **<Alt>+o k.** Schauen Sie sich den ***CockPit***-Bildschirm an, dann sehen Sie sofort, wie das gemeint ist (***CockPit*** → 0.11, Start mit Befehl **cp** auf C:\ALLEGRO). Pfad- und Dateinamen sind in diesem Handbuch zumeist groß geschrieben, um sie besser erkennbar zu machen. *ABER:* Unter UNIX sind alle Dateinamen komplett in Kleinbuchstaben zu schreiben, und so werden sie auch geliefert.

1. Konfiguration (→ Anhang A.) µo k

allegro hat kein fest eingebautes Schema von Datenfeldern oder festgelegten Eingabeformularen. Sie können selber entscheiden, welches **Kategorienschema** Sie benutzen wollen, welche **Datensatztypen** es gibt, was alles in **Erfassungs-Abfragelisten** und –**Formularen** vorkommen soll. Auch die **Meldungstexte**, **Button-Aufschriften**, die **Sprache** der Menüs und andere Einzelheiten sind variabel. Viele dieser Angaben stehen in einer **Konfigurationsdatei** (Typ .CFG). Mitgeliefert werden die Konfigurationen **$A.CFG**, **D.CFG** (MAB1), **M.CFG** (MAB), **$P.CFG** (Pica), **$U.CFG** (MARC), **N.CFG** (neutral). Sie können mehrere Datenbanken mit derselben Konfiguration betreiben oder mit je einer eigenen. Änderungen an der Konfiguration führt man mit einem Bearbeitungsprogramm aus (man nennt so ein Programm "Texteditor", → 3b). Die Konfiguration ist allerdings nur die eine Hälfte einer Datenbankdefinition, die andere ist die **Index-Parameterdatei**:

2. Parameterdateien = Vorschriften fürs System µd e...

Die Konfiguration enthält nur Angaben zur *Struktur* der Daten und einige *Grundeinstellungen* für die Funktion der Programme. Damit ist noch nicht festgelegt, wie die Daten z.B. am Bildschirm erscheinen, wie sie sortiert, umgewandelt oder ausgedruckt werden können, oder über welche Kriterien man wie zugreifen kann (die Indexierung). In diesen Bereichen bleibt viel Raum für Kreativität, und das Werkzeug dazu sind eben die **Parameterdateien**. Sie enthalten so etwas wie Vorschriften und sind geschrieben in einer Art Programmiersprache, auch **Datenmanipulationssprache** genannt, die speziell für ***allegro*** geschaffen wurde. Es gibt jedoch drei solche Sprachen für drei Aufgabenbereiche:

a) Die **Exportsprache** (→ Kap.10) : in dieser Sprache beschreibt man und teilt man dem System mit:
(in Klammern stehen Namen von geeigneten Standard-Parameterdateien, die man sich anschauen kann)

- wie die Daten am Bildschirm angezeigt werden sollen (z.B. D-0.APR oder D-1.APR) µd e
- wie eine gedruckte Liste aussehen soll (z.B. P-NORMAL.APR oder P-W.APR)
- wie eine Datenmenge zu sortieren ist (z.B. S-PT.APR)
- in welcher Form Daten in eine andere Datei zu schreiben sind (z.B. E-1.APR).
- wie der → Index einer Datenbank aussehen soll (z.B. CAT.API) µd i

Die Exportsprache hat Ausdrucksmittel für die Anordnung der Elemente, für deren Vorbehandlung (Zerlegen, Zusammenfügen, Ersetzen), für Interpunktion und Umbruchgestaltung sowie Zeichen-Umcodierung. Außerdem gibt es Sprungbefehle, Bedingungsprüfungen und Unterprogramme.
Export-Parameterdateien haben die Typkennungen .APR, .API und .APT.

b) Die **Importsprache** (→ Kap.11) : damit beschreibt man, wie "Fremddaten" aussehen und wie sie in ***allegro***-Daten umgewandelt werden sollen. Damit wird nicht jeder Anwender konfrontiert. Import-Parameterdateien haben die Typkennung .AIM. Beispiel: OCLC.AIM zur Umwandlung von MARC-Daten. µd imp

c) Die **Skriptsprache** FLEX (→ Kap.2.3) : Diese gibt es nur für das Windows-System und die Web-Oberflächen. Mit FLEX kann man nicht nur Daten bearbeiten, sondern Vorgänge aller Art programmieren. Ein FLEX-Skript ist ein kleines Programm, das innerhalb des Systems ausgeführt wird, um eine Aufgabe zu lösen. **a99: h flex**

Auch Parameterdateien und FLEX-Skripte sind Textdateien, die man mit einem → Texteditor (siehe 3b.) erstellt und verändert. Das ***CockPit*** ruft automatisch einen solchen Editor auf (→ 0.11 und Anh.D), wenn man über das Menü "Dateien" eine solche Datei bearbeiten will. [**a99**: `h param` eingeben, dann den Typ wählen]

3. Editor

Damit ist ein Programm oder Programmteil gemeint, mit dem man Datensätze oder Dateien am Bildschirm bearbeiten kann. Es gibt viele solche Programme, aber keins eignet sich für *alle* Daten. Word und WordPerfect eignen sich z.B. nicht für *allegro*-Daten. Es hängt von der Struktur der Daten ab, ob man sie mit einem bestimmten Editor bearbeiten kann oder nicht. Der ***allegro***-Anwender braucht zwei Editoren:

a) den **integrierten Editor** (Funktion 'E' zum Bearbeiten, → Kap.1.5.4 und Kap.3),
Damit kann man keine anderen, sondern **nur** ***allegro***-Daten bearbeiten, und zwar immer zu einem Zeitpunkt nur einen Datensatz: wenn er auf dem Anzeigeschirm steht; im DOS-Programm gibt man 'E' wie "Editor". Das Windows-Programm ***a99*** hat mehrere Bearbeitungsfunktionen, unter anderem einen neuen Formular-Editor (Alt+#). Einerseits gibt es die typischen Eigenschaften und Funktionen wie etwa Copy-and-Paste, andererseits bietet das Programm aber auch umfassende Möglichkeiten zur anwendungsspezifischen Anpassung und Ausgestaltung mit Hilfe des Skriptsystems FLEX;

b) einen **Texteditor**. Damit lassen sich Textdateien des einfachsten Typs, sog. "ASCII-Dateien", bearbeiten. (Man muß wissen: Textdateien bestehen aus *Zeilen*, Datensätze aber aus *Datenfeldern*, und ein Datenfeld kann viel länger als eine Bildschirmzeile sein.) Zum Typ Textdatei gehören die → Parameter- und Konfigurations-dateien. Deshalb braucht der ***allegro***-Anwender einen Texteditor. MS-DOS enthielt ab 5.0 mit EDIT einen ganz brauchbaren Texteditor. Nicht brauchbar für diesen Zweck sind Word und verwandte Produkte. Wenn man vom ***CockPit*** aus eine Datei zur Bearbeitung anfordert, kann das mit jedem Texteditor geschehen. Es wird zwar einer namens X mitgeliefert (→ Anhang D.), aber Sie können auch z.B. EDIT benutzen (wegen der Einbindung ins ***CockPit*** siehe → 0.11.6). Unter UNIX verwendet man in der Regel den Editor `vi`., für Windows ist Notepad verwendbar oder ein kostenloser Editor namens `WinVi` (im Netz zu finden).
In ***a99*** kann man auch das Anzeigefeld zum Bearbeiten von Textdateien benutzen. Beschrieben ist das in der Fortbildungslektion 9-2: http://www.allegro-c.de/fb/allegrofb-Kap9.2.htm

Nochmals: ***allegro***-Datenbankdateien **nicht** mit einem Texteditor bearbeiten, sonst gehen sie kaputt.

4. Datenbank `µr b`

Wie die Datensätze aussehen, das hängt ab von der → Konfiguration (Kategorienschema), die Register aber werden beschrieben in der → Index-Parameterdatei, z.B. `cat.api`. Der Name dieser Datei ist zugleich der Name der Datenbank, also z.B. `cat`. Die Datensätze, die der Anwender eingibt, landen in Dateien vom Typ `.ald` (oder z.B. `.nld`, wenn man mit der Konfiguration `n.cfg` arbeitet!) Deshalb müssen diese Dateien unbedingt und vor allen anderen gesichert werden. Außerdem gehören zur Datenbank der → Index (Typ `.adx` bzw. `.ndx`) und noch eine Satztabelle (vom Typ `.tbl`). Beide können erneuert werden, indem man die Datenbank aus den `.ald`-Dateien neu aufbauen läßt. Benutzt wird die Datenbank mit dem Programm ***a99*** (Windows) oder PRESTO (DOS). Der darin → integrierte Editor ermöglicht Neueingabe und Korrektur. Auch das Volltext-Suchprogramm SRCH (→ Kap.4) kann die Datenbank lesen: es durchsucht die Daten zeichenweise, kann auch → exportieren, ermöglicht aber keine Korrekturen und Neueingaben. Die Programme INDEX und UPD können neue Daten, die man z.B. aus fremden Quellen bezogen hat (→ Kap.5), einer Datenbank einverleiben. Eine neue Datenbank legt man per ***CockPit*** an (**`µr a`**) oder mit dem Hilfsprogramm PRONTO (***a99*** : **`h newdb`**). Sind schon Daten in der richtigen Struktur vorhanden, läßt man vom Programm INDEX eine Datenbank daraus machen (→ Kap.7).
In → Kap.0.8 steht eine Liste der Dateien, die zu einer Datenbank gehören.

5. Index (→ Kap.1.4, Kap. 2.1, Kap.7) `µr o i`

Jede Datenbank braucht einen Index. Das ist meistens nur eine Datei; bei der DemoBank heißt sie z.B. `cat.adx`. (Ab V27.2 kann es mehr als eine Indexdatei geben.)
Wenn man sich die Datenbank als Nachschlagewerk vorstellt, enthält der Index das oder die alphabetischen Register. Bis zu 11 solche Register kann eine Indexdatei enthalten. Man blättert darin, indem man (ohne Befehl) ein gesuchtes Wort oder dessen Anfang eingibt. Mit Druck auf die Taste <Alt> und eine Ziffer schaltet man auf eines der anderen Register um. Indexdateien entstehen und wachsen automatisch mit der Datenbank. Wenn man mit dem → Editor an den Daten etwas ändert, wird der Index automatisch auch geändert. Wie die Register aussehen, das steht in der zur Datenbank gehörigen Index-→ Parameterdatei. Diese muß man ändern (mit einem → Texteditor), wenn man grundsätzliche Änderungen am Index vornehmen möchte. Anschließend muß man aber den Index erneuern lassen - mit ***CockPit*** kein Problem (→ Kap.0.11.2 und Kap.7; ***a99***: `h org` eingeben).

6. Export = Ausgabe = Formatierung (→ Kap.6 und Kap.10) µf 4

Export bedeutet: Daten in eine Datei schreiben oder auf ein Gerät (Drucker oder Bildschirm) ausgeben. Entscheidend ist immer, in welcher *Form* das geschehen soll, und die Vorschrift dafür steht in einer Export-→Parameterdatei (siehe 2b.). In die Programme ***a99***, PRESTO, SRCH, INDEX und IMPORT ist eine **Exportfunktion** eingebaut, d.h. jedes dieser Programme kann exportieren. Es nimmt dazu eine Parameterdatei her und wendet die darin stehenden Vorschriften auf die anstehenden Daten an.
Über sog. **Optionen** (→ Kap.12) kann man den Programmen beim Aufruf mitteilen, welche Parameterdatei auf welche Daten anzuwenden ist (***CockPit*** macht das automatisch). Das Ergebnis ist eine Liste, eine Bildschirmanzeige, ein Index, etc.

7. Ergebnismenge (→ Kap.1.4 und 1.5)

Wenn man auf eine → Datenbank über einen → Index zugreift, kann man die zu bestimmten Registereinträgen gehörigen Datensätze mit den logischen Kombinationstasten '+' (UND), '/' (ODER) bzw. '-' (NICHT) zu Teilmengen der Datenbank zusammenfassen (bis zu 16.000 Sätzen (DOS), Begrenzung möglich mit dem Befehl **`mr`** in der Konfiguration. ***a99*** : 64.000). Eine Ergebnismenge kann man durchblättern, umsortieren, exportieren (z.B. drucken), aber auch global verändern oder löschen. (Mehr dazu: `h reslt` eingeben in ***a99***)

8. Datensicherung (→ Kap.0.7, ***a99*** : **`h backp`**) µr s / w

Hardware- oder Softwareschäden können Datenbanken ruinieren. Wer gewissenhaft Sicherungskopien (auf Disketten oder Bandkassetten) macht, hat wenig zu befürchten: ***allegro*** zeichnet alles auf, was an der Datenbank gemacht wird. Wenn der GAU kommt, hat man die Sicherungskopie und die sog. **LOG-Datei**, in der die letzten Änderungen protokolliert sind. Ein eigenes Programm, **UPD** (→ Kap.9), fügt beides wieder zusammen und stellt so den Zustand vor dem Fiasko wieder her. In ***a99***: `h backp` , dann "Datenbank restaurieren" wählen.

9. Programmaufrufe (→ Kap.12) µm

allegro besteht aus mehreren Programmen mit unterschiedlichen Aufgaben (→ Kap.0.4). Diese werden entweder vom ***CockPit*** oder von ***a99*** automatisch gestartet und wissen dann, was sie tun sollen. Der Experte kann ihnen die nötigen Angaben mit eine direkten Aufruf mitgeben. Kleines Beispiel: Der folgende Befehl

```
srch -d abc.alg -e PRINT/liste -s0 -f4 -v0
```

bewirkt einen Aufruf des Volltext-Suchprogramms SRCH (→ Kap.4). Alle sog. "Optionen" beginnen mit einem Bindestrich, das Zeichen hinter dem Bindestrich ist eine Befehlskennung (→ Kap.12.2). Bearbeitet wird die Datei `abc.alg`. Erzeugt wird die Datei LISTE. In der Parameterdatei PRINT.APR ist beschrieben, wie LISTE aussehen soll. `-s0` bedeutet: jeder Satz kommt auf die Liste, `-v0` heißt: keine Anzeige der Daten während des Durchlaufs. Solche Programmaufrufe kann man in **Batch-Dateien** zu umfangreichen Abläufen zusammensetzen. Batchdateien (auch **Stapeldateien** oder bei UNIX **Shell Scripts** genannt) erstellt und bearbeitet man mit einem → Texteditor. PR-LIST.BAT und QUEX.BAT sind Beispiele dafür; diese enthalten mehrere Programmaufrufe, unter anderem auch Sortiervorgänge.
Für die Windows-Programme gibt es statt der Optionen die **INI-Datei**; Beispiel: `a99.ini`.

10. Benutzeroberfläche (Windows-Version: → Kap.2)

Es wurde ein hohes Maß an Flexibilität angestrebt. Menüs und Hilfeseiten sind Textdateien und können mit einem → Texteditor verändert oder sogar übersetzt werden. Mitgeliefert wird auch eine englische Version. Eigene Menüs können auf verschiedene Art erstellt werden. Dazu kann die MS-DOS-Batchprogrammierung (unter UNIX die Shell-Skriptsprache) benutzt werden (→ Kap.12), evtl. angereichert durch zusätzliche Software-Tools.
Ferner gibt es das DOS-***CockPit*** (→ Kap.0.11). Es ermöglicht die Vorbereitung und Steuerung aller Vorgänge über Menüs in einer auch von anderer Software gewohnten Manier. Standardprozeduren wie Neu-Indexieren, Sichern und Wiederherstellen von Datenbanken, Produktion von sortierten Listen u.a. lassen sich dann durch wenige Tasten starten, man kann aber auch eigene Programme und Prozeduren in die Menüs einbeziehen. Das Windows-System hat das Hauptprogramm ***a99.*** Mit der Skriptsprache FLEX kann man auch Oberflächenfunktionen gestalten. Das OPAC-Programm ***alcarta*** ist ein abgespecktes ***a99.*** FLEX ist zugleich die „Job"sprache für alle Web-Oberflächen: damit werden u.a. alle Zugriffe auf die Daten erledigt.

Die Windows-Programme folgen dem gewohnten Standard und sind daher in ihren wichtigsten Funktionen schneller erlernbar. Es ist möglich, auf einem PC dieselbe Datenbank in einem DOS-Fenster zu öffnen und zusätzlich mit einem der Windows-Programme. ***a99*** und ***alcarta*** können sogar die DOS-Programme bei Bedarf nebenbei starten (Menü "Dateien | DOS-Programm" – jedoch nicht mehr auf den 64bit-Windows-Plattformen).

0.1 Allgemeines über Dateien

a99-Online-Hilfe: : **`h dateien`**

Dieser Abschnitt soll besonders Einsteigern eine genauere Vorstellung davon geben, was Dateien eigentlich sind, und wie insbesondere die Titeldaten (ehemals Katalogzettel) gespeichert werden.
Ein Computerspeicher ist nicht einfach ein Behälter voller Daten, in dem sich der Rechner dann schon "irgendwie" zurechtfindet und stets die richtigen Sachen hervorholt, anzeigt oder zu Papier bringt. Vielmehr ist so ein Speicher so etwas wie ein Aktenschrank, in dem viele Ordner für verschiedenste Zwecke stehen, und in jedem Ordner besteht eine zweckmäßige Untergliederung. Es gibt somit bei jedem Rechnersystem genau festgelegte, unterschiedliche Arten von Daten, und dafür muß es jeweils eigene Behältnisse geben, **Dateien** genannt, die sich in Struktur und Inhalt unterscheiden. Jede Datei hat ihren Namen. Damit man schon am Namen einer Datei erkennen kann, welche Art Daten sie enthält, hängt man an den Namen eine **Typbezeichnung** ("**extension**") aus 1 - 3 Zeichen und nennt diese auch kurz den **Typ** einer Datei.
Wichtig: In diesem Buch werden Dateinamen oft mit Großbuchstaben geschrieben, weil sie dann optisch besser auffallen. Unter DOS und Windows kann man sie so oder so eingeben, das macht keinen Unterschied, unter UNIX bestehen jedoch alle ***allegro***-Datei- und Programmnamen vollständig aus Kleinbuchstaben. *Also:* Wenn irgendwo steht `.ALD` oder `CAT.STL` oder dergl., dann ist das in der Realität immer `.ald` bzw. `cat.stl`.

Datensätze

Was früher auf Karteikarten gestanden hätte, speichert ***allegro*** in Dateien vom Typ "`.ald`" bzw. "`.alg`". An diesen Typen erkennt man also auf einer Platte die ***allegro***-Dateien, wenn man sich das Inhaltsverzeichnis ("directory") ansieht. Beide Dateitypen sind so etwas wie Karteischubladen, aber was ist der Unterschied?

`.ald` - Dateien — sind **D**atenbank-Dateien, die für den direkten Schnellzugriff über einen alphabetisch sortierten Index aufbereitet sind. Für diesen Zugriff sind zwei weitere Dateien nötig (siehe unten: Indexdatei vom Typ .ADX und Adressentabelle, Typ .TBL), und für die automatische Erstellung diesesr beiden wird eine Index-Parameterdatei des Typs .API gebraucht.

`.alg` - Dateien — sind dagegen **G**runddateien ohne alphabetischen Schnellzugriff über Index, sozusagen unsortierte Zettelkästen. Deshalb wird, etwas unpräzise, auch von "offline"-Dateien gesprochen. Solche Dateien können als Ergebnis (= Ausgabedatei) einer Volltextsuche (Programme 4 und 1) oder eines Importvorgangs (Programm 5) entstehen, man kann aber auch mit dem Programm 2/3 solche Dateien schreiben und bearbeiten. Mit Hilfe der Programme 7 (INDEX) und 9 (UPD) können solche Grunddateien dann in Datenbank-Dateien umgewandelt oder in eine vorhandene Datenbank eingemischt werden. Sehr wichtig: Grunddateien können sortiert werden (→ Kap.8). Das Windows-Programm *a99* kann solche Dateien in seinen Offline-Speicher laden, wo man sie dann bearbeiten oder in die Datenbank einmischen kann.

Der Buchstabe hinter dem Punkt, das **`a`** von **`.a`**`ld` und **`.a`**`lg`, wird **Konfigurationsbuchstabe** genannt und bezieht sich auf das verwendete Kategoriesystem. Das **`a`** steht für das ***allegro***-Standardschema (→ Anh.B), das vom ehemaligen Schema des Niedersächsischen Katalogisierungsverbundes (sog. NMN-Schema, bis 1992) abgeleitet ist. Wenn man mit einem anderen Schema (einer anderen Konfiguration) arbeitet, z.B. mit **`u`** für "MARC21", lautet der Dateityp "`.ulg`". Wenn in diesem Handbuch von **.ALG-Dateien, .APR-Dateien** usw. die Rede ist, gilt entsprechendes immer auch für .ULG-Dateien, .UPR-Dateien etc. Deshalb wird immer, wenn es nicht speziell um ein bestimmtes Schema geht, *`c`* oder x und `.cld`, `.xlg`, `.cpr` usw. geschrieben statt `.ald`, `.alg`, "`.apr`"... Denken Sie sich also für das *`c`* oder *`x`* jeweils ein **`a`** oder Ihren eigenen Konfigurationsbuchstaben.

Struktur der Datensätze

Jede ***allegro*-Datei** (Typ `.cld` oder `.clg`) kann eine größere Anzahl (viele tausend) **Datensätze** enthalten. Ein Datensatz, oft einfach nur **Satz** oder auch **Aufnahme** genannt, entspricht i.a. bei Buchdaten einer Titelaufnahme, also einem Katalogzettel.
Bei "mehrbändigen Werken" kann so ein Satz aus einem **Hauptsatz** und mehreren **Untersätzen** (Bandaufführungen) bestehen. Letztere können wiederum hierarchisch untergliedert sein, d.h. jeder Untersatz kann seinerseits Untersätze zweiter Stufe enthalten usf. bis zur Stufe 6. Diese Möglichkeit ist eine Besonderheit des Systems ***allegro***. (Ein einfaches Beispiel dafür finden Sie im nächsten Abschnitt unter 0.2.2, wo die Struktur genauer dargestellt ist.)
Ab Version 11.2 können Haupt- und Untersätze auch getrennt gespeichert und über **Satzverknüpfungen** miteinander gekoppelt sein, und dies wird auch empfohlen. Verknüpfungen können seit V14 auch zwischen Sätzen verschiedener Typen bestehen, z.B. zwischen Titelsätzen und Körperschafts-Stammsätzen. Somit ist eine **Mehrdateistruktur** realisierbar (→ 10.2.6.7), d.h. unterschiedliche Satztypen in derselben Datenbank.

Ein Datensatz entspricht zwar einer Titelaufnahme, sieht aber völlig anders aus. Was auf einer Katalogkarte steht, ist für ein Programm ein viel zu wenig strukturierter Text. Programme brauchen die Daten in anderer Form, und zwar zerlegt in logisch sinnvolle Einzelteile. Das sieht so aus:
Jeder Datensatz hat eine variable Anzahl **Datenfelder** (auch **Kategorien** genannt). "Variabel" heißt: so viele wie nötig.
Jede Kategorie kann innerhalb eines Satzes mehrfach auftreten und jeweils bis zu 10.000 **Zeichen** (auch **Bytes** genannt) enthalten. Wenn sie nichts enthält, also in einem Satz nicht mit Inhalt belegt wird, beansprucht sie auch keinen Speicherplatz, und zwar noch nicht einmal ein einziges Byte. Andererseits kann eine Kategorie auch noch in sich unterteilt sein (mit Hilfe von Steuerzeichen) in **Teil-** oder **Unterfelder** (engl. "subfields"), vor allem beim MARC-Format. In diesem Handbuch steht zumeist **Teilfeld**, obwohl Unterfeld sachlich korrekter wäre.
Weitere Einzelheiten zur Struktur der Daten im nächsten Abschnitt.
Damit aber die Programme überhaupt mit den Daten arbeiten können, brauchen sie noch etliche zusätzliche Dateien:

- **Einstellungen** für das Arbeiten an einer Datenbank stehen in einer **`.ini`-Datei** (DOS: `.opt`-Datei für das ***CockPit***)
- Die **Konfigurationsdatei** (Anh.1) beschreibt die Datenstruktur: welche Datenfelder (das **Kategorienschema**) es gibt und andere grundlegende Eigenschaften; `$a.cfg` ist die Standard-Konfigurationsdatei.
- **Parameterdateien** für Indexdefinition, Bildschirmanzeige, Druckerausgabe, und viele andere Zwecke.
- Zu den `.ald`-Dateien einer Datenbank gehören eine **Indexdatei** und eine **Satztabelle**: beide werden automatisch erzeugt und verwaltet, erstere ist jedoch durch den Anwender weitgehend beeinflußbar ("parametrierbar").
- Eine **Kurztiteldatei** kann das Suchen im Index erleichtern: statt größere Treffermengen einzeln durchzublättern, gibt die Kurztiteldatei eine schnelle Übersicht: mit einer Zeile je Titel überblickt man mehr als 20 Datensätze gleichzeitig in einem Fenster. Unter DOS funktioniert eine Datenbank auch ohne die Kurztiteldatei.
- Eine **Restriktionsdatei** ermöglicht das Einschränken von Suchergebnissen: mit Kriterien wie Erscheinungsjahr, Sprache oder Publikationstyp (sofern man solche Angaben erfaßt hat), kann man große Ergebnismengen of sehr wirkungsvoll verkleinern, bevor man sich die Titel ansieht. Für das Funktionieren der normalen Zugriffe ist diese Datei nicht notwendig, manche Datenbanken haben keine.
- Dateien mit den **Menütexten** (UIF-Dateien, z.B. UIF1GER) und Erklärungen zu den Funktionen (H-Dateien oder "HELP-Texte", z.B. H10). Für Windows-Programme: RTF-Dateien mit vielen eingebauten Funktionen.

Es muß dabei immer erkennbar sein, auf welchem Kategorienschema eine Datenbankdatei oder eine Parameterdatei beruht. Deshalb gibt es den schon erwähnten **Konfigurationsbuchstaben** (Standard **`a`**) als Bestandteil des Dateityps.
Diese und weitere Dateitypen werden im folgenden näher beschrieben (→ 0.3), zunächst wird aber versucht, das ***allegro***-System als einen Schichtenbau zu veranschaulichen, auf dessen Ebenen sich gleichartige Objekte befinden, die sich jeweils zu Objekten der nächsthöheren Schicht zusammensetzen. Hierarchische Schichtenmodelle ermöglichen es in der Informatik, auch komplexe Zusammenhänge noch übersichtlich darzustellen (z.B. das OSI-7-Schichten-Modell der Datenkommunikation).
In den Kapiteln 10 und 11, darauf sei hier schon hingewiesen, wird ebenfalls mit einem hierarchischen Schichtenmodell versucht, die umfangreichen Export- und Importparametersysteme überschaubar zu machen.

Bemerkungen zu den Dateien für Kenner
Die Grunddateien vom Typ `.clg` dürfen (im Gegensatz zu den `.cld`-Datenbankdateien) mit den normalen Befehlen des Betriebssystems kopiert und verkettet (COPY), gelöscht (ERASE) und umbenannt (RENAME) werden. Ändern Sie den Typ `.clg` jedoch nicht, weil die Programme daran die Art der Dateien erkennen!
Mit den DOS-Befehlen SORT und FIND kann der Systemkenner durchaus auch gewisse Sortierungen und Auswertungen an `.clg` -Dateien vornehmen, nicht aber an `.cld` -Dateien: diese sollte man niemals eigenmächtig ändern.
Auch der DOS-Befehl TYPE zum Betrachten der Daten auf dem Bildschirm ist möglich, wobei aber die Anzeige ohne Zeilentrennung erfolgt und daher unübersichtlich ist. Besser benutzt man z.B. das Programm SRCH, wobei man als Suchbegriff einfach nur die Ziffer '0' eingibt (die kommt in jedem Datensatz vor); dann laufen die Daten am Bildschirm durch. Mit dem Windows-Programm ***a99*** kann man solche Dateien komfortabel betrachten und bearbeiten (Menü "Datei | Weitere Offline-Datei laden").
Direkte Bearbeitung der `.clg` - oder gar `.cld` -Dateien mit anderer Software (z.B. Textprogrammen) ist zwar sehr problematisch, da es keine übliche Zeilenstruktur gibt (Datenfelder sind unterbrechungslose Zeichenfolgen!), aber es ist auch gar nicht nötig, denn die Exportfunktion (→ Kap. 4 u. 6) kann Dateien von beliebiger Struktur herstellen. Auch solche, die einem Text- oder Tabellenprogramm zugänglich sind. Konfigurations- und Parameterdateien sowie Menü- und Hilfstexte können dagegen mit jedem beliebigen Textprogramm oder ASCII-Editor manipuliert werden (→ 0.3).

0.2 Mehr über Datenspeicherung und -struktur

Online: **h files**

Bevor nun alle Dateitypen im einzelnen genau erklärt werden, folgt hier die Darstellung des erwähnten Schichtenmodells.

Wie behält man in einem Speichersystem die Übersicht? Daß es baumartig angeordnete **Verzeichnisse (directories, Ordner)** gibt, weiß jeder auch nur einigermaßen eingeweihte PC- oder UNIX-Anwender. Was aber steht alles in einem Verzeichnis, d.h. in dem dazugehörigen Plattenbereich? Mit dem Befehl **dir** (MS-DOS) bzw. **ls** (UNIX) sieht man nur eine Liste von Namen mit gewissen anderen Angaben, unter denen man Datum und Uhrzeit noch erkennt, den Rest aber ohne Vorkenntnisse nicht zu enträtseln vermag. Relativ leicht lernt man bei Windows und DOS, wie Programme und Dateien auseinanderzuhalten sind (die Typen .EXE, .COM und .BAT sind Programme), aber dann hört die Anschaulichkeit auf: irgendwelche Hinweise über Inhalt und Funktion der Dateien und deren Zusammenhänge und Abhängigkeiten untereinander sind den nackten Namen nicht anzusehen, schon gar nicht, wie es drinnen aussieht in den Dateien, wie sie also in sich strukturiert sind. Windows versucht mehr Anschaulichkeit mit den Icons zu erreichen (im "Explorer"), das ist jedoch nicht perfekt sondern kann sogar trügerisch sein (das Icon gehört nicht wirklich zur Datei und es beruht nur auf dem Datei*typ*, z.B. .DOC, nicht auf dem tatsächlichen Inhalt).

0.2.1 Dateien-Hierarchie

Es ist hilfreich, wenn man das ganze Dateisystem mit dem Umfeld, das zur ***allegro***-Software gehört, in eine Anzahl von Schichten oder Ebenen gliedert, die inhaltlich und funktional hierarchisch aufeinander aufgebaut sind. Das sieht so aus:

Gesamter Datenraum auf dem Rechnersystem : Programme und Dateien aller Art, angeordnet in **Verzeichnissen (directories)**, etwa wie Aktenordner

allegro-System : "Ordner" mit allen Programmen, Systemdateien, Anwenderdateien und Hilfsdateien, die zu ***allegro*** gehören; meistens C:\ALLEGRO

Anwender-**Datenbanken** mit anwendungsspezifischen Parameterdateien, also quasi Unterabteilungen des ***allegro***-Ordners. Eine Datenbank kann, muß aber nicht, in einem Unterverz. des ***allegro***-Programmverzeichnisses stehen. Eine Datenbank kann eine Gruppe von bis zu 255 .xld-Dateien umfassen.

Einzelne ***allegro*-Datenbank** dbn: 4teiliger Kern mit den Typen .cld, .cdx .tbl und .stl, dazu Parameterdateien, mindestens dbn.cpi und d-*.cpr, und datenbankspezifische Hilfsdateien (UIF... und H...-Dateien)
Jede Datenbank sollte ein eigenes Verzeichnis haben, z.B. c:\allegro\katalog

einzelne Anwender**datei** (Typ .cld) = Kette von N **Satzgruppen**:
Satzgr.1 CR **Satzgr.2** CR **Satzgr.3** CR ... **Satzgr.N** CR

einzelne **Satzgruppe**
= **Hauptsatz [+ hierarchische Untersätze]**

einzelner **Satz** : Kette von **Datenfeldern**, z.B.
1xxxx**20 Quantum reality**0**40 Herbert, Nick**0**74 London**0**75 Wiley..**

einzelnes **Datenfeld (Kategorie)** = Zeichenfolge
= Kategorienummer Text Endezeichen 0
z.B. **40 Herbert, Nick**0

CR ist ein Steuerzeichen (Carriage Return) und trennt zwei Datensätze. Es gibt noch weitere Steuerzeichen, die der Anwender ebenfalls nicht kennen muß, z.B. Feldendezeichen, hier als 0 dargestellt, und Satzanfang/Hierarchie-Kennung, hier als 1 dargestellt. Jeder Datensatz in einer Datei des Typs .ALD hat ausserdem, hier als xxxx dargestellt, eine interne Satznummer aus vier Bytes direkt hinter dem Code 1 . Grunddateien (Type .ALG) sehen genauso aus, nur ohne die interne Satznummer. Auf der nächsten Seite wird dies noch genauer gezeigt.
Zu einer Datenbank können bis zu 255 Dateien des Typs .ald gehören. Jede Datei kann bis zu 16 MB oder einem Vielfachen davon wachsen. Bei der Erfassung kann man bei jedem neuen Datensatz die Datei anwählen, in die er geschrieben werden soll (Kap. 1.5.4, Befehl F). Jeder Datensatz kann beliebig viele Felder und Untersätze haben und insgesamt bis zu 40K groß sein.
Am Programmverzeichnis, z.B. c:\allegro, hängen die beiden Verzeichnisse help und flex mit den Hilfedateien und den FLEX-Skripten. Datenbankverzeichnisse können auch ganz woanders liegen.

0.2.2 *allegro*-Datenstruktur

Dieser Abschnitt beschreibt noch etwas genauer die Struktur der "eigentlichen" Dateien, d.h. der Typen `.ald` und `.alg`, in denen die Anwenderdaten stehen.
Wer muß das wissen? Wer "nur" katalogisiert oder einen ***allegro***-Katalog benutzt, kann diesen Abschnitt übergehen. Dafür interessieren muß sich aber, wer Fremddaten importieren, d.h. in ***allegro***-Daten umwandeln will (Kap. 5 und 11) und wer ***allegro***-Daten mit eigener Software erstellen will, als sog. "ASCII"-Datei, um sie dann mit dem Programm UPD oder mit dem ***avanti***-Server in eine Datenbank einzubringen.
Normal erfaßte Datensätze stehen als zusammenhängende Zeichenfolge in einer Datei des Typs `.ald`. (Das Wort "zusammenhängend" ist wichtig: der Datensatz ist nicht in logische oder physische Teile oder Blöcke oder Tabellenspalten o.ä. zerlegt, sondern bildet eine einzige Zeichenfolge, mit Code 0 als Trennung zwischen den Feldern) Und wenn ein Satz beim Bearbeiten verlängert wird? Dann merkt das die Software (PRESTO, UPD, *a99*) und verlagert den gesamten Satz an eine andere, passende Stelle; wenn es innerhalb der Datei keine gibt, an das Ende der Datei. Der "alte", kürzere Datensatz wird vorn mit dem ASCII-Code 09 statt 01 gekennzeichnet und erhält einen Indexeintrag im Register 1 unter "//" mit seiner Länge (in Bytes) als fünfstelliger Zahl dahinter, z.B. `//00249` für einen Datensatz, der 249 Byte lang ist. Diese Schlüssel braucht das Programm, um geeignete Stellen zum Speichern neuer oder verlängerter Sätze zu finden, also den freigewordenen Platz wieder sinnvoll zu nutzen.
Und so sieht ein Datensatz in einer Grunddatei aus: (Typ `.alg`)

```
[1]00 654321[0]20 Was können wir wissen? [0]31 Evolutionäre Erkenntnistheorie[0]40 Vollmer,
Gerhard[0]74 Stuttgart[0]75 Hirzel[0]76 1985[0]77 Bd.1.2.[0]
[2]1 1=Bd. 1[0]20 ¬Die¬ Natur der Erkenntnis : Beiträge zur Evolutionären Erkenntnisthe
orie[0]25 Mit einem Geleitw. v. Konrad Lorenz[0]77 337 S.[0]87 3-7776-0403-8[0]90 2647-3611[0]
[2]1 2=Bd. 2[0]20 ¬Die¬ Erkenntnis der Natur : Beiträge zur modernen Naturphilosophie[0]77
350 S. [0]87 3-7776-0404-6[0]90 2650-9705[0] [cr]
```

Ganz korrekt ist dieser Eindruck nicht, denn es gibt in dieser Datei keine Zeilentrennungen: alles steht in einer langen Zeichenfolge hintereinander, nur ganz am Ende stehen die ASCII-Codes 13 10 ([cr]) Die Codes [0], [1] und [2] sind Steuerzeichen, und zwar die ASCII-Codes 00, 01 und 02, nicht die Ziffern '0', '1' und '2'. Sie bezeichnen das Feldende, den Satzanfang und den Untersatz-Anfang. Hier ist ein hierarchischer Satz mit zwei Untersätzen zu sehen! Es könnte weitere Unterstufen geben, die dann mit [3]02 ..., [4]03 ... etc. beginnen. Empfohlen wird dies nicht, das Arbeiten damit ist schwierig.
Derselbe Satz kann für die neuere Software (***avanti***-Server, Windows-Programme) auch so aussehen: (zum externen Bearbeiten und zum Einlesen aus einer Datei; innerhalb der Datenbankdatei bleibt es bei der oben gezeigten Struktur.)

```
#00 654321
#20 Was können wir wissen?
#31 Evolutionäre Erkenntnistheorie
#40 Vollmer, Gerhard
#74 Stuttgart
#75 Hirzel
#76 1985
#77 Bd.1.2.
#01 1=Bd. 1
#20 ¬Die¬ Natur der Erkenntnis : Beiträge zur Evolutionären
 Erkenntnistheorie
#25 Mit einem Geleitw. v. Konrad Lorenz
#77 337 S.
#87 3-7776-0403-8
#90 2647-3611
#01 2=Bd. 2
#20 ¬Die¬ Erkenntnis der Natur : Beiträge zur modernen Naturphilosophie
#77 350 S.
#87 3-7776-0404-6
#90 2650-9705
```

Dateien mit dieser Struktur, sog. "Externdateien", müssen den Typ **`.cdt`** haben, damit die Programme sie erkennen.
Keine Leerzeilen innerhalb eines Satzes!
Jedes Datenfeld muß hier auf einer neuen Zeile, mit einem # auf der ersten Position, beginnen. Zeilenumbrüche sind nicht nötig, d.h. jedes Feld kann eine einzige lange Zeile sein. Wenn jedoch Umbrüche auftreten, müssen diese bei einem Leerzeichen stattfinden, und die Folgezeile muß mit diesem Leerzeichen beginnen. Wenn man diese Form mit anderer Software aus Fremddaten herstellen kann, kann das ***allegro***-System sie einlesen. Die Zeichen müssen dem internen Code entsprechen, also DOS-ASCII, Windows-ANSI oder UTF-8!
Die Programme merken beim Einlesen selbst, und zwar an den Feldern #00 und #01, wo ein Satz bzw. Untersatz beginnt. (In einem anderen Schema, wie z.B. `$p.cfg`, sind es andere Nummern, z.B. #0100 und #70XX.) . Nur wenn es keine solchen Felder zur Markierung des Satzbeginns gibt, sind zwei Sätze durch eine Leerzeile zu trennen, sonst merkt das Programm es nicht.

0.3 Dateitypen

Tip für a99: **`h files`** und **`h adm`**

Wozu dieser Abschnitt?
Alle Dateitypen sind hier zusammengestellt und kommentiert, mit denen ***allegro*** arbeitet. Das ist wichtig für Systemverwalter und Entwickler. "Normalanwender" werden diese Kenntnisse nicht so dringend brauchen. Wenn Ihnen aber einmal eine Datei auffällt, warum auch immer, können Sie hier nachschlagen, was es damit auf sich hat.
Nochmals der Hinweis: unter UNIX sind alle Namen und Typen vollständig klein zu schreiben, also z.B. `cat.api` statt `CAT.API`, für DOS und Windows ist es egal. Hier sind es Großbuchstaben, die Namen sind dann besser erkennbar. Das kleine **`c`** in den Namen steht jeweils für den Konfigurationsbuchstaben, im Standardfall ist es **`a`**.

Sie finden hier Angaben zu den Aspekten (und Antworten auf die Fragen):

Inhalt	der Dateien ("Was steht drin?"),
Struktur	("Wie sind die Dateien aufgebaut?"),
Bezug / Standort	("Womit hängt das zusammen? Wo müssen/können diese Dateien liegen?"),
Funktion	("Wozu wird das gebraucht?"),
Entstehung	("Woher kommt die Datei, wie entsteht sie oder wer legt sie an?"),
Zugriff	("Wer oder welches Programm kann/darf/muß daran etwas machen?"). Hauptsächlich geht es dabei um den **Schreibzugriff**.

Nicht alle Angaben sind für das Arbeiten mit ***allegro*** zum Verständnis notwendig, das Handbuch soll aber jedem an Details Interessierten eine möglichst vollständige Dokumentation bieten.
Ein anschauliches Bild der wichtigsten Zusammenhänge zeigt Anhang C. Eine weitere Übersicht in 0.8 zeigt, welche Dateien zu einer konkreten Datenbank gehören.
Für andere Programme zugänglich, vor allem Textprogramme, sind nur die Typen, bei denen als Struktur "ASCII-Textdatei" angegeben ist. Unter ***CockPit*** ist angegeben, wie man bequem an die Dateien herankommt (→ 0.11).

Typ	Bedeutung/Inhalt:	Zugang über ***CockPit***

`.CFG` Konfigurationsdatei (→ Anhang A) **`µo k`**

`.CFL`

Inhalt: **Kategorienschema** (z.B. A.CFG als Standard, D.CFG für "MAB2", U.CFG für MARC21), Angaben zu **Datumsfeldern, IdNummern, Dialogsprache, Abfrageliste** ("Erfassungsmaske") für die Dateneingabe. Ferner die Liste der am Titelanfang zu übergehenden Artikel.

Struktur: ASCII-Textdatei

Bezug: Wichtig ist der erste Buchstabe des Namens, der **Konfigurationsbuchstabe** genannt wird. Alle von einem Kategorienschema c.CFG abhängigen Dateien haben den Konfigurationsbuchstaben 'c' als ersten Buchstaben ihrer Typenbezeichnung. Z.B. arbeiten die Parameterdateien der Typen .APR oder .API nur mit der Konfiguration A.CFG zusammen (siehe unten). In Netzwerken ist es auch eine Möglichkeit, jedem Mitarbeiter "seine" .CFG-Datei mit eigener Abfrageliste auf sein eigenes Verzeichnis zu legen. Denn:

Standort: Die Programme suchen die .CFG zuerst auf dem Datenverzeichnis, dann auf dem Verzeichnis, von dem gestartet wurde, und wenn sie dort auch nicht steht, auf dem Programmverzeichnis.

Funktion: Alle Programme mit Ausnahme von ASORT und QRIX suchen und lesen die Konfigurationsdatei beim Start, um zu erfahren, wie die Anwenderdaten strukturiert sind. Als Standard nehmen alle Programme A.CFG an. Mit dem DOS-Befehl `set -k=p` kann man z.B. P.CFG als Standard vorgeben. Die Option -k (→ Kap.12.1) ermöglicht die Wahl einer spezifischen Konfiguration bei jedem Programmaufruf. Es kann unterschiedliche Versionen einer Konfiguration geben: wenn z.B. mehrere Bearbeiter mit verschiedenen Abfragelisten und Änderungsberechtigungen arbeiten sollen, erstellt man dafür je eine Konfiguration. Nur der Anfangsbuchstabe des Namens muß derselbe sein, also z.B. A1.CFG, A17.CFG oder ADELHEID.CFG. Alle drei beziehen sich auf dasselbe Kategorienschema A.CFG und können mit denselben Parameterdateien der Typen .API und .APR betrieben werden.
Eine gleichnamige Datei mit dem Typ `.cfl`, also z.B. `a.cfl`, ist eine funktional nicht notwendige Hilfsdatei. Sie dient nur für a99 und acon dazu, einen Datensatz mit Feld- und Unterfeldbezeichnungen anzeigen zu können. Ausprobieren: in a99 `X cfga` eingeben. In der Datei `a.cfl` erkennt man leicht, wie die nötigen Angaben aussehen müssen.

Entstehung: Mehrere werden mitgeliefert (A, D, M, PICA, U) und können beliebig verändert werden. Eigene, neue Konfigurationen können erstellt werden (→ Anh.A).

Zugriff: Mit jedem Texteditor zu erstellen und zu bearbeiten.

Typ Bedeutung/Inhalt: Zugang über ***CockPit***

`.INI` Initialisierungs-Datei (nur für die Windows-Programme ***a99/alcarta***)

Inhalt: Angaben zu einer Datenbank und ihrer Benutzung

Struktur: Eine für Windows-Software übliche Liste von Schlüsseln, die die Eigenschaften der Datenbank und die Modalitäten ihrer Benutzung beschreiben. Eine vollständige Dokumentation findet man in der Datei **`a99.ini`**, die man auch als Vorlage für eigene Versionen nehmen kann.

Zugriff: Wenn `xyz.ini` die Angaben zu einer Datenbank enthält, kann man diese mit dem Befehl `a99 xyz` bzw. `alcarta xyz` öffnen. Zu jeder Datenbank kann es mehrere INI-Dateien mit unterschiedlichen Benutzungsmerkmalen geben. Der Name `xyz` muß nichts mit dem Datenbanknamen zu tun haben.

`.cLD` *allegro*-Datenbank-Datei **`µr b`**

Inhalt: In solchen Dateien stehen die "echten" Daten ("Titelaufnahmen"), die der Anwender erfaßt hat.

Struktur: Datensätze sind sequentiell angeordnet, die Felder in der Reihenfolge der Kategorienummern, wie durch die .CFG-Datei festgelegt. Jeder Satz beginnt mit der automatisch von 1 aufsteigend vergebenen internen Satznummer (4 stellig, hexadezimal). Ansonsten stimmt die Dateistruktur mit derjenigen der Grunddateien (siehe .cLG) überein.
Die Einzelheiten der Struktur sind in 0.2 beschrieben.

Bezug: Zum Zugriff auf die Daten werden die .cDX- und die .TBL-Datei gebraucht. Zu einer Datenbank *dbn* kann es bis zu 255 .cLD-Dateien geben. Die Dateinamen setzen sich aus *dbn* und der Nummer zusammen: z.B. `cat_17.ALD`. Der Name *dbn* darf deshalb nicht länger als 4 Zeichen sein.

Standort: Diese Dateien stehen zusammen mit der .TBL-Datei in einem eigenen Ordner, dem Datenbankverzeichnis des Anwenders. Hier stehen in der Regel auch die .ADX- und die .STL-Dateien, jedoch könnten diese auch anderswo gelagert sein.

Funktion: Datenspeicherung. Der Platz zum Speichern wird automatisch verwaltet von den Programmen PRESTO bzw. a99, acon und UPD, mit denen die Daten verändert werden können.

Entstehung: Entsteht beim Aufbau einer Datenbank aus vorhandenen Daten mittels Programm INDEX und wird bei der Datenerfassung im Programm PRESTO oder a99 oder acon automatisch erweitert.

Zugriff: Für Bearbeitung und evtl. Export: Programm 1 (PRESTO) und 9 (UPD), *a99* und *avanti*;
lesend mit oder ohne Export auch per Volltextsuche (Programm SRCH); zum Aufbau und Ergänzung einer Datenbank mit Programm INDEX.

`.cDX` Index-Datei (→ Kap.1.4, 7, 10) **`µr o i/c`**

Inhalt: Suchbegriffe (Zugriffs-Schlüssel) zu einer Datenbank, aufgeteilt in bis zu 11 Register, die sich alle in einer einzigen solchen Datei befinden. Die maximale Größe dieser Datei reicht aus für mehrere Millionen Datensätze. Mit **MultiX** wurde es ab V17 möglich, mehr als eine solche Datei anzulegen. Dazu gibt es eine eigene Beschreibung: `h multix` eingeben in a99.

Struktur: "Präfix-B*-Baum" : das ist eine in der Informatik speziell für Indexdateien entwickelte Struktur. Hier ist sie realisiert mit variabler Schlüssellänge und "Frontkomprimierung" (spart Platz: je größer die Datenbank, umso mehr). Zu jedem Schlüssel sind die internen Nummern der zugehörigen Datensätze gespeichert.

Bezug: Arbeitet nur mit einem Schema c.CFG zusammen. Zu jeder Datenbank *dbn* gehört eine Index-Datei *dbn*.cDX, eine Index-Parameterdatei *dbn*.cPI und eine Satztabelle *dbn*.TBL, ferner als Hauptteil eine oder mehrere Dateien *dbn*_k.cLD. Dabei ist k eine Zahl zwischen 1 und 255.

Standort: Die Datei muß auf dem Datenverzeichnis stehen, mit Option -y kann ein anderes vorgegeben werden (siehe Kap.12).

Funktion: Nötig für den Datenbank-Schnellzugriff (Kap.1); kann inhaltlich nach Wunsch des Anwenders individuell gestaltet werden (Kap.7 und 10). Dazu dient die Parameterdatei des Typs .cPI (s.u.).

Entstehung: Wird durch INDEX erstellt und durch PRESTO und UPD während des Zugriffs automatisch verwaltet. Erneuerung möglich durch ***CockPit*** **`: µr o i`** bzw. in a99 : `h org`

Zugriff: Nur automatisch per PRESTO, a99, acon, UPD, INDEX, SRCH und QRIX. Das letztgenannte ist einerseits ein Hilfsprogramm, das von INDEX aufgerufen wird, es kann aber auch einige nützliche Funktionen an einem Index selbständig ausführen (→ Kap.7.5).
Von eigenmächtigen, schreibenden Zugriffen mit Editoren, Software-Tools oder selbstgeschriebenen Programmen auf eine .ADX-Datei ist abzuraten.

Typ Bedeutung/Inhalt: Zugang über ***CockPit***

.TBL Satztabelle **µr o t**

Inhalt: 4 Byte je Datensatz für die Zuordnung der Satznummer zur realen Adresse.
Die ersten zwei Bytes am Anfang der Datei dienen Steuerzwecken. Das erste Byte ist normalerweise 0, während eines Schreibvorgangs wird es auf 1 gesetzt, damit niemand gleichzeitig in die Datei schreiben kann. Per ***CockPit*** kann man es verändern: **µr o s** setzt es auf 1 und sperrt damit die Datenbank für Schreibzugriffe, **µr o e** (entsperren) gibt sie wieder frei. Das gleiche tun die Hilfsprogramme UNFREE und FREE und der FLEX-Befehl **set tbl** .
Das zweite Byte hat den Wert 3. Es ist aber größer, wenn die maximale Dateigröße mehr als 16MB sein soll (siehe dazu Kap.10.2.1.3, Befehl `ii=k`). Steht da 4, entspricht dies `ii=2`, damit kann jede Datendatei 2x16MB groß werden, usw.

Struktur: Sequentiell in aufsteigender Reihenfolge der Satznummern. Wenn N eine Satznummer ist, findet man an der Position 4*N-2 die Adresse des Satzes in Form von 4 Bytes: Dabei ist das erste Byte die Dateinummer (1..255), die anderen 3 Byte geben die Position des Satzes in der Datei an.

Bezug: Sie bildet eine funktionale Gesamtheit mit .cDX und .cLD! Von der Konfiguration hängt die Tabelle nicht ab - sie würde bei jeder Konfiguration gleich aussehen. Daher kommt 'c' nicht in der Typbezeichnung .TBL vor.

Standort: Bei den .cLD-Dateien, zu denen sie gehört, also im Datenbankordner.

Funktion: *dbn*.TBL ist Bindeglied zwischen *dbn*.cDX und den Datenbank-Dateien *dbn*_k.cLD, indem sie jeder Satznummer die reale Adresse zuordnet. Der Zugriff geht so: Im Index findet PRESTO die Satznummer, und ein Zugriff auf die .TBL-Datei liefert dann die Adresse.

Entstehung: Wird automatisch mit der Datenbank aufgebaut (per Programm 7 = INDEX) und automatisch aktualisiert. Erneuerung möglich durch ***CockPit*** **µr o t** bzw. *a99*`:h org`

Zugriff: Nur automatisch aus den Programmen 1, 4, 7 und 9 sowie `acon`.
Von jedem direkten Eingriff durch den Anwender wird abgeraten.

.STL Kurztitelregister **µr o k**

Inhalt: Eine bis zu 250 Zeichen lange Zeile für jeden Datensatz. (Nur die ersten 72 sind in APAC, PRESTO zu sehen, in ***a99/alcarta*** mehr)

Struktur: Sequentiell in aufsteigender Reihenfolge der Satznummern, ohne Steuerzeichen. Die Position der Kurzzeile zu einem Datensatz ergibt sich aus dessen Satznummer multipliziert mit der für den Kurztitel vorgegebenen Länge (Parameter `i0` in der Index-Parameterdatei). Daher kann es je Datensatz nur genau eine Kurzzeile geben. Die Zusammensetzung der Kurzzeile ist mit den Mitteln der Parametrierung frei gestaltbar (→ Kap.10). Für die Demo-Datenbank wurde die Konstruktion "Sachtitel /Verfasser Jahr Signatur" gewählt.

Bezug: Die Datei bildet eine funktionale Gesamtheit mit .cDX, .cLD und .TBL. Im Gegensatz zu diesen ist sie jedoch entbehrlich: wenn sie fehlt, gibt es eben keine Kurzanzeige.
Standort: siehe .cDX (und Kap.12, Option **-y**).

Funktion: Die Kurzzeile wird angezeigt, wenn die Kurzanzeige per <Shift+F8> oder <Shift+F9> verlangt wird (→ 1.4) bzw. im Ergebnismengenfenster von ***a99/alcarta***.
Eine Datenbank funktioniert auch ohne .STL-Datei (bis Version 11.2 gab es keine).

Entstehung: Entsteht automatisch mit der Datenbank (per Programm 7 = INDEX) und wird, wo nötig, automatisch aktualisiert. Allerdings müssen in der Index-Parameterdatei dafür besondere Vorkehrungen getroffen sein (→ 10.2.6.3): es müssen die Befehle `i0=72` und `ak=zz+0` vorhanden sein und in der Kategorieliste ein Abschnitt mit der Sprungmarke `#-0`, wo geregelt ist, welche Angaben in der Kurzliste stehen sollen. (Sonst hat man keine Kurzanzeige)
Erneuerung jederzeit möglich durch ***CockPit*** **µr o k** bzw. *a99*`:h org` *Kurzanzeige erneuern*

Zugriff: Nur automatisch aus den Programmen 1, 7 und 9 sowie `acon`.
Direkte Eingriffe sind nicht sinnvoll, können allerdings keinen Schaden an den Daten anrichten, auch keine Folgeschäden außer fehlerhaften Kurzzeilen.

Typ Bedeutung/Inhalt: Zugang über ***CockPit***

.RES Restriktionsdatei `µr o x`

Inhalt: für jeden Datensatz eine feste Anzahl von Zeichen.
Struktur: wie .STL, nur mit entsprechend anderem Inhalt.
Bezug: Die Datei bildet eine funktionale ***Gesamtheit*** mit .cDX, .cLD und .TBL. Im Gegensatz zu diesen ist sie jedoch entbehrlich: wenn sie fehlt, gibt es eben keine Restriktionen.
Standort: siehe .cDX .
Funktion: Die Angaben in dieser Datei werden gebraucht, wenn eine Suche eingeschränkt werden soll, z.B. nach Kriterien wie Erscheinungsjahr oder Sprache (→ 1.4.6).
Entstehung: entsteht automatisch mit der Datenbank (per Programm 7 = INDEX) und wird, wo nötig, automatisch aktualisiert. Allerdings müssen in der Index-Parameterdatei dafür besondere Vorkehrungen getroffen sein (→ 10.2.9): es müssen die Befehle `ir=k` und `ak=zz+/` vorhanden sein und in der Kategorieliste ein Abschnitt mit der Sprungmarke `#-/` .
Erneuerung: jederzeit möglich durch ***CockPit*** **`: µr o x`** bzw. `a99:h org`
Zugriff: nur automatisch aus den Programmen 1, 7 und 9 sowie `acon`.
Direkte Eingriffe sind nicht sinnvoll, können allerdings keinen Schaden an den Daten anrichten, auch keine Folgeschäden außer fehlerhaften Zugriffen.

.cLG *allegro*-Grunddatei (→ Kap. 3.2) `µf 2/3`
.cDT Externdatei (→ Kap. 0.2.2)

Inhalt: Anwender-Datensätze, strukturiert gemäß Konfiguration c.CFG
Struktur: Der Aufbau ist derselbe wie bei der .cLD-Datei, es fehlen aber die Satznummern. (Einzelheiten siehe 0.2.2 und 11.0.1. Struktur .cDT ist leichter mit externer Software zu erzeugen.)
Bezug: Mit Hilfe des Programms INDEX kann man .cLG-Dateien in eine Datenbank umwandeln bzw. in eine vorhandene Datenbank einspeisen. Diese Dateien haben keinen festen Standort., weil sie nicht zu einer Datenbank gehören.
Funktion: .cLG-Dateien gehören nicht zu einer Datenbank. Sie sind deshalb in der Regel ein Zwischenprodukt für die Weiterverarbeitung mit SRCH oder INDEX ("Export"). Grunddateien können sortiert werden (mit dem Programm ASORT, siehe 0.4). Die meisten Verfahren der Listenproduktion erzeugen zuerst sortierfähige Grunddateien und exportieren diese nach dem Sortieren erneut mit SRCH. *Beispiele:* PR-LIST.BAT oder QUEX.BAT.
Entstehung: als "download" (Funktion F4) aus einer Datenbank, als Ergebnis einer Volltextsuche oder eines Imports (Ausgabe-Parameter S-*.cPR oder I-1.cPR), oder auch durch manuelle Eingabe und Bearbeitung (Hauptfunktion 2 = Programm ALLEGRO, "offline"-Erfassung).
Zugriff: Lesend mit SRCH (Volltextsuche) und INDEX (Datenbankaufbau);
Schreibende Bearbeitung nur sequentiell ("offline") mit dem Programm a99.
Mit IMPORT können strukturelle Umwandlungen, etwa Änderungen an den Kategorien, vorgenommen werden (→ 11.2.3.1, Pauschal-Import). Das bedeutet, ***allegro***-Daten können für grundlegende Änderungen quasi re-importiert werden:
UPD und **a99** können eine .cLG-Datei lesen und in eine Datenbank einmischen, wobei vorhandene Datensätze durch neue ersetzt und neue Sätze hinzugefügt werden (siehe Kap. 9). Die Struktur .cDT kann auch mit jedem Texteditor bearbeitet werden.
Tip: Mit Befehl `rdwr 1 1000000 X.alg X.txt H1 0` Umwandl. in ASCII-Datei.

.cIM Import-Parameter-Dateien (→ Kap.5 und 11) `µd i..`

Inhalt: Beschreibung eines fremden Datenformates und genaue Anweisungen, wie dieses in das ***allegro***-Format c.CFG umzuwandeln ist. Z.B. enthält OCLC.UIM die Beschreibung der Umwandlung von OCLC-Daten in das Schema U.CFG (= USMARC-Standardschema).
Struktur: ASCII-Textdatei, mit jedem Texteditor zu erstellen und zu bearbeiten.
Bezug: *Standort:* gesucht werden diese Dateien zuerst auf dem Quellverzeichnis (wo die einzulesenden Fremddaten stehen), dann auf dem Programmverzeichnis.
Funktion: Steuerung der Datenumwandlung im Programm 5 (= IMPORT).
Entstehung: einige werden mitgeliefert (DBDISK.AIM, OCLC.AIM, ...), weitere können vom Anwender beliebig erstellt werden.
Zugriff: mit jedem Texteditor zu erstellen und zu bearbeiten. Neben den Befehlen können beliebige Kommentare darin vorkommen.

Typ Bedeutung/Inhalt: Zugang über ***CockPit***

.cPx Export-Parameter-Dateien (→ Kap. 10) **µd e ..**

Inhalt: Genaue Vorschriften über die Form einer Datenausgabe (sog. "Export"), formuliert in der ***allegro***-Exportsprache.

Struktur: ASCII-Textdatei.

Bezug: Jedes Programm außer dem Sortierprogramm (Nr. 8) benötigt für die Ausgabe von Daten Parameterdateien.
Standort: Die Programme suchen jede solche Datei zuerst auf dem Datenverzeichnis (Quellverzeichnis), wenn sie dort nicht ist, dann auf dem Startverzeichnis (wo man das Programm aufgerufen hatte), und zuletzt noch auf dem Programmverzeichnis.

Funktion: Jede Ausgabe, ob Bildschirmanzeige, Papierausdruck oder "download" (auch Indexdatei!), wird durch Parameter gesteuert. Auswahl und Reihenfolge der Datenfelder, Interpunktion, Zeichenumwandlungen, Zeilen- und Seitenumbruch u.v.m. können gesteuert werden.

Entstehung: einige werden mitgeliefert (D-1.APR, I-1.APR, E-1.APR S-*.APR, P-*.APR, Prototypen x-0.APR), weitere können auf dieser Grundlage vom Anwender beliebig erstellt werden.

Zugriff: mit jedem Texteditor zu erstellen und zu bearbeiten. Neben den Befehlen können beliebige Kommentare vorkommen.

Das ***x*** steht für die 3 Typen **R, T** und **I** dieser Dateien: (→ Kap.10 und 6/7)

.cPR eigentliche **Produktions**-Parameterdatei.

Die Programme 1 - 3 benutzen automatisch die Parameterdatei D-1.cPR für die formatierte Anzeige auf dem Bildschirm. Wenn Sie diese Datei verändern, bekommen Sie eine andere Standardanzeige. I-1.cPR produziert eine Ausgabe im Original-***allegro***-Format, S-*.cPR erstellen sortierfähige Dateien. Wenn I-1 zur Erstellung einer Exportdatei benutzt wird, bekommt diese automatisch den Typ .cLG, da es sich um Dateien mit der ***allegro***-Grundstruktur handelt, die sich für eine Weiterverarbeitung z.B. mit dem Programm SRCH oder zur Einspeisung in eine andere Datenbank mittels INDEX eignen.

.cPT **T**abelle : kann solche Teile enthalten, die in mehreren Parameterdateien identisch sind (z.B. eine Zeichentabelle für die Druckersteuerung (→ 10.2.4) oder eine Stopwortliste (→ Kap.10.2, Ladebefehl t). Grundsätzlich kann eine .cPT-Datei alles enthalten, was in einer .cPR- oder .cPI-Datei stehen kann. Die sog. **Druckertreiber** werden mit .cPT-Dateien realisiert. Diejenigen .APT-Dateien auf der Lieferdiskette, deren Namen mit **P-** beginnen, sind Beispiele für Druckertreiber (→ 0.11.5, 10.2). Sie sind unabhängig von der Konfiguration, d.h. man kann z.B. P-EPSON.APT in U-EPSON-PPT umbenennen und hat den geeigneten Druckertreiber für das U-Schema.

.cPI **I**ndex-Parameterdatei (→ Kap.7 u. 10) : könnte auch **Datenbank-Definition** genannt werden; dient zur Produktion von Indexdateien und für den Schnellzugriff zur Aktualisierung der Indexdatei.
Prototyp: INDX.API. Diese können Sie als Muster verwenden. Sie enthält alle für eine Index-Parameterdatei wichtigen Teile und ist ausführlich kommentiert.

Prototypen
Für die wichtigste Unterabteilung, die .cPR-Dateien, wurde eine Klassifizierung in 10 Typen mit festgelegten Namen entworfen, z.B. SORT für die Aufgabe der Sortieraufbereitung, DISPLAY für die Bildschirmanzeige. Am Anfang von Kapitel 10 steht die Tabelle dieser Typen. Für Konfiguration A wurde sodann zu jedem Typ eine Parameterdatei geschrieben, die alle charakteristischen Merkmale hat und auch eine genau beschreibbare, nachvollziehbare Aufgabe ausführt. Diese Dateien werden als **Prototypen** bezeichnet, weil man sie als Muster für eigene Entwürfe benutzen kann. Die Namen dieser Dateien bestehen aus dem Anfangsbuchstaben des Typs und "**-0**", also z.B. **S-0.APR** und **D-0.APR**. Wenn man im ***CockPit***-Menü "Dateien / Exportparameter" die Funktion "Neue Datei vom selben Typ erstellen" wählt, bekommt man eine Kopie des Prototyps und hat damit schon die wesentlichen Teile beisammen, die man für die jeweilige Aufgabe braucht.

.FLX/.JOB **Skripte** für a99/alcarta bzw. acon. Siehe Kap. 2.3.2 Doku: **h flex**

FLEX ist die Skriptsprache des Windows- und Web-Systems. Damit werden alle höheren Vorgänge programmiert, bis hin zu Subsystemen wie Ausleihe und Erwerbung. Solcherlei Funktionen sind also gar nicht fester Bestandteil des Systems, sondern sind eigene, frei veränderliche Unterprogramme in der Sprache FLEX.

Typ Bedeutung/Inhalt: Zugang über ***CockPit***

.cPH Phrasen-Datei (Windows: **PHRASE.A99**) µd p

Phrasen sind Textbausteine, die man sich zur Erleichterung der Erfassungsarbeit anlegen und speichern kann. Phrasen sind unabhängig vom Kategoriesystem, daher ist der Typ immer .APH (→ Kap.3.5.3, Befehl #p). Phrasendateien sind ASCII-Textdateien.
Die während einer Sitzung definierten Phrasen werden bei Verlassen des Programms unter dem Namen **PHRASE.APH** gespeichert und bei der nächsten Sitzung automatisch wieder geladen. Man braucht also nur dann eigene Phrasendateien anzulegen, wenn man für besondere Arbeiten ganz bestimmte Phrasen benötigt.
Standort: PHRASE.APH wird auf dem Startverzeichnis des Nutzers abgelegt. Im Mehrplatzbetrieb richtet man am besten für jede Erfassungsperson ein eigenes Verzeichnis ein, von dem aus dann der Start erfolgt. Auf dem nämlichen Verzeichnis werden dann also die ganz persönlichen Phrasen abgelagert.

.ASP Tabelle für das Suchprogramm (→ Kap.4) µd s

Sie dient dazu, z.B. die Umlaute aufzulösen und die Groß-/Kleinschreibung bei der Suche zu ignorieren und Sonderzeichen zu eliminieren. Ohne diese Datei müßten Sie den Suchbegriff ganz korrekt eingeben. Im Bereich der Sonderzeichen kann es sinnvoll sein, wenn Sie an dieser Datei Änderungen vornehmen. Diese Datei ist unabhängig von der Konfiguration und heißt immer `S1.ASP`. Als ASCII-Datei ist sie mit jedem Texteditor zu bearbeiten.

.LOG Sicherungsdatei ("Logbuch") zu einer Datenbank µr s/w

Alle Veränderungen (Transaktionen) werden darin protokolliert. Nach Datenverlust kann das Programm UPD eine BACKUP-Kopie mit der .LOG-Datei wieder zu einer aktuellen Datenbank zusammenführen (→ 0.7). Mit dem Zusatzprogramm LOG2ALG kann daraus eine Grunddatei erstellt werden (→ 0.4).
Die Datei dbn.LOG entsteht automatisch bei der Arbeit an der Datenbank dbn. Sie können und sollten daran keine direkten Eingriffe vornehmen.
Standort: Datenbankverzeichnis. Mit Option -L*dateiname* kann sie anderswo liegen.

H... HELP-Texte (ohne Typbezeichnung). µd h

.RTF

Sie können diese Texte ebenfalls verändern und eigene selbst schreiben. Für jede Datenbank *dbn* können Sie sich eigene Dateien (jeweils bis zu 24 Zeilen) unter den Namen H*dbn* bzw. HC*dbn* anlegen, die dann durch Shift+F1 vom Funktionsmenü (→ 1.5) bzw. vom ***allegro***-Editor aus aufgerufen werden. Wenn diese Dateien fehlen, werden H1D bzw. H1C statt dessen aufgerufen.
Standort: wie Parameterdateien.
Die RTF-Dateien sind nur für ***a99/alcarta***., die H-Dateien für die DOS-Programme. Sie können auf C:\ALLEGRO\HELP liegen. **h help** eingeben!
Eine Sprachbezeichnung kann an den Namen angehängt werden, z.B. H10ENG für die englische Version von H10. Wenn eine andere Sprache als Deutsch (GER) eingeschaltet ist, werden zuerst die zu dieser Sprache gehörigen Hilfeseiten gesucht. Wenn z.B. H10ENG nicht existiert, wird H10 geladen.
Für jede Kategorie #xyz kann man eine Hilfeseite anlegen.
Mitgelieferte Standard-Hilfeseiten für das DOS-System:
In dieser Tabelle bedeuten: a = Zugriffsberechtigung (0..3 = Option -a), i = momentanes Register,
k = Kategoriegruppe (0..9), *dbn* = Datenbankname

Taste	ANZEIGE	REGISTER i	EDITOR (→ Kap.3.5.3)
F1	HDSPa, H10 Anzeigefunktionen	HINXa, H11 Registerfunktionen	H Editorbefehle
ShF1	H*dbn*, H1D Datenbank-Hilfe	Hii, H*dbn*, H1D Registerspezifische Hilfe	HC*dbn*, H1C Datenbankspez. Hilfe
StrgF1	H1B Blätterfunktionen	H12 Trunkierung, Endemarken	H0B weitere Editor-Hilfe
AltF1	H1F Anzeige-Funktionstastenbelegung	H1I Register-Funktionstastenbelegung	H0A weitere Editor-Hilfe

Sind zwei Namen angegeben, sucht das Programm zuerst die erste und nimmt ersatzweise die zweite.
Außerdem gibt es noch die Hilfeseite HFM für die Auswahlliste (→ Kap.0.5), ferner kann es Dateien HRESi geben (i = 1,2,..), und zwar Hilfeseiten für die Restriktionen. Diese erscheinen in APAC, wenn bei der Menü-Eingabe einer Restriktion F1 gedrückt wird (→ 10.2.9).

Typ Bedeutung/Inhalt: Zugang über ***CockPit bzw. a99***

`UIF...` **User Interface File = Menütexte** (ohne Typbezeichnung → 0.6) **`µd m`**

Diese Dateien enthalten die Texte der Menüs, Fehlermeldungen, Fragen und Instruktionen. Jeder Anwender kann daran Änderungen vornehmen, jedoch nur an den Texten, nicht an den als Zahlen und Sonderzeichen erscheinenden Steuerzeichen (z.B: %s oder %d). UIF1ENG enthält z.B. den englischen Text zum Programm 1 = PRESTO. Die Datei UIFA... ist für APAC und ALFA, UIFC... für ***CockPit***, UIFE... für *a99/alcarta*
Bearbeitung als Textdatei mit jedem Texteditor.
Standort: wie Parameterdateien, d.h. datenbankspezifische Exemplare legt man ins Datenverzeichnis.

`ii`*`nnnn`* **Zwischendateien (temporär) bei der Indexproduktion**: **`µf q`**

Der Anwender bekommt sie normalerweise nicht zu sehen, da sie nach erfolgreicher Indexierung verschwinden. Wenn aber eine Indexierung nicht korrekt durchgelaufen ist oder INDEX mit manueller Unterbrechung gestartet wurde, sind evtl. diese Dateien vorhanden, nicht aber die erwünschte Indexdatei von Typ .cDX. Das Programm QRIX mit Option -fq kann die II-Dateien dann zu der .cDX verarbeiten (→ 7.5) Wie der Befehl genau aussehen muß, steht dann in der Datei PROTOK. Hinweis: Mit einem Hilfsprogramm `qr.exe` kann man sich den Inhalt einer ii-Datei als Text ausgeben lassen. *Standort:* Datenverzeichnis.

`.BAT` **Stapeldatei**: (unter UNIX entspricht dies den "shell scripts") **`µm b`**

kein ***allegro***-, sondern ein MS-DOS-Dateityp, auch "Batch File" oder "Stapeldatei" genannt. Eine solche Datei, mit jedem Texteditor zu erstellen und zu bearbeiten, kann beliebige MS-DOS-Befehle und Programmaufrufe enthalten. Die ***allegro***-Programme (→ 0.4) können in solche Skripte eingebettet werden. Alle Variationen (Optionen) der Programmaufrufe werden in Kap.12 genau beschrieben. Stapeldateien bieten eine Möglichkeit, eigene Menüsysteme oder Arbeitsumgebungen zu schaffen, indem man häufig benötigte Abläufe (z.B. Listenproduktionen, Auswertungsprogramme, Importvorgänge) und Datenbankaufrufe zusammenfaßt.
Standort: Im Prinzip beliebig, meistens auf dem Programmverzeichnis.

`CP.OPT / DEFAULT.OPT`

`CP.PRE` **Vorgaben für *CockPit*** (d.h. für das DOS-Programm ACP.EXE): **`µo v`**

Darin stehen alle anwenderseitig änderbaren Einstellungen des ***CockPit***. Auch dies ist eine ASCII-Textdatei. Die mitgelieferte CP.OPT enthält Kommentare (abgedruckt in → 0.11.6).
Standort: Diese Datei wird auf dem **Start**verzeichnis gesucht, d.h. dort, wo man ACP.EXE oder CP.BAT gestartet hat. Hängt man die Aufrufoption **-oxyz** an, nimmt ***CockPit*** die Datei xyz (Name völlig beliebig) als Vorgabendatei, sonst wird nach CP.OPT gesucht, wird sie nicht gefunden, nach DEFAULT.OPT.
Man startet das Programm ACP.EXE normalerweise mit Hilfe von CP.BAT, d.h. man gibt den Befehl **cp**. Darin steht ein Aufruf `acp ... -oCP.OPT ....`
Alle anderen Programme, z.B. PRESTO, werten diese Datei nicht aus, nur ACP.EXE.
In der Datei CP.PRE stecken ein paar Vorgaben für den nächsten CockPit-Start. In dieser Datei bewahrt sich ***CockPit*** die aktuellen Einstellungen (Datenverzeichnis, Datenbankname, Konfiguration) auf, wenn man aussteigt. Wenn ACP dann wieder gestartet wird, liest es CP.PRE, so daß man da fortfahren kann, wo man zuletzt gearbeitet hatte. Kann ohne Bedenken gelöscht werden.

`LISTE` **Standard-Ausgabedatei** **`µr e 4/5`**

Wenn man per ***CockPit*** Listen produziert (→ Kap.6), entsteht jedesmal eine Datei dieses Namens, d.h. bei jeder erneuten Produktion wird sie überschrieben. Form und Inhalt hängen von den gewählten Parametern ab.

`.FLX/.JOB` Skripte für ***a99/alcarta*** bzw. für `acon`. Siehe Kap. 2.3.2 **`h flex`**

`FILELIST.RTF` **Dateiliste für die Windows-Programme** **`h filelist`**

Alle für ***a99/alcarta*** notwendigen Dateien sind in dieser Liste dokumentiert. Man findet darin Angaben, wozu die Dateien dienen, wo sie stehen müssen oder *können*, in welche man eingreifen muß etc.

`.VW` ViewListen: ein Konzept des Windows-Systems. **`h view`**

***`dbn`*`.sgf`** **Signaldatei**. Beginnt sie mit der Ziffer '1', verweigern die Direktzugriffs-Programme a99 und alcarta den Dienst. Als Meldung für den Nutzer wird dann der Text ausgegeben, der hinter der '1' steht.
Standort: Datenverzeichnis.

Tabelle der Dateien

Online-Hilfe: **h files**

dbn steht für den Datenbanknamen des Anwenders (bis zu 4 Zeichen!).
Wichtig: Die wirklichen Namen bestehen alle vollständig aus **Kleinbuchstaben**.
Wenn man nicht mit A.CFG arbeitet, sondern z.B. mit D.CFG, muß es .DPR statt .APR heißen… (K = gehört zum Kernsystem)

Name.Typ	K	Bedeutung und Funktion
$A.CFG oder **A.CFG; A.CFL**	x	Standardkonfiguration; enthält die Definition des Kategorienschemas, außerdem die Abfrageliste. Ab Version 13 (`$A.CFG`) erheblich erweiterte Möglichkeiten. `A.CFL`: Hilfsdatei f. a99, optional
CCC.BAT/org.bat		wird von ***CockPit*** / ***a99*** automatisch erstellt und startet den gewünschten Prozeß
name.**INI**	x	Initialisierungsdatei für **a99/alcarta** : Einstellungen für die Benutzung einer Datenbank
CP.OPT	x	Vorgaben für das DOS-***CockPit***, anwenderspezifische Einstellungen (früher DEFAULT.OPT)
***.FLX / *.JOB**	x	Skripte (sog. FLEXe) des Windows-/Web-Systems (Dokumentation: **h flex** eingeben)
***.VW**	x	ViewListen des Windows-Systems (Dokumentation: **h view** eingeben)
H... / *.RTF	x	Hilfetexte; vom Anwender erweiterbar; Typ .RTF nur für ***a99/alcarta***
D-WRTF.APR	x	Standard-Anzeige-Parameterdatei für a99/alcarta.. (DOS-PRESTO: `D-1.APR`)
PR-LIST.BAT	x	Stapeldatei für Listenproduktion im DOS-System. Wird von ***CockPit*** gestartet, wenn die Routine "exportieren/1" gewählt wird. Die mit F4 exportierten Daten werden verarbeitet und unter dem Namen LISTE ausgegeben.
SR-LIST.BAT	x	Funktioniert wie PR-LIST.BAT, nur mit Volltextsuche statt Schnellzugriff
P-*.APR	x	Parameterdateien für Druckformatierung; Auswahl per ***CockPit***, gewählte Datei wird kopiert auf PRINT.APR
P-*.APT	x	Druckertreiber; Auswahl per ***CockPit*** "Optionen/Drucker", gewählte Datei wird kopiert auf PRINTER.APT
S-*.APR	x	Parameterdatei für die Sortierung von Listen. Auswahl per ***CockPit***, gewählte Datei wird kopiert auf SORT.APR
S-*.APT	x	Parameterdatei für die Vollständigkeit (Kategorieumfang) einer Liste. Auswahl per ***CockPit***, gewählte Datei wird kopiert auf SELECT.APT
UIFi*lng*	x	"User Interface Files" = Textdateien mit den Menütexten für die einzelnen Programme. Vom Anwender veränderbar. i=Programmkennung, *lng*=Sprachcode
***.AIM**	x	Import-Parameterdateien (ASCII-Text) für die Umwandlung von Fremddaten
dbn`.API`	x	Index-Parameterdatei für die Datenbank *dbn* (z.B. CAT.API)
dbn_n.**ALD**		Datendatei, enthält die erfaßten Datensätze, n=1..255 (Felddefinitionen in $A.CFG)
***dbn*.ADX**		Index, enthält alle Register, die in der .API des Anwenders definiert sind
***dbn*.TBL**		Tabelle d. Satzadressen, stellt für Zugriff Verbindung zw. .ADX und .ALD her
***dbn*.STL**		Kurztitel-Register. Definition in .API
***dbn*.RES**		Restriktionen-Datei (ab V15). siehe →10.2.9
***dbn*.APT**		Überschriften der Register. Wichtig bei Datenbankwechsel mit Alt+A (→10.2.1.3) in PRESTO
***dbn*.SGF**		Signalfile. ***a99*** startet nicht, wenn '1' an erster Position dieser Datei.
***dbn*.LOG**		Aufgezeichnete Transaktionen; nötig zur Rekonstruktion nach Absturz
***.ADT**		Externdatei mit Zeilenstruktur, einfacher als .ALG, lesbar für *a99, alcarta, avanti* (→ 0.2.2).
***.ALG**		Grunddatei (wie .ALD, aber ohne Satznummern); entsteht z.B. bei Import
S1.ASP	x	Tabelle der Zeichenumwandlung für die Volltextsuche (Prog.4 = SRCH.EXE)
`*.TTF / *.FON`		Schriften für DOS bzw. Windows (→ Anh. E)

0.4 Programme

Wozu dieser Abschnitt?
allegro-C ist kein einzelnes großes Programm, sondern es ist ein **Softwarepaket**, das aus etlichen Programmen und vielen verschiedenen Dateien (→ 0.3) besteht. Das Kernsystem von etwa 10 Programmen wird durch einige Spezialprogramme ergänzt. Dieser Abschnitt stellt die Kernprogramme vor. Jedes wird in einem der Kapitel 1 bis 9 genau beschrieben. Einige obsolete Programme des DOS-Systems sind auch noch erwähnt, weil sporadisch noch im Einsatz.
Die Dateinamen sind in der Wirklichkeit komplett klein geschrieben, hier nur die neueren, für 16bit-DOS nicht geeigneten!
Alle DOS/Windows-programme haben den Dateityp `.exe`, die UNIX-Versionen nicht.
Zu den Windows-Programmen (***a99*** und ***alcarta***) → Kap.2.
Die Programme werden von Fall zu Fall auf unterschiedliche Art aktiviert:

Entweder [DOS] Man startet das ***CockPit*** mit dem Befehl **`cp`** (→ 0.11), dann erscheint das Menü "Routinen"; mit <Alt>+f schaltet man ins Menü "Funktionen" und ruft jedes Programm mit seiner Ziffer auf; dann sind vor dem Start in einem Untermenü einige Felder auszufüllen, ***µf***

oder [DOS] Man ruft nur das gerade benötigte Programm in einem Kommandofenster mit seinem Namen auf und beantwortet die Fragen, die es stellt,

oder [auch Windows] Man stellt Stapeldateien (UNIX: Shell-Skripte) zusammen (→ Kap.12), in die man die Programmaufrufe mit den für die eigene Anwendung nötigen Optionen einbettet. Dies kann bis zur Gestaltung eigener Menüoberflächen gehen. Die Windows-Programme kann man per Icon vom Desktop starten.

Die Beschreibung eines Programms mit seinem Ablauf und seinen Funktionen steht im entsprechenden **Kapitel mit derselben Nummer**. Die Windows-Programme werden im Kap.2 dargestellt. Kapitel 3 beschreibt den integrierten Editor für die Dateneingabe und -bearbeitung. Dieser ist in die DOS-Programme 1, 4 und 5 eingebaut.

0 **`ACP`** ***CockPit*** (→ Kap.0.11) *[nur DOS]*
Die "Benutzeroberfläche" für die Steuerung aller Vorgänge einer DOS-***allegro***-Umgebung
Funktionen: **Einstellen** und Bearbeiten von Optionen und Parametern
Start von Programmen und Prozessen (Stapeldateien)
Routineaufgaben (Datensicherung, Wiederherstellung bestimmter Dateien)

1 **`PRESTO`** **Schnellzugriff**
`MENUED` **... mit Menü-Erweiterung** *nur DOS*
`APAC` **OPAC-Programm**
`ALFA` **... mit Einfach-Ausleihe (s.u.)**
APAC ist eine PRESTO-Version nur für den lesenden Zugriff, geeignet als OPAC-Programm.
Wichtig für Netze: APAC und alcarta benötigen keine Berechtigung für Schreibzugriff.
Zugriff auf: Datenbankdateien (Typ .cLD) in Kombination mit Indexdatei (Typ .cDX, .STL und .RES) und Satztabelle (Typ .TBL)
Zugriffsart: Schlüssel (Namen, Stichwörter, Titel, Schlagwörter...) mit logischer Kombination durch UND, ODER, NICHT
Funktionen: **Retrieval** mit **Browsing** (die eingeschränkte Version APAC enthält nur diese Funktionen)
Bearbeitung und formatierte **Ausgabe** gefundener Datensätze
Erfassung neuer Daten mit sofortiger Index-Aktualisierung

2 **`allegro`** *nur Windows*
Das Windows-Start- und Hauptprogramm
`a99` identisch mit ALLEGRO
`alcarta` alcarta.exe ist das Windows-OPAC-Programm
Zugriff und Zugriffsart wie PRESTO/APAC
Beide Programme bauchen eine INI-Datei, in der die notwendigen Angaben stehen.
Beide können Makro-Routinen abarbeiten, die in der Sprache FLEX geschrieben sind.

3 **`avanti`** **Serverprogramm + Konsolprogramm** *Win und Linux/Solaris*
+ **`acon`** Notwendig für Web-Anwendungen, acon führt Aufträge aus
Dokumentation: `http://www.allegro-c.de/avanti` , in a99: `h acon`
`acon` kann ab V32 das Programm UPD voll ersetzen. Es braucht dazu das Skript `update.job`. (Hilfe: `h acon` in a99 eingeben)
Zugriff auf: *allegro*-Datenbanken
Funktionen: Ausführung von FLEX-Jobs, von avanti über das Web oder mit direktem Aufruf

4 `srch` Volltext-Suche und Export [ab 2012 auch als 32bit-Progr., alte Version: srch16]

Zugriff auf: Dateigruppen
Zugriffsart: sequentiell (auf .cLG- und .cLD-Dateien)
Funktionen: **Volltext-Suche** nach Zeichenfolgen, mit logischer Kombination.
Beliebige Ausgabe (Export) der Ergebnisse, fungiert deshalb zugleich als **EXPORT-Programm** mit oder ohne Selektion. Ein eigenes Programm namens EXPORT gibt es deshalb nicht.

5 `import` Fremddaten-Konvertierung [ab 2012 als 32bit-Programm, alte Version: import16]

Zugriff auf: sequentiell auf beliebig strukturierte Fremddaten
Funktionen: Umwandlung fremder Daten in ***allegro***-Dateien (Typ .cLG) mit Hilfe von Import-Parameterdateien (Typ .cIM). IMPORT schreibt nicht in die Datenbank!
Zum Einordnen in eine Datenbank (Typ .cLD) ist anschließend Programm 7 = INDEX zu benutzen (s. Kap.7) oder UPD (s. Kap.9).

7 `index` Index-Produktion [ab 2012 als 32bit-Programme; alte Versionen: index16, qrix16]

q `qrix` Index-Management

Zugriff auf: Dateigruppen (Typen .cLG und .cLD)

Zugriffsart: sequentiell

Funktionen: Generierung einer Datenbank mit Index-Datei (Typ .cDX), Satztabelle (Typ .TBL), Kurztitelregister (Typ .STL) sowie Datenbankdateien (Typ .cLD). QRIX wird von INDEX bei Bedarf automatisch aufgerufen, hat aber auch eigene Funktionen.
INDEX-Teilfunktionen:

- Einarbeitung neuer Dateien in eine Datenbank (Option `-f7`);
- Neu-Indexierung einer vorhandenen Datenbank (mit oder ohne Beibehaltung der alten Satznummern: Optionen `-f7, -fn, -fi, -ft, -fs`);
- Wiedergewinnung von ungenutztem Platz in .cLD-Dateien (`-fr`)
- QRIX: Index kompaktieren, einzelne Register herausnehmen

8 `asort` Sortierprogramm [ab 2012 als 32bit-Programm; alte Version: asort16]

Zugriff auf: einzelne Dateien (***allegro***-Grund- und Extern-Dateien oder auch schlichte Textdateien)
Funktion: alphanumerisches Sortieren von beliebig großen Dateien mit variabel langen Sätzen bzw. Zeilen. Das Programm kann durch andere, z.B. GNU-Sort, ersetzt werden.

9 `upd` Datenbank-Management [ab 2012 als 32bit-Programm; alte Version: upd16]

[früher **`UPDATE`**; ab 2010 in *Win/UNIX*: startet `acon.exe mit update.job`]

Zugriff auf: .cLD- .cDX- und .TBL-Dateien mit Ein-/Ausgabe,
.LOG- bzw. .cLG-Dateien als Eingabe

Zugriffsart: Satznummern bzw. "Primärschlüssel"

Funktionen: a) **Playback :** Datenbank-Regenerierung nach "Absturz"
b) **Mischen :** Einspeichern neuer Daten (.cLG) mit Ersetzung von vorhandenen Sätzen durch neue (sonst → 7).

c) **Check :** "Dublettenkontrolle"

Auf den folgenden zwei Seiten werden weitere Programme erwähnt, die nicht alle zum Kernsystem gehören.
1992 erstmals ausgeliefert wurde das Programm **ORDER**, mit dem man für die **Monographienerwerbung** die Funktionen Bestellkatalogisierung und Akzession automatisieren kann. Für die Ausleihe gab es ***aLF*** (= ***allegro***-Leih-Funktionen). Diese Programme waren Erweiterungen des Programms 1, hatten also dieselben Zugriffsmöglichkeiten. Beide Programme sind ab 2008 durch FLEX-Pakete ersetzt, mit denen das Windows-Hauptprogramm ***a99*** dieselben Funktionen übernehmen kann. Ein weiteres Paket, **`ZAboM`**, wurde für die Zeitschriftenverwaltung entwickelt. Diese Pakete sind in der Standard-Installation alle enthalten. Mehr dazu: in a99 eingeben **`h alf`**, **`h order`** bzw. **`h zabom`**. Es gibt jeweils einen Hilfetext (auf HILFE..HILFE klicken).

Zusatzprogramme

(bis auf LOG2ALG nur für DOS verfügbar)

Unter Windows werden diese Programme zunehmend durch die FLEX-Technik in das Hauptprogramm ***a99*** integriert. Die DOS-Programme werden deshalb nicht weiterentwickelt.

aLF — Ausleihprogramm

(Hierzu existiert eine eigene Dokumentation: **`h alf`**)

Zugriff auf: Datenbank *dbn* wie PRESTO ab 2008 neue Realisierung mit FLEX in a99, voll kompatibel

ORDER — Erwerbungsprogramm

(eigene Dokumentation. Online: **`h order`**)

Zugriff auf: Datenbank *dbn* wie PRESTO ab 2008 neue Realisierung mit FLEX in a99, voll kompatibel

ALFX / ORDERX — Ausleih- bzw. Erwerbungs-Systemdaten pflegen **[ab 2008 obsolet]**

Menügeführte Erfassung und Pflege aller Systemdaten. In a99 neu realisiert mit FLEX.

ALFA — Einfach-Ausleihprogramm

(nur Ausleihe/Rücknahme. ***a99***: **`h alfa`**)

Arbeitet wie APAC, wird genauso gestartet. Mit TAB wird die Ausleihfunktion ausgelöst. Mit <Alt>+F10 schaltet man zwischen Ausleihe und Rücknahme um. Benutzer und Datum werden einfach an das Feld #90 angehängt. Wenn es ein anderes sein soll, z.B. #91: Start mit **`-C#91`**. Benötigt wird UIFAGER (Darin auch Kommentare). ***a99***: Flips in D-WRTF.APR.

[REF] — Referentenprogramm

(hierzu existiert eine eigene Dokumentation) **nur bis V28**

Zugriff auf: Datenbank *dbn* wie PRESTO (wird genauso gestartet)
Zugriffsart: wie PRESTO
Funktionen: a) Erfassung und Pflege von Sacherschließungs-Stammdaten: Schlagwort-Normsätze, Klassifikation, Aufstellungssystematik, Sprach- und Regionalschlüssel.
b) Eingabe und Bearbeitung aller Sacherschließungselemente zu Titelaufnahmen. Dabei kann z.B. die SWD als Ressource benutzt werden.
Benötigt wurde UIFRGER (Darin auch Kommentare für Anpassungen)

[INVENT] — Inventarisierung von Exemplaren und Bänden **nur bis V26**

Zugriff auf: Datenbank *dbn* wie PRESTO
Zugriffsart: wie PRESTO
Funktionen: Erfassung und Bearbeitung von Exemplarsätzen und Zeitschriften-Einzelbänden, vor allem für das Ausleihsystem, aber auch von Bestandssätzen für Zeitschriften. Die Datensätze werden als getrennte Sätze gespeichert, die Verknüpfung zum Titelsatz geschieht über die Titel-Identnummer. Die Sonderdunktionen werden immer mit der TAB-Taste ausgelöst.

cfgconv — Alte CFG (vor V13) in neue verwandeln

(s. Anhang A.5)

Zugriff auf: Eine Konfigurationsdatei `x.cfg` in alter Struktur (vor V13)
Funktion: Eine `$x.cfg` in neuer Struktur wird daraus erstellt.

log2alg — .LOG-Datei in .ALG-Datei umwandeln

ab V33 32bit, auch UNIX

(Online: **`h super`** , dann LOG)

Zugriff auf: Die Protokolldatei *dbn*.LOG der Datenbank *dbn*
Zugriffsart: sequentiell
Funktion: Die Datensätze, die als neue, geänderte und gelöschte Sätze während der Arbeit an einer Datenbank in der Datei *dbn*.LOG protokolliert wurden, werden in eine Grunddatei *dbn*.ALG überführt. Dabei verschwindet die interne Satznummer, die nur vom Programm UPD für die Wiederherstellung nach einer Zerstörung gebraucht wird. Gelöschte Sätze erhalten ein zusätzliches Feld `#u1 @@@@@`, neue Sätze werden mit `#u1 #####`*nnn* gekennzeichnet (*nnn* = Dateinummer, in der der Satz gespeichert wurde). Wenn es einen Primärschlüssel gibt, kann man *dbn*.ALG mit `upd -fm11` in eine andere Datenbank einmischen lassen, um dort dieselben Vorgänge nachzuvollziehen. *dbn*.ALG hat außerdem den Vorteil, daß man es mit dem Volltextsuchprogramm SRCH durchsuchen kann.

Hilfsprogramme (die meisten sind EXE-Programme, einige ab Win'7/64 hinfällig)

ANSI.COM **Bildschirmtreiber für DOS-Fenster** [16bit]
Damit die Farben u.a. richtig dargestellt werden, muß vor dem Start eines der DOS-Programme dieses Hilfsprogramm gestartet worden sein; es ist eingebaut in das Startprogramm cp.bat (→ Kap. 0.10)

AW **Zeichensatz-Tabelle einblenden** (DOS)
Residentes Programm, das mit Alt+w jederzeit aktiviert werden kann. Es erscheint die Tabelle des aktuellen Zeichensatzes. Mit dem Cursor kann man jedes Zeichen auswählen, mit <Enter> holt man es in den aktuellen Bildschirm.

GK **Get Key (Tastendruck abwarten,** DOS**)** (Beispiele siehe in Kap. 12.2.4)
In jeder DOS-Stapeldatei einsetzbar. Aufruf z.B. [ersetzbar durch `choice /c`]

```
echo  Weitermachen? j / n
gk /G jn
```

Nur mit 'j' oder 'n' geht es weiter. ERRORLEVEL ist dann 1 bzw. 2.

JUMPS **Sprungmarken auflisten** (von Heinrich Allers) [16bit]
Dieses Programm liest .CFG-Dateien sowie Parameterdateien (Typen .cPR und .cPI) und stellt die darin vorkommenden Sprungbefehle und Sprungmarken fest. Ergebnis ist eine Tabelle der vorkommenden Sprungmarken mit Fehlerprotokoll (Sprünge ohne Ziel, Marken ohne Sprung, ...)

rdwr **Lesen + Schreiben** (ab V33: 32bit, auch für Linux/Solaris)
Dieses Programm liest eine sequentielle Datei (Zeilentrennung mit Code 10 = LineFeed) und schreibt ihren Inhalt ab einer vorgegebenen Zeile und schreibt dann den Inhalt bis zu einer anderen vorgegebenen Zeile in eine neue Datei. Der Aufruf erfolgt mit

`rdwr FROM TO INFILE OUTFILE [Hn] [0]`

Die Zeilen *FROM* bis *TO* werden aus *INFILE* in die Datei *OUTFILE* kopiert. Will man .ALD oder .ALG-Dateien in ASCII-Dateien überführen, gibt man zusätzlich H5 bzw H1 an, um die Steuerzeichen am Satzanfang zu übergehen, sowie **0**, um einen Zeilenvorschub am Feldende zu bekommen. Als Satzende gilt 13 10 oder 10 (UNIX), d.h. in .ALD- und .ALG-Dateien gilt ein ganzer Datensatz (nicht jedes Datenfekd) als"Zeile". Man schreibt "nul" statt *OUTFILE*, wenn man nur die Anzahl der Zeilen der Datei wissen will.

qr **ii-Datei entschlüsseln** (32bit, auch Linux/Solaris)
Liest eine Index-Zwischendatei `ii`*k* und gibt sie als Textdatei aus (k=1,2,…).

ACREPLY.COM **Benutzereingabe abwarten**. (Beispiele siehe in Kap. 12.2.4 und z.B. in PRONTO.BAT)
In jeder Stapeldatei einsetzbar. Die Eingabe des Benutzers (Zeichenfolge, abgeschlossen mit <Enter>) wird in eine Batchdatei namens acantw.bat geschrieben. Wenn man diese dann aufruft mit call acantw.bat, wird die Antwort in die Environment-Variable REPLY kopiert.

SNIFFER **Konsistenz einer Datenbank untersuchen** [nur DOS 16bit] ***a99***: `h check`

Zugriff auf: Die Dateien *dbn*.TBL, *dbn*_*.*c*LD, *c*.CFG
Zugriffsart: sequentiell. Ersatz für Win: Menü `h check`
Funktionen: Das Programm kann prüfen, ob die in der .TBL stehenden Adressen der Datensätze stimmen, ob in den .cLD-Dateien die Satzanfänge korrekt sind, und ob die Datensätze keine unerlaubten Kategorien enthalten. Entdeckte Fehler werden protokolliert. Außerdem kann SNIFFER gesperrte Sätze finden und wieder freigeben.

UNFREE/FREE **Datenbank sperren / wieder freigeben** (*a99*: **`h org`**)
Wenn man nicht mehr speichern kann, weil endlos die Meldung "bitte warten" kommt, dann braucht man FREE.EXE: **`free dbn.tbl`** als Befehl eingeben, dbn=Name der Datenbank.
Ersatz in *a99*: `h org, FLEX:  set tbl loc/free`

Tabelle der Programme

Die folgende Seite ist als Kurzreferenz gedacht und faßt alle Programme nochmals tabellarisch zusammen, und zwar mit Angabe ihrer Funktion und der zugehörigen Dateien. Außerdem wird angegeben, was zum Kernsystem gehört (K) und ob es nur unter Windows läuft (w), und welche Programme auch für UNIX/Linux verfügbar sind (X). Mit (P) sind (ganz rechts) alte Programme gekennzeichnet, die von PRESTO abgeleitet waren und dieselben Voraussetzungen brauchten.

Tabelle der Programme (K=Kernsystem, w=Windows, d=DOS16, X=UNIX-Version, s = Sprachcode, z.B. GER)

Die Nur-DOS-Programme sind hier in Großbuchstaben geschrieben; unter UNIX dieselben Namen ohne `.exe`

Name	K	X	Funktion	Dazu nötige Dateien
ACP.EXE **CP.BAT**	**kd**		CockPit-Programm (DOS-Startplattform) Damit startet man es	CP.OPT, UIFCs,
acon.exe	**kw**	x	Konsolprogramm zum Ausführen von Jobs	`avanti.con, *.job`
a99.exe **alcarta.exe**	**kw**		Windows-Programme (a99 = allegro) `alcarta.exe` identisch mit `opac.exe`	*.INI, UIFEs, *.RTF, *.FLX
ALF.EXE **ALFA.EXE**	**d** **kd**		Ausleih-Programm (DOS, aufwendig) [obsolet] Einfach-Ausleihe (nur Ausgabe/Rücknahme)	UIFLs, (P) UIFAs
ALFX.EXE	**d**		Hilfsprogramm: Bearbeitung der aLF-Systemdaten [obs.]	
allegro.exe	**kw**		Das Windows-Hauptprogramm, identisch mit a99 [Bis 1998 war ALLEGRO.EXE ein anderes Programm]	*.INI
APAC.EXE	**k**		allegro-OPAC-Programm	UIFAs, (P)
asort.exe	**k**	x	Sortierprogramm für .ALG- u. ASCII-Dateien	UIF8
avanti.exe	**k**	x	Datenbankserver f. Web-Anwendungen, braucht acon	UIFSs, avanti.con
a30.swf	**w**	x	RIA-Programm auf Basis Adobe Flex	*.job / avanti+acon
cfgconv.exe	**k**	x	Umwandlung alte x.CFG (bis V12) neue `$x.cfg`	
import.exe	**k**	x	Umwandlung von Fremddaten in `.alg`-Daten (Einmischen macht UPD oder INDEX!)	UIF0s, UIF5s, .CFG, .AIM, I-1.APR o.a.
index.exe	**k**	x	Aufbau einer Datenbank aus .ALG-Dateien, Erneuerung des Index einer Datenbank	UIF0s, UIF7s .CFG, .API
log2alg.exe	**k**	x	.LOG-Datei in .ALG-Datei wandeln	
ORDER.EXE	**d**		Monographien-Erwerbung (DOS, obs.); dazu gehört:	UIFos (P)
ORDERX.EXE	**d**		Hilfsprogramm: Bearbeitung der Systemdaten [obsolet]	
PRESTO.EXE **REF.EXE**	**kd**		Datenbank-Zugriff und -Bearbeitung MENUED = Overlay-Version (bei Speichermangel)	UIF0s, UIF1s, .CFG, D-1.APR, .API, UIFR..
qrix.EXE	**k**	x	Hilfsprogramm für INDEX.EXE, außerdem: Kompaktieren / Auslisten eines Index	UIFQs, .API
srch.EXE	**k**	x	Volltextsuche und Export	UIF4xs .CFG, (.APR), S1.ASP
upd.EXE	**k**	x	Neue Daten oder .LOG-Datei einspeisen, `+update.job`	UIF0/9s, .CFG, .API
Hilfsprogramme:				
GK.COM	**kd**		Tastenabfrage in Stapeldatei (DOS, ERRORLEVEL)	
JUMPS.EXE	**kd**		Sprungbefehle und -marken untersuchen	
rdwr.exe	**k**	x	Seqentielle Dateien zeilenweise lesen + schreiben	
ACREPLY.COM	**kd**		zum Einsatz in Stapeldateien (Benutzereingabe abwarten, in Environment-Variable speichern)	
SNIFFER.EXE	**d**		Untersuchen von Datenbanken auf Inkonsistenzen	SNIFFER.INI
X.EXE	**d**		Editor für ASCII-Dateien (Makros in X.MAC)	

0.5 Auswahl-Listen (DOS)

Wann immer es in den DOS-Programmen notwendig ist, aus einer Anzahl von Dateien eine oder mehrere auszuwählen, geht dies mit demselben Verfahren. Die Windows-Programme haben dafür das übliche Verfahren, das man von anderen Programmen kennt: eine Dateiauswahl-Box.
(Nur im ***CockPit*** geht die Dateiauswahl etwas anders, und zwar mit einem "Leuchtbalken", → 0.11.4).

1. Oft erscheint eine solche Liste automatisch, wenn das Programm schon "weiß", wo die Dateien liegen. Weiter geht es dann bei Punkt 2.
Wenn es nicht klar ist, auf welchem Laufwerk und/oder Verzeichnis sich die Daten befinden, mit denen zu arbeiten ist, wird die Frage gestellt:
(Laufwerk:)(Pfad\)
Dabei sind mehrere Arten von Antworten möglich:
Wenn die gewünschten Daten auf demselben Laufwerk und Pfad sind, von dem aus man das Programm gestartet hatte, drückt man nur Enter. Wenn sie auf einem Unterverzeichnis des aktuellen Verzeichnisses stehen, braucht man nur dessen Namen einzugeben, nicht die komplette Pfadbezeichnung.

Beispiele für Eingaben an dieser Stelle:

<Enter> Die Dateien auf dem aktuellen Verzeichnis werden angezeigt,

`xyz`
- wenn es auf dem aktuellen Verzeichnis Dateien des gesuchten Typs gibt, deren Namen mit "xyz" beginnen, werden diese gezeigt
- sonst wird geprüft, ob es ein Unterverzeichnis "xyz" innerhalb des aktuellen Verzeichnisses gibt; dann werden die darin befindlichen Dateien genommen

`xyz\`
- wenn beides zutrifft und man möchte gezielt das Unterverzeichnis ansprechen, muß man ausdrücklich "xyz\" eingeben.

`e:\xyz` So kann auch ein Pfad auf einem anderen Laufwerk angewählt werden
"xyz" kann auch ein mehrfach gestuftes Unterverzeichnis sein, und es kann mit einer Zeichenfolge enden, die mit dem Anfang bestimmter Dateinamen übereinstimmt, wobei man '?' als Maskierungszeichen verwenden kann. Also würde man mit "bd\???k" alle Dateien des Unterverzeichnisses BD auswählen, deren Name an vierter Stelle das Zeichen 'k' hat. Somit kann im Bedarfsfall sehr differenziert ausgewählt werden.
Man vergleiche dazu auch die Möglichkeiten der automatischen Auswahl, wenn man die Programme als Makros benutzt (→ Kap.12.1, Option -d*).

2. Es erscheint ein Menü mit den verfügbaren Dateien des relevanten Typs, die das Programm auf dem Datenverzeichnis findet.
Die Namen werden in bis zu fünf Spalten nebeneinander angezeigt, und zwar alphanumerisch sortiert. Die Auswahl wird getroffen, indem man mit den Cursortasten zu den gewünschten Dateiamen fährt, diese mit einem Pluszeichen + markiert und <Enter> drückt, nachdem man den oder die betreffenden Namen markiert hat. Die Funktion der Tasten Home, End, 'a' (=alle), 'c' (=clear) und PgUp / PgDn (wenn die 5 Spalten nicht reichen!) erklärt sich von selbst.
Mit **F5** ist als Orientierungshilfe die erste Zeile der Datei abrufbar, wenn es sich um eine Textdatei (z.B. Parameterdatei) handelt. Die Zeile erscheint dann unterhalb der Liste.
Mit **F2** (nicht bei **-a0**) schaltet man um auf die entsprechenden Dateien desselben Typs (z.B. .APR) auf dem **Start**verzeichnis, nochmaliges **F2** schaltet zum **Programm**verzeichnis. Erneutes **F2** schaltet wieder zum **Datenbank**verzeichnis, usw. Ferner: F1 = Hilfe, F10 = Bearbeitung der Datei. (s.a. 10.3.3)
Mit **F10** wird die angewählte Datei in den X-Editor geladen (→ Anh.D) für sofortige Bearbeitung.

Wenn man gelegentlich "nicht an die Daten rankommt", obwohl man genau weiß, daß sie da sind, prüfe man, ob die Hintergrundvariablen -D und/oder -K anders besetzt sind. Man stellt dies fest, indem man auf MS-DOS-Ebene den Befehl **set** gibt. Dann werden diese (und andere) Variablen angezeigt. Mit `set -D=` und `set -K=` beseitigt man sie.

0.6 Menütexte, Sprache

Zum Hilfekonzept von ***a99***: **`h help`** eingeben.

Die Menütexte, Fragen und Fehlermeldungen der Programme sind in Textdateien ("ASCII Files") abgelegt und somit für Anwender zugänglich. Die Namen dieser Textdateien beginnen mit UIF (für "User Interface"), dann kommt eine Ziffer (= Programmnummer), und daran hängt dann noch (ohne Punkt!) die Sprachbezeichnung: GER für Deutsch und ENG für Englisch werden mitgeliefert, SPA für Spanisch und DUT für Niederländisch wurden von Anwendern erstellt.

`UIF0s`	Texte für den Programmkern (wird von allen Programmen außer QRIX und ***CockPit*** benötigt)
`UIF1s`	Texte für Programm 1 = PRESTO
`UIF4s`	Texte für Programm 4 = SRCH
....	z.B. `UIF7s` für das Programm 7 = INDEX
`UIFCs`	Texte für ***CockPit***
`UIFAs`	Texte für APAC und ALFA
`UIFQs`	Texte für QRIX
`UIFSs`	Texte für **`avanti`** und **`acon`**
`UIFE``s`	Texte für **`a99`** und **`alcarta`** (diese ist im ANSI-Code, die anderen in DOS-ASCII)

Beispiele (aus `UIF1GER`):

```
152 "ist bereits gelöscht"
156 "Datenbank: %s   Datei#: %d   Satz#: %ld"
158 "Satzlänge: %d" 13 10
186 "Online-Katalog der Bibliothek ...   "
```

In der Zeile 186 können Sie also den Namen Ihrer Bibliothek einsetzen. Diese Zeile erscheint als Kopfzeile des Funktionsmenüs im Programm APAC und bei PRESTO -a0 . Für a99/alcarta wird diese Angabe in die INI-Datei geschrieben, und zwar mit dem Befehl `DbTitle=...`

Wenn Sie Veränderungen vornehmen wollen, sind folgende Punkte zu beachten:

1. Jede Zeile beginnt mit einer Nummer, die auf keinen Fall verändert werden darf. Die Reihenfolge der Zeilen ist allerdings beliebig. Eine Zeile steht für jeweils eine Meldung. Die Länge der Zeile kann man mit einem geeigneten Texteditor auch über 80 Zeichen ausdehnen. Fortsetzungszeilen gehen nicht!
2. Die in Anführungszeichen stehenden Texte sind im Prinzip beliebig veränderbar. Zu achten ist evtl. auf die Länge. Verkürzungen sind unkritisch, Verlängerungen mit Vorsicht durchzuführen, da in manchen Fällen andere Bereiche des Bildschirms dann überlagert werden könnten. **Nicht verändern** darf man diejenigen Teile innerhalb von Anführungszeichen, die mit dem Zeichen '%' beginnen, z.B. Angaben wie %ld oder %s. An solchen Stellen erscheinen dann aktuelle Werte des Programms.
3. Außerhalb der Anführungszeichen stehen Steuerzeichen in Form von Dezimalzahlen, getrennt durch genau ein Leerzeichen. *Vorsicht:* wenn man mehr als ein Leerzeichen macht, fällt der Rest der Zeile weg (wie in den Parameterdateien). Prinzipiell dürfen auch hier Veränderungen und Ergänzungen vorgenommen werden, z.B. Farbumschaltungen nach ANSI-Standard, jedoch muß man dabei genau wissen, was man tut. Auskunft gibt jedes bessere MS-DOS-Handbuch. Die Werte 13 10, die sehr häufig vorkommen, stehen für „neue Zeile“, die Zahl 27 steht für das ESCAPE-Zeichen und leitet immer eine Befehlsfolge für die DOS-Bildschirmsteuerung ein (sog. ANSI-Escape-Sequenz). Diese Befehle ermöglichen es z.B. auch, eine bestimmte Meldung auf eine bestimmte Bildschirmposition zu setzen.

Die Programme laden die Textdateien in der per Option `-l` vorgewählten Sprache (→ Anhang A, .CFG-Datei), standardmäßig Deutsch (= GER). Gesucht wird zuerst auf dem Daten- , dann auf dem Programmverzeichnis. Wenn UIFxGER nicht existiert, wird auf denselben Verzeichnissen nach UIFx gesucht, also ohne Sprachkennung.
Nur von der Satzanzeige des Schnellzugriffs aus (→ Kap.1.5) kann die Sprache auch während des Programmlaufs umgeschaltet werden, indem man **F9** und dann die Sprachbezeichnung eingibt. Wenn die betreffenden UIF-Dateien nicht existieren, erhält man eine Fehlermeldung.
Für die Hilfeseiten (→ 0.3) gelten ähnliche Regelungen bzgl. der Sprache. Wenn ENG als Sprache eingestellt ist, sucht das Programm z.B. bei F1 auf Anzeigeschirm (1.5) nach H10ENG; wenn es dies nicht findet, wird H10 genommen.

0.7 Datensicherung: Konzept und Methoden

Im Normalfall geben Sie in ***a99 ein :*** **`h backp`**

Bedenken Sie: Ihre Daten sind die weiche Frucht harter Arbeit - Sicherung tut not. Die unbedingt zu sichernden Daten stehen nur in den `.xld`-Dateien - die anderen lassen sich rekonstruieren. Veränderungen an diesen Dateien werden nur durch ***a99,*** PRESTO (und dessen Varianten) und UPD in einer Datei `cat.log` protokolliert (wenn `cat` der Name Ihrer Datenbank ist). Nehmen wir als Beispiel an, diese Datenbank befindet sich auf `c:\allegro\katalog`.

- **DOS-System**: Der ***CockPit***-Pilot hat es im Normalfall leichter: er aktiviert nur **`µr s`** bzw. **`µr w`** (→ 0.11.2) und wählt dann das "**Private Backup**" bzw. "**Private Restore**" (das sind die Prozeduren PBACKUP.BAT und PRESTORE.BAT).
- **Windows-System:** `h backp` eingeben (Anleitung und Menü erscheint) und gewünschte Funktion wählen. Dazu wird die hier gegebene Beschreibung nicht gebraucht, sie kann nur zum Verständnis evtl. hilfreich sein.

Um im Katastrophenfall die Datenbank rekonstruieren zu können, geht man folgendermaßen vor. Hier ist ein möglicher "Handbetrieb" beschrieben, um den interessierten ***Allegrologen*** ein detailliertes Verständnis zu vermitteln.

1. Anfertigen einer Sicherungskopie (DOS-Standardmethode) **`µr s`**
Wir nehmen an, ein Ordner `f:\back` ist für die Sicherung vorgesehen.
Man wählt mit dem Befehl `cd` das Verzeichnis `katalog` an, auf dem sich die Datenbank befindet. Dann gibt man die Befehle

```
rename cat.log xxx.log   (.LOG-Datei darf nicht mit gesichert werden!)
copy cat*.* f:\back      (alle zur Datenbank gehörigen Dateien kopieren)
del xxx.log              (jetzt .LOG-Datei löschen)
```

Zu empfehlen: das Laufwerk `f:` befindet sich auf einem Server in einem anderen Raum.
Es ist **sehr** wichtig, die .LOG-Datei **nicht** mit zu sichern, sondern gleich nach dem Kopieren zu löschen, bevor man wieder an der Datenbank arbeitet. (Auch wenn Ihnen dies nicht einleuchtet, machen Sie es unbedingt trotzdem so!) Bei der nächsten Bearbeitung wird automatisch eine neue .LOG-Datei begonnen. **Nur diese** braucht man zum Rekonstruieren, denn sie enthält die *nach* der Sicherung angefallenen Daten!

2. Man arbeitet mit der Datenbank. In gewissen, frei bestimmbaren Abständen wiederholt man den Vorgang 1. **oder** kopiert die .LOG-Datei ihrerseits an einen anderen Ort (siehe unten).

3. Wenn irgendwann durch Plattenfehler, Stromausfall oder andere Störungen die Datenbank "abgestürzt" und nicht mehr brauchbar ist (keine Registeranzeige oder offensichtliche Fehler), löscht man die unbrauchbar gewordenen Dateien, **aber nicht** die .LOG-Datei, und speist die Sicherungskopie ein.

Das geht so: man gibt nach Anwählen des Datenbank-Verzeichnisses die folgenden Befehle:

```
del cat*.cld
del cat.tbl
del cat.cdx
copy f:\back\cat*.*
```

und führt danach die Schritte 4. bis 6. aus.

Alternativ, oder wenn man keine Sicherungskopie gemacht hat, die .cLD-Dateien aber noch da sind, kann man auch so vorgehen: man löscht CAT.cDX und CAT.TBL und regeneriert dann mit dem Programm 7 (= INDEX) die Datenbank aus den noch vorhandenen .cLD-Dateien (→ Kap.7). Dazu nutzt man die Funktion **`-fi`** des Programms INDEX, dabei bleiben auch die Satznummern erhalten! Mit diesem Schritt wäre die Rekonstruktion abgeschlossen, d.h. die folgenden Schritte entfallen dann.

4. Danach kopiert man die .LOG-Datei, falls sie beim Zusammenbruch verschwunden war (das ist sehr unwahrscheinlich), von ihrer Sicherungsdiskette auf das Datenverzeichnis. Sie muß `cat.log` heißen.

5. Schließlich gibt man, vom Programmverzeichnis aus, diesen Befehl

```
upd -fp -dkatalog\cat -ucat.log -n1                    µr w
```

Daraus kann man sich einen Batch namens `playback.bat` machen.
Oder (bequemer!) man setzt ***CockPit*** ein (→ 0.11.2). In ***a99*** findet man die Funktion "Restaurieren" auch auf dem Quick-Menü (`v quick` eingeben).

6. Nach Ablauf dieses Programms wird die Datenbank wieder in dem Zustand sein, den sie vor dem Zusammenbruch hatte (nach der letzten noch fehlerfreien Speicherung).

7. Gelegentlich beginnt man wieder neu bei 1, wenn die .LOG-Datei eine gewisse Größe erreicht hat. Eine vorgeschriebene Grenze gibt es nicht – das Maximum ist 2GB, es ist aber besser, lange vorher zu sichern, weil bei sehr großen LOG-Dateien das Restaurieren im Fall des Falles entsprechend lange dauert.

Wenn man LOG-Dateien auf Disketten oder auf ein anderes Laufwerk bzw. Dateisystem (UNIX) kopiert, muß man über die Vorgänge sorgfältig Buch führen und sich für eine von drei Methoden entscheiden:

A Man kopiert jeweils die gesamte LOG-Datei und löscht sie **nur** zum Zeitpunkt einer Gesamtsicherung (Vorgang 1.) Diese einfachste Methode empfiehlt sich, wenn man genügend Platz auf der Platte hat. Man beachte aber die unten folgende Anmerkung.

B Man numeriert die LOG-Dateien (cat1.LOG, cat2.LOG, ...) und **löscht sie nach dem Kopieren**.
Dann muß man die Vorgänge 4. und 5. im Bedarfsfall mehrfach ausführen, zuerst mit cat1.log ...

C Man kopiert die jeweils neu entstandene LOG-Datei an das Ende der beim letzten Mal gesicherten LOG-Datei (z.B. mit copy /b A:cat.LOG + C:cat.LOG) und **löscht sie nach dem Kopieren**. Dann brauchen 4. und 5. nur einmal ausgeführt zu werden. So geht es mit ***CockPit***: Gesamtsicherung mit **µr s b**, für die **erste** .LOG-Datei **µr s L** , dann immer **µr s +** ("additiv") bis zur nächsten Gesamtsicherung.
Man beachte aber:

Wichtige Anmerkung:
Nur Methode B kommt in Frage, wenn man mit INDEX zwischendurch neue Dateien einspielt. Die Regenerierung einer Datenbank muß dann so erfolgen, daß die LOG-Dateien in chronologischer Folge eingespielt werden und die neuen Dateien dazwischen, ebenfalls in korrekter Folge, erneut mit dem Programm INDEX einsortiert werden. Man führe also Vorgang B jeweils vor einem INDEX-Lauf durch, kopiere auch die neuen Dateien auf eine Sicherungsdiskette und numeriere sie ebenfalls chronologisch.
Es wäre zu riskant, das INDEX-Programm ebenfalls seine Vorgänge in der LOG-Datei protokollieren zu lassen. Ein Absturz während eines INDEX-Laufes würde eine anschließende Regenerierung sehr schwierig gestalten und nicht immer zuverlässig ermöglichen. Wenn ein solcher Absturz passiert, führe man das PLAYBACK-Verfahren durch und wiederhole den INDEX-Lauf.
Wenn man mit UPD statt mit INDEX neue Daten einspeist, ist keine besondere Aktion vonnöten, denn UPD protokolliert wie PRESTO alles in der .LOG-Datei. Einen abgestürzten UPD-Lauf können Sie wiederholen - wenn der Index danach noch in Ordnung ist, denn die schon vorhandenen Sätze werden nicht verdoppelt.
Der Platzbedarf für die .LOG-Datei ergibt sich daraus, daß jeder Datensatz bei jedem Bearbeitungsvorgang, den man mit dem Speicherbefehl (F10 oder #r) abschließt, vollständig kopiert wird. Besonders dann, wenn man mit dem Platz sparsam umgehen muß, überprüfe man nach mehreren längeren Bearbeitungssitzungen die Größe der .LOG-Datei und wende nicht die Methode A an, weil dabei die .LOG-Datei auf der Platte bleibt und ständig wächst. (Wie oben schon erwähnt: ***CockPit*** paßt auf und macht Sie ab einer einstellbaren Dateigröße darauf aufmerksam (→ 0.11.6, Befehl L).

Eine **völlig andere Sicherungsmethode** ist diese:
Wenn man zwei oder mehr getrennte (nicht vernetzte) Rechner hat, auf denen dieselbe Datenbank installiert ist, gehe man so vor:

- Bearbeitung findet nur auf Rechner 1 statt, auf Rechner 2 etc. wird die Datenbank nur lesend benutzt. Rechner 2 kann z.B. auch ein UNIX-System sein.
- Zu gewissen Zeiten kopiert man die .LOG-Datei von Rechner 1 auf eine Diskette und speist sie mit dem UPD-Programm in die Datenbank auf Rechner 2 ein. Dazu genügt ein Befehl:

  `upd -d dbp\dbn -fp -u a:cat.log`

  den man auf Rechner 2 von C:\ALLEGRO aus gibt, wobei `dbp` der Datenbankpfad und `dbn` der Datenbankname ist. Die Diskette mit cat.LOG muß zu dem Zeitpunkt in Laufwerk A: sein. Natürlich wird man für einen Routinebetrieb diesen Befehl in einem Makro, z.B. mit dem Namen NEWDATA.BAT verstecken.
- Auf Rechner 1 löscht man anschließend die .LOG-Datei. Die Datenbank auf Rechner 2 stellt nun quasi die Sicherungskopie dar!

Auf einem lokalen Netz mit Mehrplatzzugriff liegt diese Situation natürlich nicht vor, denn dort hat man nur ein Exemplar der Datenbank, an dem von mehreren Plätzen aus gearbeitet werden kann. Es kann aber sein, daß einer der angeschlossenen Rechner (der nicht im selben Raum steht!) eine genügend große Platte hat, die man zur Sicherung benutzen kann. Das ist dann schneller und bequemer als alles andere. man kopiert nur schlicht das ganze Datenverzeichnis und löscht dann die LOG-Datei.

Im Kap. 9.2 wird erklärt, wie man zwei Datenbanken **gegenseitig** aktualisieren kann, wenn in beiden katalogisiert wird.

Nochmals:
Das Windows-System macht die Sicherung viel einfacher. Man gibt in a99 ein: **`h backp`** und es erscheint eine Erläuterung, in der man den jeweils gewünschten Vorgang anklickt: **Sicherungskopie** bzw. **Datenbank restaurieren**. Beides findet man auch im Quick-Menü (`v quick`).

0.8 Was gehört alles zu einer Datenbank?

a99: `h filelist`

Hier ist zusammengestellt, welche Dateien zum Kern und Umfeld einer ***allegro***-Datenbank gehören. Wenn man z.B. eine Datenbank mit allem Zubehör sichern, auf ein anderes System kopieren oder auch verbreiten will ("Database Publishing"), ist diese Aufstellung als Checkliste brauchbar. (Eingeklammert: die nur in Nichtstandardfällen notwendigen Dateien). Bei den einzelnen Dateien ist angegeben, ob und wie sie im Katastrophenfall **regenerierbar** (d.h. automatisch wiederherstellbar) oder (manuell-intellektuell) **rekonstruierbar** sind, oder ob nicht.

Nicht aufgeführt sind die Programme (Typ `.exe`). Soll eine Datenbank auf einen anderen Rechner kopiert und dort nur zur Recherche benutzt werden, genügt das Programm `alcarta.exe` (Windows) bzw. APAC.EXE (DOS). Ansonsten müßte auf dem Zielrechner das normal erworbene und komplett installierte ***allegro***-System vorhanden sein.

Nochmals der Hinweis: Unter **UNIX** sind alle Namen vollständig mit **Kleinbuchstaben** zu schreiben.

Nehmen wir mal an, die Datenbank heißt **`dat`** und benutzt die Konfiguration **`k.cfg`**. Es ist hier unerheblich, auf welchem Unterverzeichnis die Dateien angesiedelt ist. Empfehlung: alle datenbankspezifischen Dateien auf das Datenverzeichnis bringen, alles anderen auf das Programmverzeichnis. (Ersetzen Sie **`dat`** durch Ihren eigenen Datenbanknamen und evtl. **`k`** durch Ihren Konfigurationsbuchstaben, z.B. **`a`** oder **`n`**.)

Kernbestandteile

Checkliste für *a99*/*alcarta*: **`h filelist`**

`k.cfg` die *Konfiguration* (oder **`$k.cfg`**): Kategorienschema und Abfrageliste; regenerierbar (→ Anh.A)

`dat.kpi` *Index-Parameterdatei*, **`dat`** ist damit zugl. *Datenbankname*; beschreibt die Art und Struktur der Zugriffsregister. Bei genauer Kenntnis der Datenbank und der Parametersprache rekonstruierbar (→ 10).
Die Namen `k` und `dat` finden sich in anderen Dateien wieder:

`dat_n.kld` die *Datendateien* (*n* = 1...255), d.h. hier stecken die Daten drin; vollständig regenerierbar, aber **nur** aus einer Sicherungskopie mit Hilfe von UPD und der aktuellen `.log`-Datei (→ 0.7)

`dat.kdx` *Indexdatei*; zusammen mit `dat.tbl` regenerierbar : benötigt werden die `.kld`-Dateien, `dat.kpi` und das Programm INDEX (Optionen -fi und -fn, → Kap.7), dabei wird auch `dat.stl` erneuert sowie `dat.tbl` und `dat.res`

`dat.tbl` *Satztabelle*; regenerierbar durch das Programm INDEX mit Option `-ft`. (***CockPit*** **`µr o t`**)

`dat.stl` *Kurztitelregister*; regenerierbar durch das Programm INDEX, und zwar mit der Funktion "Kurztiteldatei erneuern" (Option `-fs` : ***CockPit*** **`µr o k`**). Datenbank funktioniert ohne.

`dat.res` *Restriktionsdatei*; regenerierbar durch Programm INDEX mit Funktion "Restriktionsdatei erneuern" (Option `-fx` : ***CockPit*** **`µr o x`**). Datenbank funktioniert ohne.

`[dat.log]` *Sicherungsdatei* (→ 0.7); Wird nur benötigt, wenn eine beschädigte Datenbank wiederherzustellen ist.
Bei einer Datensicherung ***nicht*** mit sichern, mit ausliefern ist völlig überflüssig!
Nach Sicherung löschen, sie entsteht anschließend neu, sobald etwas gespeichert wird.

`d-1.kpr` *Anzeigeparameter.* (→ Kap.10), für Windows: `d-krtf.kpr`; für Web: `d-khtm.kpr`, rekonstruierbar

Peripherie

Die folgenden Dateien sind bei hinreichender Dokumentation alle rekonstruierbar, außer den UIF-Dateien ist keine zwingend erforderlich!

`CP.OPT` Datei mit den Vorgaben für das ***CockPit*** (→ 0.11), falls man damit arbeitet. Diese Datei kann auch anders heißen; sie wird über CP.BAT aufgerufen. (Muß auf Startverz. liegen)

`dat.ini` Einstellungen für die Windows-Programme. Dokumentation in `a99.ini` (→ Kap.2.5)
`dat.frm` Formulare für die Erfassung und Bearbeitung von Datensätzen
`HDAT` Datenbankspezifische Hilfeseite (PRESTO); erscheint bei Shift+F1 vom Funktionsmenü.
`datger.rtf` Die Entsprechung für die Window-Programme
`HCDAT` datenbankspezifische Hilfeseite bei Shift+F1 im Editor (PRESTO, Hinweise für die Erfassung)

`h*` *spr* evtl. abgewandelte H-Dateien (*spr* = Sprachkennung: `ger, eng`, etc.)
`uif?`*spr* evtl. abgewandelte UIF-Dateien (`uif0, uif1, uife, uifs,` *spr* = Sprachkennung)

`*.kpr` evtl. datenbankspezifische Export-Parameterdateien für die Listenproduktion etc.

`*.kpt` zugehörige Tabellendateien, z.B. **`o.kpt, d.kpt, p-*.kpt`** (z.B. Druckertreiber, → 10.2)

`*.kim` evtl. spezifische Import-Parameterdateien für die Umwandlung von Fremddaten
`*.bat` evtl. Stapeldateien für bestimmte Produktionen, z.B. `CP.BAT, PR-LIST.BAT`.

0.9.1 Schnelle Antworten auf häufige Fragen (für DOS)

*Online-Hilfe in **a99** mit Antworten für **a99**:* **`h ac0-9`**

Was ist bei der Installation zu beachten? Das steht alles im Abschnitt 0.10.

Wie muß man beim Versionswechsel die Datenbank umwandeln? Gar nicht, auch nicht beim Wechsel von DOS auf Windows. Die neuen Programme verstehen auch die alten Daten.

Wie bearbeite ich einen Datensatz? Satz aufblättern, dann den Großbuchstaben 'E' drücken. (→ 3.1)

Wie kopiere ich einen Datensatz? Satz aufblättern, 'C' drücken, dann bearbeiten und mit F10 speichern.

Wie lösche ich einen Datensatz? Satz aufblättern, Taste <Entf> drücken, mit 'j' bestätigen.

Und wie mache ich das rückgängig? Satz wieder aufblättern (mit <Bild↑>), dann 'E' F10 'j'.

Wie lösche ich eine Kategorie? Im Editormodus den Befehl #v benutzen, z.B. `#v27`, um `#27` zu löschen (→ 3.3).

Und wenn das aus Versehen passiert ist? Mit Befehl #t zurückholen: z.B. `#t27` .

Kann ich den Kurztitel ändern oder löschen? Nein, der wird automatisch erzeugt und korrigiert. Wenn die Kurztitel eine andere Struktur haben sollen: Änderungen in der Indexparameterdatei im Abschnitt **`#-0`** machen, dann vom ***CockPit*** **`µr o k`** (Organisieren / Kurzanzeige erneuern)

Kann ich die unbelegten Felder eines Datensatzes anzeigen lassen? Nein. Diejenigen, die zur Abfrageliste gehören, kann man nachträglich abfragen lassen mit Befehl #E. / **Windows:** In den Formularen sieht man, welche Felder leer sind, Alt+#.

Wieviele Sätze enthält meine Datenbank? Auf dem Anzeigebildschirm Taste <Ende> drücken, dann erscheint der letzte Satz und oben sieht man die letzte Satznummer. Die gelöschten Sätze sind dabei mitgezählt. / Win: Global / Datenbank-Info

Wieviele Sätze sind gelöscht? Im Register 1 eingeben: F6 `//?` Die Zahl steht danach links neben // .

Wie wird man die gelöschten Sätze endgültig los? Sie werden automatisch wieder zum Speichern neuer Sätze benutzt. Allerdings kann man die Datenbank auch "entlüften" (→ 0.11.2 "Organisieren / R"), danach sind sie weg.

Wieviele Einträge stehen in den Registern? Index kompaktieren, dann wird am Ende ein Protokoll erstellt, angezeigt und in der Datei PROTOQ gespeichert. Darin steht auch eine Statistik der gelöschten Sätze. Achtung beim Mehrplatzbetrieb: Nur kompaktieren, wenn keiner an der Datenbank arbeitet.

Soll man regelmäßig kompaktieren oder den Index erneuern? Notwendig ist es nicht. Der Index wird vielleicht etwas schneller, weil die Datei kleiner wird. Erneuern ist nur nötig, wenn man die Indexparameter verändert oder wenn der Index nicht mehr funktioniert (z.B. nach Stromausfall).

Kann man die Schlüssel sehen, die alle zu einem Datensatz gehören? Ja, mit F7, wenn der Satz in der Anzeige steht.

Daten einmischen: Ist INDEX oder UPD besser? Wenn Datensätze durch neue ersetzt werden sollen, geht das nur mit UPD, *a99* oder *avanti*. INDEX darf man nicht im Mehrplatzbetrieb einsetzen (→ Kap.7).

Kann man die ganze Datenbank als Ergebnismenge nehmen? Ja, mit dieser Tastenfolge: <Pos1> '(' <Ende> ')' .

Kann ich Fremddaten oder Altdaten übernehmen? Im Prinzip ja. Studieren Sie Kapitel 5.

*Kann man eigene Routinen in das **CockPit** einbauen?* Zeilen mit R oder S in die OPT-Datei einbauen (→ 0.11.6)

*Welcher Aufruf gehört zu einem **CockPit**-Menüpunkt?* Bei "eigenen Routinen" sieht man das, wenn man nach dem Anwählen <Tab> statt <Enter> drückt. Ansonsten: ***CockPit*** mit **`acp`** statt `cp` starten, dann den Menüpunkt auswählen und starten. Es entsteht eine Batchdatei CCC.BAT. Schauen Sie hinein.

*Wie kann man Menüpunkte aus dem **CockPit** entfernen?* Die zugehörige Zeile in der Datei UIFCGER entfernen.

Wie erstellt und testet man Parameterdateien? Das steht in Kap. 10.1 und 10.3.

Wie kann man Eingabedaten prüfen lassen? Es gibt einige eingebaute Prüfmöglichkeiten (→ Anh. A.1.2). Mit Parametrierkenntnissen kann man eigene Prüfungen programmieren (→ Kap. 10.2.8)

Wie gibt man Sonderzeichen ein? Vor Programmaufruf `aw` geben, im Programm dann Alt+w (→ Anh.E).

Kann man Registerauszüge drucken? Mit dem Programm QRIX (→ 0.11.1 Funktion q, auch Kap. 7.9 und 1.4.1).

Wie lang darf ein Datenfeld sein? Bis zu 10000 Zeichen. *Jedes?* Ja. Mehr über Grenzen: http://www.allegro-c.de/grenzen.htm

Wieviele verschiedene Datenfelder kann es geben? Über 1000. S. dazu: Anh. A.1.3)

Kann jedes Datenfeld mehrfach belegt werden? Ja. Man kann dies jedoch in der CFG-Datei einschränken (Anh.A).

Muß man zuerst irgendwie Platz reservieren, wenn eine neue Datenbank entstehen soll? Nein, die Programme benötigen am Anfang nur sehr wenig Platz und erweitern die Dateien ständig automatisch.

Kann man Änderungen einer Datenbank auf eine andere übertragen? Siehe Kap. 0.7, letzter Abschnitt.

Ist es möglich, zwischen zwei oder drei Datenbanken umzuschalten? Ja, aber man muß richtig starten → Kap. 1.3.2.

Welche Satztypen gibt es? Das ist allein Sache der Parametrierung, vor allem der Indexparameter. Die Software behandelt ansonsten alle Datensätze gleich.

Können im Netzwerk zwei oder mehr Updates gleichzeitig laufen? Ja. Dann aber beim Start von UPD nicht die Option **`-F`** verwenden (→ Kap.9.1) und jedem UPD mit `-x`*name* seine eigene Protokolldatei zuweisen.

Wie muß man eine Datenbank für UNIX/Windows ändern? Gar nicht, nur an den Parametern sind evtl. Änderungen nötig.

0.9.2 Änderungen : Was? Wo? Wie?

zusätzlich für *a99/alcarta* : **`h wastun`**

Hier finden Sie einen schlagwortartigen, alphabetischen Überblick, was Sie als Anwender alles ändern **können** und wo Sie bei Bedarf eingreifen **müssen**.
Alle Änderungen an Parameterdateien etc. sind mit jedem **Textprogramm** oder **Texteditor** ausführbar. Bei Textsystemen wie WORD oder WordPerfect oder WinWord ist darauf zu achten, die Dateien als "ASCII files" oder "DOS-Text" abzuspeichern. Empfehlenswert sind aber eher solche Editoren, die in die bekannten DOS-Hilfssysteme wie NORTON COMMANDER oder PCTools integriert sind. Der mitgelieferte Texteditor X, der auch vom ***CockPit*** aus normalerweise aufgerufen wird, ist im Anhang D beschrieben. Viele Experten schwören noch immer darauf.

Nehmen wir an, Sie arbeiten an einer Datenbank namens CAT, die auf der Konfiguration $A.CFG beruht. (Wird eine andere Konfiguration verwendet, z.B. P.CFG, muß in den Dateitypen jeweils 'A' durch 'P' ersetzt werden, also .PPR statt .APR etc. Für die Bildschirmanzeige verwenden Sie D-1.APR, für die Druckausgabe (Option -q) P-KARTE.APR. Zum Sortieren von Recherche-Ergebnissen gibt es die Parameter S-*.APR.

Abfrageliste (zur Datneingabe) : ist enthalten in Konfigurationsdatei **$A.CFG** (→ Anh.A.1) **`µo k`**

Anzeige der Datensätze am Bildschirm : PRESTO-Parameter **D-1.APR** (→ Kap.10) */Win' a99*: `d-wrtf.apr` **`µd e`**

Benutzeroberfläche : ***CockPit*** bietet eine Vielzahl von Funktionen in übersichtlichen Menüs, ermöglicht aber eigene Anpassungen und Erweiterungen (→ 0.11). Möglich sind aber auch eigene Menüprogramme auf der Basis von Batchfiles (→ 12). a99: „Füllhorn“ oder Alt+4

Bildschirmfarben : einstellbar mit den w-Befehlen in der Konfigurationsdatei **A.CFG** (→ Anh.A.3) **`µo k`**

CockPit : Anpassungen macht man in der Datei CP.OPT (früher DEFAULT.OPT) (→ 0.11.6) **`µo v`**
Die Texte der Menüs stehen in der Datei UIFCGER **`µd m`**

Codierung : Zeichencodes für die Druckausgabe oder andere Exporte lassen sich individuell ändern in der Parameterdatei, die man für die Druckausgabe benutzt (→ 10.2.4), s.a. **Druckertreiber**

Datenbank-Definition, Datenformat : siehe **Kategoriesystem** und **Index**

Druckertreiber : "Druck" fällt unter "Export", deshalb sind Druckertreiber innerhalb der Exportsprache realisiert (→ 10.).
Es sind Dateien des Typs .APT (→ 0.3, 0.11), z.B. P-DSKJET.APT. **`µo d`**

Druckformatierung : Parameterdatei ändern (z.B. P-NORMAL.APR bei Listendruck, P-KARTE.APR bei Kartendruck), die für die Ausgabe benutzt wird (→ 10.)

Einstellungen für die Windows-Programme : `INI-`Datei `(a99.ini` als Muster und Dokumentation)

Einzelplatzbetrieb : → Mehrplatzbetrieb

Ergebnismenge : globale Korrekturen in der Datenbank: F10 (→ 1.5.4, 3.3 Befehl #X) / ***a99***: Menü Global

Fehlermeldungen : Die Meldungstexte stehen in den UIF-Dateien (→ 0.3, siehe auch **Anh.C.2**) **`µd m`**

Formulare für die Windows-Erfassung und Bearbeitung von Daten: Datei CAT.FRM **`h adm`**

Funktionstasten-Belegungen : Grund- und ALT-Belegung sollte man nicht ändern, SHIFT- und CTRL-Belegung außer F1 möglich (→ 1.5, 2.6). (Die UNIX-Version erfordert andersartige Belegungen!)

Füllzeichen : Code und Anzahl einstellbar mit Befehl F bzw. f in $A.CFG (→ Anh. A.1.3)

Grunddatei : (Typ .ALG) Bearbeitung mit Programm 3 = ALLEGRO (→ 2.4) **`µf 2/3`**

Hilfetexte : (H-Dateien) frei bearbeitbar mit jedem ASCII-Texteditor (→ 0.3, 0.6) / .rtf: WordPad **`µd h`** / ***a99:*** **`h help`**

Index : Art und Gestalt der Eintragungen sind in einer Index-Parameterdatei festgelegt, z.B. CAT.API. **`µd i`**
Bei nachträglichen Änderungen an dieser Datei muß man den Index oder wenigstens diejenigen Teil-Indizes neu generieren (→ Kap.1.2 und Kap.7, Option -fi), deren Einträge betroffen sind. Auch die Kurzanzeige und die Restriktionen werden durch die Index-Parameterdatei bestimmt: es gibt in CAT.API Abschnitte, die mit den Sprungmarken **`#-0`** und `#-/` beginnen; dort sind die Strukturen der Kurzanzeige (Datei CAT.STL) und der Restriktionen (CAT.RES) mit Hilfe der Exportsprache definiert.

Interpunktion : Diese ist in Form von "Präfix-" und "Postfix-Angaben" in einer Export-Parameterdatei definiert. Z.B. ist das im Normalfall für die Bildschirmanzeige die Datei D-1.APR, für die Druckausgabe z.B. P-KARTE.APR (→ 10.2.0)

Ja-Nein-Codes : bei fremdsprachigen Versionen müssen u.U. die Werte für die Antworten "ja" und "nein" bei Ja-Nein-Abfragen geändert werden: Befehl Y in A.CFG (→ Anh.A).

Kategoriesystem : In der Datei A.CFG steht das Schema des "konsolidierten Formates"; dieses kann mit Hilfe eines Texteditors geändert werden, oder man erstellt oder verwendet eine andere Konfiguration, z.B. N.CFG für das Neutralformat oder $U.CFG für USMARC (→ Anh. A, B)
Vorwahl durch Option bzw. Umgebungsvariable -k (→ Kap.12).

Köpfe der Katalogkarten : Definitionen stehen in P-KARTE.APR, siehe Befehl **`ak`** (→ 10.2.1)

Kurzanzeige : siehe Index

Listengestaltung : Auswahl einer Datei P-*.APR (evtl. Erstellung einer neuen Datei dieses Typs, → Kap.6) **`µo 3`**

Mehrplatzbetrieb : die Software ist mehrplatzfähig auf jedem PC-Netz. Evtl. muß man in CP.OPT eingreifen **`µo v`**

Menüsystem : siehe **Benutzeroberfläche**

Nichtsortierzeichen : Code bzw. Modus: Befehle N bzw. n in A.CFG (→ Anh.A.1.3)

Ordnung in sortierten Listen : in den Dateien S-*.APR werden die Sortierbegriffe durch den Befehl **`ak`** in Verbindung mit den Sonderkategorien **`#u1`** und **`#u2`** gesteuert. So läßt sich die Ordnung präzise festlegen. (→ 6.1, 6.4, 10.2.1, 10.2.6) **`µo 1`**

Paßwörter : nur beim Windows-Programm a99 möglich. Mehr dazu: **`h npw`** eingeben

Pflichtkategorien : Abfragezeilen mit '!' und '?' in A.CFG (→ Anh.A.2) **`µo k`**

Phrasen : werden automatisch (in der Datei PHRASE.APH / Win: PHRASE.A99) gespeichert und geladen. .APH-Dateien sind aber auch per Textprogramm bearbeitbar! (→ 0.3, 3.3 Befehl #p) **`µd p`**

Qualität der Daten : evtl. entscheidend zu verbessern durch Fremddaten-Übernahme, vor allem aus Nationalbibliographien, Verbundsystemen und Literaturdatenbanken (→ 2.0, 5.0)

Regelwerk : ***allegro*** ist nicht auf RAK fixiert. Es duldet jede eigenmächtige Interpretation oder sogar Änderung von Katalogregeln, allerdings nicht mühelos: sofern davon irgendeine Ausgabeform oder Indexeinträge betroffen sind, hat man die zugehörige Export-Parameterdatei (.APR-Datei) zu ändern (→ 10.)

Reports / Sortierte Listen : per Exportparametrierung (→ 6. und 10.)

Skriptsprache: FLEX für a99 und acon (avanti). In a99 eingeben: `h flex`

Speicherbedarf : Die DOS-Programme arbeiten im 640K-Bereich. Wenn hier zu wenig frei ist: Bedarf der geladenen Geräte- und Netzwerktreiber und anderer residenter Programme prüfen. Anpassung des Speicherbedarfs evtl. mit den m-Befehlen in der .CFG-Datei (→ Anh.A.1). Prüfung des aktuellen Zustands mit <Alt+F7> vom Anzeigebildschirm aus. *Tip:* Wenn PRESTO.EXE wegen Speichermangel nicht läuft, statt dessen MENUED.EXE versuchen.

Sprache der Menüs : falls entsprechende UIF-Dateien vorhanden (→ 0.6): Vorwahl durch Befehl 'l' in der A.CFG oder Option bzw. Umgebungsvariable `-l` verwenden (→ 12.). Default ist GER.

Startprozedur : `_start.flx` (auf dem Datenverzeichnis) wird von a99/alcarta sofort nach dem Start ausgeführt.

Stopwörter : die Tabelle SWL1.APT kann erweitert, gekürzt und beliebig geändert werden (→ 0.3, 10.2.1).

Übersetzung : Menü- und Hilfetexte (UIF*- bzw. H*-Dateien) können und dürfen übersetzt werden, siehe **Sprache**

Vorbearbeitung von Fremddaten : Umfangreiche Möglichkeiten durch eine eigene "Importsprache" (→ 5., 11.)
Datenumwandlung auch mit FLEX möglich (`h fremd` eingeben in ***a99***).

Wiederholbarkeit von Kategorien : grundsätzlich immer gegeben, kein Eingriff nötig (→ 3.1); Alternative: Mehrfacheinträge innerhalb einer Kategorie, mit definiertem Trennzeichen (Empfehlung: "; " oder "¶"). Begrenzte Wiederholbarkeit einstellbar in der Konfiguration (→ Anh.A.1.2)

Windows-Programme : Deren Einstellungen stehen in einer INI-Datei. Beispiel und Beschreibung: `a99.ini`

Zeichensatz : für Bildschirm und Tastatur können bei VGA-Bildschirmen die Zeichensätze softwaremäßig verändert werden, ohne daß dies ***allegro*** stören würde; für den Drucker: siehe **Codierung**. Der Normzeichensatz DIN 31628/2 wurde realisiert. Man schaltet den Zeichensatz ein mit dem Befehl `ostwest.bat`. Mit <Alt>+w erscheint dann jederzeit eine Hilfstabelle mit allen Zeichen.
Derselbe Zeichensatz wird für **Windows** als TTF-Schrift bereitgestellt (→ Anh.E). Sind diese Schriften installiert, benutzen *a99* und *alcarta* sie automatisch. Aber auch in jeder anderen Windows-Software, z.B. WinWord, kann man die Schriften nutzen, das erleichtert den Datenaustausch mit anderen Anwendungen.

0.10 Installation

Online-Hilfe in ***a99*** *:* **`h inst`**

Installationspaket ***Achtung:*** **Installation immer als "Administrator" ausführen!**
Die aktuelle Version des Installationspakets finden Sie immer unter **`http://www.allegro-c.de/aktuelle-version`**
Für UNIX und avanti gibt es eigene Anleitungen, die von der Homepage aus leicht zu finden sind:
avanti: `http://www.allegro-c.de/doku/avanti`
UNIX: `http://www.allegro-c.de/doku/unix.htm`

Hier soll es nur um die Windows-Version ab V26 gehen, für die ein **Windows-System** (ab 2014: Win'7 oder 8) erforderlich ist. Irgendwelche besonderen Hardwareanforderungen gibt es nicht, sonstige Softwarekomponenten werden nicht gebraucht.
Falls Sie ohne Internet-Zugang sind: Wenn Sie die DVD der aktuellen Version erhalten haben, starten Sie das darauf befindliche Programm `start.bat` (falls das nicht automatisch beim Einlegen passiert). Dann wird das Programm nach der mitgelieferten, 13stelligen Schlüsselzahl fragen, um das "Gesamtpaket" mit dem Namen **`INST-ALL.EXE`** von der DVD entschlüsselt zu kopieren. Das ist ein Windows-typisches Installationsprogramm. Darin befinden sich dann die eigentlichen Programme. Die Version auf dem Server ist in der Regel aktueller. Beim Ablauf des Installationsprogramms ist nur an einer Stelle evtl. einzugreifen, und zwar um den Namen des Zielverzeichnisses zu ändern. Vorgegeben wird `c:\allegro`. Wer damit leben kann, braucht nur noch zu klicken und die Installation ist erledigt.
Ab V26 haben Sie die Wahl zwischen dem Normalpaket INST-ALL.EXE und dem sog. Neutralpaket namens INST-NEU.EXE.
Die Programme unterscheiden sich nicht, nur die mitgelieferten Demo-Dateien sind unterschiedlich:
`INST-ALL.EXE` : (`http://www.allegro-c.de/doku/form2004/`) **Konsolidiertes Format** `$A.CFG`
`INST-NEU.EXE` : (`http://www.allegro-c.de/doku/neutral/`) **Neutralformat** `N.CFG`
Für den Einsteiger hat das Neutralformat den Vorteil, daß man noch nicht einmal eine eigene Datenbank anlegen muß, sondern sofort in die Demo-Datenbank hineinkatalogisieren kann. Zum Löschen der Beispielsätze genügt ein Klick, den man jederzeit ausführen kann. Mehr zur Frage des Datenformats in Anhang A.

Wohin mit allem?
Alle Programme können auf einem Server installiert und gemeinsam genutzt werden. Lokal (also auf jedem PC) muß man **nur** die Schriften und bei Bedarf ein oder mehrere Icons installieren. Die Schriften liegen in Form von 4 TTF-Dateien auf dem Programmverzeichnis (`a-times.ttf` etc.), außerdem gehört `a-dos.fon` dazu, das ist die Schriftart für die DOS-Fenster.
Tip: Einen interaktiven Überblick über das allegro-Dateisystem erhält man in a99 mit Eingabe von `h files`.

Verzeichnis- und Dateinamen
Man vermeidet Komplikationen, wenn man folgende Konventionen streng einhält: Programme und Datenbanken nicht auf tief verschachtelte Verzeichnisse mit langen Namen legen, insbes. keine Namensteile mit mehr als 8 Zeichen verwenden, Verzeichnisnamen ohne Punkt, alle Namen komplett in Kleinbuchstaben halten, Konvention 8.3 einhalten, keine Leerzeichen, Unterstriche oder Umlaute in Pfad- und Dateinamen.
Ein Datenbankname (= Name der Indexparameter, → 0.8) soll nicht länger als 4 Zeichen sein (damit die Namen der Datenbankdateien, z.B. `cat_123.ald`, nicht länger als 8 werden können). Daran ist zu denken, wenn man eine eigene Datenbank mit eigener, neuer Index-Parameterdatei anlegt: auf deren Namen kommt es dann an. Verwendet man die Standards, hat man entweder den Namen CAT (von `CAT.API`) oder BANK (von `BANK.NPI`).
Hinweis: Die Namen sind alle intern, d.h. Endnutzer oder auch Katalogisierer müssen sie nicht kennen oder gar intuitiv verstehen können! Daher sind solche Einschränkungen kein Problem.

DOS-Fenster
Schrift mit Sonderzeichen: "Eigenschaften / Schriftart" und dann 8x13 oder 10x19 wählen. (Nur verfügbar, wenn die Schrift `A-DOS.FON` installiert wurde, siehe oben.)
Beim Schließen des Eigenschaften-Fensters: Bestätigen mit "Verknüpfung, die dieses Fenster aufruft, ändern"
Wenn man Copy&Paste ermöglichen will: Rechte Maustaste linke obere Ecke des DOS-Fensters, Eigenschaften / Optionen, und dann das Häkchen bei "Quick-Edit-Modus" setzen.

Schreibberechtigung
Jeder Nutzer, auch alcarta-Nutzer (OPAC-Programm) braucht ein Verzeichnis mit Schreibberechtigung. Zu empfehlen ist das TEMP-Verzeichnis auf dem eigenen PC. Dieses wird als Default verwendet. Soll es ausnahmsweise ein anderes sein, kann man in der INI-Datei die Variable `DbAux` mit dem Namen des gewünschten Verzeichnisses belegen.
Wer das Programm a99 nutzt, braucht volle Schreibberechtigung auch auf dem Verzeichnis, wo die Datenbank liegt.

Netz
Datenverzeichnis für alle a99-Nutzer zum Schreiben und Lesen, für alcarta-Nutzer zum Lesen freigeben. Alle dort liegenden Dateien, falls nötig, für gemeinsame Nutzung freigeben (Novell: Attribut sh = sharable, Befehl: flag *.* sh, Windows-Server: entsprechende Einstellungen vornehmen in den Gruppenrichtlinien).
Auf dem Programmverzeichnis muß Lese- und Ausführrecht bestehen. Die Programme sind netzwerkfähig, eine Einzelplatzversion gibt es gar nicht.
Das sog. "Caching" im Client-PC muß generell abgeschaltet werden. Je nach System muß man das zentral oder auf jeder Workstation machen (z.B. Novell: "client file caching enabled=off" in der autoexec.ncf).
Keine Netware-Clients von Microsoft verwenden.

Desktop-Verknüpfung für die eigene Datenbank
Nach dem Vorbild der DemoBank, die automatisch bei der Installation angelegt wird. Das "Ziel: " der Verknüpfung sollte immer so aussehen:
a99 *ProgDir*`\allegro.exe` *DbDir*`\`*ininame*
alcarta *ProgDir*`\allegro.exe opac` *DbDir*`\`*ininame*
Wobei z.B. *ProgDir* ein Pfadname wie `x:\allegro` sein könnte, *DbDir* etwas wie `k:\daten\katalog` und *ininame* z.B. `xyz` (wenn es eine INI-Datei namens `xyz.ini` auf dem *DbDir* gibt).

Dann in den "Eigenschaften" des Icons noch die Eintragung "Ausführen in:" auf ein Verzeichnis setzen, wo Schreibrecht besteht. Das kann z.B. `%temp%` sein.

DOS-Programme (nur bis Win'7/32 lauffähig)
Damit diese laufen, ist es nur erforderlich, das Treiberprogramm `ANSI.COM` einmal vor dem Aufruf des ersten DOS-Programms zu starten. Dies ist in das CockPit-Startprogramm `cp.bat` eingebaut und wird beim Erstellen von Batchdateien per CockPit oder a99 jeweils berücksichtigt, d.h. nur bei eigenen Batchprozeduren ist darauf zu achten.
DOS-Programmspeicher: mindestens 590K " maximale Größe für ausführbares Programm" sollte man haben, meistens ist das der Fall.
Wenn Sie ***allegro*** **nicht auf dem Verzeichnis C:\ALLEGRO** installiert haben, ist es notwendig, in der Datei CP.OPT die Einstellung "Programmverzeichnis" zu ändern. Diese Datei ist kommentiert (→ 0.11.6), und Sie sehen sofort, wo man eingreifen muß. Auch Ihr Datenbankpfad ist dort einzutragen, und zwar wo

```
d C:\ALLEGRO\DEMO2
```

steht. In der Datei `$A.CFG` (→ Anh.A.4) nehmen Sie Änderungen an den Speichergrößen, Bildschirmfarben etc. vor.

Windows-Programme
Hinweise zur Einrichtung der spezifischen Dateien → **Kap.2.5.** Insbesonders, was man tun muß, um eine eigene, ältere Datenbank mit den Windows-Programmen nutzen zu können.

Upgrade-Installation
Das Installationspaket (siehe oben) ganz genauso starten wie bei Neuinstallation, dann werden alle veralteten Dateien ersetzt, auch die Demo-Datenbank. Außer `$A.CFG` und `CP.OPT`, die sehr oft nutzerseitig verändert werden. Anhand eines Registry-Schlüssels stellt das Installationsprogramm fest, ob und wo eine ältere Installation vorhanden ist. Wenn diese Angabe nicht stimmt, dann ändern, sonst durchklicken. Im Normalfall können Sie also ein „Upgrade" in wenigen Minuten erledigen. Eigene Daten bleiben selbstverständlich unberührt und müssen auch nicht konvertiert oder sonstwie geändert werden.
Achtung: Wenn man gerade ein Fenster mit der Demo-Datenbank offen hat, dann dieses zuerst schließen.
Empfehlung: Wenn man Änderungen macht an Standard-Parametern (.APx-Dateien), die auf dem **Programm**verzeichnis liegen, dann diese auf das eigene **Daten**verzeichnis kopieren. Ebenfalls UIF-Dateien und Dateien der Verzeichnisse HELP und FLEX.
Hinweis: Nur die Programme `a99.exe` und `alcarta.exe` können nicht im laufenden Betrieb, also wenn gerade jemand damit arbeitet, überkopiert werden. Das passiert dann beim nächsten Start des Windows-Systems. Sie sind im Installationspaket zusätzlich unter anderen Namen enthalten: `allegro.exe` bzw. `opac.exe`.

Demo-Paket auf Vollpaket aufrüsten
Wer sich bereits das Demopaket installiert und damit echte Arbeit begonnen hatte, kann ohne Problem genauso verfahren, wie gerade beschrieben: `INST-ALL.EXE` nehmen oder `INST-NEU.EXE` und ausführen.

Vorher nur ganz alte DOS-Version gehabt?
Auch das ist kein Problem. Allenfalls wenn die Version *sehr* alt ist, bis V12, muß etwas mehr getan werden (→ **Anh.A.5**).

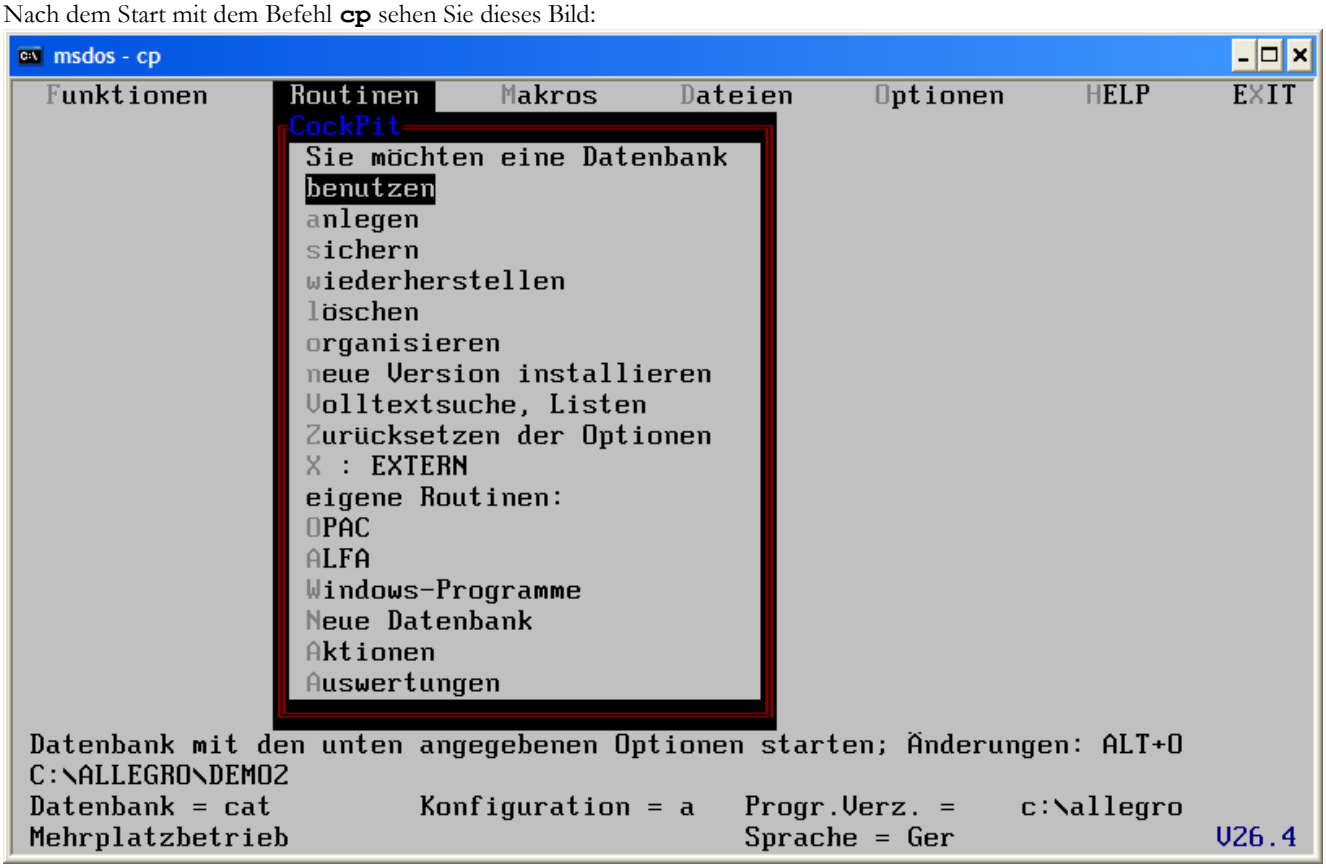
0.11 *CockPit* : Startprogramm des DOS-Systems

***a99** : statt dessen Menüpunkt* **"?"**

Der Name ***CockPit*** soll andeuten, daß es sich um so etwas wie eine Pilotenkanzel handelt, in der man mit Hilfe geeigneter Instrumente alles überwachen und steuern kann. Allerdings wird dadurch dieses Systemhandbuch keineswegs überflüssig.
CockPit erleichtert die Routinearbeit, indem es eine Reihe von Vorgängen, die man immer wieder braucht, auf wenige Knopfdrücke reduziert. Es ist ferner vom Anwender erweiterbar: es ermöglicht, eigene Prozeduren, die man als "Batchfiles" geschrieben hat, mit einzubinden. Es bietet aber wenig Unterstützung für Gestaltungsaufgaben: Datenbankdefinition, Export- und Importprogrammierung vereinfacht es kaum. Viel mehr leistet da das Windows-System, z.B. mit dem Export-Komfortmenü. Ansonsten ist solide Systemkenntnis immer noch wertvoll, und dafür gibt es dieses Systemhandbuch.
Die Datenbankarbeit kann unter Windows komplett mit dem Programm ***a99*** geschehen. Die Datenbank*verwaltung* kann aber immer noch nach alter Art mit dem ***CockPit*** in einer DOS-Box gemacht werden, wenn man sich darin zu Hause fühlt.

Startbild

Nach dem Start mit dem Befehl **cp** sehen Sie dieses Bild:



Die Angaben auf den drei untersten Zeilen stammen aus der Vorgabendatei CP.OPT - dazu unten noch mehr.
Der Leuchtbalken steht zu Anfang auf der wichtigsten Funktion: **benutzen**. Somit reicht ein weiterer Druck auf <Enter>, um in die vorgegebene Datenbank per Schnellzugriff sofort einzusteigen.

Dieses Menü funktioniert so, wie es von vielen DOS-Standardprogrammen bekannt ist:

- Mit den Cursortasten bewegt man den Leuchtbalken auf- und abwärts, mit <Enter> aktiviert man den gewählten Menüpunkt. Oder: Schnellwahl durch Betätigen des Anfangsbuchstabens.
- Zum Umschalten auf ein anderes Pulldown-Menü fährt man mit <Cursor links> und <Cursor rechts> oben an der Menüzeile entlang. Oder: <Alt> + Anfangsbuchstabe, also z.B. <Alt>+d für das Menü "Dateien".

>>> Als symbolische Kurzbezeichnung verwendet dieses Handbuch das Zeichen **'µ'** für "Menü". Z.B. bedeutet die Angabe **µr o i** , daß man die Tasten <Alt>+r o i hintereinander betätigen soll.

- Aus Untermenüs entkommt man stets ohne Folgen mit <Esc>
- Über "EXIT" oder <Alt>+x verläßt man das ***CockPit***.
- *Besonderheit:* Die Funktion **z** auf dem Routinenmenü stellt die Grundoptionen auf den Fußzeilen auf die Werte, die in der Vorgabendatei stehen.

Hinweis: Die Windows-Programme lesen die CP.OPT nicht, sie beziehen ihre Einstellungen aus einer Datei vom Typ INI. Vorlage und Dokumentation dafür ist die Datei A99.INI.

1. DOS-Programm PRESTO

0.11.0 Die *CockPit*-Hauptmenüs

Zunächst ein schlagwortartiger Überblick, die nachfolgenden Seiten erklären jedes Menü im Detail.

Funktionen Aufruf der einzelnen ***allegro***-Programme mit der Möglichkeit, über programmspezifische Menüs alle Aufruf-Optionen (→ Kap.12) einzustellen

Routinen oberer Teil: Routine-Verwaltungsaktionen an der ausgewählten Datenbank
unterer Teil: eigene Prozeduren, definiert mit den Befehlen R und S in CP.OPT
Dahinter stecken beliebige, eigene Batchdatien.

Makros vorhandene Batchfiles können gesichtet, gestartet oder bearbeitet werden.

Dateien alle Parameterdatei-Typen und andere Hilfsdateien des Systems können aufgelistet, betrachtet und bearbeitet werden. Dazu wird der Editor X benutzt (oder Ihr eigener, siehe oben). Es erscheint jeweils die Liste der vorhandenen Parameterdateien des betreffenden Typs, und zwar zuerst die auf dem Datenverzeichnis; mit dem Menüpunkt "umschalten" bekommt man die entsprechenden Dateien des Programmverzeichnisses. (Dazu muß aber in der CP.BAT der Aufruf von ACP mit der Option `-a2` erfolgen.)

Optionen Hier sind die auf den zwei Fußzeilen (Statuszeilen) stehenden Angaben veränderbar:

- **Datenverzeichnis** (bei Anwahl eines anderen Verzeichnisses stellen sich die Statuszeilen automatisch auf die dort liegende Datenbank ein)
- **Konfiguration** (hier auch Bearbeitung der .CFG-Dateien)
- **Sprache** (GER und ENG werden mitgeliefert)
- **Laufwerke** (nach Umschaltung: "Datenverzeichnis" anwählen)
- **Vorgaben** (Bearbeitung der CP.OPT-Datei - mit sofortiger Wirkung)
- **Ressourcen** (Anzeige wesentlicher Speichergrößen)
- **Druck-Optionen** Einstellmöglichkeiten für Listenproduktionen (Auswahl bestimmter Parameterdateien für Sortierfolge, Ausführlichkeit, Listengestaltung und Druckertreiber.

Diese Dateien gehören zum *CockPit*

CP.BAT Hiermit startet man das ***CockPit***. Es ruft dann selbsttätig das eigentliche Programm ACP.EXE auf.

ACP.EXE Wird folglich nur indirekt, eben durch CP.BAT, aufgerufen. Dieser Umweg ist notwendig, damit das ***CockPit*** nach jedem Vorgang erneut die Kontrolle übernehmen kann.

CP.OPT Diese Datei enthält die Voreinstellungen, mit denen ***CockPit*** dann arbeitet. Alle Einstellungen lassen sich ändern: `CP.OPT` ist eine Textdatei und kann, wie alle Parameterdateien, mit jedem Texteditor geändert werden. Dafür ist sogar ein eigener Menüpunkt vorgesehen: der Punkt **v** im Menü "Optionen". Die Datei ist ausführlich kommentiert und in 0.11.6 abgedruckt.

CP.PRE Einstellungen der vorigen Sitzung. Diese Datei löschen, dann gelten wieder alle Einstellungen aus CP.OPT

UIFCGER Das deutsche und das englische "User Interface File" (→ 0.3).

UIFCENG Wie bei den anderen Programmen stehen darin die Menütexte, Fragen und Meldungen, die am Bildschirm erscheinen. Auch bei diesem Programm kann man also Änderungen und Übersetzungen durchführen. **Unerwünschte Menüpunkte** löscht man heraus, dann erscheinen sie gar nicht!
Sehr zu achten ist auf die Längen der Texte, die in vielen Fällen keine großen Änderungen vertragen - die Resultate sollte man immer prüfen. Soll ***CockPit*** auf Englisch laufen: in der CP.BAT ergänzt man die Zeile `acp ...` durch **`-l eng`**.

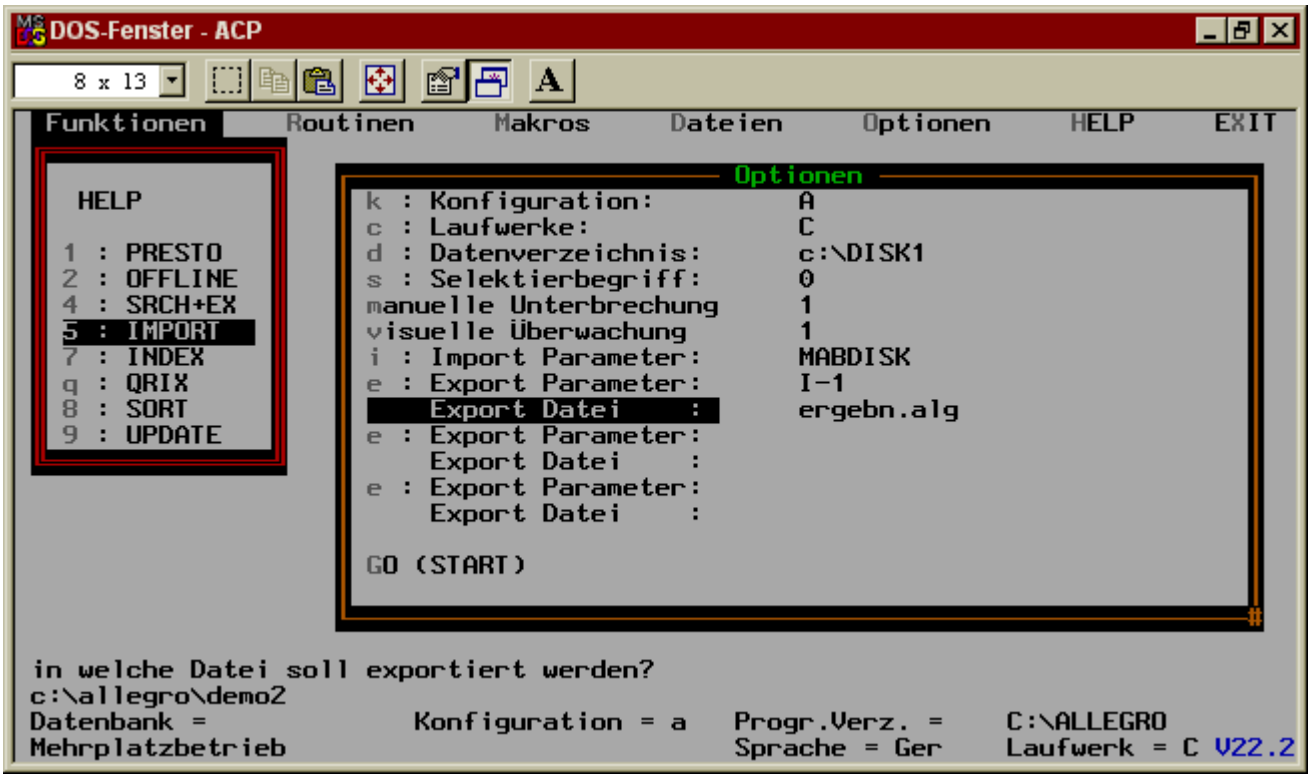
Mitgeliefert werden ferner
(jedoch gehören sie nicht eigentlich zum ***CockPit***, sondern sind durch andere Programme ersetzbar)

X.EXE Ein DOS-Texteditor, der für die Bearbeitung der Parameter- und Konfigurationsdateien benutzt wird; kann durch einen anderen Editor ersetzt werden, z.B. Microsoft-EDIT (dann Befehl E in CP.OPT ändern)
Anleitung: siehe Anhang D

V.COM Ein DOS-"View"-Programm zum Betrachten von Textdateien; kann gleichfalls durch etwas entsprechendes anderes ersetzt werden (dann Befehl V in CP.OPT ändern)

0.11.1 µf : Das Menü "Funktionen"

Die einzelnen Einstellungen nimmt man so vor:



- man bewegt den Leuchtbalken im linken Teil des Untermenüs "Optionen",
- man drückt <Enter>, wenn man den betreffenden Punkt bearbeiten will,
- es erscheint eine Frage oder eine Eingabeaufforderung oder ein weiteres Untermenü, aus dem man etwas auszuwählen hat - so z.B. bei dem Punkt "Import-Parameter".
- Ist alles richtig eingestellt, bewegt man den Leuchtbalken auf **`GO (START)`** und drückt nochmals <Enter>.

Eine Besonderheit bei Funktion 9 = UPD: man stellt vorher über **`µo d`** als Datenverzeichnis dasjenige ein, wo die einzuspeisende .ALG- oder .LOG-Datei liegt, und dann im Update-Menü dasjenige, auf dem die zu aktualisierende Datenbank liegt.

CockPit produziert nun eine Batchdatei namens CCC.BAT. Darin steht der folgende Aufruf:

```
import -f5 -dC:\DISK1 -iDBDISK -kA -lGER -e I-1/ergebn.alg -s0 -m1 -v1
```

Anschließend wird CCC.BAT ausgeführt, also IMPORT mit allen erforderlichen Angaben gestartet. Nur die umzuwandelnde Datei (die auf A:\FREMD stehen muß) ist anschließend noch auszuwählen (→ Kap.0.5). Nach Ablauf des Importvorgangs sorgt CCC.BAT dafür, daß ***CockPit*** wieder aktiviert wird.

Man könnte den IMPORT-Aufruf genauso mit der Hand geben oder IMPORT ohne Optionen starten (dann fragt es selbst die nötigen Angaben ab - man muß aber dann wissen, was gemeint ist). Der Vorteil vom ***CockPit*** liegt darin, daß die Optionen für den Programmabruf überschaubar und kommentiert zur Auswahl und kontrollierten Eingabe und Bearbeitung angeboten werden. (Ein ähnliches Beispiel wird in Kap.5.1 noch näher erläutert.)

1. DOS-Programm PRESTO

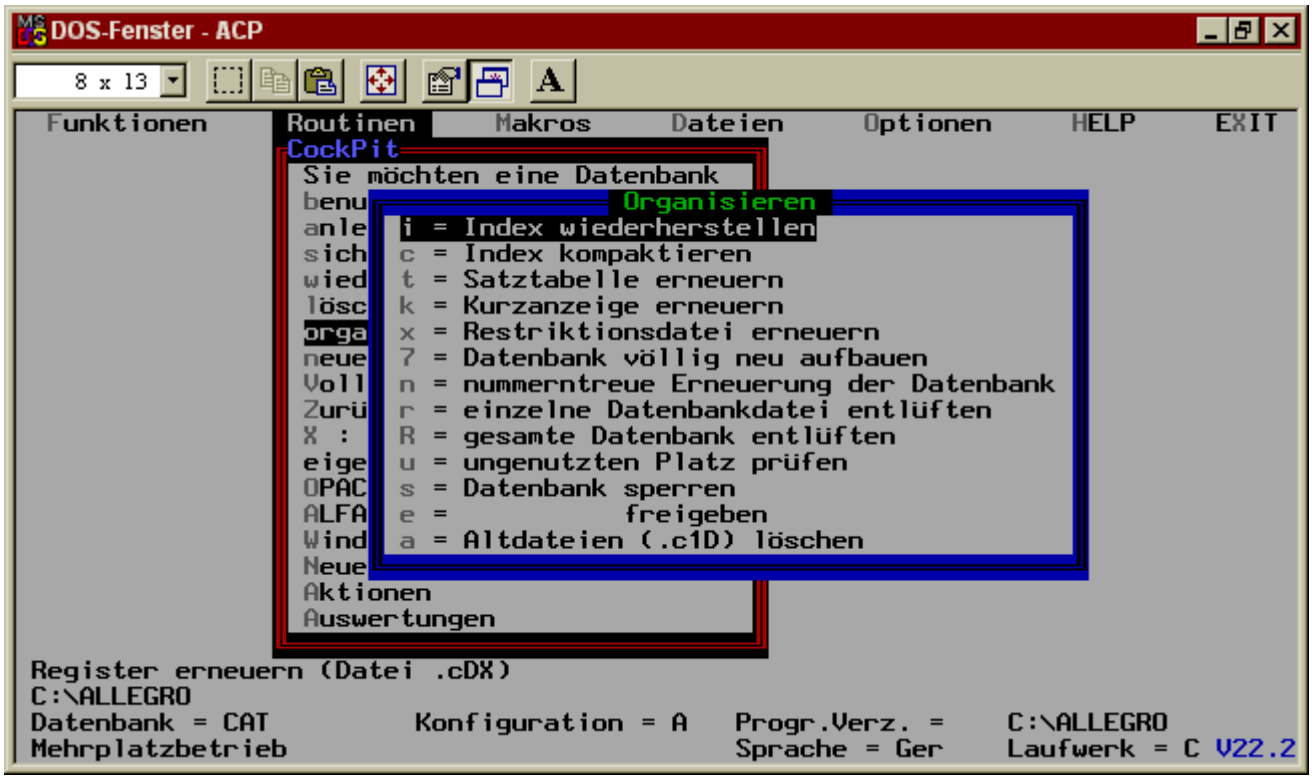
0.11.2 µr : Das Menü "Routinen"

Dieses erscheint als erstes, weil es das wichtigste Menü ist, und der Leuchtbalken steht auf "benutzen". Das Menü enthält die Routineprozeduren (daher heißt es so), die an einer Datenbank oft oder gelegentlich auszuführen sind.
Der **obere Teil** bietet fest vorgegebene Prozeduren, die man unmittelbar auf jede Datenbank anwenden kann.
Der **untere Teil** ("eigene Routinen") ist vom Benutzer frei variierbar. Die Menüpunkte (bis zu 6 Hauptpunkte mit je bis zu 10 Unterpunkten) sind in die Datei CP.OPT einzutragen (→ 0.11.6). Darunter können auch Programmaufrufe sein, die mit ***allegro*** nichts zu tun haben. Wer konsequent ist, kann seine gesamte Programmumgebung unter dem ***CockPit*** zusammenfassen und alles vom Menü "Routinen" aus starten, denn dieses erscheint immer wieder automatisch nach Abschluß eines Programmlaufs, nimmt aber dem aufgerufenen Programm keinen Platz weg.
Die Standardpunkte des Routinen-Menüs sind weitgehend selbsterklärend, deshalb hier nur eine Kurzdarstellung: (bis auf "anlegen" und "neue Version installieren" bezieht sich alles auf die gewählte Datenbank, die in den Fußzeilen ersichtlich ist)

benutzen	PRESTO aufrufen, um an der Datenbank zu arbeiten (→1.3)
anlegen	völlig neue Datenbank anlegen (→ 1.1)
sichern	BACKUP-Kopie machen oder .LOG-Datei kopieren (→ 0.7)
wiederherstellen	Playback (.LOG-Datei einspielen, → 0.7)
löschen	zugehörige Dateien (.ALD, .TBL, .STL, .ADX, .LOG) beseitigen
organisieren	... siehe unten ...
neue Version installieren	wenn eine neue ***allegro***-Diskette geliefert wird (→ 0.10)
Volltextsuche / Listen	Produktion von sortierten Listen (siehe auch 0.11.5)
Zurücksetzen der Optionen	Statuszeilen werden wieder auf die Werte der Datei CP.OPT gesetzt.
X : EXTERN	startet sofort und ohne Nachfrage die Datei EXTERN.BAT

Zu den Punkten **sichern** und **wiederherstellen** siehe Kap. 0.7!

Von größter Bedeutung für den Umgang mit einer ***allegro***-Datenbank ist das Untermenü "**Organisieren**". Hier sind Prozeduren zusammengefaßt, die zwar fast alle durch Programmaufrufe mit den richtigen Optionen auch bisher schon ausführbar waren. Jetzt aber sind sie auf einen Knopfdruck reduziert. So sieht es aus, wenn man **µr o** wählt:



Im Kapitel 7 erfahren Sie, welche Bewandtnis es ganz genau mit den einzelnen Punkten dieses Menüs hat. Für den "Normalgebrauch" reicht eine Kurzbeschreibung, die wir auf der folgenden Seite geben:

Datenbank (re)organisieren — Online in *a99*: `h org`

Die meisten dieser Vorgänge benötigen einigen Platz für Zwischenspeicherungen. Wenn es auf Ihrer Platte eng geworden ist, beachten Sie die angegebenen Werte. Im Mehrplatzbetrieb: Funktionen außer **u**, **s, e** und **a** nur aktivieren, während die Datenbank gerade nicht anderweitig benutzt wird.

i **Index erneuern** (schließt **k** und **x** ein; wenn .TBL nicht existiert, dann auch **t**) : Dieser Punkt wird benötigt, wenn
a) der Index nicht mehr funktioniert (er erscheint einfach nicht, meist mit Fehlermeldung), oder
b) eine Änderung an der Index-Parameterdatei ausgeführt wurde (**µd i**). Eine geänderte Indexdefinition bewirkt nicht automatisch eine Neugenerierung des Index, sondern dafür gibt es diesen Menüpunkt.
Vor dem Start sollte etwa doppelt so viel Platz frei sein, wie der aktuelle Index beansprucht. In der Regel wird weniger gebraucht, außer wenn man starke Erweiterungen an der Index-Parameterdatei vornimmt.

c **Index kompaktieren** : in größeren Abständen kann es sein, daß der Index sich vergrößert hat, da er durch viele Löschungen oder andere Aktionen intern Leerräume enthält und dadurch unnötig viel Platz beansprucht und evtl. auch etwas langsamer wird. Wenn man sich die Größe unmittelbar nach dem Aufbau notiert und dann gelegentlich vergleicht, wird man einen Eindruck gewinnen, wann diese Funktion sich lohnen könnte. Die Ausführung geht wesentlich schneller als die Funktion i, bei der die Schlüssel alle neu berechnet werden müssen! Beim Start sollte in etwa so viel Platz frei sein, wie der aktuelle Index selbst einnimmt. Meistens reicht zwar weniger, doch ist das nicht vorhersehbar. Funktion **c** bringt nichts, wenn man gerade vorher **i**, **7** oder **n** gemacht hat.

t **.TBL-Datei** : Die .TBL-Datei wird neu produziert. Das ist höchstens dann notwendig, wenn sie unbeabsichtigt gelöscht wurde, oder wenn die Fehlermeldung "kein Zugriff unter dieser Satznummer" kommt, denn diese deutet darauf hin, daß die .TBL-Datei möglicherweise nicht in Ordnung ist. Geht sehr schnell.

k **Kurztitelregister erneuern** : Nur die .STL-Datei (das Register 0) wird erneuert oder erstmals angelegt. Dies geht sehr schnell, ca. 10 bis 30 Datensätze je Sekunde. Es braucht 72 Bytes je Datensatz.

x **Restriktionsdatei erneuern** : Nur die .RES-Datei wird erneuert oder erstmals angelegt (falls eine solche in den Indexparametern definiert ist, → 10.2.9). Geht genauso schnell wie .STL.

7 **Datenbank völlig neu aufbauen** : Index, .TBL- und .STL-Dateien werden gelöscht, die .ALD-Dateien hintereinander eingelesen und auf neue Dateien geschrieben, die Schlüssel berechnet, die gelöschten Dateien werden neu erstellt. Gelöschte Datensätze verschwinden völlig, alle Sätze werden neu durchnumeriert, die Dateinummern bleiben aber erhalten, wenn man mehrere .ALD-Dateien hat. Anschließend sind die alten .ALD-Dateien noch unter Dateityp .c8D vorhanden. Man sollte die alten Dateien dann, wenn alles korrekt gelaufen ist, unbedingt löschen, also nachdem man gesehen hat, daß der Index wieder funktioniert. Dazu gibt es Funktion 'a'.
Speicherplatz: es muß vorher doppelt soviel Platz vorhanden sein, wie die Datenbank momentan einnimmt.

n **Nummerntreue Erneuerung** : wie Funktion 7, aber die Satznummern bleiben erhalten. Zwangsläufig gibt es dann unbesetzte Nummern (unter denen vorher gelöschte Sätze standen). Unter dem Sonderschlüssel `/[0]` im Index 1 sind diese Nummern registriert und werden im weiteren Verlauf wiederverwendet.
Die Funktionen **7** und **n** brauchen relativ viel Zeit. Sie schließen **i** und **R** mit ein.

r R **Entlüftung** einer einzelnen .ALD-Datei bzw. der ganzen Datenbank, d.h. Beseitigen gelöschter Sätze. Bei **r** kann man anschließend auswählen, welche der .ALD-Dateien entlüftet werden sollen, wenn man mehrere hat, ansonsten wird die gesamte Datenbank entlüftet. Dabei werden die .ALD-Dateien gelesen und neu geschrieben, jeder Satz erhält die Anzahl Füllzeichen, die in der .CFG-Datei festgelegt ist (→ Anh.A.1, evtl. vorher ändern!). Satznummern bleiben erhalten. Für gelöschte Sätze und die Ausgangsdateien gilt dasselbe wie bei Funktion **n**. Das Entlüften braucht nur einen Bruchteil der Zeit des Neu-Indexierens. Vorher muß etwa genauso viel Platz frei sein, wie die zu entlüftenden Dateien beanspruchen. In der Regel werden die Dateien kleiner - das ist ja auch der Sinn. Wie groß die Platzersparnis etwa sein wird, erfährt man mit Funktion **u**:

Empfehlung: vor `7,i,n,r,R` prüfen, ob alle .ALD-Dateien zur Datenbank gehören und keine .ALG vorhanden ist.

u **ungenutzten Platz prüfen** : diese Funktion stellt (mit Hilfe des Programms QRIX, → Kap.7.5) die Gesamtlänge und -anzahl der in der Datenbank vorhandenen gelöschten Sätze fest, aufgeschlüsselt nach Dateien. Das geht sehr schnell und bietet eine Entscheidungshilfe, ob man eine Entlüftung durchführen sollte.

s **Datenbank sperren** : diese zwei Funktionen sind im Mehrplatzbetrieb wichtig, denn

e **Datenbank entsperren** : eine gesperrte Datenbank erlaubt immer noch den Lesezugriff, d.h. OPAC-Benutzer merken nichts. Während der Sperre kann man gefahrlos z.B. eine Sicherungskopie, Entlüftung oder Index-Kompaktierung machen. Wenn während der Sperrung jemand einen Schreibvorgang versucht, wird durch die Meldung `bitte warten` der Sperrzustand angezeigt. Ansonsten erscheint diese Meldung höchstens einmal für wenige Sekunden, wenn unmittelbar vorher jemand anders eine Speicherung veranlaßt hat.
Soll der Zugang zur Datenbank völlig verhindert werden (nur für ***a99/alcarta***), muß man eine Signaldatei einrichten (→ Kap.0.3, Dateityp .SGF, s.a. Programme UNFREE und FREE in Kap.0.4)
Wenn ausnahmsweise beim Einzelplatz endlos **bitte warten** kommt und kein Speichern möglich ist, läßt sich das mit der Funktion **e** beheben. Der Fall ist allerdings selten und auf vorangegangene Fehler zurückzuführen.

a **Alte Dateien löschen** : Wenn man eine der Aktionen `7,n,r,R` gemacht hat, bleiben am Ende die alten Datendateien unter dem Dateityp .c8D zurück. Diese müssen gelöscht werden, bevor man erneut eine der genannten Aktionen machen kann; ***CockPit*** macht selber darauf aufmerksam. Dafür gibt es diesen Menüpunkt.

1. DOS-Programm PRESTO

0.11.3 µm : Das Menü "Makros"

Mit "Makros" sind hier Batchdateien gemeint (.BAT). (Das Windows-System hat eine ganz andere Makrosprache, genannt FLEX.)
Das Menü besteht normalerweise aus nur zwei Punkten:

Produktion starten : Die Liste der vorhandenen Stapeldateien (= Batchdateien, Dateityp .BAT) wird aufgeblättert. Das Listenmenü ermöglicht ein Umschalten zwischen Daten- und Programmverzeichnis. Man setzt den Leuchtbalken auf den gewünschten Namen und startet den Vorgang mit <Enter><Enter>.
Ohne Probleme kann man auch solche Batchdateien starten, die mit ***allegro*** nichts zu tun haben.
Vor dem Start wird ***CockPit*** völlig aus dem Arbeitsspeicher entfernt, es nimmt also dem aufgerufenen Programm keinen Platz weg.

Makros bearbeiten : Dieselbe Liste wird aufgeblättert, aber jetzt kann man sich die einzelnen Stapeldateien zwecks Durchsicht oder Bearbeitung vornehmen, auch neue produzieren. Die Möglichkeiten sind dieselben wie sie im Menü "Dateien" für die Parameterdateien angeboten werden.

Darunter können noch weitere "eigene Routinen" erscheinen, wenn der Platz im Hauptmenü "Routinen" dafür nicht reicht (siehe dort).
In der Praxis braucht man nur die wichtigsten Vorgänge an das Menü "Routinen" anzubinden (→ 0.11.6). Alle anderen Stapeldateien, die man irgendwann erstellt hat, kann man ohne sonstige Vorkehrungen über das Menü "Makros" aufrufen. Ein Problem ist höchstens die
Parameterübergabe : ***CockPit*** ermöglicht auch eine Übergabe eigener, beliebiger Parameter an aufgerufene Stapeldateien. Es legt aber außerdem vor deren Start die wichtigsten Angaben im DOS-Environment ab, und zwar unter den Namen

`-B`	Name der Datenbank	z.B. `-B=CAT`
`-D`	Name des Datenverzeichnisses	z.B. `-D=C:\ALLEGRO\KATALOG`
`-K`	Konfiguration	z.B. `-K=PICA`
`-K1`	Konfiguration, 1. Buchstabe	z.B. `-K1=P`
`-L`	Sprache	z.B. `-L=GER oder -L=ENG`
`-P`	Programmpfad	z.B. `-P=C:\ALLEGRO`

Um eigene Parameter zu übergeben, braucht man nur in der **R** oder **S**-Zeile, die für einen Batchjob angelegt wurde (→ 0.11.6), an den Namen der .BAT-Datei hinten ein Fragezeichen und einen Fragetext anzuhängen. Vor dem Start dieser Batchdatei wird dann die Frage gestellt und die Antwort wird dem Batch als Parameter %1 übergeben.

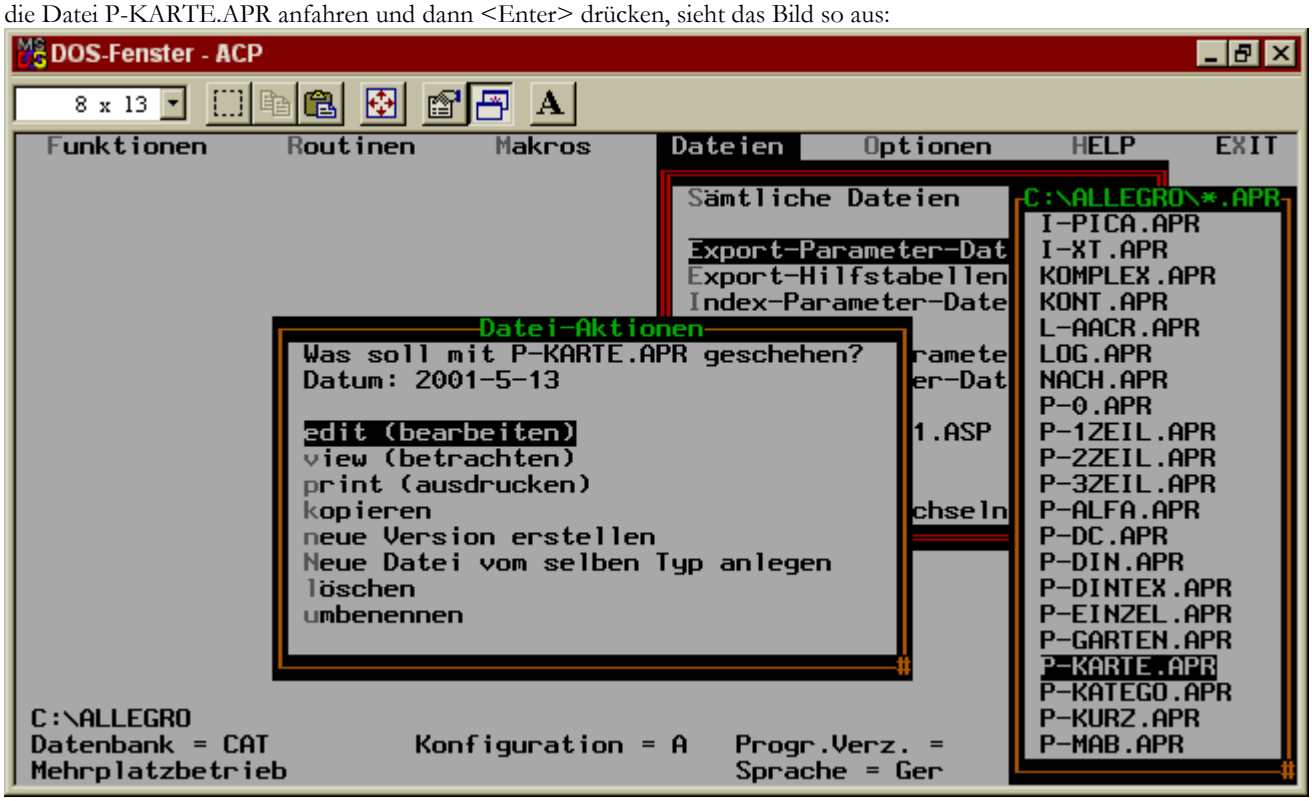
0.11.4 µd : Das Menü "Dateien"

a99: **`h files`** (allg.) und **`h adm`** (speziell)

Um die Übersicht in den vielen Dateitypen zu gewinnen, die zur ***allegro***-Umgebung gehören, ist dieses Menü besonders hilfreich. Alle Typen lassen sich getrennt und alphabetisch geordnet sichten und dann bearbeiten, umbenennen, kopieren oder löschen. Das alles wäre sonst nur über DOS mit den entsprechenden Kenntnissen und gutem Gedächtnis zu machen, oder aber mit Fremdhilfsmitteln wie Norton-Commander, der jedoch keine systemspezifischen Hilfen anbieten kann.
Jeder Unterpunkt dieses Menüs bringt eine alphabetische Liste auf den Schirm, die eben genau die entsprechenden Dateien enthält. Das Blättern in diesen Listen bietet einigen Komfort:

- Wenn die Liste zu lang für den Bildschirm ist, blättert man mit den Tasten <Bild↓> und <Bild↑> sowie <Pos1> und <Ende>, wie man es erwartet, auf- und abwärts.
- Sucht man einen ganz speziellen Namen: nur die ersten paar Buchstaben eintippen. Der Balkencursor springt bei jedem Buchstaben sofort auf den ersten dazu passenden Namen. Bei Verschreiben korrigiert man mit der Rücksetztaste. Jederzeit kann man aus diesem Blättermodus wieder mit den Cursortasten anderswo hinspringen.
- Der Menüpunkt "Umschalten", oberhalb der Liste stehend, ermöglicht ein Hin- und Zurückschalten zwischen den auf den unteren Zeilen angegebenen Verzeichnissen: Datenbank- und Programmverzeichnis sowie dem Startverzeichnis. Dazu muß man wissen: die ***allegro***-Programme suchen eine angeforderte Parameterdatei immer zuerst auf dem **Datenbank**verzeichnis. Wenn sie dort nicht ist, wird automatisch auf dem **Start**verzeichnis gesucht, dann auf dem **Programm**verzeichnis. Die hier gebotene Umschaltmöglichkeit ist deshalb eine besonders wichtige, dem Konzept entsprechende Hilfe. **Zu empfehlen** ist, die datenbankspezifischen Parameterdateien und die Konfiguration stets auf dem zugehörigen Datenverzeichnis anzuordnen, also der Datenbank direkt anzugliedern. Parameterdateien von allgemeiner Bedeutung, die man für mehrere Datenbanken nutzen kann (z.B. I-1.APR und E-1.APR), sind dagegen auf dem Programmverzeichnis besser aufgehoben, nutzerspezifische auf dem Startverzeichnis des Nutzers.

Wenn wir nun auf dem Menü "Dateien" als Beispiel den Punkt "Export-Parameter" anwählen, auf der dann erscheinenden Liste die Datei P-KARTE.APR anfahren und dann <Enter> drücken, sieht das Bild so aus:



Ein weiterer Druck auf <Enter> würde den (evtl. vom Benutzer gewählten, → 0.11.6) Texteditor aufrufen und diesem die Bearbeitung von P-KARTE.APR übergeben.
Bei Wahl von "sämtliche Dateien" erscheinen nur dann wirklich alle, wenn mit **`-a3`** gestartet wurde.
Ansonsten hat das Untermenü "Datei-Aktionen" die folgenden Funktionen:

view (betrachten) hiermit kann man sich die angewählte Datei am Bildschirm anzeigen lassen. Benutzt wird dazu ein "View-Programm" namens V.COM. Der Anwender kann es durch ein anderes ersetzen, evtl. auch durch einen alternativen Editor (→ 0.11.6)

print (ausdrucken) ***CockPit*** gibt den Befehl `print P-KARTE.APR` . Statt PRINT kann ebenfalls in CP.OPT ein anderes Hardcopy-Programm (oder auch irgendetwas völlig anderes) eingesetzt werden.

kopieren ein neuer Name wird vom Benutzer abgefragt; dieser Name kann eine Pfadangabe enthalten. ***CockPit*** kopiert P-KARTE.APR dann auf diesen Namen. Wenn es den aber schon gibt, muß man bestätigen, ob die betreffende Datei überschrieben werden soll.

neue Version erstellen:
Auch hier muß zunächst ein Name eingegeben werden, dann wird der Texteditor aufgerufen und die angewählte Datei zur Bearbeitung vorgelegt. Abgespeichert wird sie aber unter dem vorher angegebenen neuen Namen.

Neue Datei vom selben Typ: Fast dasselbe, aber man bekommt eine Kopie der Prototyp-Datei P-0.APR. (→ 10.0)

löschen Die Bedeutung dieser Funktionen ist unmittelbar verständlich. Sicherungen sind eingebaut:
umbenennen Das Löschen muß bestätigt werden, bzw. wenn der neue Name schon existiert, also die Umbenennung nicht stattfinden kann, wird man informiert.

Wenn man den obersten Punkt des Menüs "Dateien", den Punkt "sämtliche Dateien", anwählt, werden alle auf dem betreffenden Verzeichnis stehenden Dateien angezeigt. Im Untermenü "Datei-Aktionen" erscheinen dann zusätzlich Datum und Uhrzeit der Datei-Erstellung sowie deren Größe. Außerdem gibt es einen weiteren Menüpunkt:

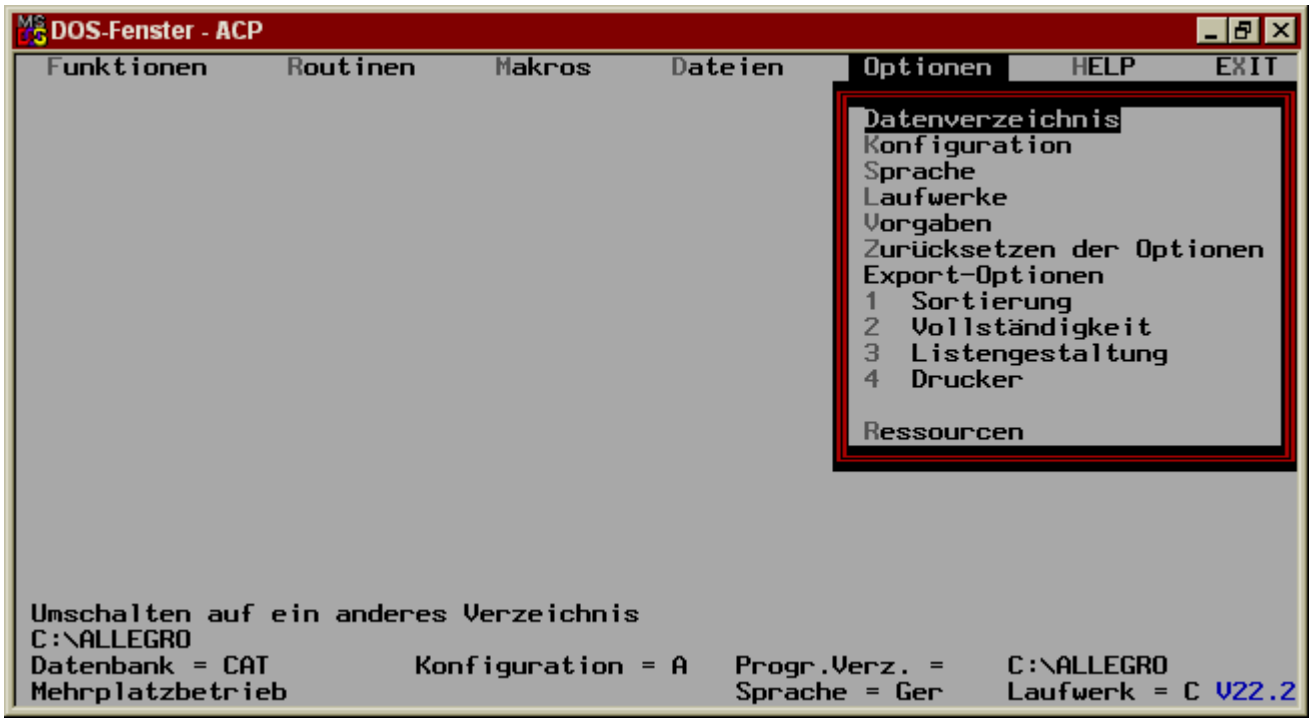
diagnost Anzeige der Datei in Form eines Dezimal-Dump, d.h. alle Zeichen werden mit ihrem dezimalen Codewert angezeigt. Sehr nützlich ist dies für denjenigen, der einen Import programmieren will und die genauen Werte der Satz- und Feldbegrenzungen nicht kennt, denn diese müssen in den Importparametern als Dezimalwerte angegeben werden (→ 11.2).

0.11.5 µo : Das Menü "Optionen"

*Einstellungen bei **a99** :* **`h a99.ini`**

Statt "Optionen" kann man auch "Einstellungen" sagen. Man verändert hier die grundlegenden Einstellungen, die für die aktuelle Sitzung gelten, man kann aber auch die Standardeinstellungen für künftige Sitzungen umdefinieren.

CockPit liest beim Start über CP.BAT die Datei CP.OPT (→ 0.11.6). Darin stehen die **Vorgaben**, die in den Statuszeilen am unteren Rand erscheinen und mit denen dann die Hauptroutinen arbeiten. Änderungen der Vorgaben wirken sich somit auf spätere Sitzungen aus. Wenn man aber etwa mehrere Datenbanken mit unterschiedlichen Charakteristika betreibt, wird man nicht jedesmal zuerst die Vorgaben ändern und dann ***CockPit*** verlassen und neu starten wollen. Deshalb lassen sich auf dem Menü "Optionen" alle Grundeinstellungen während einer Sitzung verändern. Und so sieht das Menü aus:



Zu den Punkten im einzelnen:

Datenverzeichnis : Eine Auswahlliste der Verzeichnisse erscheint. Man wählt daraus genauso wie bei den Datei-Auswahllisten. Die Angaben "Datenverzeichnis", "Konfiguration" und "Datenbank" auf den unteren Zeilen ändern sich entsprechend dieser Wahl, falls auf dem Verzeichnis eine Datenbank liegt, auch erfährt man dann deren Größe. Anschließend beziehen sich die "Routinen" auf diese Datenbank. Mit **`µr b`** springt man sofort in die Benutzung derselben. So geht der Umstieg auf eine andere Datenbank sehr schnell.

Konfiguration : Die vorhandenen .CFG-Dateien werden zur Auswahl oder Bearbeitung angezeigt wie beim Menü "Dateien". In der untersten Zeile links ändert sich dann die Eintragung, wenn man eine andere Auswahl trifft. Anschließend werden im Menü "Dateien" nur die zu dieser Konfiguration gehörigen Parameterdateien gezeigt.

Sprache : Nur GER und ENG sind tatsächlich wählbar. Ansonsten haben Anwender UIF-Dateien für Spanisch und Niederländisch erstellt. Eigene Übersetzungen sind möglich. Angepaßte Versionen der UIF-Dateien können unter eigenen "Sprachbezeichnungen" angelegt und über den Unterpunkt "USR" gewählt werden.

Laufwerk : Umstellen des aktuellen Laufwerks; anschließend Datenverzeichnis wählen.

Vorgaben : ***Sehr wichtig!*** Bearbeitung der Vorgabendatei, in der Regel CP.OPT (→ 0.11.6, nächste Seite). Das sind die Grundeinstellungen, mit denen das Programm startet, z.T. sind sie auf den Fußzeilen zu sehen.

Ressourcen : Zeigt die Größe des für ***allegro*** verfügbaren Arbeitsspeichers und des freien Plattenraums.

Druck-Optionen : Diese 4 Punkte sind wichtig für die Produktion sortierter Listen. Sie finden eine genaue Beschreibung dieser Möglichkeiten im Zusammenhang mit dem Export in den Kapiteln 4 und 6.
Dieselben Punkte sind unter "Routinen , Volltextsuche/Listen" (**`µr v`**) zu finden.

0.11.6 Die *CockPit*-Vorgabendatei CP.OPT

Auf den folgenden zwei Seiten ist die mitgelieferte Datei CP.OPT abgedruckt. Sie enthält genügend viele Kommentare, um weitere Beschreibungen fast überflüssig zu machen. Über **µo v** kommt man in die Bearbeitung dieser Datei und kann die nötigen Anpassungen vornehmen. Die Kommentare werden in anderer Schriftart dargestellt, die eigentlichen Befehle dagegen in **Courier**. Sie können ansonsten Kommentare auch daran erkennen, wie in den Parameterdateien, daß die Zeile mit einem Leerzeichen beginnt bzw. innerhalb einer Zeile 2 Leerzeichen vorangehen.
Unerwünschte Menüpunkte können Sie entfernen: nehmen Sie die zugehörigen Zeilen aus der Datei UIFCGER heraus!

```
    CP.OPT = Datei mit Variablen für den Start von CockPit
     Für Mehrplatzbetrieb: in CP.BAT im Aufruf von ACP die Option -S wegnehmen
    !  Ergänzen Sie in der Datei CP.BAT
         -M  für Monochrom-Bildschirme,(und zwar anhängen an die Zeile  acp ...)
         Alle Zeilen, die mit Leerzeichen beginnen, sind Kommentare!
         ═══════════════════════════════════════

c C      wenn Sie z.B. auf D: arbeiten, D statt C einsetzen [nicht mehr relevant]
l GER    Sprache der Menütexte (diese Zeile darf nicht weiter unten stehen)
C cp     Name der Batchdatei, die nach einer Routine wieder aufgerufen wird

         In die folgende Zeile tragen Sie Ihr eigenes Datenverzeichnis ein:
d C:\ALLEGRO\DEMO     wiederholbar (2fach oder 3fach)
  ---------------
         aber dann muß der Datenbankname, z.B. \cat noch angehängt werden
         und in die nächste das Programmverzeichnis:
P C:\ALLEGRO
  ----------
         Kommentar und Tip für MS-DOS-Kenner:
         CP.BAT  ist ein Batch-File, in dem ACP.EXE aufgerufen wird.
         Wenn man ACP direkt aufruft, werden die angewählten Programme
         nicht gestartet, sondern es entsteht ein neues Batch-File
         namens CCC.BAT, in welchem der Programmaufruf mit den gewählten
         Optionen steht; am Ende wird wieder CP.BAT aufgerufen. Die Datei CCC.BAT
         kann man als Vorlage für eigene Stapeldateien nehmen!

         ALLGEMEINE OPTIONEN:
         (diese erscheinen in den 2 Fusszeilen)
         Für die Beispiel-Datenbank werden folgende Einstellungen mitgeliefert:
         ACHTUNG:
         Die Funktionen zum Löschen, Regenerieren etc. sind nur verfügbar, wenn man
         ACP mit Option -a2 bzw. -a3 startet. Bauen Sie das auf Wunsch in CP.BAT ein!
a 2      Zugriffsberechtigung  (0: Menü "Funktionen" entfällt, 3: alle Rechte)
  a 320  wenn 3 Datenbanken gleichzeitig, dann z.B. so
  A 3          Berechtigungsstufe für CockPit, falls von a abweichend
b CAT          Datenbankname   (Name Ihrer Datenbank, d.h. Indexparam.)
k A            Verwendete Konfiguration  (z.B. U für USMARC)
         hier wird $A.CFG genommen, wenn dies nicht existiert, dann A.CFG
E x
         Editor-Aufruf für die Parameterdateien etc.
         (schreiben Sie z.B.      E Notepad , falls Sie Notepad benutzen wollen
                                  E EDIT  für den Editor von MS-DOS 5)
         (Default ist sonst das mitgelieferte X.EXE: die Anleitung dazu steht in der Datei X.DOC)
V v
         Viewer-Aufruf
         (falls Sie ein besonderes Programm zum Betrachten der Dateien benutzen wollen, z.B. DEBUG - Default ist
         sonst das mitgelieferte V.COM)
B B:
  für das Backup kann hiermit ein Laufwerk vorgegeben werden  (default ist A☺
p D-NEU  Name der Anzeigeparameter; Default ist D-1.APR
```

1. DOS-Programm PRESTO

H PRINT (PRINT ist default)
zum Ausdrucken von Parameterdateien etc. kann man eine HARDCOPY-Funktion vorgeben.
Achtung: wenn man PRINT benutzt, muß man vor Start von ACP wenigstens einmal den Befehl PRINT geben, damit der residente Teil geladen ist. Unter Novell wird man evtl. NPRINT nehmen.
Man könnte auch H TYPE geben, dann würde die angewählte Datei am Bildschirm erscheinen!

L 1200000
maximale Logfile-Größe in Byte. Wenn die Datei .LOG größer wird, informiert CockPit Sie bei jedem Start darüber, daß eine Sicherung sinnvoll wäre.

```
*************************************************************************
* Außerdem können Sie eine Anzahl eigene Menüpunkte ins CockPit einbauen: *
  Die eigenen Menüpunkte werden an das Menü ROUTINEN angehängt.
  Die dafür nötigen Zeilen beginnen mit R oder S
```
ACHTUNG a99: Mit `X cp` kann man diese Routinen reaktivieren und auch in a99 nutzen
```
  R MenuItem, Batch?Frage, InfoLine       Hauptpunkt, bis zu 16 Stück
                    wenn mehr als 6, dann kommen die weiteren auf Menü Makros
  S MenuItem, Batch?Frage, InfoLine       Submenü-Punkt, bis zu 10 je R-Hauptpunkt
   mit:
        MenuItem    = Bezeichnung des Menüpunkts
        Batch = Name des aufzurufenden Batch-Files.
                          Hier kann auch ein MS-DOS-Befehl stehen
       (Batch-Files müssen auf dem aktuellen Verzeichnis liegen, sonst Pfadangabe nötig).
        Bei R KANN hier '&' stehen, dann MÜSSEN S-Zeilen folgen
        InfoLine    =     dieser Text erscheint auf der Hilfszeile, sobald der
                          Menüpunkt aktiviert wird.
        Frage       = Frage an den Benutzer, bevor Batch gestartet wird
                    Die Antwort auf die Frage wird als Parameter %1 an Batch weitergegeben.
```
Die folgenden Zeilen sind Beispiele, die zeigen, wie die Struktur aussehen muß. Vergleichen Sie dazu, wie die Menüpunkte im Menü "Routinen" herauskommen, insbes. wie die Untermenüs funktionieren:

```
R 1 Einfache Anwendungen, &, Pakete QUEX, QUANT, PRONTO, EXPEX (ab V14c)
S 1 Sortierte Listen, QUEX.BAT, Menuegesteuerte Produktion von Titellisten
S 2 Statistik-Auswertungen, QUANT.BAT,  Quantitative Auswertungen
S 3 Neues Projekt anlegen, PRONTO.BAT, Menuegefuehrtes Anlegen e. neuen Datenbank
S 4 Parameter-Experimente, EXPEX\EXPEX.BAT, Exportsprache lernen mit MARC!

R 2 OPAC, apac -e I-1/DOWNLOAD.%-K1%LG, Online-Katalog
R 3 MENUED, menued -a3 -n1 -S , Editor mit Komfort
R 4 ALFA, alfa -a3 -n1 -S -s"|3m", Einfach-Ausleihe

R 5 Download, &, Erzeugen von Downloads und ihre Verarbeitung
S Download mit PRESTO, P-DOWN.BAT, Daten aus der Datenbank herunterladen in eine Grunddatei
S Download mit SRCH, S-DOWN.BAT, Mit SRCH e. Grunddatei aus Datenbank herausziehen
S Sortierte Liste, S-LIST.BAT, aus Download-Datei eine sortierte Liste produzieren
S Unsortierte Liste, U-LIST.BAT, letzte Download-Datei unsortiert ausgeben
S Liste anzeigen, v liste, zuletzt produzierte Liste anzeigen
S Registerauschnitte, QR.BAT?Welches Register, Teile des Index ausgeben

R 6 Import-Aktionen, &, Fremddaten-Import und Einarbeitung in eigene Datenbanken
S 1 DEUTSCHE BIBLIOTHEK DISK, dbdisk.bat, DB-Diskettendaten importieren
S 2 DEUTSCHE BIBLIOTHEK CD, dbcd.bat, DB-Daten von CD-ROM importieren
S 3 VLB-Daten, vlbkonv.bat, Daten aus dem Verzeichnis Lieferbarer Bücher
S 4 OCLC, oclc.bat, MARC-Daten vom OCLC oder von der LC
S 5 PICA, pica.bat, Pica3-Daten vom GBV

R 7 Aktionen, &, besondere Routinen
S MS-DOS Exkursion, COMM.BAT, Ausstieg in's MS-DOS - Rückkehr mit EXIT
S Indexparameter, edit-ixp.bat, Indexparameter-Datei bearbeiten
S Anzeigeparameter, edit-d1.bat, Datei D-1.APR bearbeiten
S Konfiguration, edit-cfg.bat, CFG-Datei bearbeiten
```

```
S Sniffer, sniffer -a3 -x, Untersuchung der Datenbank auf formale Fehler
S Erw. Zeichensatz, ostwest.bat, Umstellen auf den OSTWEST-Zeichensatz
S Norm-Zeichensatz, VGANORM.bat, Rücksetzen auf den VGA-Zeichensatz
```

Wenn an der Position *Batch* kein Batchfile, sondern ein EXE- oder COM-Programm eingesetzt wird, kann der Name ohne .EXE bzw. .COM angegeben werden, .BAT darf dagegen nicht fehlen. (Vgl. oben die Zeilen unter R 7)

```
  ********************************************************
 * Voreinstellungen für die einzelnen "allegro"-Programme *
            Hier können für die einzelnen Programme eigene Optionen stehen, die dann im Menü 1 "Funktionen"
             benutzt werden. Hinter einer Zeile mit der Kennziffer des Programms folgen Zeilen mit dem
            Optionsbuchstaben und dem zuzuordnenden Wert:
1   presto      (Optionen unterhalb dieser Zeile gelten nur im Menü "Funktionen")
s a             Index wird zuerst bei 'a' aufgeklappt
 p ANZEIGE      statt D-1.APR soll ANZEIGE.APR benutzt werden
 q P-KARTE      für den Druck (mit F2) die Parameter P-KARTE.APR

4   srch
f 4
d c:\allegro\beispiel
m 1

5   import
v 0
e i-1
...
```

Nachwort zu Kapitel 0

allegro wird oft als weniger "anwenderfreundlich" empfunden als einige andere Systeme. Der Einstieg gilt deshalb als schwierig. Nach einiger Zeit sieht man jedoch, daß dies der Preis der Flexibilität ist. Die Einübung erfordert einige Zeit und genaues Formulieren der eigenen Wünsche und Vorstellungen. Dieses jedoch sollte am Anfang **jeder** DV-Anwendung stehen.
Wahrscheinlich wünschen sich die meisten Interessenten eine strengere oder mehr ins Detail gehende Menüführung. Das ***CockPit*** ging ein gutes Stück in diese Richtung, und die Programme für Erwerbung und Ausleihe setzten ebenfalls dessen Menü- und Fenstertechnik ein. Die Windows-Programme **a99** und **alcarta** haben dann neue Maßstäbe gesetzt (siehe Kap.2). Wer länger mit ***allegro*** arbeitet, weiß die Freiheitsgrade und das schnelle Arbeiten zu schätzen, und daran soll es auch in Zukunft keine Abstriche geben. Es handelt sich bei ***allegro***, wie man auch sagen könnte, um ein Katalogisierungssystem für "Erwachsene", die ihre Spielregeln kennen und im eigenen Interesse danach handeln. Auch ein noch so ausgefeiltes Menü- und Formularsystem ersetzt ja nicht die Regelwerkskenntnisse. Ob und wie z.B. ein Urheber einzugeben ist, und ob der, den man eingegeben hat, tatsächlich einer ist, das kann unmöglich alles über Hilfetexte abrufbar gemacht und schon gar nicht automatisch überprüft werden.
Da alle ***allegro***-DOS-Programme als Module aufrufbar und von anderen Prozessen (vor allem Batchprogrammen oder UNIX-Shellprozeduren) aktivierbar sind (→ Kap.12), können Anwender, die das ***CockPit*** nicht einsetzen wollen, sich komplette Menüsysteme schaffen, in denen alle Abläufe, die im Normalbetrieb vorkommen, mit wenigen Tasten aktiviert werden können.
Im Windows-System werden neue Funktionen und Menüs mit der schon erwähnten FLEX-Sprache erstellt. Auf viele davon wird in diesem Handbuch an den passenden Stellen hingewiesen.
Benutzeroberflächen sind im übrigen besonders problematisch hinsichtlich der Portabilität. Für ***allegro*** wurde angestrebt, daß es weitgehend identisch unter MS-DOS, Windows und UNIX lauffähig ist und Datenbanken unmittelbar zwischen diesen Plattformen ausgetauscht werden können. Dazu ist es notwendig, Oberfläche und Kern in den Programmen so zu trennen, daß nur relativ kleine Programmteile betriebssystemspezifisch zu schreiben sind. Inzwischen hat sich diese Bemühung ausgezahlt, nachdem es etliche Jahre langsamer voranging, als wenn man sich ausschließlich auf ein bestimmtes Betriebssystem konzentriert hätte.
allegro-Kataloge im WWW gab es schon Ende 1999 weit über 50. Einen Ersatz für das konventionelle Programm PRESTO, das Hauptprogramm beim Katalogisieren, können diese aber nicht bieten, und auch für lokale Netze oder für ein Intranet braucht man mehr Effizienz, als ein Web-Interface bieten kann. Deshalb wurden die Windows-Programme ***a99*** und ***alcarta***.entwickelt. Sie lösten ab Anfang 2000 die Programme PRESTO und APAC weitgehend ab (geeignet ab Windows'95 aufwärts). Es gibt jedoch kein Entweder-Oder, denn die alten und die neuen Programme können im lokalen Netz gleichzeitig auf dieselbe Datenbank zugreifen. Auch in V34 sind die DOS-Programme nochmals aktualisiert und kompatibel. Der Anwender hat die Wahl! Gegen Ende 2013 kam dann aber eine Web-Anbindung namens ***a35*** hinzu, die ein größeres Potential hat als alle früheren. ***a35*** basiert auf den gängigen Standards HTML5, PHP, Javascript, CSS3 und UTF-8 und ist damit in allen zeitgemäßen Browsern arbeitsfähig. Für den Ausbau und die spezifischen Anpassungen braucht man nur die Skriptsprache FLEX, mit der alle Zugriffe auf die Datenbank programmiert werden können.

1. DOS-Programm PRESTO

1 Schnellzugriff (DOS-Programm PRESTO)

Abgelöst vom Windows-Programm ***a99***, → Kap. 2

1.0 Aufgabenstellung und Konzept

Dieses Kapitel antwortet auf die zentralen Fragen: Wie komme ich an die einmal erfaßten Daten wieder heran? Wie, und wie schnell, kann ich etwas wiederfinden und wie kann ich daran Veränderungen vornehmen? Sind neu eingegebene Daten sofort voll verfügbar? Diese letzte Frage ist uneingeschränkt mit Ja zu beantworten, zu den zwei anderen ist die Antwort nicht so kurz…

allegro bietet drei Zugriffsmethoden an, die im Mehrplatzbetrieb gleichzeitig arbeiten können:

1. **Schnellzugriff** über einen **Index** mit alphabetischen Registern, in denen man blättern kann. Das ist die normale, meistbenutzte Methode. Die DOS-Hauptprogramme hierfür sind **PRESTO** und **APAC**, die anderen (MENUED, ALFA, REF, aLF, ORDER, INVENT) können dasselbe und sehen genauso aus, haben aber noch zusätzliche Funktionen. PRESTO und APAC werden unter Windows durch ***a99*** und ***alcarta*** abgelöst (siehe Kap. 2), können aber weiter benutzt werden. Für UNIX (Solaris und Linux) gibt es ein weniger komfortables **`presto`**
2. **Volltext-Zeichenfolgensuche** = Programm SRCH (wird beschrieben in Kap.4)
   Diese Methode eignet sich nicht für einen OPAC, dafür ist der Zugriff zu langsam.
3. **Client/Server-Zugriff** mit dem ***avanti***-Server als Kernprogramm. Damit werden die WWW-Kataloge realisiert. Dieses Handbuch geht darauf nicht ein, es gibt dazu eine eigene Dokumentation auf dem Webserver.

Es ist alles so vorbereitet, daß man mit der Methode 1 sofort beginnen, d.h. Daten erfassen und dann gleich in alphabetischer Folge darauf zugreifen kann. Besonders Eilige können vom ***CockPit*** aus mit den Menüpunkten **`µr a`** und **`µr b`** sofort eine Datenbank anlegen bzw. eine vorhandene benutzen (→ 0.11).

Der Index besteht aus bis zu 11 **Registern** (man könnte auch **Suchbegriffslisten** sagen), die alle gewünschten **Suchbegriffe** enthalten, also z.B. Namen, Stichwörter, Titel, Serientitel, Schlagwörter, Signaturen, Notationen, Erscheinungsorte und -jahre, Verlage, ISBN/ISSN etc.

Welche Suchbegriffe (auch **Schlüssel** genannt) es bei einer Datenbank tatsächlich gibt und wie sie aussehen, das steht in einer zur Datenbank gehörigen **Index-Parameterdatei** (Typ *.d*PI, → Kap.0.3, Kap.10 u. 7).

Für den "Normalfall" gibt es mehrere vorbereitete Indexstrukturen, die man sofort benutzen kann. Folgende werden mitgeliefert:
(*Wichtig:* die Entscheidung für das eine oder andere Modell ist **nicht endgültig**; die einmal erfaßten Daten kann man später evtl. in ein anderes Schema exportieren und mit einer anderen Parameterdatei neu indexieren. Das erfordert aber in jedem Fall große Sorgfalt und einige Kenntnisse der Parametrierung.)

`CAT.API` ein sehr aufwendiges, in der Praxis **bewährtes** und **optimiertes** Modell, ebenfalls für das Standardschema A.CFG, d.h. die Datenstruktur ist dieselbe. Als Variante für die Musik-Katalogisierung existiert BOL.API (***bolero***-Konzept).
Diese Struktur wird jedem **empfohlen**, der keine wohlerwogenen Gründe für eine andere Entscheidung hat.
Die mitgelieferte Beispieldatenbank beruht auf diesem Schema.

Index: 9 verschiedene Register:
Personen / Körperschaften / Stich- und Schlagwörter / Sachtitel / Zeitschriften- und Serientitel / Klassifikation:Jahr/Typ / Signatur / Verlag:Jahr / ISBN u.a. Identifikationsnummern .

`NMN.API` ein besonders schlichtes Modell, gleichfalls verwendbar mit dem Standard-Kategorienschema (A.CFG); geeignet für Dokumente aller Art, insbesondere Bücher und Zeitschriftenaufsätze. Zu empfehlen höchstens noch für ganz schwache PCs.

Index: "Kreuzregister" (alles in einem Alphabet) für Personen- u. Körperschaftsnamen, Titelstich- und Schlagwörter, Serientitel, Schlagwörter; Normdaten für Namen, … (Anh.B)
Zum Standardschema passend gibt es z.B. die Import-Parameter MAB2.AIM für DDB-Disketten-daten und OCLC.AIM für USMARC-Daten der LoC oder vom OCLC, mit denen die DB-Sachgruppen bzw. die Library-of-Congress-Signaturen mittels Konkordanzlisten auf einen Nenner gebracht werden (→ Anh.B.5).

`BANK.NPI` Ab V26 gibt es das modernere "Neutralformat" N.CFG und dazu eine Parametrierung mit neuen Eigenschaften. Dazu findet man alles unter http://www.allegro-c.de/doku/neutral/ .

MAB.DPI verwendbar mit dem DB-MAB2-Kategorienschema ($D.CFG)
Index: Namen, Titelstichwörter, Titel, Serien, ISBN.
Anm.: MAB2 wird 2012 als Austauschformat durch MARC21 ersetzt

PICA.PPI verwendbar mit dem Pica.Kategorienschema ($P.CFG), das dem Format Pica3 des niedersächsischen Verbundes entspricht.
Index: Personennamen, Titelstichwörter, Titelanfang, Körperschaften, ISBN, Erscheinungsorte u. -länder, Sachgebiete, Signaturen

MARC.UPI zu verwenden mit der Konfiguration $U.CFG für MARC21.
Index: ähnlich wie bei CAT.API

Zur Standardkonfiguration A.CFG bietet Anhang B eine etwas gekürzte Darstellung des Kategorienschemas mit einigen Beispielen und Hinweisen auf die Indexgestaltung. Die vollständige, aktuelle Dokumentation des Formats ist im Web:
http://www.allegro-c.de/doku/form2004/

Die Indexdateien, die mit Hilfe der Parameter erzeugt werden, haben die Typkennung *dbn.c*DX, wobei *dbn* Ihr selbstgewählter Datenbankname und *c*.CFG Ihre Konfiguration ist (→ 0.3). *Beispiel:* bei Verwendung von A.CFG und CAT.API bekommt der Index den Namen CAT.ADX, bei Verwendung von N.CFG und BANK.NPI heißt der Index BANK.NDX.
Sie können sich völlig andersartige Kategoriesysteme und Schlüssel ausdenken. Dazu ist es nötig, eine eigene *.c*PI-Parameterdatei anzulegen oder eine der genannten zu modifizieren: diese Dateien sind ausführlich kommentiert, das hilft Ihnen bei der Abwandlung von Details. Fast nichts ist fest eingebaut - deshalb kann der Job aber leider nicht leicht sein! Auch nachträglich sind Änderungen an der Indexgestaltung möglich, nur muß man den Index dann neu "generieren" (→ Kap.7).

Die Parameterdateien enthalten jeweils auch Listen von Zeichenumwandlungsvorschriften (p- und q-Befehle, → Kap. 10.2.4). Sie können auch hier Änderungen vornehmen. Normalerweise werden die Indexeinträge in Kleinbuchstaben und mit aufgelösten Umlauten generiert. Man kann genausogut mit Großbuchstaben arbeiten oder bestimmte Eintragungstypen in Groß-, andere in Kleinbuchstaben verlangen. So wird z.B. manchmal, vor allem in Kreuzregistern, die Signatur in Großbuchstaben indexiert - sonst würden die Signaturen im Index ins Stichwortalphabet eingereiht.
Zu jeder Konfiguration kann man eine Kopie der Standard-**Stopwort-Tabelle** `SWL1.APT` verwenden (also z.B. `SWL1.UPT`), die von der Index-Parameterdatei aus mit dem Befehl `tswl1` hinzugeladen wird. Diese Liste ist beliebig erweiterbar. Sie muß mit einer Befehlszeile 'N' beginnen und enden und alphabetisch geordnet sein. Soll ein Wort aus dieser Liste im Ausnahmefall trotzdem ins Register, setzen Sie bei der Erfassung das Zeichen "Klammeraffe", also '@', davor. (Z.B. UNO oder MIT würden sonst als Stopwörter behandelt!) Umgekehrt wird ein Wort, das mit dem Nichtsortierzeichen (normalerweise '¬') markiert ist, nicht ins Register übernommen (→ Anh.A.1.3), auch wenn es kein Stopwort ist.
Empfehlenswert ist, daß Sie für Ihre Datenbank *dbn* eine spezifische Stopwortliste *dbnstop.c*`PT` anlegen, und diese zusammen mit der Parameterdatei *dbn.c*`PI` in das Unterverzeichnis zu legen, wo sich die Datenbank befindet. In *dbn.c*`PI` muß dann der Befehl `t`*dbnstop* statt `tswl1` stehen. Auch die Konfiguration *c*`.CFG` kann auf dem Datenverzeichnis liegen. Dann sind alle zur Datenbank gehörigen Elemente im selben Verzeichnis.

1.1 Anlegen einer neuen Datenbank

***a99*: `h newdb` eingeben**

Wie eröffnet man eine völlig neue Datenbank? Wer noch keine Daten hat, kann das Schnellzugriff-Programm PRESTO auch sofort als Erfassungsprogramm benutzen. Dieser Abschnitt beschreibt mehrere mögliche Einstiege. Die Indexdatei entsteht in jedem Fall automatisch während der Erfassung.

Hinweis für Anfänger:
Startbefehl für das DOS-System ist **cp**, nicht **allegro**. Das Programm `allegro.exe` ist identisch mit `a99.exe`. (Vor langer Zeit war es einmal anders.)

1.1.1 Der schnelle Standard-Einstieg (DOS)

Illustrierte Schnell-Anleitung für Windows: http://www.allegro-c.de/schnell.htm

Das hier beschriebene Verfahren funktioniert nur in einer DOS-Umgebung, nicht mit Windows'7/64!

Empfehlung: Wenn Sie Buch- und Aufsatzdaten erfassen wollen und keine ganz ausgefallenen Wünsche haben, können Sie sofort mit der Erfassung beginnen. Sie benutzen dann das Standardschema A.CFG und eine der dazu mitgelieferten Indexdefinitionen, vorzugsweise CAT.API. Ab V26 gibt es als Alternative das "Neutralformat": dazu erfahren Sie alles unter der Web-Adresse http://www.allegro-c.de/neutral/ .

Abwandlungen der Register (z.B. für die Signaturen) und Erweiterungen des Schemas sind später immer noch in weiten Grenzen möglich. Auch der Umstieg vom A- zum N-Format!

So wird's gemacht: Sie gehen im DOS-Fenster in Ihr Programmverzeichnis und tun dann genau folgendes:

Sie geben ein: **`cp <Enter><Enter> a`** (***CockPit*** starten, Funktion "Datenbank anlegen")
(wenn *nicht* Konfiguration A gewünscht, dann zuerst Konfiguration wählen, z.B. N)
`KATALOG` als Datenbankpfad angeben (Verz. C:\ALLEGRO\KATALOG wird angelegt)
`CAT` als Index-Parameterdatei auswählen (bei geringen Ansprüchen **`NMN`**)
`g <ENTER>` Menüpunkt GO (START) aufrufen

CockPit startet dann das Programm PRESTO mit den gewählten Vorgaben.

Die "Abfrageliste" (→ Kap.3.2) erscheint, damit Sie die erste Titelaufnahme eingeben können. Nach deren Abspeicherung (mit der Funktionstaste F10) ist auch der Index (→ 1.4) schon benutzbar. Nehmen Sie sich jetzt den Abschnitt 1.5 vor. Sie drücken auf **I** und es kommt die Abfrageliste für das Eingeben des nächsten Datensatzes.

Es entstehen auf `\ALLEGRO\KATALOG` folgende Dateien, wenn man CAT gewählt hatte (→ 0.3 und 0.8):

`CAT_1.ALD` enthält die eigentlichen Daten

`CAT.ADX` die Indexdatei (die alphabetischen Register)

`CAT.STL` das Kurztitelregister (zur Benutzung → 1.4.5)

`CAT.RES` die Restriktionsdatei (zur Benutzung → 10.2.9) [ab V15]

`CAT.TBL` die Satztabelle (für interne Zwecke)

`CAT.LOG` die Sicherungsdatei (für den Katastrophenfall, → 0.7). Alle Eingaben und Korrekturen werden darin protokolliert.

Hintergrundinformation:

Dies ist unter DOS der einfachste Weg, eine neue Datenbank zu beginnen. Man bedient sich hierbei einer fertigen Konfiguration (A.CFG) und einer fertigen Index-Parameterdatei (CAT.API) sowie der Parameterdatei D-1.APR für die Bildschirmanzeige. Diese drei sind zu verändern, wenn die Datenbank anders aussehen soll. (Beim Neutralformat verwendet man statt dessen die Dateien N.CFG, BANK.NPI und DISP-WIN.NPR.)

Das ***CockPit*** ruft, wie gesagt, PRESTO auf und teilt ihm diese Einzelheiten in Form von "Optionen" mit (→ Kap.12.).

Tip für MS-DOS-Kenner: Starten Sie ***CockPit*** mit dem Befehl **`acp -ocp.opt`** statt mit **`cp.bat`**, dann erzeugt es eine Datei CCC.BAT, was es sonst auch täte, startet sie aber nicht. Sehen Sie sich diese Datei an und vergleichen Sie den Aufruf von PRESTO mit den in Kap.12 beschriebenen Optionen.

Andererseits können die einzelnen Programme des ***allegro***-Pakets (→ 0.4), so auch das hier benutzte PRESTO, jeweils einfach mit ihrem Namen aufgerufen werden. Sie fragen dann vom Benutzer alle nötigen Einzelheiten ab, wie z.B. den Namen der Datenbank. Sie sehen das bei der individuellen Methode, die in 1.1.3 beschrieben wird. Wer ein eigenes Projekt mit ganz neuen Strukturen entwickeln will, sollte noch folgendes lesen:

1.1.2 PRONTO : Menügeführter Datenbank-Neuentwurf

Hat man ein relativ einfaches Projekt, bei dem keine unterschiedlichen Satztypen auftreten und nur bis zu 20 verschiedene Felder vorkommen, kann man ein "Fertigteilverfahren" anwenden. Man startet die Stapeldatei PRONTO.BAT, entweder direkt mit der Hand oder vom ***CockPit*** (unter Menüpunkt "Einfache Anwendungen"). Dann läuft ein weitgehend selbsterklärender Dialog ab. Man hat eine Reihe von Fragen zu beantworten, danach stellt das Programm alle notwendigen Dateien aus vorgefertigten Teilen selbständig zusammen und startet dann das ***CockPit*** oder PRESTO mit den richtigen Vorgaben, damit man sofort mit der Dateneingabe beginnen kann.

Wenn Sie sich für die Hintergründe und Abläufe dieser Methode interessieren oder diese modifizieren wollen, lesen Sie die Dokumentation zu PRONTO; sie steht in der Datei PRONTO.TXT.

1.1.3 Neue Datenbank "zu Fuß" anlegen (ohne Menüführung) [DOS]

Im Windows-Programm ***a99*** gibt es ein Verfahren unter dem Menü
„Datei / Weitere Datenbank öffnen / Neue Datenbank anlegen“

0. *Empfehlung:* Studieren Sie, falls noch nicht geschehen, Kap.0, um einen Überblick über die Funktionen, das Dateisystem und die Terminologie zu gewinnen. Sie sehen dann, daß Sie mit verschiedenen Kategoriesystemen und Indexdefinitionen arbeiten können. Für einen individuellen Datenbankaufbau sind diese Vorgaben alle variierbar. In den Abschnitten 0.8 und 0.9 ist zusammengestellt, welche Dinge zu einer Datenbank gehören, und wo und wie man Änderungen machen kann.
1. Wählen Sie ein Kategoriesystem *c* , mit dem Sie arbeiten wollen, d.h. eine der vorhandenen Konfigurationsdateien des Typs .CFG. Fortgeschrittene nehmen darin Anpassungen vor oder erstellen sich eine eigene Konfiguration (→ Anh.A). Wenn Sie *c*=A wählen, dann brauchen Sie weiter nichts zu tun, *sonst* setzen Sie in die ***CockPit***-Vorgabendatei CP.OPT den Befehl **k** *c* ein, damit künftig diese Konfiguration immer automatisch verwendet wird (→ Kap.0.11.6).
2. Wählen Sie einen Datenbanknamen mit **bis zu 4 Buchstaben**; symbolisch schreiben wir dafür *dbn*.
3. Als nächstes wird eine Datenbankstruktur = Index-Parameterdatei (Typ .*c*PI) gebraucht. Sie legen eine neue Index-Parameterdatei *dbn*.*c*PI an, indem Sie eine der vorhandenen kopieren (z.B. mit
   `COPY CAT.API` *dbn*`.API`). Fortgeschrittene ändern diese Datei evtl. ab oder schaffen sich eine eigene (notwendig, wenn eine eigene *c*.CFG verwendet wird!) (→ Kap.7 u. 10)
4. Wenn ein eigenes Schema *c* verwendet wird, muß außerdem eine Parameterdatei namens D-1.*c*PR vorhanden sein, sonst können Sie die Daten nur in Kategorieform betrachten (→ 10.). Mitgeliefert werden D-1.APR, D-1.PPR, D-1.UPR. Die neue Datei D-1.*c*PR kann zunächst ganz einfach sein und später ausgebaut werden.
5. Wählen Sie einen Namen für ein Unterverzeichnis, auf dem dann Ihre Datenbank liegen soll. (Dieser Name kann bis zu 8 Zeichen lang sein.) Symbolisch schreiben wir dafür *sub*.
6. Legen Sie Ihr Unterverzeichnis mit diesem Befehl an:

   md *sub*

   Es kann *sub* = *dbn* sein, muß aber nicht; es können auch auf einem Unterverzeichnis hinterher mehrere Datenbanken mit unterschiedlichen Namen und gleicher Konfiguration liegen!
7. Stellen Sie für ***CockPit*** die Namen des Unterverzeichnisses und der Datenbank als Vorgaben ein, indem Sie in die CP.OPT die Zeilen

   d C:\ALLEGRO*sub*

   b *dbn*

   einsetzen (→ Kap.0.11.6). Mit **µr z** werden diese Einstellungen im ***CockPit*** aktiviert.
8. Starten Sie das Schnellzugriff-Programm wie unter 1.1.1 beschrieben (mit Ihren individuellen Einstellungen statt der Standardangaben) oder geben Sie auf DOS-Ebene den Befehl
   presto -a3 -i*dbn* **-n1 -k***c* **-d** *sub* **-S**
   (die Angabe **-a3** bedeutet: volle Zugriffsberechtigung; **-S** steht für "Einzelplatz"; → Kap. 12)

   Mit diesem Befehl (den ***CockPit*** automatisch geben würde) teilt man dem Programm PRESTO alles mit, was es zum Anlegen der gewünschten Datenbank braucht. Diese entsteht auf dem Verzeichnis *sub*, und die eingegebenen Daten befinden sich anschließend in der Datei *dbn*_1.*c*LD .
   Zur Erinnerung: ***allegro***-Dateien werden mit dem Namen der Datenbank und einer angehängten Zahl zwischen 1 und 255 bezeichnet, dazwischen steht ein Unterstrich '_', der Dateityp ist .*c*LD. Also wäre XY_32.MLD eine ***allegro***-Datenbankdatei zum Schema M.CFG und der Index-Parameterdatei XY.API. Mit der Option **-ni** beim Start von PRESTO gibt man an, in welche Datei die neuen Datensätze sollen.

1.2 Datenbank-Generierung

So nennt sich der Vorgang, **aus bereits vorhandenen Daten** eine Datenbank zu erstellen, im Gegensatz zum völligen Neubeginn (→ 1.1). Sie erfahren in diesem Abschnitt alles, was man grundsätzlich zu dem Thema wissen sollte.

Für diese Aufgabe gibt es die Programme INDEX, UPD und IMPORT (→ Kap.7, 9, 5).
Die folgenden vier Situationen kann es geben:

a) **Grunddateien** (→ 0.3), die mit dem Erfassungs- und Redigierprogramm erstellt oder mit dem Import-Programm (→ Kap.5 und 11) aus Fremddaten gewonnen wurden, soll das Programm INDEX in eine Datenbank umwandeln **oder in eine schon vorhandene einspeisen** (→ Kap.7). Notwendig ist dazu eine Index-Parameterdatei (Typ .cPI). Entweder man benutzt eine der mitgelieferten (s.o.) oder fertigt eine eigene an. Die Ausgangsdateien müssen die Typkennung .cLG haben, also z.B. .ALG beim Standardschema.
Wenn Ihre Daten noch nicht in diesem Format vorliegen: siehe d).
Mit ***a99*** kann man Grunddateien und sogar die einfacheren Externdateien (Typ .cDT, → Kap.0.2) direkt einlesen und die Datensätze vor dem Abspeichern noch betrachten und bearbeiten (Menü "Datei | Weitere Offline-Datei", Datei auswählen, dann den Button [Daten in Bearb.]).

b) **Existierende Datenbank-Dateien** (Typ .cLD), sollen neu indexiert werden. Diese Aufgabe fällt an, wenn man Änderungen an der Indexdefinition (an der .cPI-Datei) vorgenommen hat. Das System merkt nämlich keinesfalls selbst, daß nun der Index anders aussehen muß, und es führt diese Änderungen nicht automatisch aus! Der Anwender muß selbst das Programm INDEX veranlassen, den Index neu aufzubauen. Dazu nutzt man die Funktion **`-fi`** des Programms, wobei die anderen Teile der Datenbank (.cLD-Dateien und die .TBL-Datei) unverändert bleiben. Am bequemsten geht es, wenn man im ***CockPit*** die Routine "organisieren / Index erneuern" startet. (**`µr o i`**)

c) **Neue Daten** sollen in eine vorhandene Datenbank eingespeist werden, und zwar nicht wie unter a) einfach hinzugefügt, sondern bereits vorhandene Sätze sollen durch die neuen ersetzt werden. Diese Aufgabe wird "Update" genannt und im Kap. 9 behandelt. Es gibt dafür das Programm UPD, mit dem man Daten auf unterschiedliche Art einmischen und auch einen Dublettencheck ausführen kann. Diese Aufgabe kann auch der ***avanti***-Server übernehmen, seit es ihn gibt.

   Mit ***a99*** kann man Grunddateien und Externdateien (Typ .cDT, → Kap.0.2) mit der FLEX-Methode einlesen und einmischen lassen. Dazu gibt es den FLEX-Befehl "update", der dieselbe Arbeit wie UPD leisten kann,

d) **Fremddaten** (aus anderen Quellen stammend oder eigene Daten aus einem anderen System) sind zu einer ***allegro***-Datenbank umzuwandeln oder in eine solche einzuspeisen. Dazu muß erst einmal das Programm IMPORT die Fremddaten in die richtige Form bringen, d.h. eine Grunddatei daraus machen. Die Struktur solcher Dateien wird in Kap.0.2.2 beschrieben. Dann liegt Situation a) bzw. c) vor, d.h. die Programme INDEX bzw. UPD sind für das Einspeisen zuständig. (Der Name IMPORT ist etwas irreführend: dieses Programm kann nichts in eine Datenbank hineinschreiben, sondern nur Fremddaten in eine Grunddatei umwandeln. → Kap.5 und 11).

Kapitel 5 beschreibt den Vorgang der Umwandlung von Fremddaten, Kapitel 11 die dazu nötige "Importsprache", mit der man präzise die Umwandlung programmieren kann und muß. Diese Aufgabe ist nichts für Einsteiger, denn es gibt dabei keine Menüführung.

Kapitel 7 beschreibt den genauen Ablauf des INDEX-Programms und seine unterschiedlichen Möglichkeiten, und in welchen Situationen man es auf welche Weise einsetzt.

Kapitel 9 führt in die Möglichkeiten des Updating ein: wie bringt man Daten aus anderen Quellen in eine Datenbank ein? Ausnahmsweise kann UPD auch aus Grunddaten eine Datenbank neu aufbauen.

Kapitel 10 erläutert die Details der Generierung von Zugriffsschlüsseln, also die Gestaltung der Index-Parameterdatei. Dies ist eine Teilaufgabe des Gesamtkomplexes "Export" und wird deshalb im Zusammenhang mit der Exportsprache behandelt. Dieses Kapitel muß man intensiv studieren, wenn man eine eigene Datenbankstruktur neu entwickeln will (insbes. → 10.2.1.3, 10.2.6.3).

1.3 Benutzung einer Datenbank (DOS)

Nehmen wir als Beispiel an, Sie haben Ihre Datenbank angelegt auf einem Datenverzeichnis namens `C:\ALLEGRO\KATALOG`, angehängt also an das Programmverzeichnis. Für die Windows-Programme (→ Kap.2) wird eine `.ini`-Datei gebraucht, um eine Datenbank benutzen zu können. Hier beschreiben wir, wie man mit dem DOS-Programm PRESTO einsteigt.

1.3.1 Einstieg per *CockPit* oder von Hand

Am schnellsten kommen Sie hinein, wenn Sie im ***CockPit*** die "Vorgaben" auf Ihre Datenbank einstellen (siehe oben Kap. 0.11.6). Dann brauchen Sie nur nach Erscheinen des ***CockPit***-Hauptmenüs noch einmal <Enter> zu drücken - schon kommt die Registeranzeige. Was Sie dann alles tun können, beschreiben die Abschnitte 1.4 und 1.5.
Wenn die Vorgaben nicht richtig eingestellt sind: Im Menü "Optionen" (<Alt>+o) Ihr Datenverzeichnis (KATALOG) auswählen, dann zurück zum Menü "Routinen" mit <Alt>+r

- Sie sind nur an der Datenbankbenutzung interessiert? Gehen Sie gleich zu Abschnitt 1.4 und 1.5 über.
- Wer tiefer in die Materie einsteigen will, lese auch noch den Rest dieses Abschnitts, denn für eventuelle Anpassungen ist zum genaueren Verständnis der Vorgänge einiges zu sagen:

1. Die ***allegro***-Programme können von jeder Platte und von jedem Verzeichnis aus gestartet werden. Zweckmäßig ist, sich zuerst auf das Verzeichnis einzuwählen, auf dem die Daten liegen. Das kann ein Unterverzeichnis von C:\ALLEGRO sein, es kann aber auch jedes andere Verzeichnis auf jeder anderen Platte sein. Günstig ist dann, wenn in der PATH-Variable von MS-DOS der Name des Programmverzeichnisses, meist C:\ALLEGRO, registriert ist (→ 0.10 Installation).

2. Wenn Sie ***CockPit*** nicht benutzen, und wenn Sie nicht A.CFG als Konfiguration verwenden, sondern z.B. P.CFG, dann geben Sie zuerst unter MS-DOS den Befehl
`set -k=p`
Alle Programme wissen dann anschließend Bescheid. Oder Sie starten jedes einzelne Programm mit der Option **`-kp`**.

3. Sie können PRESTO auch "von Hand" starten: Sie geben den Befehl

`presto -a3 -S -n1 -d`*datenpfad*

wobei *datenpfad* die korrekte Angabe des Verzeichnisses ist, auf dem sich Ihre Datenbank befindet.
Wenn der Start vom Programmverzeichnis C:\ALLEGRO aus erfolgt und die Datenbank auf C:\ALLEGRO\KATALOG liegt, kann man `-dkatalog` angeben.
Für DOS-Kenner: Wenn C:\ALLEGRO in der MS-DOS-Variablen PATH enthalten ist, kann man auch auf dem Datenverzeichnis starten. Dann genügt `presto -a3 -S .`

- wenn es auf dem Datenbankpfad bereits mehrere Datenbanken gibt, erhalten Sie eine Auswahlliste, die wie üblich (→ 0.5) behandelt wird. Wenn es nur eine gibt, nimmt das Programm diese. (Wenn es keine gibt: Fehlermeldung "keine Datenbank auf Verzeichnis ...".)
- `Dateinummer für neu erfaßte Daten?` **`1`** (oder andere Zahl von 1 bis 255)
  Wenn Sie während der nachfolgenden Sitzung neue Datensätze erfassen, müssen diese in einer ***allegro***-Datei gespeichert werden (Typ .cLD). Eine ***allegro***-Datenbank kann bis zu 255 solche Dateien enthalten, der Index umfaßt aber alle gemeinsam. Sie können bei jeder Sitzung eine neue beginnen oder immer dieselbe verwenden.

  Die Numerierung muß nicht aufsteigend sein und nicht mit 1 beginnen. Bestimmte Nummern können Sie z.B. für bestimmte Zwecke reservieren. Während der Erfassung kann man die hier gewählte Nummer auch wieder ändern, und zwar über die Taste F auf dem Anzeigeschirm (→ 1.5).

4. Ein Index-Ausschnitt erscheint, und Sie können mit der Arbeit an Ihrer Datenbank beginnen. Alle Funktionen sind in 1.4 und 1.5 beschrieben.

5. Nach Ausstieg aus dem Index- oder Anzeigeschirm mit F8 erscheint wieder das ***CockPit***, wenn man dieses zum Starten benutzt hatte.

Zwei Datenbanken gleichzeitig benutzen

Dazu gibt man beim Aufruf von PRESTO zweimal die Option **-d***dbp**dbn* an, wobei *dbp* der Pfad der jeweiligen Datenbank ist und *dbn* ihr Name. Zum Beispiel:

```
presto -a30 -n1 -dkatalog\cat -df:\allegro\swd\swd
```

Damit wird die Datenbank CAT auf dem Unterverzeichnis KATALOG als erste, die Datenbank SWD auf dem Verzeichnis F:\ALLEGRO\SWD als zweite angesprochen. Mit <Alt>+a schaltet man während des Betriebs zwischen beiden um. Auf der ersten hat man alle Berechtigungen, auf der zweiten nur Leseberechtigung (das wird mit `-a30` gesteuert).
Wenn auf dem angewählten Verzeichnis mehr als eine Datenbank ist, werden diese zur Auswahl angeboten.
Genauso aufzurufen ist das sog. OPAC-Programm **APAC**. Die Unterschiede sind:

- es gibt keinen Schreibzugriff, d.h. es Eingaben und Korrekturen sind unmöglich. Wenn Sie den Aufruf von APAC in eine Stapeldatei z.B. mit dem Namen KATALOG.BAT oder in ein beliebiges Menüsystem einbauen, wird der Zugang für einen Endbenutzer sehr einfach.
- Die Funktionstasten sind anders belegt: F1 bis F9 schalten die Register 1 bis 9 ein. Bringen Sie eine Beschriftung über den Funktionstasten an, z.B. "Namen" über F1, "Titel" über F4 etc., dann sinkt der Erklärungsbedarf ganz drastisch. Die dann noch nötigen Erklärungen liefert das Programm selbst.
- Es gibt zwei Menüs: Auf dem Registerbildschirm oder dem Anzeigebildschirm kann man <Esc> drücken, dann erscheint das zugehörige Menü an der rechten Seite. Es ermöglicht die Wahl aller Funktionen und Register.
  Die Menütexte stehen in der Datei UIFAGER (bzw. auf Englisch in UIFAENG). In dieser Datei nimmt man Anpassungen vor, wenn die eigene Datenbank z.B. andere Register hat. Menüpunkte, die man nicht anbieten möchte (z.B. nicht besetzte Register), läßt man in der UIFA einfach fort.
- Mit Option `-M` erscheint nach dem Start als erstes das Menü. Sonst erst bei Druck auf <Esc>.
- Export ist nur möglich, wenn beim APAC-Aufruf eine Option `-e` gegeben wird oder `-a1`.
- Wenn die Option `-a00` gegeben wird, ist <Alt>+a nicht möglich, d.h. auf die zweite, beim Start angegebene Datenbank, kann nicht eingesehen werden. Sinnvoll ist das, wenn aus der zweiten Datenbank nur Sätze nachgeladen werden sollen (für die Bildschirmanzeige).

Genau wie PRESTO ruft man auch die Programme ALFA und MENUED auf (nur für DOS verfügbar). Diese haben zusätzliche Funktionen, die man mit der TAB-Taste aktiviert:

ALFA Wenn auf einer Titelanzeige TAB gedrückt wird, kann man das angezeigte Buch ausleihen. Wenn es mehrere Signaturkategorien enthält (für jedes Exemplar eine), erscheint ein Auswahlmenü, damit man wählen kann, welches Exemplar ausgeliehen werden soll. Mit <Alt>+F10 schaltet man auf Rückgabe um bzw. von Rückgabe wieder zurück auf Ausleihe.
Taste '#' aktiviert ein Erfassungsmenü für "Notaufnahmen" von unkatalogisierten Büchern

MENUED hier erhält man mit TAB ein Menü mit Bearbeitungsfunktionen, die sich selbst erklären.

Unter Windows braucht man diese Programme nicht mehr. Ihre Funktionen lassen sich durch die FLEX-Technik von ***a99*** und ***alcarta*** realisieren und dann viel besser an lokale Wünsche anpassen und ausbauen.

1.3.2 Aufruf des Programms in Stapeldateien

Das Programm PRESTO für den Schnellzugriff kann innerhalb von Stapeldateien mit allen notwendigen Optionen aufgerufen werden (→ Kap.12). Dadurch wird ein vollautomatischer Start von PRESTO ermöglicht. Der Anwender kommt dann mit einem einzigen Befehl bis zum Registerbildschirm (→ 1.4).
Beispiel: Sie haben Daten und Index in einem Unterverzeichnis KATALOG des ***allegro***-Programmverzeichnisses. Auf dem Unterverzeichnis H:\FREMD liegt die Datenbank DAT. Wenn Sie dann folgenden Befehl geben (oder diesen in eine Batch-Datei einbauen):

```
presto -f1 -d katalog\cat -d h:\fremd\dat -n1 -a31 [-kc] [-S]
```

dann brauchen keine weiteren Fragen beantwortet zu werden. Datenbank CAT erscheint zuerst, mit <Alt>+a schaltet man auf DAT um. Hier geben wir nur ganz knapp die wichtigsten Optionen an, für die anderen konsultieren Sie Kap.12:

-a*b* Angabe der **Berechtigungsstufe** ("access level"), dabei bedeutet:
b= 0 : keine Schreibberechtigung, nur Lesezugriff
1 : Schreibzugriff nur auf die mit Option -n angegebene Dateinummer
2 : voller Schreibzugriff auf die gesamte Datenbank
3 : zusätzliche Berechtigung zum globalen Ändern (Befehl '_') und für Indexmanipulationen
31 : Berechtigung 3 für die erste, Berechtigung 1 für die zweite Datenbank (s. **-d** unten).

-e P/F Vorwahl von **Export-Parametern** und **Ausgabedatei** für Export (mit F4)
P : Name einer Parameterdatei (z.B. I-1 oder PRINT)
F : Name einer Ausgabedatei, in welche die Ergebnismenge hineingeschrieben werden soll.

-n m **Neue Sätze** werden in die Datei m geschrieben (bei Erfassung mit den Funktionen **I** und **C**). Wenn die Datenbank z.B. BUCH heißt, ist das also die Datei BUCH_m.ALD. Existiert diese noch nicht, wird sie neu angelegt. (Vgl. Frage 1.3.2)

-s X **Index-Einstieg**: Ohne diese Option wird nach dem Start ein zufälliger Abschnitt des Index angezeigt. Mit -s"|3nordpol" kann man z.B. erreichen, daß zu Beginn der Abschnitt im Register 3 angezeigt wird, der mit "nordpol" beginnt.

-x n **Schwelle**: für den Index wird der Schwellenwert n vorgegeben, d.h. bei Einträgen mit mehr als n Datensätzen wird statt der korrekten Zählung ">n" angezeigt. Man erreicht dadurch eine Beschleunigung der Indexanzeige und weniger Plattenzugriffe bei großen Datenbanken, in denen es Tausende von Sätzen zu einem Indexeintrag geben kann.

-d *pfad\dbn*
Angabe der **zu benutzenden Datenbank**. Kann doppelt oder dreifach auftreten, dann ist Umschaltung mit <Alt>+a zu den anderen Datenbanken möglich. Voraussetzung: beide beruhen auf derselben Konfiguration. (Die Namen können verschieden sein, auf den Inhalt kommt es an!)
Soll in beiden geschrieben werden, müssen auch die Indexparameterdateien identisch sein, und man muß z.B. **-a31** setzen, sonst wird für die zweite Datenbank **-a0** angenommen.
Mit drei Datenbanken muß dreimal Option -d gegeben werden und es müssen drei Ziffern hinter Option -a stehen. Der Kopierbefehl 'C' auf dem Anzeigemenü kopiert immer in die erste Datenbank.

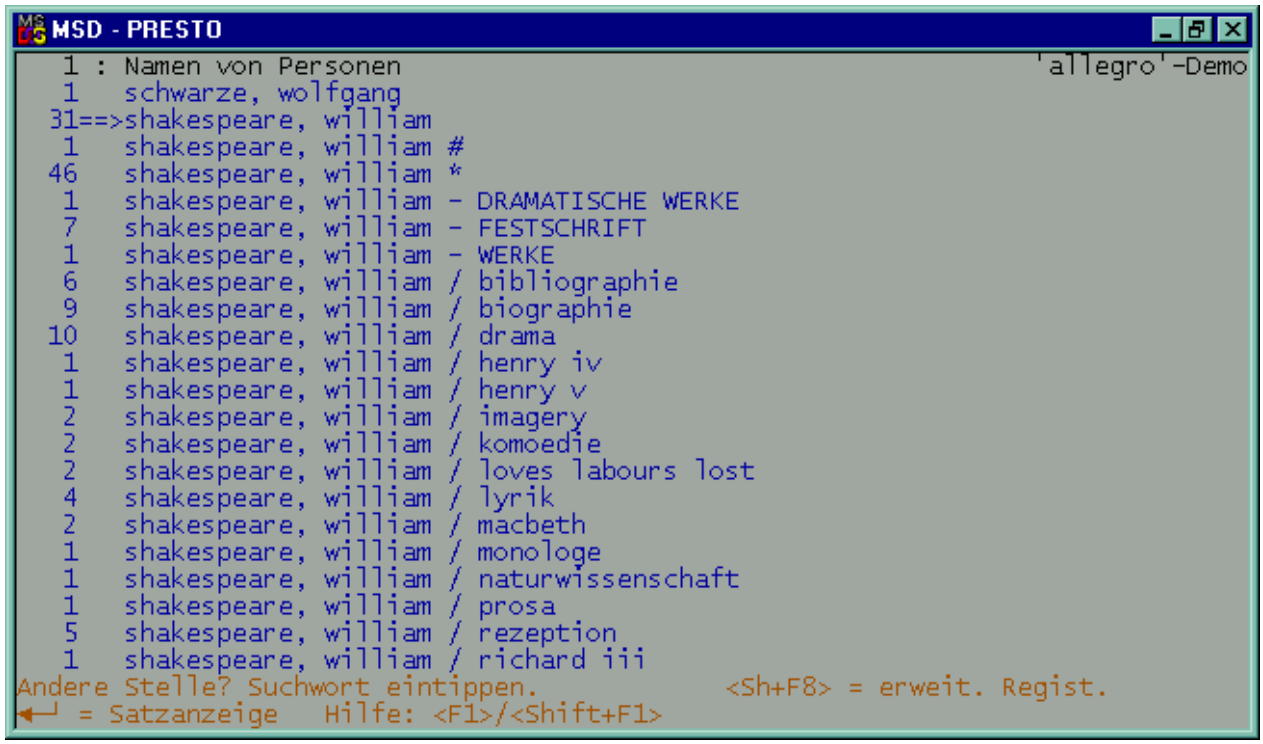
Ein Beispiel für den Aufruf von PRESTO sieht man in der Stapeldatei PR-LIST.BAT, mit der man eine Listenproduktion durchführt. Sehen Sie sich diese Datei an. Wenn Sie mit der MS-DOS Stapelprogrammierung vertraut sind, werden Sie leicht erkennen, wie Sie daran Veränderungen für Ihre Anwendung vornehmen können.

-S **Einzelplatzbetrieb** : notwendig, wenn das Programm nicht im Mehrplatzmodus arbeiten soll. Auf alten PCs kann es Probleme mit dem Speichern geben, wenn man ohne **-S** auf einem Einzelplatzgerät startet. Den korrekten Betrieb im Netz sollte der Netzverwalter im übrigen dadurch sicherstellen, daß er den Zugang nur über Stapeldateien oder Verknüpfungen mit den korrekten Aufrufen ermöglicht.
Ab V15 ist **-S** generell nicht mehr nötig.

1.4 Der Register-Bildschirm (DOS)

Sie sehen nach dem Einstieg in eine Datenbank im oberen Teil einen Ausschnitt des Registers 1, unten auf zwei Zeilen ein paar kurze Benutzungshinweise.
a99 und ***alcarta*** zeigen in ihrem Indexfenster dieselben Register und haben auch fast dieselben Funktionen (Kap.2).

Der Index einer CAT-Datenbank hat 11 Register. Dieses Beispiel zeigt einen Einblick in das Register 1:



Was alles in den Registern steht und wie es aussieht, das regelt die **Index-Parameterdatei**. In diesem Falle ist das die Datei CAT.API: sie definiert die 11 Register, wobei die Nummer 1 das Personenregister ist. Ganz anders arbeitet z.B. NMN.API: diese definiert ein "Kreuzregister", bei dem alle Suchbegriffe in einem Alphabet stehen. Man sieht also: Es können in einem Register unterschiedliche Arten von Begriffen (gemischt aus mehreren Kategorien), mehrteilige Eintragungen und auch Verweisungen vorkommen. Wünscht man Veränderungen an den Registern, sind Eingriffe in die Datei CAT.API bzw. NMN.API notwendig (Aufruf: ***CockPit*** **`µd i`**). Dazu muß man Kap. 10 studieren.

Es folgt die Beschreibung, was man mit dem Index alles machen kann. Die Windows-Programme benutzen denselben Index und haben fast dieselben Funktionen (s. Kap. 2).

Taste	Funktion

1.4.1 Browsing, Blätterfunktionen

↓ ↑ Der Zeiger → , der zunächst in der zweiten Zeile steht, wird mit den Cursortasten an den Eintragungen entlang nach unten oder oben bewegt. Sobald er auf einen für Sie relevanten Begriff zeigt, bringt Ihnen die Taste <↵ oder <Cursor links> den ersten, zu diesem Begriff gehörigen Datensatz auf den Anzeige-Bildschirm (→ 1.5). Von dort kehren Sie mit <↵ hierher zurück.
Wenn der Zeiger das untere oder obere Ende überschreitet, verschiebt sich alles in die betreffende Richtung - die Zählung wird jedoch beim "Scrollen" nach unten nicht aktualisiert! Dies ist deshalb so eingerichtet worden, weil das Weiterblättern dann aus internen Gründen viel schneller geht. Mit Druck auf '*' bekommt man die Zahlen nachgeliefert.

Taste	Funktion

<┘ (=<Enter>)
<–

ZUGRIFF: drücken Sie die Taste <Enter> oder <Cursor links> :
Es erscheint der erste zu dem betreffenden Suchbegriff vorhandene Datensatz, darunter zwei Hinweiszeilen (→ 1.5). Wenn es nicht die richtige ist: mit der Leertaste oder Taste <Cursor nach unten> wird der nächste Satz geholt, die zum selben Begriff gehört. Wenn zu diesem Begriff keine weiteren Datensätze existieren, erscheint automatisch der erste zum nächsten Begriff gehörende Satz, d.h. Sie blättern alphabetisch durch sämtliche Suchbegriffe vorwärts oder rückwärts. In der dritten Zeile von unten (→ **1.5)** erscheint jeweils der gerade aktuelle Begriff (der in dem angezeigten Datensatz an irgendeiner Stelle vorkommt).
Mit **<┘** geht es von da aus wieder zur Indexanzeige zurück.

→

Referenzierung: <Cursor rechts> wirkt normalerweise wie <Cursor links>, es sei denn, der Zeiger zeigt auf eine Verweisung der Form **`xxx -> yyy`** . Dann bewirkt die Taste <Cursor rechts> wie auch **<┘**, daß der Index an der Stelle yyy aufgeblättert wird, d.h. man springt im Register von der Verweisungsform zur Ansetzungsform. Wie '->' können auch andere Zeichen wirken (10.2.1.3). Wenn man also mehrteilige Schlüssel konstruiert, dann setze man eines dieser Zeichen zwischen die Bestandteile. Daraus ergibt sich die beschriebene "Referenzierung" im Register ganz automatisch. In der Beispieldatenbank finden Sie z.B. Namensreferenzen im Index 1 und Verweisungen von Begriffen im Register 3 auf Notationen im Register 7. In diesen Fällen beginnt der rechte Teil mit **`|7`**: das Programm blättert dann um ins Register 7.

Merkhilfe:

Die Wirkung der Cursortasten wird schnell plausibel: Ein Katalog ist eine Art Nachschlagewerk; und wenn man ein Nachschlagewerk als Buch mit Register vor sich liegen hat, klappt man es nach

- **rechts** um und blättert im **Register,** und nach
- **links,** wenn man eine Stelle im **Hauptteil** aufschlagen will.

Bild↑
Bild↓

Seitenweise blättern. Diese Tasten wirken genauso, wie Sie es erwarten:
es wird die nächste bzw. vorige "Seite" (= 22 Zeilen) des Index aufgeblättert. Achtung: das Rückwärtsblättern geht auffällig langsamer: das Programm wurde in Vorwärtsrichtung optimiert, da man dies erheblich häufiger braucht. Außerdem kann man durch Eingabe eines neuen Anfangspunktes (F6 oder unmittelbare Eingabe einer Zeichenfolge) sehr schnell auch ein größeres Stück zurückblättern.

<Alt>+Ziffer

Zwischen den einzelnen Registern einer Datenbank schaltet man mit Hilfe der **<Alt>**-Taste und der Registernummer um. So wechselt man z.B. bei einer CAT-Datenbank mit **<Alt>+3** in das Stich- und Schlagwortregister, mit **<Alt>+8** in das Signaturenregister etc. Mit **<Alt>+0** kommt Register 10 (nicht im OPAC), mit **<Alt>+ß** Register 11 (nur bei Berechtigung **`-a3`**). Das kann man zu jedem Zeitpunkt machen, und zwar auch auf dem Anzeigeschirm. Aufgeblättert wird in dem betreffenden Register jeweils dieselbe Alphabetstelle, an der man bei dem vorher benutzten Register gerade stand. Hat man also einen Namen im Namensregister aufgeblättert und geht in das Wortregister, so findet man dort sofort denselben Namen, falls er als Titelstichwort vorkommt.
Logische Kombinationen (→ 1.4.4) sind registerübergreifend!

<Alt>+a

Datenbank wechseln. Wenn beim Aufruf von PRESTO etc. zweimal die Option -d gegeben wurde, schaltet man mit <Alt>+a zur jeweils anderen Datenbank um. Das Register der zweiten Datenbank erscheint in anderer Farbe, so daß man optisch sofort orientiert ist. Weitere Einzelheiten → 1.3.2. Für APAC und ALFA gilt: mindestens `-a1`, sonst kein Umschalten.

Taste	Funktion
a..z **A..Z** **0..9**	**Neuen Suchbegriff eingeben:** Wenn man an der falschen Stelle des Registers ist, braucht man nichts anderes zu tun als den Anfang der gewünschten Stelle einzugeben, dann **<┘** . In der Mitte erscheint, **sobald man das erste Zeichen tippt** (also ohne einen Befehl!), ein Fenster mit der Frage "neuen Suchbegriff eingeben: " und dahinter das eingegebene Zeichen. Man tippt weiter, drückt <┘, und sofort erscheint der gewünschte Registerabschnitt. Hat man nicht exakt getroffen: Vorgang wiederholen oder mit <Bild↓> weiterblättern. ***Sonderfunktion:*** beendet man den Suchbegriff mit **'?'**, wird dadurch die **Trunkierung** an dieser Position eingeschaltet (→ 1.4.2).
F6	Die Taste F6 benutzt man nur (und gibt danach das Suchwort ein), wenn der Suchbegriff nicht mit Buchstabe oder Ziffer anfängt. Die Zugriffs-Schlüssel der gelöschten Sätze beginnen z.B. mit **"//"**. Wenn man die gelöschten, aber noch nicht wieder überschriebenen Sätze (→ 1.5, Funktion **Del**) durchblättern will, dann gibt man **F6 // <┘** : es erscheint die Liste dieser Schlüssel, geordnet nach der Länge der Sätze (hinter **"//"** 5stellig angegeben). Im Normalfall sind die Register in Kleinbuchstaben angelegt und mit aufgelösten Umlauten. Beachten Sie das bei der Abfrage. Allerdings kann der eingetippte Suchbegriff automatisch umcodiert werden: Dazu muß etwas in der Index-Parameterdatei getan werden (→ 10.2.1.3). So ist erreichbar, daß man zu "mueller" kommt, obwohl man "Müller" eintippte. Es ist also tatsächlich so, als ob man in einem gedruckten Register blättert, nur noch besser: lesen Sie unten die Erläuterungen zu "Endemarken", Trunkierung, logischen Kombinationen und Ergebnismengen.
Shift+F6	Wenn man eine Folge von sehr langen Einträgen (z.B. Körperschaftsnamen oder Serientiteln) vor sich hat, die sich alle nur am Ende unterscheiden, ist **Shift+F6** sehr bequem: man erhält im Eingabefenster in der Mitte die aktuelle Eintragung (auf der gerade der Pfeil steht) und kann daran etwas ändern, also z.B. die Bandnummer am Ende des Serientitels.
F2	**Index-Abschnitt drucken**. Der gerade angezeigte Ausschnitt des Registers wird gedruckt. Ansonsten verwendet man zum Ausdrucken der Indexdatei das Hilfsprogramm QRIX. Wenn man gerade eine Ergebnismenge als Kurzliste anzeigen läßt (mit Shift+F9), bekommt man diese Liste mit F2 ausgedruckt.
F5	Mit dieser Taste springt man **zurück zur letzten Titelanzeige** (→ 1.5). Das wird man nur tun müssen, wenn man versehentlich im Register gelandet war. Dann erscheint wieder der zuletzt angezeigte Datensatz, bevor man in den Index wechselte.
Strg+F10	**Schwelle** der Häufigkeitszählung ändern. Bei großen Datenbanken bringt es einen Geschwindigkeitsvorteil, eine "Schwelle" vorzugeben. Gibt man z.B. "100", so wird bei Einträgen mit mehr als 100 Datensätzen ">100" statt der tatsächlichen Zahl angezeigt. **Negative Schwelle:** Gibt man z.B. den Wert **-5** als Schwelle an, so werden nur Einträge mit mehr als 5 Datensätzen gezeigt. Nützlich ist das z.B. im ISBN-Register: man gibt -1 und erhält die doppelt vorkommenden ISBNs und kann kontrollieren, ob Dubletten vorliegen. (Tip: mit dem Programm QRIX kann man eine Liste derjenigen Registereinträge erstellen, deren Häufigkeit oberhalb einer beliebigen Schwelle liegt (→ Kap. 7.5).
%	Mit Schwellenwert 0 oder Druck auf '%' schaltet man die Zählung überhaupt ab, dann geht die Registeranzeige am schnellsten. Die Funktionen der Trunkierung und Ergebnismengenbildung bleiben von der Schwelleneinstellung unberührt. Mit der Option -x (→ Kap.12) kann man schon beim Start eine Schwelle vorgeben.
*****	Wenn die Zahlenanzeige fehlt (weil '%' betätigt wurde oder man zu früh eine andere Taste gedrückt hat, als die Zählungen noch nicht erschienen waren, kann man sie mit '*' nachliefern lassen. Bei großen Datenbanken und starker Trunkierung braucht die Ermittlung der Zahlen evtl. einige Sekunden. Sobald man eine Taste drückt, wird der Vorgang aber unterbrochen, also braucht man auf langsamen Rechnern nicht abzuwarten, bis alle Zahlen da sind.
Shift+F8	**Das erweiterte Register**. Wenn die Datenbank ein Kurztitelregister besitzt (→ Kap.10), kann man an jeder Stelle in jedem Register mit der Kombination **Shift+F8** dieses Register anzeigen lassen. Beginnend mit der Stelle, auf der gerade der Zeiger steht, listet dann das Programm unter den Registereinträgen die Kurztitelzeilen auf. Mit **Esc** geht es zurück zur normalen Registeranzeige. Im Abschnitt 1.4.5.2 wird diese Funktion mit einer Beispielseite noch ausführlicher dargestellt.

Taste	Funktion

1.4.2 Trunkierung, Endemarken

Verkürzte (= trunkierte) Registeranzeigen sind eine Eigenheit, die Sie vermutlich von keinem anderen System kennen.
Es gibt 2 Arten der trunkierten Anzeige - feste und variable Trunkierung:

F10 **Trunkierung** einschalten. Nach Druck auf F10 erscheint die Frage "Position?", die mit einer Zahl zwischen 2 und 72 zu beantworten ist oder mit einem beliebigen Zeichen.
Wenn es eine **Zahl** ist, wird die Registeranzeige auf entsprechend viele Zeichen gekürzt (trunkiert). Z.B. bekommt man mit dem Wert 4 alle Eintragungen, die mit "chem" beginnen, auf eine Zeile komprimiert und sieht links daneben auch sofort die Anzahl.
Dasselbe passiert, wenn man `chem?` statt `chem` als Suchbegriff eingibt.
Gibt man nach **F10** ein **Zeichen** statt einer Zahl ein, so wird dieses als **Endemarke** benutzt (siehe unten) und die variable Trunkierung durchgeführt. Speziell das **Leerzeichen** als Endemarke bewirkt, daß die Einträge auf das erste Wort verkürzt werden (wirksam nur bei Titel-, Namens- und Schlagwortregistern, wo Einträge aus mehreren Wörtern bestehen können).

< > **Trunkierung verschieben**. Die vorher mit F10 eingestellte Position wird um 1 nach links bzw. nach rechts verschoben. Die Anzeige verändert sich entsprechend. Bei variabler Trunkierung wirken diese Tasten nicht.

Die variable Trunkierung läßt sich auch ohne **F10** direkt einschalten, indem man das als Endemarke gewünschte Zeichen betätigt:

, . ; : **Endemarken** - sog. **variable Trunkierung** einstellen.
Shift+F10 Endemarke in der aktuellen Zeile automatisch bestimmen
Die Satzzeichen **`,;.:`** und das **Leerzeichen** können als sog. "Endemarke" (engl. "truncator") eingegeben werden. Das ist ein speziell für ***allegro*** neu erfundenes Konzept. Ein Effekt tritt nur ein, wenn eines dieser Zeichen in den Registereintragungen vorkommt. Die Wirkung ist folgende: Nach Eingabe von z.B. ',' wird die Registeranzeige am Komma abgeschnitten und die bis zum Komma identischen Einträge werden summiert. Bei Personennamen bedeutet das: sämtliche "Müllers" werden auf eine Zeile komprimiert. Bei Schlagwörtern könnte man z.B. den Bindestrich oder den Schrägstrich als Endemarke verwenden, wenn man Schlagwortketten mit solchen Trennzeichen hat. Bei Serientiteln kommt natürlich das ';' zum Einsatz, um Serienstücke auf eine Zeile zusammenzuziehen. Davor erscheint in dem Fall die Anzahl der vorhandenen Stücke. Ein weiteres Anwendungsbeispiel: In der Beispieldatenbank ist #30a für eine Klassifikation benutzt. Im Register 7 findet man die Notationen und mit Komma daran angeschlossen die Erscheinungsjahre. Die Folge ist, daß das Material unter einer Notation chronologisch geordnet ist. Wenn sodann **','** als Endemarke benutzt wird (Shift+F10 oder Komma drücken), werden alle zu einer Notation gehörigen Einträge zu einer Anzeigezeile zusammengezogen und man sieht davor die Gesamthäufigkeit.

F7 **Trunkierung aufheben = Expandieren**. Mit **F7** hebt man jede Trunkierung wieder auf - die Liste wird wieder "expandiert", und zwar an der Stelle, wo der Pfeil gerade steht. Bei Umschalten auf ein anderes Register wird jede Trunkierung aufgehoben.

F7 F7 Wenn ein Kurztitelregister vorhanden ist, kann die Expansion noch weiter gehen: das erste **F7** macht die Trunkierung rückgängig, das zweite schaltet das "erweiterte Register" ein. Es wirkt also genau wie **Shift+F8** (→ 1.4.5.2). Wenn gerade keine Trunkierung eingeschaltet ist, schaltet schon ein einzelnes **F7** das erweiterte Register ein. Welche Methode man benutzt, um von der Trunkierung zum erweiterten Register zu kommen, ist gleichgültig.

Taste	Funktion

TAB **Rechts-/Linksverschiebung** bei überlangen Registereinträgen

Shift+TAB Wenn man mit Indexlängen über 72 arbeitet (Parameter `il`, → 10.2.1.3), wird der Abschnitt oberhalb 72 Zeichen zunächst nicht sichtbar. Mit **TAB** verschiebt man die gesamte Anzeige um 40 Zeichen nach rechts, mit **Shift+TAB** zurück. Die kürzeren Einträge bleiben allerdings unverschoben stehen. Zunächst mag das irritieren, im praktischen Betrieb ist es aber kein Problem, da man den Blick auf die tatsächlich interessierenden Einträge richtet. Sobald der kritische Abschnitt sichtbar wird, ist man bereits zufrieden und nimmt die anderen Einträge kaum noch wahr. Beim Umschalten auf ein anderes Register wird die Verschiebung aufgehoben.

Die **aktuelle Registerzeile** wird **vollständig sichtbar** gemacht. Diese Funktion benutzt man, wenn es überlange Indexeinträge gibt, die in der Normaldarstellung auf 72 Zeichen gekürzt werden. Man erhält die Zeile, in der der Zeiger steht, vollständig in einem Fenster angezeigt.

<Esc> x **Phrasenspeicherung** (nicht im Programm APAC: dort löst <Esc> das Menü aus). Bei der Erfassung ist es eine willkommene Arbeitserleichterung, wenn man Namen, Serientitel etc., die im Register vorkommen, übernehmen kann. Man stellt den Pfeil auf die gewünschte Eintragung, drückt <Esc> und dann irgendeinen Buchstaben. Der Registerbegriff wird dann als Phrase (→ Kap.3.5.3, Befehl #p) zwischengespeichert. Eine Finesse ist hierbei diese: wenn man einen Großbuchstaben drückt, wird der Registerbegriff in Großschreibung umgewandelt, und zwar jeweils der erste Buchstabe jedes Wortes. Bei Namen ist das in den meisten Fällen die korrekte Form, die man wirklich benötigt. Bei Körperschaftsnamen oder Serientiteln wird man anschließend einzelne Zeichen wieder mit der Hand in Kleinbuchstaben umwandeln müssen. Wichtig ist, daß die Übernahme der vorhandenen Schreibweise automatisch die Konsistenz des Registers unterstützt. Somit kann das alphabetische Register auch gewisse Funktionen der sog. "authority control" (Normenkontrolle) übernehmen. Wichtig ist, daß man auch vom Editor aus mit **F6** in den Index springen und sich während der Erfassung von dort Daten holen kann! Mit **<┘** kommt man zurück in die Erfassung und kann die Phrasen verwenden.

Das Bequemste: Im Editor **F6** drücken, im Register den Zeiger auf eine Zeile setzen, die man übernehmen will, dann <┘ <Strg>+<┘ , und die Registerzeile wird kopiert.
Enthält die Registerzeile eine Stammsatznummer mit dem Zeichen '_' vorweg, wird nur diese Nummer dann kopiert, und das ist genau das, was man braucht.

1.4.3 Direkte Indexeingriffe

Entf **Registereintrag löschen**
Nur bei Zugriffsberechtigung 3 (→ Kap.12, Option -a) darf man diese Funktion ausüben. Sie schafft schnelle Abhilfe bei nutzlosen Eintragungen, jedoch nur temporär (s.u.).

Einfg **Register ergänzen**
Nur bei Zugriffsberechtigung >= 2 erlaubt. Das Programm fragt nach einer Eintragung und sortiert diese als Schlüssel ein. Dieser Schlüssel wird mit dem zuletzt angezeigten Datensatz verbunden. Insbesondere Nützlichkeitsverweisungen (mit **`->`** in der Mitte) können so ad hoc eingebracht werden, wodurch man die Referenzierung ausnutzt.

Aber: Die Funktionen **Entf** und **Einfg** haben keine Rückwirkung auf die Datensätze, sondern wirken sich nur im Index aus. Nach einem Neuaufbau des Index (→ Kap.7) müßte man die Manipulationen wiederholen. Nur Änderungen in den Datensätzen bewirken dauerhafte Indexänderungen. Es wird daher empfohlen, diese Funktionen nur ausnahmsweise zu benutzen, oder aber den Index nie zu erneuern (das ist unrealistisch). Nur die Funktion `-fc` des Hilfsprogramms QRIX, mit dem man einen Index kompaktieren kann (→ Kap.7; ***CockPit***: **`µr o c`**), ist in dieser Hinsicht unproblematisch: sie läßt die manuellen Einträge drin, denn sie greift nicht auf die Datensätze zu, sondern schreibt nur die Indexdatei um.

Vorschlag: Um dauerhafte Verweisungen im Index zu erzielen, die nicht an bestimmte Datensätze gebunden sind, lege man eigene Verweisungssätze an. Dazu wählt man eine sonst nicht benutzte Kategorie, etwa #9s, und gibt darin, durch Semikolon getrennt, exakt die Einträge ein, die in den Registern erscheinen sollen, jeweils mit der Registernummer **`|i`** vorangestellt. Diese Kategorie läßt man so indexieren, daß alle durch Semikolon getrennten Angaben unverändert eingetragen werden. Z.B.

```
#9s |5centralblatt -> zentralblatt
```

Taste Funktion

1.4.4 Logische Kombination, Ergebnismengen

Das DOS-Programm hat keinen Such- oder Findbefehl mit "booleschen" Operatoren (AND, OR, NOT), wie man es vielleicht von anderen Systemen kennt. (**a99** und **alcarta** haben solche Befehle: drücken Sie den Fernglas-Button.) Das Suchen geschieht vielmehr stets durch direkten Einblick in die Register. Der Vorteil: man sieht immer, was es wirklich gibt in der Datenbank. Das logische Kombinieren ist aber dennoch möglich, und sogar einfacher als mit Befehlen: man muß nur die richtigen Tasten drücken. Und zwar die folgenden:

/ + - **Logische Kombinationen**

Index-Eintragungen können mit den Booleschen Operatoren ODER UND und NICHT beliebig miteinander kombiniert werden (registerübergreifend). Dadurch bildet man eine **Ergebnismenge**: das Programm ermittelt die zu der jeweiligen Begriffskombination gehörige Teilmenge der Datenbank. Man setzt dazu lediglich den Pfeil auf eine zu kombinierende Eintragung und drückt eine der "logischen" Funktionstasten:

/ **ODER** die zum Registerbegriff gehörigen Datensätze werden zur Ergebnismenge hinzugefügt (Vereinigungsmenge). Wenn man '/' in der Kurzanzeige benutzt, kann man einzelne Titel in die Ergebnismenge übernehmen.

+ **UND** es wird die Schnittmenge aus den zum aktuellen Indexbegriff gehörigen Sätzen und der Ergebnismenge gebildet. Anders gesagt: die Einträge der aktuellen Zeile und der aktuellen Ergebnismenge werden durch logisches UND verbunden.

- **NICHT** = Gegenteil von UND: die Elemente der Schnittmenge werden aus der Ergebnismenge herausgenommen (Komplementärmenge). Wenn man '-' in der Kurzanzeige betätigt, kann man einzelne Titel aus der Ergebnismenge herausnehmen.

So entsteht jeweils eine neue Ergebnismenge (oder die leere Menge), die in weiteren Schritten mit neuen Begriffen (auch trunkierten Begriffen) beliebig kombiniert werden kann.

Tip: bei sehr großen Mengen zuerst die Registerstelle mit der kleineren Menge wählen.

<RÜCKTASTE>

("BACKSPACE") : Ergebnismenge "vergessen". Alle Kombinationen werden rückgängig gemacht; der Aufbau einer neuen Ergebnismenge kann beginnen.

Wenn diese Taste mehrfach gedrückt wird, werden der Reihe nach zuerst die Kurzanzeige abgeschaltet (s. 1.4.5), dann die Ergebnismenge vergessen, dann die Werte der Restriktion und der Trunkierung zurückgesetzt (falls welche eingestellt sind).

Wenn noch keine Ergebnismenge existiert, wirken alle drei logischen Funktionen gleich: die zum Registerbegriff gehörige Teilmenge wird zur Ergebnismenge gemacht.

Logische Kombinationen sind **registerübergreifend**, d.h. zwischen zwei Kombinationsbefehlen kann man mit **Alt+Ziffer** auf einen anderen Index umschalten, wenn man mehrere hat, und dort z.B. zunächst anders trunkieren, bevor man die Kombination auslöst.

Eine Ergebnismenge kann bis zu 16.000 (sechzehntausend) Eintragungen enthalten. Bei knappem Arbeitsspeicher kann es sein, daß man die maximal mögliche Größe reduzieren muß, damit PRESTO überhaupt läuft (→ Anh.A.1, Befehl `mr`).

Mit Ergebnismengen kann man drei Dinge machen: darin **blättern**, die Menge als ganzes **exportieren**, oder **globale Veränderungen** darin vornehmen. Das Exportieren und das Globale Ändern beschreibt ausführlich der Abschnitt 1.5.

Das Wichtigste ist das **Blättern**, das Besichtigen der Datensätze, die zur Ergebnismenge gehören. Dazu gibt es die **Kurzanzeige**. Der folgende Abschnitt beschreibt sie und zeigt ein Beispiel.

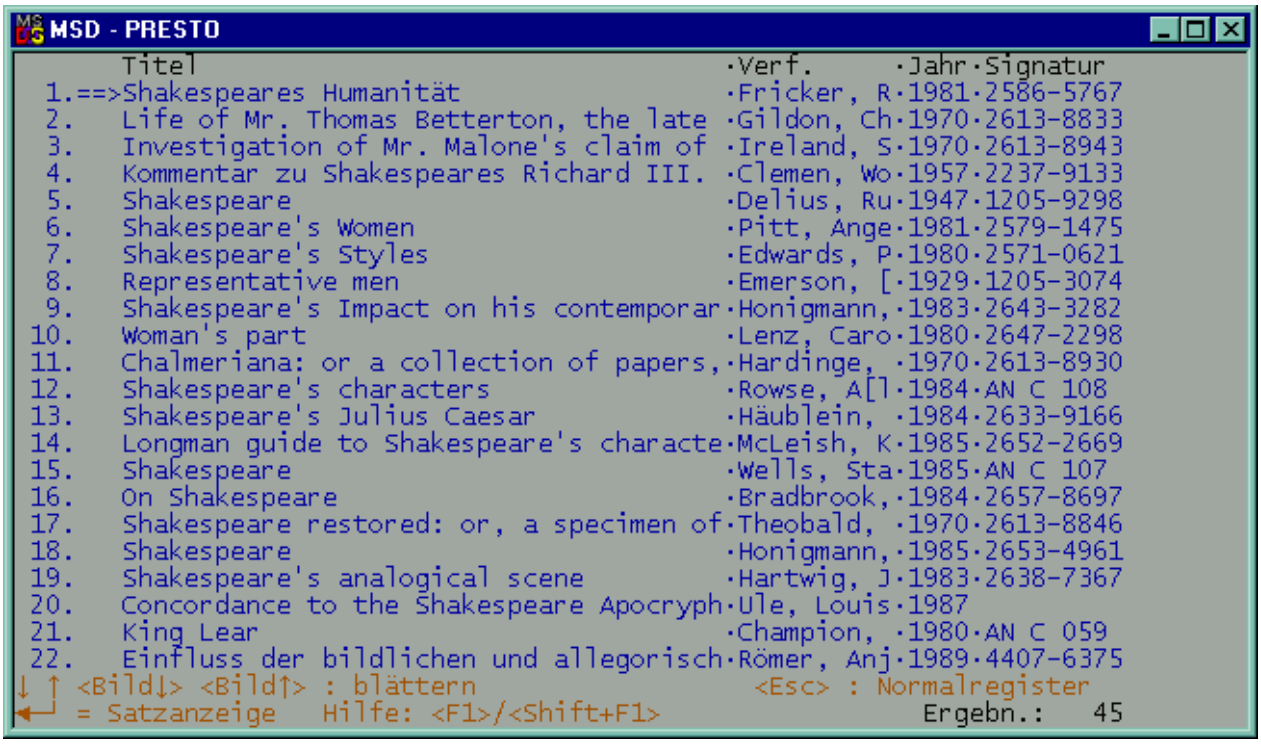
1.4.5 Kurztitelanzeigen

1.4.5.1 Die Kurzliste zur Ergebnismenge

Was eine "Ergebnismenge" ist und wie man eine anlegt, wurde unter 1.4.4 beschrieben.
Wenn man eine angelegt hat, sieht man unten rechts die Meldung `Ergebnisse: nnn`. Die Ergebnismenge besteht also aus nnn Datensätzen. Wie bekommt man die Titel zu sehen? Bis Version 12.1 ging das so: Man wechselt in die Titelanzeige und drückt auf den Buchstaben 'd'. Der erste Titel der Ergebnismenge erscheint, und zwar als vollständige Anzeige. Mit 'd' (für "down") blättert man weiter, mit 'u' (für "up") wieder zurück. Das ist umständlich, und wenn es sich um eine größere Menge handelt, hat man mit dieser Methode wenig Freude.
Wenn aber für die Datenbank ein Kurztitelregister eingerichtet ist, geht es viel bequemer und übersichtlicher:
Auf dem Registerbildschirm drückt man

Shift+F9

und eine numerierte Kurzliste der Ergebnisse kommt zum Vorschein. Mit den Pfeiltasten setzt man den Zeiger, wie man es gewohnt ist, auf eine interessierende Zeile und drückt <Enter>.



Von der Titelanzeige kann man nun mit den Tasten ↓ und ↑ die Ergebnisse durchblättern, solange die Kurzanzeige eingeschaltet ist. Unten rechts sieht man immer, der wievielte Titel aus der Ergebnismenge gerade in der Anzeige steht.

Das DOS-Programm hat einen **Schwachpunkt**: die numerierte Kurzliste ist unsortiert. Dies wird erst durch die neueren Programme überwunden: ***a99/alcarta*** unter Windows 95 etc. (und ***avanti*** auch unter UNIX) können Ergebnismengen bis zu 64000 Sätzen anlegen und bis zur Größe von 16.000 nach jedem Kriterium ordnen, das in der Kurzanzeige vorhanden ist, also z.B. nach Verfasser, Titel, Erscheinungsjahr oder Signatur (s.o.). Außerdem können ***a99/alcarta*** Ergebnismengen aufbewahren und miteinander verknüpfen, auch diese Funktionen fehlen in PRESTO/APAC. Das "konventionelle" Programm PRESTO arbeitet so, daß die Titel in der Reihenfolge ihrer internen (nicht sichtbaren) Nummern erscheinen. Dadurch finden sich die neueren Sachen im allgemeinen am Ende der Liste. Mit der Taste <Ende> kann man sich sofort zum Ende bewegen.
Mit **F2** kann man die gesamte Kurzliste auch ausdrucken, nachdem man sie eingeschaltet hat.
Mit <Esc> schaltet man die Kurzliste wieder ab. Die Ergebnismenge bleibt aber bestehen, kann vergrößert oder verkleinert und jederzeit mit <Shift+F9> wieder besichtigt werden, bis man sie mit ↵ beseitigt.

1.4.5.2 Das erweiterte Register

An jeder Stelle in jedem Register kann man sich eine Kurzübersicht der Titel zeigen lassen, die zum aktuellen Registerausschnitt gehören: die Tastenkombination

Shift+F8

erweitert das Register. Und zwar sieht man jetzt die Registerzeilen plötzlich in Rot, und darunter jeweils die zugehörigen Titel in Kurzform. Mit der Pfeiltaste ↓ setzt man den Zeiger auf einen gewünschten Titel und drückt <Enter>, dann kommt die vollständige Anzeige des betreffenden Buches.
Mit **Esc** schaltet man das erweiterte Register wieder ab.

Der Zeiger steht zuerst auf der ersten Titelzeile. Wenn man ihn mit <Cursor nach unten> bewegt, überspringt er die Registerzeilen, zeigt also immer nur auf die Titelzeilen. Auch hier ist das Blättern in Vorwärtsrichtung optimiert, d.h. es geht schnell, mit dem Cursor oder mit <Bild↓> weiter nach unten zu fahren. Nach oben geht es, abhängig von der Speichersituation, manchmal gar nicht, sondern es wird dann auf die normale Indexanzeige zurückgeschaltet. In der Praxis ist dies nach vorliegenden Erfahrungen eine geringe Beeinträchtigung.
Auch hier gibt es einen Schwachpunkt: das Blättern klappt nicht immer. Mit <Cursor nach unten> geht es unbegrenzt, mit <Cursor nach oben> kann die Anzeige plötzlich auf normal zurückspringen, bei <Bild↓> kann es Schwierigkeiten geben, und auf der Titelanzeige kommen mit den Cursortasten keine vernünftigen Ergebnisse heraus. Das alles fällt aber kaum ins Gewicht, denn:
Die primäre Funktion ist die "Kurzliste zur Ergebnismenge" (siehe vorige Seite), demgegenüber ist das "erweiterte Register" eher eine Zugabe. Der Wert liegt im sofortigen Aufblättern einer Kurzübersicht zu einer konkreten Registerstelle, ohne erst eine Ergebnismenge bilden zu müssen. (Nebenbei gefragt: welche andere Software kann das?)

1.4.6 Restriktionen benutzen

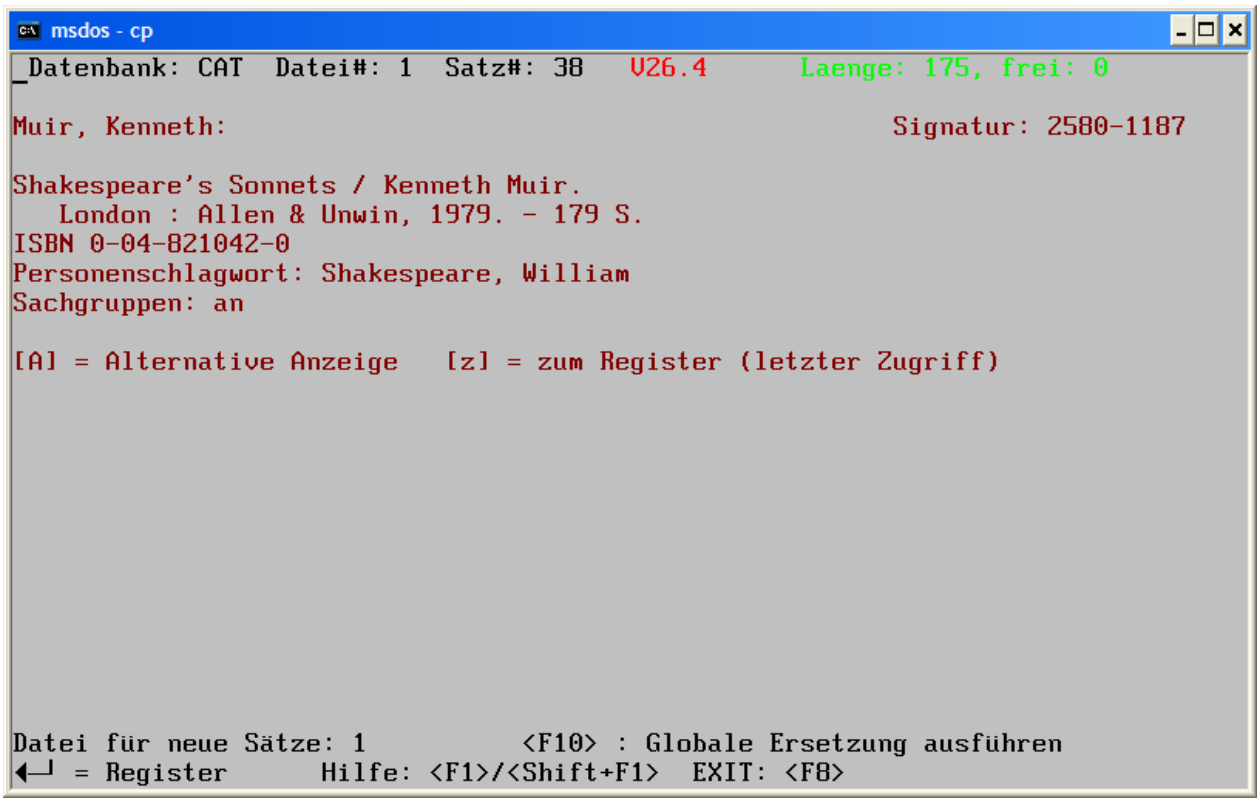
Ab Version 15 gibt es die Möglichkeit, bei der Suche Einschränkungen z.B. nach Erscheinungsjahr oder Sprache vorzunehmen. Dazu müssen Vorkehrungen in den Indexparametern getroffen sein (was in CAT.API ab V15 der Fall ist). Man nutzt die Restriktionen dann in PRESTO mit der Taste **F9**, in APAC über einen neuen Menüpunkt auf dem ESC-Menü. (Mehr dazu: → 10.2.9).
In ***a99*** und ***alcarta*** geht es bequemer über das Find-Menü (Kap.2).

1.5 Der Anzeige-Bildschirm (DOS)

Dieser Bildschirm erscheint, wenn man auf dem Registerbildschirm <↵ drückt. Sie sehen hier immer die Anzeige eines einzelnen Datensatzes. Dann können Sie eine ganze Reihe von Funktionen auf diesen Datensatz oder auch auf die vorher zusammengestellte Ergebnismenge (→ 1.4.4) anwenden. Auf der obersten Zeile teilt Ihnen das Programm den Namen der Datenbank, die Nummer der Datei und die des Satzes, seine Länge und die Anzahl freier Zeichen mit. Diese Angaben brauchen normalerweise nicht beachtet zu werden.
Dem Anzeigebildschirm entspricht im Windows-System das Anzeigefeld von ***a99/alcarta*** (s. Abb. Kap. 2)

1.5.1 Zugriff, Anzeige, Bearbeitung

Sobald man einen Suchbegriff (mit <↵ oder <Cursor links>) ausgewählt hat oder mit **F5** den Index verlassen hat, erscheint im oberen Teil des Bildschirms ein Datensatz (in Anzeigeform) und unten zwei Hinweiszeilen; mit **F1** können Sie Hilfe anfordern. Wenn man in dem Beispiel von 1.4 den Zeiger auf die Eintragung `shakespeare, william / lyrik` setzt und <↵ drückt, erhält man folgende Anzeige: (das ist der erste von drei Titeln)



Im oberen Teil steht also der gefundene Datensatz im Anzeigeformat. Wenn der Platz nicht reicht, blinkt darunter eine Meldung, die zur Betätigung der Cursortasten auffordert, um das Anzeigefenster nach unten oder oben zu verschieben bzw. mit 'x' diesen Verschiebevorgang zu verlassen und die anderen Funktionen benutzen zu können. Beim Verschieben bleibt die erste Zeile stehen. Daher ist es sinnvoll, in der Anzeige-Parameterdatei (Standard D-1.APR) die Signatur in die erste Zeile zu bringen, weil sie dann auch beim Blättern immer sichtbar bleibt.
Bei sehr langen Sätzen (mehr als 50 Zeilen; in der Regel hierarchische Sätze) teilt das Programm den Text in Anzeigeblöcke auf. Am Ende eines solchen Blocks muß man mit <Bild↓> die Anzeige des nächsten Blocks veranlassen - oder mit 'x' die Anzeige abbrechen. (Solche Probleme kennen die Windows-Programme nicht.)
Mit der Cursortaste ↓ blättert man den nächsten Satz auf, mit ↑ wieder zurück (→ 1.5.3).
Das Aussehen der Anzeige wird bestimmt von der Datei **D-1.APR** (bis V13 war es P1.APR). Die Anzeige ist unabhängig davon, über welches Register man zugreift. Wenn Sie also an der Anzeige etwas verändern wollen, ist ein Eingriff in die D-1.APR nötig. Veränderbar ist alles: die Reihenfolge der Elemente, welche Kategorien überhaupt angezeigt werden, die Interpunktion, sogar die Farben. Allerdings muß man dazu das Kap. 10 studieren.
Mit **F5** können Sie jederzeit auf die **Kategorieanzeige** umschalten - dann wird alles sichtbar, was eingegeben wurde.

Die Funktionen bilden mehrere Gruppen:

1.5.2 Zugriffsfunktionen (DOS)

<Enter> **Umschalten zum Index**

→ <Cursor rechts> wirkt wie <Enter>: das Register wird wieder "aufgeklappt", so daß man darin weitersuchen kann.

x **Expresszugriff** (ohne Umweg über den Index), zwei Möglichkeiten:

a) **mit Indexbegriff**

Wenn man sehr genau weiß, was man sucht, kommt man so am schnellsten zum Ziel: nach dem Druck auf **x** gibt man einen Suchbegriff ein (in der Form, wie er im Index stehen müßte) und drückt **<↵**. Es genügt der Anfang des Suchbegriffs. Dieser wird dann im aktuellen (momentan eingestellten) Register gesucht. Setzt man z.B. `|3` vor den Suchbegriff, wird auf Register 3 zugegriffen. Ein "Trunkierungszeichen" für den Abschluß unvollständig eingegebener Suchwörter gibt es nicht! Das Programm ermittelt den ersten Indexbegriff, der in alphabetischer Folge auf das eingegebene Suchwort paßt, und zeigt den ersten dazu vorhandene Satz. Eine Meldung wie "nichts gefunden" gibt es also auch nicht. Von hier aus kommt man mit ↓ zu den nachfolgenden Sätzen, wenn der erste nicht der gesuchte ist.

Wenn man ein '?' vor das Suchwort setzt, springt das Programm in die Indexanzeige (→ 1.4) und zeigt den Registerausschnitt, in dem das Suchwort stehen müßte.

b) **mit Satznummer**

Gibt man z.B. nach Druck auf 'x' **`#2712`** statt eines Suchbegriffs, so wird der Datensatz mit der Nummer 2712 angezeigt. Diesen sehr schnellen Zugriff kann man benutzen, wenn man eine Satznummer zufällig kennt. Mit '>' bzw. '<' kommt man zur nächsten bzw. vorangehenden Satznummer. Mit **<Pos1>** bzw. **<Ende>** kommt man zur Satznummer 1 bzw. zur letzten Nummer.

TAB mit der TAB-Taste gelangt man in eine **externe Funktion**. Unter den DOS-Programmen gibt es dafür 6 Beispiele: das Einfach-Ausleihprogramm ALFA, das Referentenprogramm REF für die Sachkatalogisierung, das Bestellprogramm ORDER, das Ausleihprogramm ***aLF***, das auch Funktionen zur Exemplarverwaltung enthält, und schließlich zwei erweiterte PRESTOs namens MENUED und INVENT. Es ist möglich, auf C-Ebene weitere derartige Funktionen zu programmieren, d.h. also ***allegro*** um weitere Funktionen zu ergänzen. Von der Titelanzeige aus kann man z.B. in ***aLF*** und ***ORDER*** dann in die Ausleih- bzw. Erwerbungsfunktionen einsteigen.

Hintergründe: Die programmiertechnischen Möglichkeiten sind in den C-Programmen vorhanden. Es gibt dafür eine eigene Dokumentation, betitelt "***allegro*** Programmers' Interface". Nur versierte C-ProgrammiererInnen können allerdings neue Funktionen auf diesem Wege programmieren. ***allegro*** kann somit als "Datenbank-Kernsystem" dienen, so daß der Programmierer ihm alles überlassen kann, was mit den Titeldaten zu tun hat und sich sich nur um die neuen Funktionen kümmern muß.

Die Windows-Programme lassen sich auf ganz andere Art erweitern: durch das FLEX-System. Damit läßt sich eine Oberfläche nach Art eines Web-Browsers schaffen, mit anwendungsspezifischen Funktionen zum Anklicken. Es existieren schon Beispiele für Ausleih- und Erwerbungsfunktionen.

( ) **Satznummernbereich als Ergebnismenge** definieren (→ 1.5.4, 1.4.4)

Man steuert die erste Nummer des gewünschten Bereichs an (mit <Pos1> die Nummer 1, mit <Ende> die letzte Nummer, mit `x #nnn <Enter>` jede andere Nummer. Dann betätigt man die Taste '**(**'. Danach steuert man die letzte Nummer des gewünschten Bereichs an und drückt '**)**'. Damit ist der Nummernbereich als Ergebnismenge festgelegt.

Tip: die Tastenfolge `<Pos1> '(' <Ende> ')'` macht die ganze Datenbank zur Ergebnismenge.

/ - **Einzeltitel zur Erg.menge ergänzen / herausnehmen** (logisches ODER / NICHT)

Der aktuelle Datensatz wird in die Ergebnismenge hineingenommen bzw. herausgenommen.

Mit diesen zwei Funktionen kann man gezielt einzelne Titel, die nicht bzw. ungewollt über die logischen Kombinationen automatisch in die Ergebnismenge gelangt sind, manuell hinzuzufügen bzw. herauszunehmen. Allerdings kann man immer auch einzelne Titel einzeln in eine Exportdatei mit **F4** "nachschieben" bzw. mit der F10-Funktion einzeln verändern. Das Herausnehmen wird daher die wichtigere Funktion sein.

1.5.3 Blätterfunktionen (Mit <Strg>+F1 kommt hierzu die Hilfeseite!) (DOS)

↓ ↑ **Blättern in Indexreihenfolge**

Leertaste Sobald ein Datensatz angezeigt wird, dient die Leertaste zum Weiterblättern in der Reihenfolge des Registers. <Cursor nach unten> ist gleichwertig, <Cursor nach oben> blättert zurück. Wenn zu einem Registereintrag mehrere Datensätze gehören, werden sie in der Reihenfolge der Satznummern hintereinander erscheinen. Wenn es sehr viele sind:

n / v **nächsten / vorangehenden** Schlüssel aufblättern:
Bei 'n' wird der erste Datensatz aufgeblättert, die zum nächsten echt verschiedenen Schlüssel gehört, bei 'v' entsprechend umgekehrt. In der Zeile über dem Menü wird immer der gerade für den aktuellen Zugriff benutzte Schlüssel angezeigt. Sie können also in Indexfolge blättern, ohne jeweils immer zur Indexanzeige zurückkehren zu müssen.

B / b **verknüpft gespeicherte Sätze durchblättern**
Dies ist gedacht für mehrbändige Werke, deren Einzelteile verknüpft gespeichert sind. Um die einzelnen Sätze bearbeiten zu können, blättert man sie mit **B** und **b** durch. Am besten vorher auf die Kategorieanzeige umschalten (mit **F5**).

Eine Ergebnismenge durchblättern

Zuerst stellt man mit den logischen Kombinationen eine "Ergebnismenge" zusammen (Beschreibung → 1.4.4). Dann möchte man diese Ergebnisse natürlich sehen. Das geht so:

Entweder: man drückt auf der Registeranzeige <Shift>+F9, dann kommt die Kurzliste, **oder:**

- man drückt <Enter>, um in die Anzeige zu kommen. Dabei wird, wie sonst auch, der erste Datensatz sichtbar, der zum aktuellen Schlüssel gehört.
- man betätigt 'd' - nun erscheint der erste Ergebnisdatensatz.

Stellen Sie sich die Ergebnismenge als Liste vor, die nach Satznummern geordnet ist. Dann wird es leicht verständlich, wie die folgenden Tastenfunktionen gemeint sind:

d / u **down / up** : in der Ergebnismenge vor-/zurückblättern
D / U **letztes / erstes** Mitglied der Ergebnismenge aufblättern
j / J **Sprung (jump)** 10% / 20% nach "unten"
r / R ... bzw. nach "oben"
s **show** : bestimmte Nummer innerhalb der Ergebnismenge ansteuern

Die Ergebnisse erscheinen immer in aufsteigender Folge der Satznummern. Da diese vom Programm fortlaufend vergeben werden, ist die Reihenfolge keine logische.

Viel besser ist es, wenn man ein **Kurztitelregister** hat. Dann ist es möglich, von der Registeranzeige aus zunächst eine Übersicht in Kurzform anzufordern. (→ 1.4.2, **Shift+F8** und **Shift+F9**)

Die Windows-Programme können dagegen Ergebnismengen sofort sortieren, und zwar nach mehreren Kriterien, z.B. Titel, Personenname, Erscheinungsjahr, Signatur.

Bild↑/Bild↓ **rück-/vorblättern in den bereits bearbeiteten Daten**
Wenn man mehrere Datensätze hintereinander gesucht und bearbeitet hatte, kann man mit diesen Tasten die Liste dieser Sätze rückwärts und vorwärts nochmals ablaufen lassen. Die Liste wird immer länger, maximal 64 Datensätze, danach werden die ältesten "vergessen" (natürlich nicht gelöscht).

L **Ergebnismenge bilden** aus den vorher bearbeiteten Datensätzen

H **Hierarchie als Ergebnismenge nehmen**. Alle mit dem aktuellen Satz verknüpften Sätze bilden werden als Ergebnismenge zusammengefaßt.

> < Hiermit kann man die Daten in **Satznummernfolge durchblättern**, d.h. in der Reihenfolge der Erfassung bzw. zum Zeitpunkt der Datenbankgenerierung. Sinnvoll ist es also nur, wenn diese Reihenfolge einen Sinn hat, wenn z.B. vor der Indexierung die Daten geordnet vorlagen.

<Pos1> **Satz Nummer 1** bzw.

<Ende> der **letzte Datensatz** wird angezeigt. (Mit **<Ende>** erfährt man also sofort den Umfang der Datenbank, weil dann oben die letzte Satznummer erscheint.)

<Alt>+a **Datenbank umschalten** → 1.4.1 Diese zwei Funktionen gehen auf dem Anzeigemenü

<Alt>+Ziffer **Register umschalten** → 1.4.1 genauso wie auf dem Indexmenü.

1.5.4 **Bearbeitungsfunktionen** (DOS)

Man löst sie durch **Großbuchstaben** aus - dies vermeidet den unbeabsichtigten Sprung in eine solche Funktion:

E **Editor** aufrufen für **Korrektur**. Der im oberen Teil angezeigte Datensatz wird zur Bearbeitung freigegeben. Es gelten dann dieselben Befehle und Eingabemöglichkeiten wie in allen Programmen, wenn man in den Editor springt (→ Kap.3). Die Meldung "Satz gesperrt" im Mehrplatzbetrieb bedeutet, daß gerade jemand anders an dem Satz arbeitet.
Mit **F10** oder Befehl #r beendet man die Bearbeitung, kann aber dann noch wählen, ob der bearbeitete Satz tatsächlich auch abgespeichert werden soll oder nicht. Neue und veränderte Suchbegriffe sind anschließend sofort in den Registern.
Mit **F8** verläßt man ohne Folgen den Editor, wenn man versehentlich darin gelandet war.

Wichtige Hinweise zum Editor für Einsteiger: (→ Kap.3.0-3)

a) Übergeben Sie jedes korrigierte Datenfeld jeweils mit <⏎ dem Programm, während der Cursor noch **innerhalb** des Feldtextes steht, erst dann wird die Änderung wirksam;
b) F1 und der Befehl #h geben Ihnen Hilfestellung.

I **Erfassung (Input)**. Sie werden aufgefordert, einen neuen Datensatz einzugeben. Einige Kategorien werden abgefragt, anschließend können Sie noch andere in beliebiger Reihenfolge eingeben, indem Sie die Feldnummern Ihres Kategorienschemas benutzen (→ Kap.3.2 **"Abfrageliste"** und Anh.B).
Die Abfrageliste sowie die erlaubten Kategorienummern stehen in der .CFG-Datei und können verändert werden (→ Anh.A). Wenn Sie die Erfassung als zunächst wichtigste Tätigkeit gründlich üben wollen, um dann richtig effizient arbeiten zu können: Nehmen Sie sich zunächst die Beispieldatenbank vor, bearbeiten Sie darin beliebige Titel und geben Sie dann auch eigene ein. Seien Sie dabei experimentierfreudig: die Beispieldaten dürfen beliebig verballhornt werden, das macht dem Programm und dem Computer nichts aus und hat keine Auswirkung auf Ihre eigenen Daten, aber Sie lernen dabei. Ganz detaillierte Erläuterungen zu den einzelnen Befehlen und Funktionen finden Sie in Kapitel 3, das Kategorienschema ist in Anhang B dargestellt; im Editor können Sie mit dem Befehl #h jederzeit eine Übersicht über das Kategoriesystem und die Bedeutung der einzelnen Feldnummern erhalten. Geben Sie z.B. #h2, um etwas über die Kategorien der Gruppe #2 zu erfahren. Ein Beispiel für die Erfassung eines **mehrbändigen Werkes** ist am Ende von Anhang B.2 zu finden.

C **Kopie (Copy)** des momentan angezeigten Satzes, um daraus durch Bearbeitung einen neuen zu machen. Zuerst kommt eine Frage nach der **Dateinummer**. Man gibt eine Zahl zwischen 1 und 255 oder <Enter>, dann wird die per Option `-n` beim Start gewählte Nummer genommen. Wie bei 'E' gelangt man in den Editor, hat dann aber der kopierte Satz zur Bearbeitung vor sich. Der Speicherbefehl (**F10** im Editor) speichert ihn dann genauso ab wie einen völlig neu erfaßten Satz. Ein Tip, wenn man diese Funktion aus Versehen betätigt hat: entweder mit F8 zum Menü zurück oder mit Befehl #V den kopierten Satz aus dem Arbeitsspeicher beseitigen und so weiterarbeiten, als ob man 'I' gegeben hätte. (Hinweis: #V löscht nichts in der Datenbank, nur im Arbeitsspeicher!)
Sonderfall: wenn man zwei oder drei Datenbanken parallel benutzt und aus der zweiten oder dritten einen Satz mit C kopiert, wird dieser anschließend in die erste gespeichert. Dies ist die bequemste Art der Fremddatenübernahme (→ 1.6). Eine andere Möglichkeit:

Shift+F5 **aktuellen Datensatz in den Hintergrundspeicher kopieren / "aufstapeln"**

Alt+F5 Dies dient als Eingabeerleichterung. Wenn anschließend mit 'I' in die Erfassung gegangen wird, kann mit dem Übernahmebefehl jede Kategorie aus dem Hintergrund in den neuen Satz übernommen werden: wenn z.B. `#40 Verfasser:` abgefragt wird, kann man mit Eingabe eines Punktes oder 't' den Kategorietext aus dem Hintergrund übernehmen (→ 3.3 Befehle #a und #t).
Mit **Alt+F5** wird der Satz auf andere dort schon vorhandene Daten aufgestapelt.

F **Datei wechseln**. Neu eingegebene Datensätze werden in die Datei (Typ .cLD) mit der beim Start angegebenen oder per Option `-n` gewählten Nummer gespeichert (normalerweise Nummer 1, → 1.3.2 und 1.6). Will man während der Erfassung auf eine andere Datei überwechseln, gebe man 'F' und danach die Nummer (zwischen 1 und 255) der gewünschten Datei. Wenn diese noch nicht existiert, wird sie angelegt. Wichtig ist diese Funktion z.B., wenn man Normdateien unter eigenen Nummern hat (→ 1.6). Die Umstellung kann mit dem Befehl F in die Abfrageliste eingebaut werden und dann automatisch und unbemerkt erfolgen (→ Anh.A.2)

Entf **Datensatz löschen**

Nach Bestätigung mit 'j' wird der aktuelle Satz als gelöscht gekennzeichnet, die zugehörigen Indexeintragungen werden beseitigt. Der Satz wird nicht sofort spurlos getilgt! Mit dem Suchschlüssel '**//**' findet man im Register 1 die gelöschten Sätze und kann sie durchblättern. Wenn der gesuchte erscheint: **E** und **F10** geben, dann ist alles rückgängig gemacht. Ansonsten werden die so gekennzeichneten Datensätze automatisch wiederverwendet, um neue Daten abzuspeichern: Wenn man in die Erfassung geht (**I** oder **C**) und dann **F10** gibt, sucht sich das Programm zum Abspeichern einen genügend langen Platz. Es kann sein, daß dann der gerade vorher gelöschte Satz endgültig dran glauben muß, d.h. überschrieben wird. Wenn keine gelöschten Sätze existieren, wird der Satz in die vorher vereinbarte Datei (→ 1.3.2) geschrieben, und zwar hinten angehängt. Er erhält die nächste Satznummer in fortlaufender Folge.

Wenn gerade eine **Ergebnismenge** existiert, fragt das Programm, ob es den aktuellen Einzelsatz löschen soll oder etwa die ganze Ergebnismenge. Voraussetzung: PRESTO war mit Option **`-a3`** gestartet worden.

F10 **Globale Änderung** (***a99***: Alt+g)

Wenn bestimmte Korrekturen oder Änderungen an mehreren Sätzen in gleicher Weise durchzuführen sind, kann dieser Befehl viel Zeit sparen. Vor seiner Anwendung muß man im Editor mit dem Befehl #X (→ 3.3) die nötigen Änderungen in Form von Ersetzungsbefehlen definieren. Bis zu 64 solche Befehle können theoretisch zugleich definiert sein. Wenn man danach auf dem Anzeigemenü **F10** gibt, werden die Ersetzungen auf den gerade angezeigten Datensatz angewendet. Wenn gerade eine Ergebnismenge existiert, wird mit der Meldung

```
wählen Sie:    1 = Einzeltitel   0 = ganze Ergebnismenge
```

zu einer Entscheidung aufgefordert. Bei Eingabe von '0' werden die Änderungsbefehle auf die gesamte Ergebnismenge angewendet, bei '1' nur auf den gerade angezeigten Satz. Wenn die Anzeige sich nicht verändert: Punkt drücken, dann wird die Anzeige erneuert und man sieht die Änderung.

Da bei einem Fehler in der Änderungsdefinition auch recht gravierende Auswirkungen denkbar sind, sollte man im Falle größerer Ergebnismengen zunächst immer einen Test mit einzelnen Sätzen machen und das Ergebnis kontrollieren.

Wegen seiner "Gefährlichkeit" gibt es für diesen Befehl eine besondere Berechtigungsstufe (siehe dazu → Kap.12, Option -a).

Mit 'x' läßt sich eine laufende Massenänderung unterbrechen und abbrechen.

Während des Ablaufs sehen Sie unten rechts einen Zähler, an dem der Fortgang der Aktion erkennbar ist. Im Mehrplatzbetrieb können Sätze übrigbleiben: wenn einer gerade gesperrt ist (weil jemand anders dran arbeitet) macht das Programm mit dem nächsten weiter. Wenn man am Ende noch Sätze findet, die nicht bearbeitet wurden, kann man sich diese ansehen oder dieselbe Aktion mit **F10** sofort nochmal starten, in der Hoffnung, daß die Sätze inzwischen wieder frei sind. Vermutlich bleiben aber höchstens solche Sätze übrig, die fälschlich gesperrt sind. Nach Freigabe mit <Strg>+z kann man auch diese noch bearbeiten.

Sie können die Ersetzungen auch **selektiv** vornehmen: mit 'd' die Ergebnismenge von oben nach unten durchblättern und bei jedem Titel, der behandelt werden soll, **`F10 1`** geben. Dann sehen Sie auch immer sofort, was herauskommt.

!!! Bei jeder Bearbeitung wird sofort der Index automatisch aktualisiert, falls durch die Änderungen irgendwelche Indexeinträge betroffen sind.

Strg+F10 **Globale Manipulation** [ab Version 14]

Dies ist potentiell die mächtigste Funktion. Sie setzt voraus, daß eine Export-Parameterdatei geladen ist. (Z.B. XYZ.APR mit Option `-exyz/nul` beim Start von PRESTO.) Darin muß eine Sprungmarke **`#-#`** vorkommen. Das Programm führt dann die Exportbefehle aus, die auf die Sprungmarke folgen, dann speichert es den Datensatz wieder ab. Innerhalb der Exportbefehle hat man die Möglichkeit, mit dem Manipulationsbefehl **`M`** Änderungen am Datensatz vorzunehmen (→ 10.2.6.3). Damit ist dies ein Verfahren, das noch weitaus mächtiger ist als die Globale Änderung.

1.5.5 Ausgabe- und Anzeigefunktionen (DOS)

F2 **Druckmenü** : Sie erhalten ein Menü mit mehreren Funktionen zum Ausdrucken des Datensatzes. Normalerweise werden Sie die Unterfunktionen 'a' (Produktion aller Ausgabesätze, die sich aus den Exportparametern ergeben, z.B. alle Karten mit den diversen Köpfen) oder '0' (z.B. Ausdruck einer Standardkarte ohne Kopf) wählen. Zur Ausgabe werden dieselben Parameter benutzt wie für die Bildschirm-Anzeige. Es sei denn, beim Aufruf von PRESTO wurde Option -q benutzt: dann werden die dadurch angeforderten Parameter wirksam. Das Druckmenü wird genauer in Kap.3.5.3 beschrieben, denn man kann es auch vom Editor aus aufrufen (→ 3.5, Befehl #d).
Wenn mehrere Export-Parametersätze geladen sind (per Optionen -q und/oder -e), kann durch Eingabe einer Ziffer einer dieser Sets angewählt werden. Bei **F3** wird dann dieser Parametersatz für die Bildschirmanzeige benutzt:

F3 **Kontrollanzeige** aller Ausgabesätze, normalerweise aller geköpften Karten gemäß P-KARTE.APR - es sei denn, man hat andere Parameter geladen oder per **F2** andere angewählt.

F4 **Export** des angezeigten Satzes bzw. der Ergebnismenge im gewählten Exportformat. (APAC: Alt+F4.) Wenn noch keins geladen ist, passiert dasselbe wie bei **Shift+F4**. Ein anderes Format ist beim Aufruf des Programms PRESTO mittels Option -e wählbar (→ Kap.12) oder mit **Shift+F4**. Der angezeigte Satz bzw. wahlweise die Ergebnismenge werden mit den betreffenden Ausgabeparametern in die gewünschte Datei geschrieben. Abbruch des laufenden Vorgangs mit 'x' ist möglich.

Sh+F4 **Laden** anderer Export-Parameter (per Auswahlliste) und/oder Ändern der Ausgabedatei (Name wird abgefragt). Während einer Sitzung können somit mehrere Exporte produziert werden.

F5 **Anzeigeform wechseln** : Umschaltung zwischen der formatierten und der kategorisierten Anzeigeform. Im OPAC-Programm: **Alt+F5**, aber nur wirksam für den aktuellen Satz.

Alt+g **Grafik aktivieren**. (Nur PRESTOG/APACG bis V15) Hiermit kann man beliebige andere Programme aktivieren, wobei der Aufruf mittels eines Sonderabschnitts in den Exportparametern programmiert werden muß. Damit können Grafikdateien, aber auch andere elektronische Ressourcen, an einen Katalog angebunden werden. Mit der FLEX-Technik in a99 sind die Möglichkeiten noch besser (`h flex`).

Es gibt außerdem ab V14b und erst recht in ***a99/alcarta*** programmierbare Anzeigefunktionen, die "Flips". Es können damit externe Programme gestartet werden. Dazu mehr in Kap.10.2.6.9.

1.5.6 Hilfsfunktionen und Sprache

<Esc> beim Programm APAC: Menü aufklappen, dort sind alle Funktionen per Leuchtbalken wählbar.

F1 **HILFE** : Diese Taste läßt die Hilfeseite H10 mit den näheren Angaben zum Menü erscheinen. Mit **Shift+F1, Alt+F1 und Strg+F1** erhält man weitere Hilfeseiten (→ 0.3).

F7 **Schlüssel anzeigen**: alle zum aktuellen Satz gehörigen Indexeinträge (Zugriffsschlüssel) werden ad hoc errechnet und auf dem Bildschirm angezeigt, einschl. der Kurzzeile (Register 0). Besonders nützlich zum Testen neuer oder geänderter Indexparameter und zur Kontrolle gleich nach einer Eingabe. Vor jedem Eintrag sieht man **`|i`** mit der Indexziffer i, von **`|0`** bis **`|:`** für Reg.10 und **`|;`** für Reg. 11.

Alt+F7 **Speicherzustand anzeigen** : wie beim Editorbefehl **`#s`** (→ Kap.3.5.3)

F8 **Ende (EXIT)**: Programm wird verlassen (**Alt+x** is gleichwertig)

F9 **Sprache wechseln** : die Frage "LANGUAGE CODE", die dann folgt, beantworten Sie mit "eng" (für Englisch), dann werden alle Menütexte, Fragen und Meldungen in Englisch ausgegeben. (Mit nochmaligem F9 und "ger" für Deutsch kommt die deutsche Anzeige zurück.) Die Dateien UIF.. enthalten die Menü- und Anzeigetexte. Eine anderssprachige Version kann jeder herstellen, der diese Texte übersetzt und die letzten 3 Buchstaben des Namens durch einen geeigneten neuen ersetzt. Wenn man z.B. POL für Polnisch wählt, müßte man UIF0POL und UIF1POL aus UIF0GER und UIF1GER mit Hilfe eines Texteditors erzeugen. Anschließend kann man als LANGUAGE CODE pol eingeben und hat die polnische "Oberfläche". Beim Start der Programme muß man die Option `-lPOL` vorgeben, um diese zu aktivieren (Empfehlung: Einbau in CP.BAT).

Alt+F10 Temporärer Ausstieg auf die DOS-Ebene. Dort steht dann allerdings u.U. nicht viel Arbeitsspeicher für andere Aktivitäten zur Verfügung. Eine evtl. offene Exportdatei wird vorher geschlossen und hinterher wieder geöffnet, daher kann man sie bearbeiten oder löschen! Mit **`EXIT`** geht's zurück.

1.5.7 Funktionstasten : Zusammenfassung (Details → 1.4 und 1.5) (DOS)

Die folgende Übersicht gilt für PRESTO. Im OPAC-Programm APAC ist die Belegung einfacher: die Funktionstasten führen von beiden Menüs aus direkt in die Register, ein paar andere Funktionen sind auf die Shift- und Alt-Ebenen gelegt, mit <Esc> erhält man ein Hilfsmenü (→ Menütexte in der Datei `UIFAGER`)

Auf dem **Registerbildschirm** (1.4) wirken die Funktionstasten so: (in Klammern: Unterschiede bei APAC)

F1	Hilfeseite H11 wird angezeigt: Erklärung der Funktionen
Strg+F1	Hilfeseite H12 aufblättern: Hinweise zum Trunkieren
Alt+F1	Hilfeseite H1I wird gezeigt: Erläuterungen zu den Funktionstasten
F2	Registerseite (die gerade angezeigte Seite) abdrucken
F5	Rücksprung zum letzten aktuellen Satz (Anzeigebildschirm)
F6	andere Stelle im Index aufblättern (APAC: **Shift+F6**)
Shift+F6	andere Stelle im Index aufblättern, aktuelle Zeile dazu verwenden und ändern
F7	Expandieren (Trunkierung aufheben) (APAC: **Shift+F7**)
F8	EXIT : Ausstieg aus dem Programm (APAC: **F10**)
Shift+F8	Erweitertes Register einschalten (wenn .STL-Datei existiert)
Shift+F9	Kurzliste der Ergebnismenge (wenn .STL-Datei existiert)
F10	Trunkierung einschalten (fest oder variabel mit "Endemarke") (APAC: **Shift+F10**)
Shift+F10	Endemarke aus aktueller Zeile automatisch entnehmen (APAC: --)
Strg+F10	Schwelle für die Häufigkeitszählung ändern
Alt+i	Umschalten auf Register i (i = 1...9 und 0 für Register 10) (APAC: statt dessen **Fi**)
Alt+a	Umschalten auf zweite oder dritte Datenbank (gilt auch für Anzeigeschirm)

Auf dem **Anzeigebildschirm** (1.5) ist die Wirkung sinngemäß ähnlich:

Alt+i	Umschalten auf Register i (i = 1...9 und 0 für Register 10) (APAC: statt dessen **Fi**)
F1	Hilfeseite H10 wird angezeigt: Erklärung der Funktionen
Alt+F1	Hilfeseite H1F wird gezeigt: Erläuterungen zu den Funktionstasten
Shift+F1	Hilfeseite Hdbn zur aktuellen Datenbank (zu NMN gehört z.B. HNMN) Sie können/müssen diese Hilfeseite selbst erstellen!
Strg+F1	Hilfe zu den Blätterfunktionen (Hilfeseite H1B)
F2	Druckmenü (→ Kap.3.5.3, Befehl #d)
F3	Anzeige in Kartenform
F4	Druck in Kartenform bzw. Export in Datei (bei Option -e)
Shift+F4	Änderung der Exportparameter und/oder Wechsel der Exportdatei
F5	Wechsel der Anzeigeform : Kategorieanzeige <-> Formatanzeige (APAC: **Alt+F5**)
Shift+F5	Aktuellen Datensatz in den Hintergrundspeicher kopieren
Alt+F5	Aktuellen Satz im Hintergrundspeicher "aufstapeln"
F6	Wechsel zum Indexmenü (letzte Indexanzeige wiederholen)
F7	Anzeige aller Indexeintragungen zum aktuellen Datensatz
Alt+F7	Anzeige des Speicherzustands (insbes. Freier Arbeitsspeicher!)
F8	Ausstieg (zurück zum ***CockPit***, wenn von dort gestartet) (APAC: **F10**)
F9	Sprache wechseln (GER = Deutsch, ENG = Englisch) (APAC: **Shift+F9**)
F10	Globale Ersetzung starten (vorher im Editor Befehl #X, → Kap.3.5.3)
Strg+F10	Globale Manipulation starten (Voraussetzung: spezielle Exportparameter)
Alt+F10	temporärer Ausstieg auf DOS-Ebene, Rücksprung mit EXIT
H	Erg.Menge aus den vorher bearbeiteten Sätzen bilden
L	Erg.Menge aus den verknüpften Sätzen bilden, die zum aktuellen Satz gehören

Anwendungsspezifisch können **Alt+g** und "Flips" dazukommen (→ 10.2.7).

Im **Editor** wirken einige Funktionstasten ebenfalls ähnlich (→ 3.4)

Die von ***allegro*** kaum benutzte Strg-Ebene der Funktionstasten kann vom Anwender belegt werden. Ohne weitere Hilfsmittel ermöglicht dies der PROMPT-Befehl von MS-DOS. Konsultieren Sie Ihr MS-DOS-Handbuch. Z.B. würde mit

```
PROMPT $e[0;95;"Erj";13p
```

die Taste Strg+F2 (= Code 0;95) so belegt, daß ein gelöschter Satz, der in der Anzeige erscheint, wieder gültig gemacht wird. (Sprung in den Editor und dann sofort der Befehl #rj zum Abspeichern.) Solche Befehle kann der MS-DOS-Versierte in eine Batchdatei einbauen, aus der die ***allegro***-Programme aufgerufen werden.

1.6 Hinweise zu Normdaten

Einsteiger können diesen Abschnitt zunächst übergehen. Lesen Sie ihn, wenn Sie an vertieften Möglichkeiten der Sacherschließung und an Normierung der Zugriffselemente interessiert sind.
Normdaten sind Hilfsdaten. Sie besteht aus **Normsätzen** (auch **Stammsätze** genannt), und jeder Normsatz enthält eine "verbindliche" Ansetzungsform mit Verweisungsformen und evtl. zusätzlichen Angaben. Anwendbar ist das Konzept auf

Namen Man legt für Personen und/oder Körperschaften je einen Datensatz an, der die für den Katalog verbindliche Namensform und eine oder mehrere Verweisungsformen enthält, bei Personen also z.B. abweichende Schreibweisen, Transliterationen, Pseudonyme etc.

Sachtitel Die sogenannten "Einheitstitel" sind normierte Titel für Werke, die in verschiedenen Ausgaben unter verschiedenen Bezeichnungen erschienen sind, insbes. in Übersetzungen. Es mag für manche Sammlungen erwünscht sein, auch hierfür eine Normdatei zu führen. Erhebliche Bedeutung hat dieses Konzept in der Musik. Dafür existiert eine Parametrierung namens ***bolero*** mit `bol.api` als Indexparametern

Serien Ein Serien-Normsatz enthält den Ansetzungstitel, Verweisungstitel und bibliographische Angaben. Wenn man Zeitschriftenaufsätze katatlogisiert, ist es günstig, Stammsätze für die ausgewerteten Zeitschriften anzulegen.

Schlagwörter oder **Thesaurus** Ein Schlagwort-Normsatz enthält ein Hauptschlagwort und dazu Synonyme und evtl. auch übergeordnete Begriffe und weitere Hinweise.

Notationen Wenn zur Sacherschließung oder Buchaufstellung eine Klassifikation verwendet wird, kann man auch für die Systemstellen Normdatensätze anlegen. Ein Normdatensatz enthält dann eine Notation und die zugehörigen Sachbegriffe und Verweisungen. Es kann sich auch um Regionalschlüssel, Sprachschlüssel, historische Codes u.a. handeln (→ Anh.B.2.6).

allegro kann für diese unterschiedlichen Satztypen einen gemeinsamen Index anlegen. Voraussetzung ist, daß die Konfiguration für alle Elemente geeignete Kategorien vorsieht.
Es ist möglich, Stammsätze und Titelsätze alle in derselben Datenbankdatei zu speichern, d.h. man braucht keine extra Normdatei. *Oder* man reserviert bestimmte Dateinummern für die unterschiedlichen Satztypen, indexiert jedoch alles gemeinsam. *Oder* man legt für die Normdaten eine eigene Datenbank an und ermöglicht das Umschalten zwischen beiden mit **Alt+a** (→ 1.4). Einige Anwender verfahren so mit der GND (früher SWD) und kopieren sich bei Bedarf einen Schlagwortstammsatz in die eigene Datenbank.
Ein Kategoriesystem `x.CFG` (z.B. `$a.cfg, $p.cfg, n.cfg`) muß alle Kategorien enthalten, die für die verschiedenen Dateitypen benötigt werden. Die Index-Parameterdatei (*dbn*`.xpi`) muß die nötigen Anweisungen enthalten, damit alle relevanten Kategorien der erfaßten Daten in das Register eingespeist werden.
Anmerkung: Die Daten der Library of Congress im Format MARC21 haben unter den Normsätzen (engl. "authority records") auch die "series records". Das sind zum Teil Hauptsätze von mehrbändigen Werken. Deshalb findet man diese leider nicht in den MARC-Titeldaten, sondern dort sind nur Datensätze für die Stücktitel.
In der Standardkonfiguration `$a.cfg` (→ Anh.B) sind Kategorien enthalten (`#2n, #3n, #4n, #6n, #8n und #3t..`), die **alle** genannten Normdateien abdecken. In der Demo-Datenbank, die mit der Index-Parameterdatei `cat.api` arbeitet, sind auch Beispiele insbesondere für die Sacherschließungs-Normdaten enthalten. Ein Normsatz dient dazu, die Verweisungen in den Registern zu erstellen. Studieren Sie insbesondere den Index 7 mit seinen Abteilungen G, H und S (für geographische, historische und Sprachschlüssel) und den dazugehörigen Stammsätzen.
Seit Version 11.2 gibt es Möglichkeiten, Satzverknüpfungen anzulegen (→ Kap.10.2.6.3, Befehl **`|im`**). Für Normdaten bedeutet dies, daß man in Titelsätzen im Prinzip nur eine Identifikation der Normsätze einzutragen braucht. Das Problem aber ist: wie kann so etwas korrekt indexiert und angezeigt werden, so daß der OPAC-Benutzer davon gar nichts merkt? Denn der hat von Normdaten noch nie etwas gehört...
Erst ab Version 14 wurde das Arbeiten mit Normdaten richtig praktikabel, vorher war es mit sehr komplizierter Parametrierung verbunden. Für Indexierung, Anzeige und Druck können Titelsatz und Normsatz nun automatisch zusammengefügt werden (→ 10.2.6.8 "Stammsatz-Verknüpfungen"). Praktisch sieht das so aus, daß z.B. in einer Personen- oder Schlagwortkategorie kein Text steht, sondern eine Nummer mit vorgesetztem '_' als Steuerzeichen, also z.B. `#40 _123` statt `#40 Tucholsky, Kurt`. In der Titelanzeige und im Register erscheint dann trotzdem der Name, nicht die Nummer - das Programm führt solche Ersetzungen automatisch aus, auch bei der Indexierung.
`nmn.api` und `cat.api`, auch die neue `bank.npi`, bieten hinsichtlich der Normdateien dieselben Möglichkeiten. Gleichwohl besteht kein Zwang, das alles komplett zu nutzen und den dafür notwendigen Aufwand zu treiben, man kann es auch wahlweise und fallweise machen. Z.B. kann man auf die Namensstammsätze verzichten und Namensverweise bei Bedarf in den Titelsätzen mit unterbringen (Verweisungsformen in den Namenskategorien mit " = " anfügen, → Anh.B); ferner muß man nicht für jede Serie eine Hauptaufnahme machen, denn der Index bringt die Stücke ohnehin zusammen! Nützlich ist ein Stammsatz aber dann, wenn die Serie den Titel ändert oder Paralleltitel hat.

In den Jahren 2012-2013 wurde bei der DNB und den Verbünden eine große Umstellung der Normdaten durchgeführt. Die vorher existierenden getrennten Normdatenbestände PND, GKD und SWD wurden in einem einheitlich strukturierten Datenbestand namens GND (Gemeinsame Normdatei) zusammengeführt. Der Austausch geschieht im Format MARC21, intern verwenden die Pica-Verbünde und die DNB das Pica-Format. In http://portal.dnb.de kann man Normsätze gezielt finden, wenn man bei einer Suche zusätzlich `gnd` eingibt.

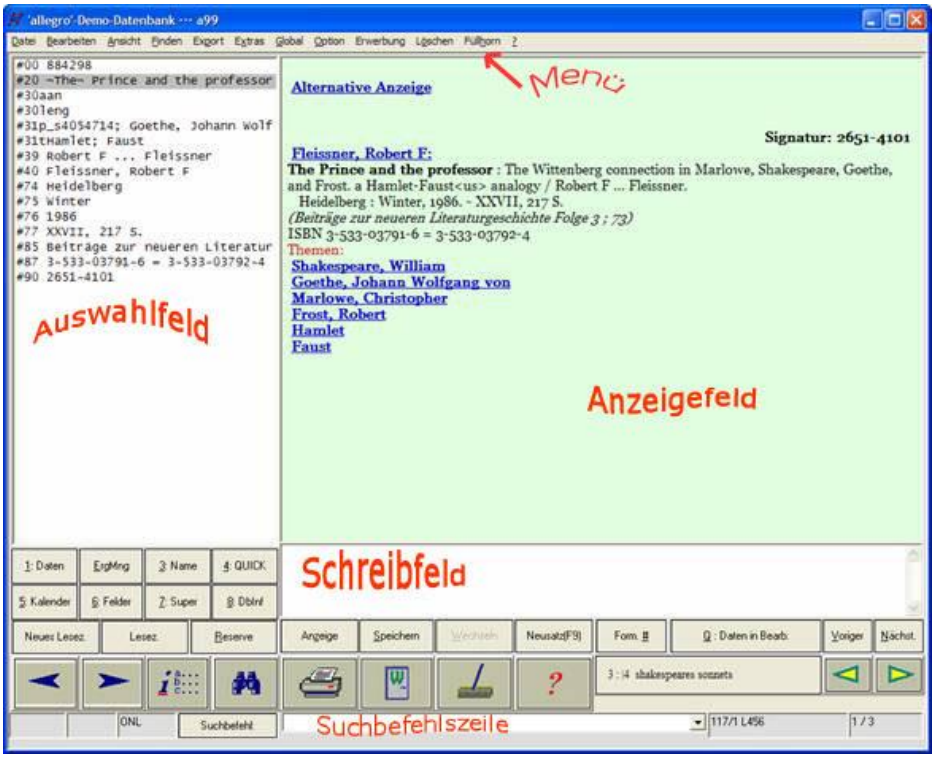
2 Hauptprogramm *a99* (Windows)

(Zugang zu allen Hilfetexten: In ***a99*** oben rechts im Menü das **?** klicken)
Dieses Kapitel beschreibt knapp die Arbeit mit den Windows-Programmen ***a99*** und ***alcarta***, die ab 1999 die DOS-Programme langsam ablösten. Letztere können aber noch immer zugleich auf dieselbe Datenbank zugreifen, ein Zwang zur Entscheidung für DOS oder Windows wurde vermieden!
Lauffähig sind die Programme ***a99*** und ***alcarta*** unter allen gängigen Windows-Versionen. *alcarta* funktioniert wie *a99*; ihm fehlen aber die Bearbeitungsfunktionen – es ist als OPAC-Programm im lokalen PC-Netz gedacht.

Wie starten?
Sie können *a99* direkt starten und dann Ihre Datenbank anwählen (Datei des Typs *.?DX, z.B. CAT.ADX). Dann sind aber nicht alle Möglichkeiten gegeben, die das Programm eigentlich bietet. Deshalb die
Empfehlung: Man erstellt eine **INI-Datei** für die eigene Datenbank. Als Vorlage nimmt man **`a99.ini`** (darin stehen Kommentare) und trägt die eigenen Vorgaben ein. Hat man daraus etwa **`katalog.ini`** gemacht, dann lautet der Startbefehl, den man mit einem Icon verknüpfen kann, z.B. so: **`c:\allegro\a99 d:\daten\opac\katalog`** . Damit können Sie dann Ihre eigene Datenbank genauso starten wie die automatisch installierte Demo-Datenbank.

2.1 Die Grundfunktionen auch: http://www.allegro-c.de/alca99/alca99.htm

Dieser Abschnitt behandelt diejenigen Elemente des Systems, die bei jeder Anwendung genutzt werden können und die man an der Oberfläche unmittelbar sehen kann. Abschnitt 2.2 behandelt "höhere Funktionen", die in der Makrosprache FLEX programmiert sind und anwendungsspezifisch modifiziert werden können.
Hier eine Momentaufnahme aus einer Arbeitssitzung an der Demo-Datenbank.



Der Datensatz, der ausgewählt wurde, der im Anzeigefeld zu sehen ist und mit dem man sich aktuell gerade beschäftigt, wird immer der "aktuelle Datensatz" genannt. Es gibt fünf Bereiche, in denen man etwas tun kann. Im Bild oben sind die Namen dieser Bereiche hinzugefügt (man sieht sonst diese Namen nicht).
Wenn Sie noch die nächste Seite genau lesen, haben Sie schon einen viel besseren Durchblick:

Anzeigefeld

Meistens steht hier ein einzelner **Datensatz**. Aber auch **Hilfetexte** werden hier eingeblendet. Texte bis zu 64K Größe kann das Anzeigefeld fassen. Mit "Cut-and-Paste" kann man daraus beliebige Teile in die Zwischenablage kopieren.
Für *Allegrologen:* Für die Erzeugung der Anzeige wird als Standard die Parameterdatei D-WRTF.APR benutzt.
TIP: Mit Alt+z oder F5 kann man umschalten zwischen der Normalanzeige und der kategorisierten (internen) Anzeige.
TIP: Im Anzeigefeld kann man auch schreiben. Was man dort schreibt, wird nicht in der Datenbank gespeichert! Und:
TIP: Der Button mit dem Drucker (unten neben [Find]) schickt das gesamte Anzeigefeld an den Drucker, mit allem, was gerade in dem Moment darin steht. Mit Alt+c kann man auch markierte Teile in die Windows-Zwischenablage übernehmen. Wer kein Textprogramm hat, kann sogar das Anzeigefeld zum Schreiben und Bearbeiten kleiner Texte benutzen. Auch

speichern kann man solche Texte, und zwar über das Menü "Datei", Unterpunkt "Anzeige speichern als..." Gespeichert wird dann im Format RTF, das man auch mit Word einlesen kann. Mit Dateityp .TXT kann man auch als ASCII-Datei speichern.

Auswahlfeld

Im Normalfall stehen hier zur Auswahl die Datenfelder (Kategorien) des aktuellen Satzes. Im Bild stehet der Balken gerade auf dem Feld #20. Mit <Enter> kopiert man es ins Schreibfeld zum Bearbeiten.
Mit der Taste <Entf> kann man einzelne Felder sofort löschen. (Drauf klicken, dann die Taste <Entf> und weg ist es.)
TIP: Im Reservespeicher findet man die Kategorien wieder, die man gerade mit <Entf> aus dem Datensatz gelöscht hatte.
So geht's: Alt+r (dann sieht man die gelöschten Kategorien), Kategorie anwählen, dann Alt+k zum Kopieren, und mit Alt+r zurück zur Anzeige des aktuellen Satzes; man sieht dann, daß die Kategorie wieder da ist.
Im Auswahlfeld können auch stehen:

- Die Liste der Ergebnismengen der Sitzung (Alt+e),
- die Kategorien des Reservespeichers (Alt+r), und
- das Datenschema (Liste der erlaubten Kategorien, Alt+y) sowie
- die Abfrageliste (nochmal Alt+y).

Schreibfeld (auch Eingabefeld)

Alt+m : Sonderzeichen-Eingabehilfe

Hier bearbeitet man Daten und gibt neue ein. Man setzt im Auswahlfeld den Balken auf die zu bearbeitende Kategorie und drückt <Enter> : der Inhalt samt Feldnummer erscheint dann im Schreibfeld.
Wichtig: Man muß <Enter> drücken, um die Eingabe dem Programm zu übergeben. Wenn man etwas geschrieben hat und dann z.B. mit der Maus woanders hinklickt, bleibt die Eingabe unwirksam.
TIP: Eine andere Arbeitsweise zum Eingeben und Bearbeiten ist möglich, wenn **Formulare** für die Datenbank eingerichtet wurden (siehe weiter unten). Dafür ist der Button [Form#] da.

TIP: Drei wichtige Sonderzeichen (→ Anh.E) sind leicht einzugeben: ▼ = AltGr+2, ¶ = Alt+**t**, ˇ = AltGr+3 (Haček),
mit **Alt+m** bekommt man eine Hilfeseite, auf der man alle Sonderzeichen auswählen kann.
Auch Phrasen gibt man hier ein: z.B. **`p s Shakespeare`** eingeben, dann hat man "Shakespeare" unter Phrase s gespeichert. Mit Eingabe von `\ s` kann man an jeder Stelle diese Phrase wieder abrufen. (Hilfetext: `h phrasen` eingeben)

Suchbefehlszeile (man könnte auch sagen "Zeile für Expertenmodus")

In diese Zeile gibt man direkte Suchbefehle ein (wenn man weiß, wie das geht; sonst Hilfe mit F1).
Im Beispiel oben ist gerade dieses eingegeben worden:

```
per beethoven? and tit sinfonie?
```

Ds sieht man unten auf der langen Schaltfläche, die Anzahl der gefundenen Sätze ist 80. Neben der Ergebnis-Schaltfläche sind zwei Pfeile: damit geht man zum nächsten oder vorigen Satz der Ergebnismenge, hier also zu Nummer 34 bzw. 36.
Suchmaschinenmodus: Wenn ein Wortregister mit dem Namen ALL eingerichtet ist, kann man hier einfach Stichwörter und Namen eingeben, wie man es bei Suchmaschinen gewohnt ist! (Beim Standardsystem ist dies der Fall.) [Ab V30.7]
*Geheim***TIP**: Wenn man F7 eingibt, sieht man in der Anzeige die zum aktuellen Satz gehörigen Registereinträge. (Wie bei PRESTO.) Das geht auch schon vor dem Abspeichern und auch bei ***alcarta.***

Der *Focus*

Der sog. Focus ist kein Symbol oder Objekt, sondern dasjenige Feld "besitzt den Focus", so sagt man, in dem sich gerade der Cursor befindet. Tasten (Zeicheneingaben und Cursorbewegungen) wirken sich nur in dem Feld aus, das gerade den Focus besitzt. Jederzeit kann man den Focus in ein anderes Feld setzen, indem man mit der Maus hineinklickt. Mit <Tab> kann man herumspringen zwischen Auswahlfeld, Schreibfeld und Befehlszeile.

Dringende Empfehlung

Der Menüpunkt **`?`** bringt ein großes Menü hervor mit einer Fülle von Komfort-Funktionen zum Anklicken. Damit wird das Leben des Anwenders stark erleichtert! (→ Kap. 2.2)

Paßwörter

Das Windows-System bietet auch die Möglichkeit, Paßwörter mit abgestuften Berechtigungen einzurichten.
Mehr dazu: geben Sie ein **`h npw`**

Noch Fragen offen

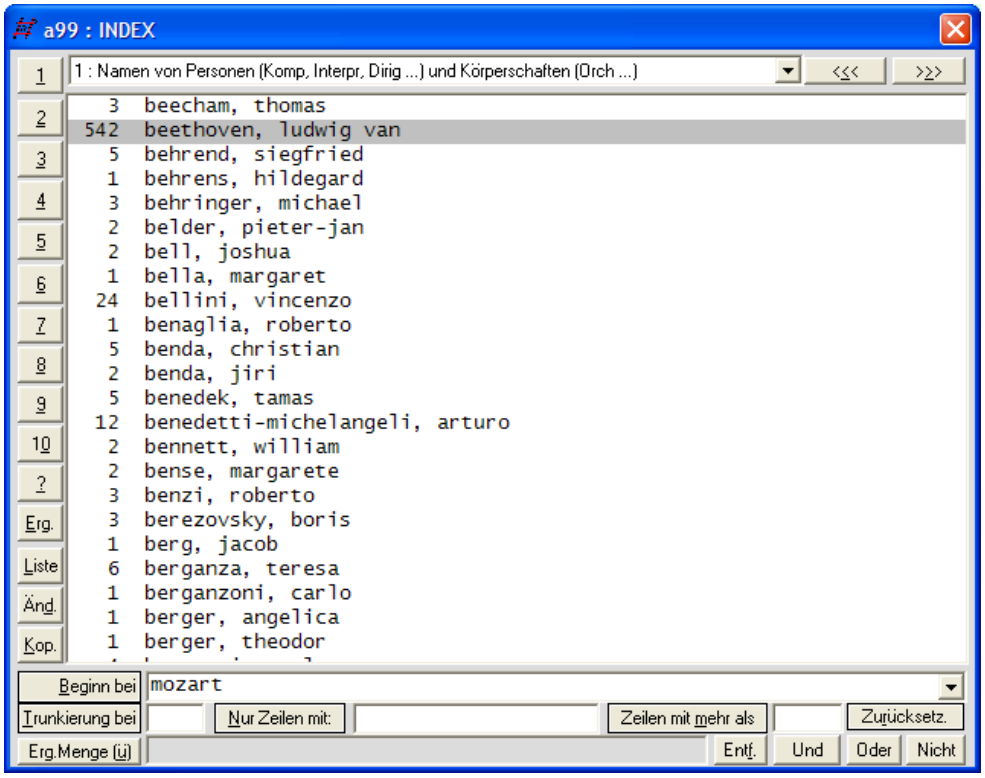
Dafür ist der rote Fragezeichen-Button da. Der bringt einen Hilfetext zur Anzeige, der speziell für die Datenbank gemacht wurde, mit der man gerade arbeitet. Der Systemverwalter kann diesen Text (eine RTF-Datei namens *Db*Name.rtf) frei gestalten, um die Arbeit mit dieser speziellen Datenbank besonders zu unterstützen. Vielleicht hat er einige hilfreiche Links darauf untergebracht, die z.B. automatisch Erfassungsformulare aufrufen oder zu besonderen Datensätzen, Ergebnismengen oder Registerstellen hinführen. Eine gut gemachte Hilfeseite kann den Lernbedarf drastisch reduzieren. Ein "Link" sieht so aus wie auf einer Web-Seite: blau und unterstrichen. Es können aber ganz andere Dinge dahinter stecken. Deshalb wird so etwas bei ***a99*** und ***alcarta*** nicht "Link" oder "Hyperlink" genannt, sondern "Flip". Ganze Befehlsfolgen, kleine Programme also, können sich hinter einem Flip verbergen. Solche Prográmmchen werden dann FLEX genannt, weil sie die Flexibilität vermehren. (s. Kap. 2.3)

2.1.1 Suchen und Blättern

Was man oben in der Abbildung sieht, ist längst nicht alles, was das Programm zu bieten hat. Es gibt vor allem noch drei weitere Fenster, die dem Nutzer beim Suchen und Auswählen helfen:

1. Das Indexfenster (ein- und ausschalten mit Alt+i oder mit dem Button [Index])

Das sieht z.B. so aus, wenn man gerade im Personenregister "Beethoven" in einer *bolero*-Daenbank aufgeblättert hat:



Achtung: Button [?] verrät noch mehr!

Im Eingabefeld steht "mozart", offenbar wurde das gerade eingetippt, um zu den Einträgen von Wolfgang Amadeus (vielleicht aber auch Leopold) zu gelangen. *Wichtig:* Sobald man einen Buchstaben tippt, erscheint derselbe gleich im Eingabefeld - man muß also nicht erst hineinklicken.

Mit den 10 numerierten Knöpfen an der linken Seite schaltet man zwischen den 10 Registern um, die es maximal gibt. Die Kopfzeile enthält eine Liste (auf das Dreieck klicken) der vorhandenen Register. Umgeschaltet wird immer auf dieselbe Alphabetstelle, was praktisch ist, wenn man z.B. zuerst das falsche Register gewählt hatte. Das Umschalten funktioniert auch mit Alt+Ziffer (wie beim alten PRESTO). *Wichtig:* Button [**Erg.**] aus der gewählten Zeile eine Ergebnismenge.

Bei Doppelklick auf eine Zeile (oder <Enter>) passiert folgendes:

a) Die Trefferzahl ist 1 : der zugehörige Satz wird in das Anzeigefeld geholt. Oder:

b) Es gibt mehr als 1 Treffer: man bekommt eine **Kurzliste** der zugehörigen Sätze (auch "Ergebnismenge" genannt; mehr dazu weiter unten), und der Balken geht eine Zeile tiefer.

TIP: Cut-and-Paste geht nicht in diesem Fenster, aber: Balken auf die erste Zeile setzen, die man kopieren will, ins Eingabefeld eine Zahl schreiben, z.B. 50, [List] drücken. Schon hat man die 50 nächsten Zeilen ab dem Balken im Anzeigefeld; dort kann man markieren und Cut-and-Paste nutzen (Strg+c / Strg+v), also den Registerausschnitt als Text übernehmen.

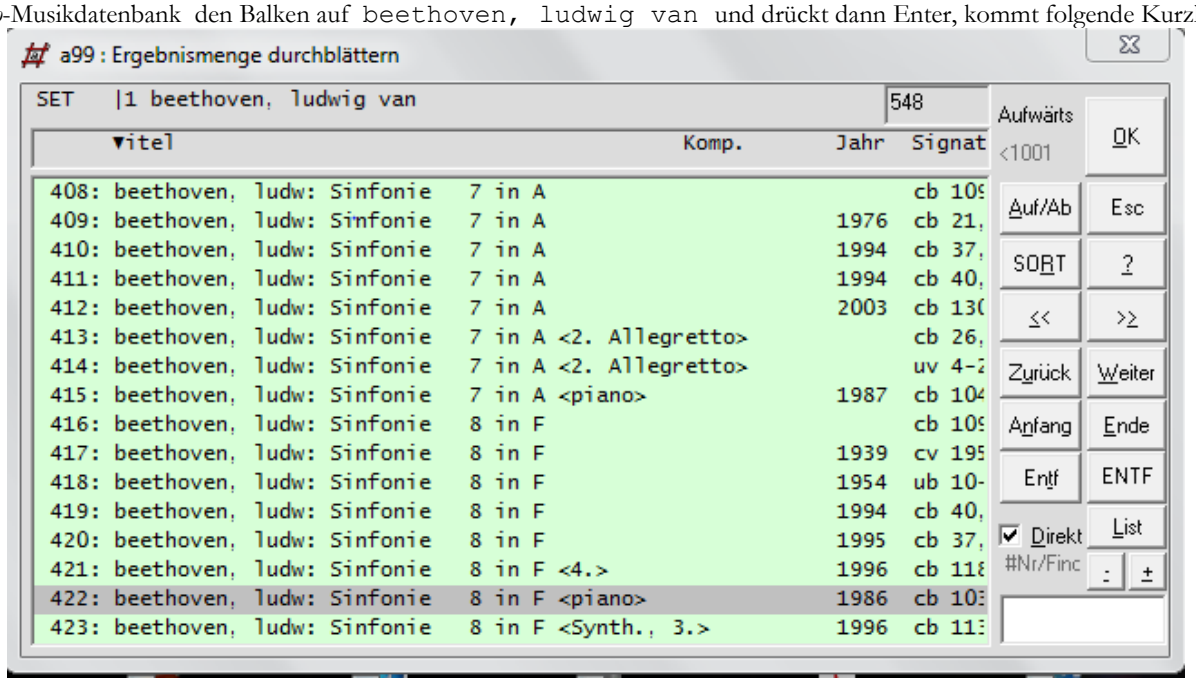
Weitere nützliche Funktionen: (mit **F1** kommt allgemeine Hilfe, mit [?] evtl. eine datenbankspezifische)

- Der Button [Kop.] kopiert die gerade gewählte Zeile in das Schreibfeld. Wenn aber von einem Formular aus in den Index gesprungen wurde, landet die Kopie in dem Formularfeld, von dem man ausging.
- Mit [Änd.] wird die angewählte Zeile nach unten ins Eingabefeld kopiert, wo man sie ändern kann, um dann an eine andere Stelle zu springen, wenn das Register sehr viele und lange Einträge hat, wie z.B. ein Titel- oder Serienregister.
- Ganz unten sieht man, welches Ergebnismenge gerade vorher gebildet wurde. Mit den Buttons [AND] [OR] und [NOT] kann man weitere Registerzeilen mit der Ergebnismenge verknüpfen lassen. Statt dessen tun es auch, wie beim alten PRESTO, die Tasten **+**, **/** und **–** . Es wird immer der Inhalt der Zeile mit dem Balken verknüpft mit der Ergebnismenge.
- Mit <Esc> oder Alt+i läßt man das Indexfenster verschwinden, mit Alt+i geht es sofort wieder auf, an derselben Stelle.
- [Liste] kopiert den Registerabschnitt, der mit der angewählten Zeile beginnt, in das Anzeigefeld (bis zu 32K Länge).
- Mit [Erg.Menge] oder Alt+ü sieht man die Kurzliste der momentanen Ergebnismenge, mit [Entf.] macht man sie leer.

Trunkierung: Drücken Sie mal im Register 1 (Personennamen) auf das Komma oder im Register 5 (Serien) auf das Semikolon: alle Einträge werden dann an dem Satzzeichen abgeschnitten. Eine andere Möglichkeit: Geben Sie im Register 1 "shakesp?" ein, also ein Fragezeichen am Ende, dann werden alle Zeilen auf diese Länge gekürzt. So erhält man einerseits ganz neue Überblicke über ein Register und kann andererseits sehr leicht Ergebnisse zusammenfassen. Trägt man etwas ein in das Feld "Nur Zeilen mit", werden nur Zeilen angezeigt, die diese Zeichenfolge enthalten. Das ist so etwas wie eine **Linkstrunkierung**.

Die Kurzliste

Das ist ein besonderes Fenster, in dem die Übersicht der aktuellen Ergebnismenge zu sehen ist. Setzt man z.B. Beispiel in der *bolero*-Musikdatenbank den Balken auf `beethoven, ludwig van` und drückt dann Enter, kommt folgende Kurzliste:



Diese Liste kann man sich an jeder sinnvollen Position sortieren lassen. Man verschiebt oben über den Titeln entlang das kleine Dreieck mit den Buttons [<<] und [>>] nach links oder rechts. , dann auf [SORT] drücken.

TIP: nur einmal auf [<<] oder [>>] klicken, dann die <Enter>-Taste niederhalten, das Dreieck verschiebt sich dann sehr schnell; mit Maus allein geht es langsam.

In diesem Fall ist nach Titel sortiert worden, dann wurde der Abschnitt der Sinfonien aufgeblättert. Mit [Auf/Ab] dreht man die Sortierrichtung um. Bei Jahreszahlen ist die umgekehrte Reihenfolge sinnvoll, dann hat man die neuesten Einträge oben.

Für Kenner: Mit Alt+s sortiert man nach der internen Satznummer.

Mit [F1] kommt eine Hilfeseite, die den Rest erklärt. Noch ein **TIP**: eine Zahl in das Feld "Nr." eintragen, dann [List]. Wie beim Indexfenster erhält man, schwupp, diese Anzahl Zeilen, ab der angewählten Zeile, in das Anzeigefenster. Gibt man aber eine Zeichenfolge dort ein und drückt <Enter>, wird im Text der Kurzliste gesucht! Mit [+] und [-] geht das dann vor und zurück so lang man will.

Das Find-Fenster (Hilfetext dazu kommt mit F1)

Wer noch nicht weiß, wie man Suchbefehle formulieren muß, kann sich das Find-Fenster aufklappen. Man drückt auf den [Fernglas]-Button (oder Alt+f) und sieht dieses: (mehr dazu auch in 10.2.6.9 "Expansion" und 10.2.9 "Einschränkungen")

(Auf so etwas mußten und müssen DOS-Anwender verzichten)

Der Nutzer hat in diesem Suchformular verlangt, im Personenregister nach "beethoven" zu suchen und zugleich nach "sinfonie" im Register der Titel- und Schlagwörter. Außerdem sollen nur Einträge von CD-Aufnahmen in die Ergebnismenge aufgenommen werden (also keine Kassetten oder anderen Medien).

Damit das Find-Fenster funktioniert, müssen in den **Indexparametern** der Datenbank die Namen der symbolischen Register eingetragen sein sowie die Bezeichnungen der Einschränkungen (sog. Restriktionen), falls solche eingerichtet sind.

Bei älteren Datenbanken ist das unter Umständen noch nicht der Fall, dann wird man keine Registernamen zu sehen bekommen, logisch. Ein Fall für den Systemverwalter (siehe Kap. 10.2.1, Befehlszeilen mit **`I`** bzw. **`R`** am Anfang).
TIP: Nach Benutzung des Formulars hat man den daraus entstandenen Befehl in der Befehlszeile stehen. Dort kann man Änderungen vornehmen und den Befehl erneut ausführen lassen.
TIP: Wenn es keine Registernamen gibt, kann man trotzdem Find-Befehle in die Befehlszeile eingeben: statt
`per beethoven? and tit sinfonie?` könnte man auch schreiben: `|1 beethoven? and |3 sinfonie?`
TIP: Hat man schon eine Ergebnismenge, kann man den "Volltext" aller Datensätze (nicht der Dokumente!) durchsuchen lassen, indem man in die Zeile ganz unten eine Zeichenfolge eingibt. Groß-/Kleinschreibung muß dabei beachtet werden.

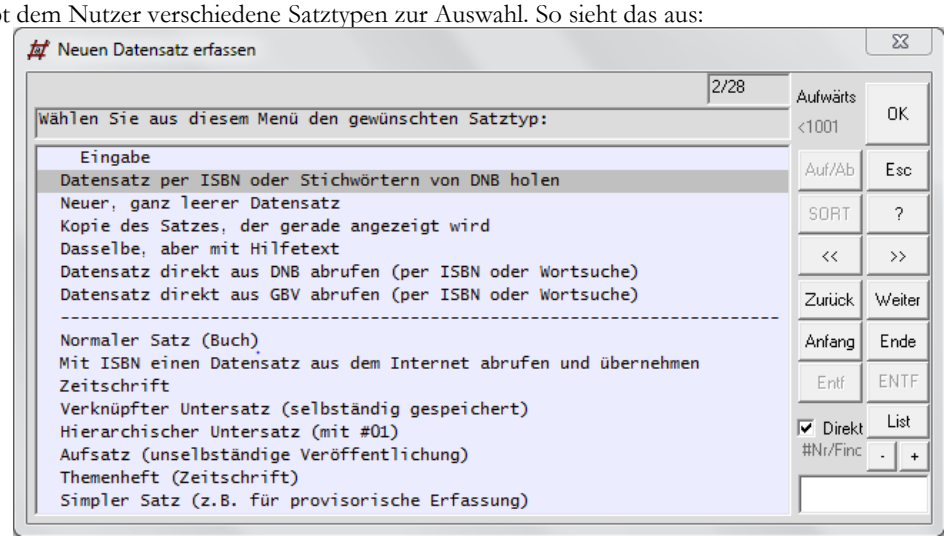
2.1.2 Eingeben und Bearbeiten

Bearbeiten vorhandener Daten
Einen schon vorhandenen Satz kann man jederzeit verändern. Im Auswahlfeld links (siehe S. 2) ist die Übersicht der Datenfelder des Satzes zu sehen (wenn nicht, dann Alt+r). So geht's (siehe auch unten unter „Formulare")

- Man wählt das zu bearbeitende Feld an (Balken drauf und <Enter>). Dann wird der Text in das Schreibfeld kopiert und läßt sich dort beliebig bearbeiten, auch mit Cut-and-Paste.
  - Nochmals <Enter>, und die Kategorie wird zurück kopiert in den Datensatz, zuerst aber geprüft, und zwar feldspezifisch: z.B. ist das erste Titelwort ein Artikel, ist die Jahreszahl plausibel, die ISBN-Prüfziffer korrekt, usw. usf.

TIP: Auch die Kategorienummer ist änderbar. Dann allerdings sind plötzlich zwei Kategorien da, denn das alte Datenfeld bleibt erhalten. Was tun? Das überflüssige anwählen, <Entf>-Taste drücken, weg ist es.
Haben Sie den Satz fertig bearbeitet? Dann [Speichern] (siehe auch unten unter „Offline-Speicher")

Eingeben neuer Daten
Das Hauptgeschäft beim Katalogisieren ist das Erfassen neuer Daten. Am schnellsten geht es mit der Übernahme geeigneter Fremddaten, aber darum geht es hier nicht (siehe Kap. 5). Man muß immer auch neue Sätze von Hand eingeben können, und darauf konzentrieren wir uns hier. Dafür ist der Button **[Neusatz]** da. Wird er betätigt, wird ein FLEX namens **`ONINPUT.FLX`** ausgeführt und gibt dem Nutzer verschiedene Satztypen zur Auswahl. So sieht das aus:



Weiter unten hat man auch noch die Wahl, die vom DOS-System bekannte Abfrageliste aufzurufen. Nicht viel anders als in PRESTO kann man damit zuerst den Satztyp wählen (Buch, Aufsatz, ...) und dann die zugehörigen Daten Feld für Feld eingeben.

Kopieren!
Oft spart es Arbeit, wenn man zuerst einen Satz aufblättert, der eine Anzahl übernehmbare Kategorien enthält. Besonders ist das der Fall, wenn man eine Neuauflage von einem bereits katalogisierten Titel hat. Wird hier mit [Ja] geantwortet, hat man einen neuen Datensatz vor sich, den man nur noch durch ein paar Korrekturen anpassen muß. Ist kein geeigneter Satz zum Kopieren vorhanden, wählt man hier [Nein]. Es wird dann im Auswahlfeld die **Abfrageliste** gezeigt. PRESTO-Anwender kennen dieses Konzept. Hier sieht es etwas anders aus, aber das Prinzip ist dasselbe: es kommt eine Reihe von Abfragen, die man durch die richtigen Eingaben im Schreibfeld beantworten muß.
Wenn Formulare eingerichtet wurden, kann man jederzeit den Button [Formulare] betätigen (Beispiel s.u.), und zwar in beiden Fällen, bei Neueingaben und bei Bearbeitungen. Ein Formular einschalten heißt **nicht**, daß dabei sofort ein neuer Satz entsteht, sondern die Eingaben der Formulare gehen alle in den aktuellen Satz über. Auf der nächsten Seite ist ein Beispiel für ein Formular zu sehen. Ein neuer Datensatz wird nur angelegt mit Druck auf [Neusatz] .

***Hilfe,* wo ist denn mein neuer Datensatz?**
Jederzeit, auch während einer Neuaufnahme, kann man [Index] drücken, blättern und sich andere Daten anschauen. Schwupp, verschwindet aber dann der neue Datensatz vom Schirm - wo ist er geblieben, wie kriegt man ihn zurück? Keine Panik: er landet im *Offline-Speicher*. (Den gibt es bei PRESTO nicht, da muß man einen Datensatz zuerst speichern, vorher kann man sich keine anderen ansehen.) Der Offline-Speicher öffnet sich mit Alt+q oder Druck auf die Schaltfläche unten rechts über der Ergebnismengen-Fläche. "Q : Daten in Bearb." steht drauf. Der letzte Eintrag in der Liste, die dann erscheint, ist die gesuchte Neuaufnahme.
Also: **Alt+q** <Ende> <Enter>, schon hat man den neuen Satz wieder vor sich.

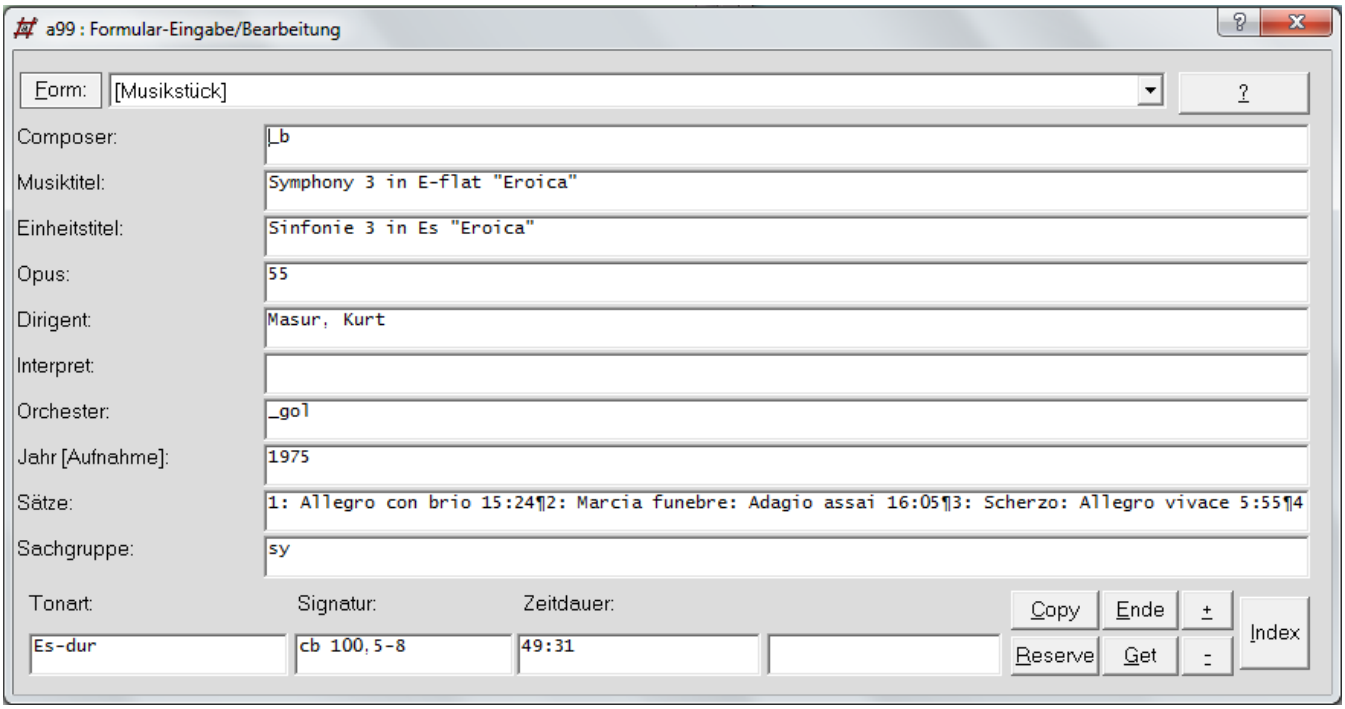
Offline-Speicher? *Online-Hilfe in* ***a99*** *:* **`h off`**

Im Offline-Speicher sammeln sich alle Sätze, die man während einer Sitzung "anfaßt", d.h. an denen man irgendeine Änderung macht, außerdem die Neuaufnahmen. Die bearbeiteten, aber noch nicht wieder gespeicherten Sätze erkennt man an der Kennung EDT vor der Kurzzeile.
Also: Anders als bei PRESTO muß man bei diesem Programm keineswegs einen Satz zuerst abspeichern, bevor man sich den nächsten vornehmen kann, sondern jederzeit kann man zu anderen Sätzen und zu Neuaufnahmen übergehen.
Mehr zu dem Thema in → Kap.**2.2.3** und in der Online-Hilfe.

Formulare

Zur Formular-Methodik: **`h form`** geben!

kann man für jede Datenbank einrichten. Wenn es welche gibt, kann man den Button [Formulare] betätigen (oder Alt+#). Sie sehen hier ein Formular, das für eine *bolero*-Datenbank eingerichtet wurde: (allgemeine **Hilfe** dazu kommt mit dem Button [?])



Hier wird mit Enter nicht das Formular geschlossen, sondern das gerade bearbeitete Datenfeld dem Programm übergeben. So kann es sofort nach der Eingabe geprüft werden, z.B. auf Richtigkeit der Prüfziffer bei der ISBN. Solche "Validierungen" können in den Indexparametern individuell parametriert sein (→ Kap.10.2.8). Wenn nicht Enter sondern Tab gedrückt wird, um ins nächste Feld zu gelangen, wird das Feld erst beim Schließen des Formulars übernommen und geprüft.
Mit [Index] springt man in die Register; dort gibt es zum Übernehmen von Angaben einen [Copy]-Button (s. oben).
Über die Auswahlzeile ganz oben kann man jederzeit eines der anderen Formulare öffnen; die Daten gehen alle in den aktuellen Satz, es entsteht nicht je Formular immer ein neuer Satz! Mit <Esc> verläßt man das Fenster, beendet also die Formulareingabe. Das Feld, in dem gerade der Cursor ist, wird nicht mehr übernommen. Anders beim Button [Ende]: da wird es übernommen.
TIP: Gibt man in ein Feld nur einen Punkt ein und dann <Enter>, wird das Feld aus dem vorigen Datensatz übernommen (genauer: aus demjenigen, der in der Anzeige stand, als man [Neusatz] drückte). Dasselbe macht [Reserve].
TIP: Verschieben Sie das Fenster so nach rechts, daß das Auswahlfeld zu sehen ist.
Und wie richtet man Formulare ein? Das erfährt man mit dem Befehl `h form` im Schreibfeld. Die Formulare werden in einer Datei *dbn*.FRM gespeichert (*dbn* = Name der Datenbank), die auf dem Datenverzeichnis liegen soll. Schauen Sie sich CAT.FRM als Beispiel an. Über das Administratormenü kommt man an die Formulardatei sofort heran (**`h adm`** eingeben).
Beispiel: Für die Bolero-Datenbank gibt es die Formulardatei BOL.FRM. Darin steht dieser Abschnitt (vgl.mit dem Bild oben):

```
[Musikstück]
#52 "Komponist: "
#20 "Musiktitel: "
#22 "Einheitstitel: "
#89o"Opus: "
#57d"Dirigent: "|1      (Die Angabe |1 bedeutet, daß bei Button [Index] das Register 1 erscheint)
...
```

Eingabe abbrechen

Mit dem Menüpunkt [Deaktivieren] kann man eine Erfassung komplett abbrechen, egal ob es eine Kopie oder eine Neu-Erfassung ist.

Sitzung unterbrechen

Im DOS-System noch völlig undenkbar: eine laufende Sitzung kann man verlassen, ohne erst die Änderungen zu speichern, und später neu einsteigen und da weitermachen, wo man aufgehört hatte.

"Wollen Sie die Sitzung später fortsetzen?" Diese Frage kommt vor dem Verlassen des Programms. Antwortet man mit [Ja], wird der gesamte aktuelle Zustand gesichert, einschließlich Ergebnismengen und Offline-Speicher. Wenn Bearbeitungen nicht vollendet wurden, kommt aber die Zusatzfrage: "Sollen die neu bearbeiteten (und noch nicht gespeicherten) Daten gespeichert werden?". Mit [Ja] kann man hier erreichen, daß wenigstens die Bearbeitungen und Neuaufnahmen gespeichert werden, . Für die nächste Sitzung werden dann immer noch die Ergebnismengen aufbewahrt, von den bearbeiteten Sätzen aber auch noch die Originalformen, die man mit [Wechseln] sogar nach dem Neustart wieder aktivieren kann! Der Offline-Speicher besteht aus zwei temporären Dateien vom Typ `.$$$` und `.tab`.

2.1.3 Die Menüs **Online-Hilfe: h menu**

Auf den Menüs findet man viele Grundfunktionen. Einige davon sind auch über Hilfedateien und darin eingebaute Flips erreichbar und werden dort meistens als komfortabler empfunden. Die höheren Exportfunktionen z.B. sind auch über ein Menü in einem Hilfetext zugänglich: geben Sie **`h exprt`**, um dieses Menü zu sehen und zu nutzen. (→ 2.2.4)

Datei [die ersten drei Punkte gibt es nur bei ***a99***]

- **Weitere Datenbank öffnen** - man sucht im Dateisystem die zu öffnende Datenbank oder eine INI-Datei dazu.
- **Schließen** - das Fenster verschwindet nicht, aber alle Dateien der Datenbank werden geschlossen.
- **Alle bearb. Daten speichern** - die während der Sitzung bearbeiteten, aber noch nicht gesicherten Sätze werden gespeichert
- **Ergebnismenge -> Datenbank** - Nur diejenigen Sätze der aktuellen Ergebnismenge, die verändert wurden oder neu sind, werden gespeichert
- **Offline-Datei -> Datenbank** - Wenn es Offline-Daten gibt (z.B. vorher mit "Weitere Offline-Datei" geladen), werden diese alle als neue Datensätze in die Datenbank überführt (s.a. Befehl `Offline=...` in der INI-Datei)
- **Anzeige speichern als ...** – Der Inhalt des Anzeigefeldes kann im momentanen Zustand (also einschl. der manuellen Änderungen, die man gemacht hat) als Datei abgespeichert werden. Wählt man den Dateityp .RTF, kann man die Datei hinterher mit WinWord einlesen, wählt man *.TXT, wird als ASCII-Datei gespeichert (für DOS geeignet).
  Diese Funktion ermöglicht es, Hilfedateien, Parameterdateien usw. im Anzeigefenster zu bearbeiten!
- **Anzeige ausdrucken** - macht dasselbe wie der Drucker-Button [wenn vorhanden, wird **`onprint.flx`** gestartet]
- **Seite einrichten** - um die Breite der Seitenränder beim Drucken einzustellen
- **DOS-Programm** - startet in einem Fenster das konventionelle Programm PRESTO bzw.APAC, wobei dann als erstes derselbe Registerabschnitt erscheint wie in **alcarta**
- **Externe Ergebnismenge laden** - Es wird nachgesehen, ob es eine Datei EXTERN.DAT gibt. Wenn ja, wird sie als Ergebnismenge präsentiert. Sie kann aus einem anderen Programm heraus entstanden sein, z.B. einem Z39.50-Client. Diese Datei muß im allegro-Externformat strukturiert sein (Handbuch Kap. 0.2.2)
- **Weitere Offline-Datei laden** - in einer Windows-Dateiauswahlbox kann man nach Dateien des Typs .ALG und .ADT suchen. Die ausgewählte Datei wird als Offline-Datei geladen und als Offline-Ergebnismenge gezeigt. Die Sätze können bearbeitet und in die Datenbank gespeichert werden (s.o. "Offline-Datei -> Datenbank).
- **Normalposition** – Hat man das Fenster in der Größe verändert: Normalgröße wieder einstellen.
- **Beenden** - Programm verlassen. Je nach Situation werden noch 1 oder 2 Fragen gestellt.

Bearbeiten [bei ***alcarta*** gibt es nur den ersten Punkt]

- **Alles markieren** - Dasselbe wie Strg+a im Anzeigefenster: anschließend kann mit Strg+c der gesamte Anzeigebereich in die Zwischenablage kopiert werden. (In WinWord kommt dann alles mit Strg+v korrekt wieder heraus.)
- **Extern** - Der aktuelle Datensatz wird in die Datei `exx.xxx` geschrieben (Export mit E-W.APR) und mit dem externen Editor (WinVi o.a., Einstellung in der INI-Datei) zum Bearbeiten vorgelegt. Bei Rückkehr aus dem externen Editor wird gefragt, ob der externe Satz geladen werden soll; er ersetzt dann den aktuellen Satz im Arbeitsspeicher.
- **Read** - Die Datei `e.adt` wird geladen und in den aktuellen Satz eingemischt. Das ist eine Erfassungshilfe. In `e.adt` können beliebige Angaben im Externformat stehen.
- **Neuaufnahme** - Siehe Schaltfläche [Neusatz]
- **EingabeFormulare** - siehe Schaltfläche [EingMask]
- **Markieren/Verschieben** - nur bei hierarchischen Sätzen anwendbar. Wenn der Cursor im Listenfeld auf einer Kategorie #01, #02 ... steht, wird diese markiert. Wenn er *innerhalb* eines Untersatzes steht (also nicht auf dessen #01...) wird der vorher markierte Untersatz dahinter verschoben. Wenn er nicht verschoben, sondern kopiert werden soll:
- **Kopieren** - Nur bei hierarchischen Sätzen: Wenn vorher ein Untersatz markiert wurde, wird dieser nun hinter den Untersatz kopiert, in dem gerade der Cursor steht.
- **Satz freigeben** – Wenn der aktuelle Satz gesperrt ist, kann man ihn hiermit wieder freigeben (Berechtigung 3 nötig)

Ansicht [Zur Erinnerung: immer kann man mit dem Printer-Button den Inhalt des Anzeigefeldes sofort ausdrucken]

- **Aktueller Satz in Druckform** - Für die Druckanzeige werden andere Parameter benutzt, wobei z.B. die für das Drucken nicht gewünschten Flips entfallen, aber auch die gesamte Struktur anders aussehen kann als bei der OPAC-Anzeige.
- **Aktuelle Erg. Menge in Druckform -** Die gesamte Ergebnismenge wird in der Druckform gezeigt
- **Kurzliste der Erg. Menge** - Macht dasselbe wie der Button [List] im Ergebnisfenster.
- **Internformat** - Der aktuelle Satz wird in kategorisierter Form gezeigt (auch: F5)
- **Anzeigeparameter wechseln** - Bei *a99* kann man in der Dateiauswahl-Box eine Datei des Typs D-*.cPR auswählen, die dann geladen und für die Anzeige benutzt wird. Bei *alcarta* wird die in der .INI-Datei eingestellte Anzeigeparameterdatei neu geladen und die Anzeige damit neu aufgebaut. Gedacht zum schnellen Testen der Anzeigeparameter.
- **Alle Erg.mengen** - Gleichwertig ist Alt+e (s. oben Punkt 5). Man erhält im Listenfenster die Liste der Ergebnismengen. Bei ***alcarta*** steht diese ständig dort, es sei denn man benutzt Alt+r oder Alt+a
- **Hintergrundspeicher / Datensatz anzeigen** - gleichwertig ist Alt+r. Siehe Button [Reserve]
- **Datensatz anzeigen** – Ist gerade ein Hilfetext in der Anzeige, holt man hiermit den Datensatz zurück. Oder mit Alt+z
- **Views** – Ein eigenes Menü erscheint. Damit kann man verschiedenste Ansichten der Erg.menge produzieren.
- **Registereinträge** – Die zum aktuellen Satz gehörigen Registereinträge werden gezeigt (auch: F7)
- **Letzter Hilfetext/Datensatz** - der zuletzt benutzte Hilfetext wird erneut gezeigt bzw. der zuletzt betrachtete Datensatz.

Finden : zum Öffnen des "Fernglas"-Menüs (Alt+f), der Indexfenster (Alt+i und Alt+j) sowie

Volltextsuche mit regulären Ausdrücken

Export

Exportieren kann man mehrere Dinge: *online-Hilfe mit größerem Menü :* **`h exprt`**

- **Aktueller Satz**
- **Aktuelle Ergebnismenge**
- **Kurzliste der Erg. Menge**
- **Ganze Datenbank**
- **Ganze Offline-Datei**

Exportiert wird mit Hilfe der Parameter `e-w.apr` in die Datei `output.dat`. Beides kann in der INI-Datei anders eingestellt werden, aber auch über zwei weitere Menüpunkte jederzeit geändert:

- **Andere Exportdatei** (Datei suchen oder neuen Namen eingeben)
- **Andere Exportparameter** (Auswahlbox mit den wählbaren Parameterdateien erscheint)
- **Ausgabedatei löschen** (Beim nächsten Exportbefehl wird dann eine neue unter demselben Namen geöffnet)
- **Komfort-Methoden** (gezeigt wird die Menüdatei `exprtger.rtf`, → 2.2.4 mit Untermenüs und Spezialfunktionen)

Extras [nur bei *a99*]

Das Menü hat zwei Teile: "Offline-Datei" und "Datenbank (ONLINE)". Jeder Menüpunkt liefert eine spezielle Ergebnismenge. Gedacht ist das Ganze zur Utnerstützung der Bearbeitungsfunktionen. Normalerweise wird man meistens den länglichen Button [Q -: Daten in Bearbeitung] mit Alt+q aktivieren, um die momentan in Arbeit befindlichen Sätze zu sehen! Bei längeren Sitzungen sind aber folgende detaillierten Funktionen von Interesse:

[Offline-Datei] : Datensätze im Arbeitsspeicher von *a99*

- **Ungültige Offline-Sätze** - die mit dem Button [Deaktivieren] ungültig gemacht wurden.
- **Geänderte Offline-Sätze** - solche, an denen während der laufenden Sitzung etwas bearbeitet wurde
- **Änd. in Erg.Mg. rückgängig** - alle Sätze der aktuellen Ergebnismenge, die während der Sitzung geändert wurden (und noch nicht in die Datenbank abgespeichert), werden wieder auf den Originalzustand gesetzt. Mit [Wechseln] kann jeder einzelne wieder auf den Änderungszustand gesetzt werden!
- **Offline-Änderungen rückgängig** - dasselbe, aber anstatt der Ergebnismenge werden alle Sätze der Offline-Datei berücksichtigt

[Datenbank (ONLINE)] : Datensätze, die schon einmal gespeichert wurden

- **IN ARBEIT befindl. Sätze** - Als Ergebnismenge erhält man alle Online-Sätze, die in der laufenden Sitzung verändert, aber noch nicht abgespeichert wurden.
- **Korrigiert+gespeichert** - als Ergebnismenge erscheinen alle Sätze, die während der Sitzung korrigiert und gespeichert wurden. Also keine neuen Sätze, sondern nur solche, die vorher schon in der Datenbank standen.
- **Gespeicherte Sätze (neu+korr.)** - Alle während der Sitzung gespeicherten Sätze
- **Neu, noch nicht gespeichert** - Alle Neuaufnahmen, die noch nicht in die Datenbank gespeichert wurden.
- **Gesperrte Sätze** - Findet alle momentan von der Datenbank her gesperrten Sätze und kann sie freigeben.

Global vor allem für Globale Änderungen und die Löschung von ganzen Ergebnismengen. Ferner: Datenbank-Info.

Option

- **Datenfont** Die Schriftart im Listenfeld, Schreibfeld, Index und Kurzanzeige kann hier verändert werden
- **Teilfeld-Hilfe** [nur *a99*] - normalerweise deaktiviert. Wenn aktiviert, wird ein Hilfsfenster für die Teilfeld-Bearbeitung angeboten. Wer mit den Teilfeldern vertraut ist, wird oft auf das Hilfsfenster verzichten wollen.
- **Farbwechsel** [nur *a99*]- Wenn deaktiviert, wechselt die Farbe im Anzeigefenster nicht. Wenn aktiviert, wechselt sie auf hellgelb: veränderter, noch nicht wieder gespeicherter Satz, bzw. hellroetlich: gelöschter Satz
- **Ergeb. aufbewahren** - Wenn aktiviert: Beim Verlassen des Programms werden alle Lesezeichen, die Liste der vorher angezeigten Daten und die Ergebnismengen gesichert, so daß bei erneutem Start diese Ergebnisse sofort wieder zur Verfügung stehen. Beim Start ist dieser Punkt nicht aktiviert!
- **Eingabeschrift+ /** - Die Schrift im Auswahlfeld und Schreibfeld kann hierdurch schrittweise vergrößert und verkleinert werden. (Diese Einstellungen bleiben nicht zur nächsten Sitzung erhalten.)
- **Anzeigeschrift+ /** - Die Schrift im Anzeigefeld vergrößern/verkleinern. Das funktioniert nur dann, wenn in die gezeigten Textdaten keine Befehle für die Schriftgröße eingebaut sind - erkennbar ist dies von außen jedoch nicht!

Löschen : Der aktuelle Datensatz wird, natürlich nur nach Rückfrage, gelöscht.

Füllhorn und **?** (auch **`Alt+h`**) **Das ? ist der wichtigste Punkt von allen → 2.2!**

Es erscheinen umfangreiche Menüs, die unter anderem die Online-**Dokumentation** zugänglich machen, aber auch alle fortgeschrittenen Funktionen wie **Listenproduktion**, **Views**, **Datensicherung**, **Reorganisieren, Administration** usw.

2.1.4 Die Buttons

Einige, wie [Speichern] oder [Drucker] sind wohl selbsterklärend, andere (wie [Neusatz] oder [Fernglas]) wurden weiter oben schon erklärt. Außerdem gibt es einen Hilfetext, den man mit dem Befehl **`h buttn`** im Schreibfeld anfordern kann.
Nicht alle Buttons sind sichtbar, dennoch haben manche davon ihre Einträge in der UIF-Datei: z.B. Alt+k, Alt+m, Alt+t.
Hier folgen deshalb nur wenige Erläuterungen zu ausgewählten Schaltflächen.

[Reserve] bzw. **[Record]**: Damit läßt man im Auswahlfeld den Hintergrundspeicher (Reservespeicher) anzeigen. Dieser enthält Kopien normaler Kategorien, die man sofort neu verwenden kann, aber auch die #u-Variablen, die u.a. für die FLEXe gebraucht werden. Wer sich auskennt, kann solche Variablen dann manuell bearbeiten. Nochmaliger Druck auf diesen Button schaltet wieder zurück zum aktuellen Datensatz.

[Wechseln] : Hiermit schaltet man zwischen der veränderten Fassung eines Datensatzes und der momentan in der Datenbank gespeicherten Fassung hin und her. Die veränderte (aber noch nicht gespeicherte) Fassung hat immer einen gelblichen Hintergrund und ist damit unmittelbar zu erkennen. Man nutzt dies, um schnell zu vergleichen, wie der gültige (tatsächlich gespeicherte) Satz aussieht, oder auch, um den veränderten wieder ungültig zu machen. Dann wird am Ende der Sitzung der bearbeitete einfach endgültig vergessen. Nur die "gelben" Sätze werden dann gespeichert. Wenn man per Button [Speichern] einen Satz von Hand speichern läßt, wird die frühere Fassung aus der Datenbank kopiert und in den Offline-Speicher übernommen. Mit [Wechseln] kann man nach dem Speichern dann immer noch den älteren wieder gültig machen!

[Neues Lesezeichen] : Einzelne Sätze kann man sich hiermit markieren. Diese Sätze bilden dann eine besondere Ergebnismenge, eben die sog. "Lesezeichen"-Sätze. Nützlich, um bestimmte Sätze schnell wiederzufinden.

[Erg.Mengen] : Dieser Button ist nicht sichtbar, kann daher nur mit Alt+e ausgelöst werden! Links kommt dann die Liste der in der Sitzung entstandenen Ergebnismengen in Sicht. Jede davon kann sofort erneut ausgewählt und benutzt werden.

[Schema] : Die Liste aller erlaubten Kategorien erscheint im Auswahlfeld. Diese Liste wird aus der Konfigurationsdatei (CFG-Datei) entnommen. Auch dieser Button ist unsichtbar und muß mit Alt+y ausgelöst werden. Nochmaliger Druck läßt dann die von DOS bekannte Abfrageliste erscheinen. Das passiert auch automatisch, wenn man über **[Neusatz]** den Erfassungsmodus für "Neue Eingabe" gewählt hat. Dann wählt man einen Satztyp und kann anschließend die zugehörigen Kategorien hintereinander eingeben, fast genauso wie im alten PRESTO

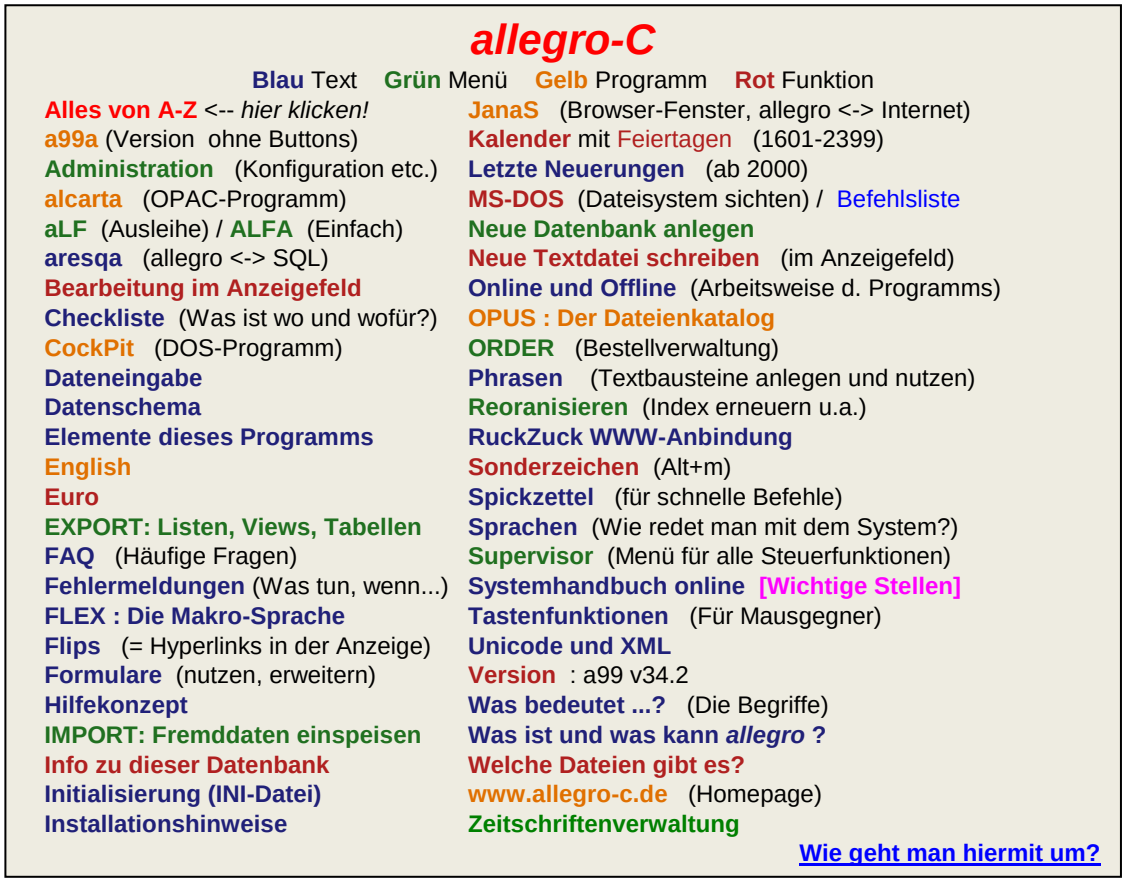
[Flip-Buttons] : Links neben dem Schreibfeld sieht man 8 Buttons, die der Anwender selber mit Makros (FLEXe) belegen kann. Ausgelöst werden sie mit Alt+1 bis Alt+8. Mit **`h xflip`** erfährt man, wie das geht.

Statusfelder : Außer den oben beschriebenen Elementen gibt es noch ganz unten fünf kleine Status-Anzeigefelder (1-3 sind links, 4 und 5 rechts):

1. **DEL** wenn ein gelöschter / deaktivierter Satz angezeigt wird
2. **EDT** wenn der Satz in der aktuellen Sitzung geändert wurde (dann ist [Wechseln] aktiv)
3. **ONLINE** wenn der Satz ein Datenbanksatz ist
   Offline wenn der Satz zur Offline-Datei gehört (Befehl Offline=... in der INI-Datei)
   NEU wenn er als Neusatz angelegt, aber noch nicht gespeichert wurde
4. Die interne Satznummer, die Datenbank-Dateinummer und die Länge des Satzes.
5. Die laufende Nummer des Satzes in der Ergebnismenge / Größe der Erg.menge

2.2 Die Komfort-Funktionen

Unter dem Menüpunkt **?** verbirgt sich viel mehr als eine Hilfefunktion. Es erscheint folgende Seite:



Hinter den farbigen Punkten dieses Menüs stecken entweder Texte (RTF-Dateien) oder FLEXe (Makros). Das braucht der Normalanwender zwar nicht zu wissen, der etwas ambitionierte und der Systemverwalter jedoch wird sich dafür interessieren. Deshalb der *Tip*: Geben Sie im Schreibfeld ein **`h doku.rtf`** , dann sehen Sie diese Datei in der internen Form, aus der man ersehen kann, was an den einzelnen Punkten "dranhängt": unter dem Menü sieht man dann die Liste der "Flips". Diese Datei, sowie auch viele andere, kann auch dazu dienen, eigene Menüs zu erstellen. Die Bearbeitung und das Speichern der Datei kann unmittelbar im Programm erfolgen (wenn man vorher **`h doku.rtf`** gegeben hat; Speichern dann am Ende mit dem Button [Speichern]). Bearbeitung ist auch möglich mit dem Editor WordPad, der speziell für RTF-Dateien gedacht ist. Von WinWord ist für diesen Zweck abzuraten!

Die vier Farben der Elemente sollen nicht irritieren, sondern deuten eine grobe Gliederung an:

Blau: Text
Hilfetexte und Dokumentationen zu verschiedenen Themen. Auf der folgenden Seite als wichtiges Beispiel der "Spickzettel" abgedruckt, im Anschluß dann das Blatt mit den Tastenfunktionen. Diese Liste erhält man online mit **Alt+c**.

Grün: Menü
Dies sind Texte mit vielen eingebetteten Funktionen, die man jeweils durch Anklicken auslösen kann.

Gelb: Programm
Diese Menüpunkte starten jeweils andere Programme, z.B. das DOS-CockPit.

Rot: Funktion
Damit werden einzelne nützliche Funktionen ausgelöst, etwa die Euro-Umrechnung, die Kalenderfunktion, oder die Anzeige von nützlicher Information zur Datenbank. Besonders vielseitig: "**Alles von A-Z**". Eine alphabetische Liste erscheint in einem eigenen Fenster, in dem man auch suchen kann. Diese Liste ist eine sog. ViewListe. Steckt hinter einer Zeile ein Hilfetext, wird er direkt angezeigt und die Liste bleibt offen, steckt ein FLEX-Makro dahinter, wird es ausgelöst, wenn man die Liste mit Enter schließt.

2.2.1 a99 - Spickzettel Online: h spick *oder* F12 *oder* Alt+c (Notepad!)

Auf dieser Seite ist kompakt aufgelistet, welche Befehle und Tastendrücke die verschiedensten Funktionen und Menüs zugänglich machen. Am besten ausdrucken und das Blatt an die Wand hängen! Speziell für die Tasten gibt es noch eine Extraseite:

2.2.2 a99 - Tastenbelegungen

Online: **Alt+c**

Alt		**Strg** ([W]=Windows-Vorgabe, nicht ändern!)
a	Ansicht (Menü)	Alles markieren [W]
b	Bearbeiten (Menü)	*(frei)*
c	Dieser Text	Copy (Anzeige, Eingabe) [W]
d	Datei (Menü)	*(frei)*
e	Erg.Liste	*(frei)*
f	Find-Menue	Find (im Anzeigefeld)
g	Global (Menü)	Weitersuchen
h	Füllhorn (allg. Hilfe)	Backspace [W]
i	Index 1	TAB (im Anzeigefeld)
j	Index 2	Neue Zeile (im Eingabefeld) [W]
k	Satz<->Reserve Feldkopie	¬ = Nichtsortierzeichen
l	Nächster Satz Erg.Menge (u voriger)	ł
m	Sonderzeichen-Hilfeseite bzw. mark/move hierarch. Untersätze	Enter (nicht belegen!) [W]
n	Nächster Offline-Satz	ñ
o	Option (Menü)	*(frei)*
p	Export (Menü)	ç
q	Übersicht Sätze in Bearb.	*(frei)*
r	Arb.Speicher<->Reserve	ř
s	Speichern (Datensatz bzw. Text)	ś
t	Externe Bearbeitung	¶ = Absatz-Endezeichen
u	Voriger Satz Erg.Menge (l nächster)	*(frei)*
v	Voriger Offline-Satz	Paste (Eingabe, Anzeige) [W]
w	Wechseln bearb.<->unbearb.	*(frei)*
x	Extras (Menü)	Cut (Eingabe, Anzeige) [W]
y	CFG<->Abfrageliste	*(frei)*
z	Interne<->Externe Anzeige (auch F5)	Undo (Eingabe, Anzeige) [W]
ä	E.ADT einlesen (zu aktuellem Satz ergänzen)	
ö	Aktuellen Satz löschen	
ü	Übersicht aktuelle Ergebnismenge	
?	Hilfetext zur Datenbank	
		TAB Auswahl →Schreib- → Befehlsfeld
#	Formulare	**F1** Hilfe (kontextbezogen)
.	Druckmenü (Alt+Punkt)	**F5** Anzeige umschalten intern<->extern
+	Nächste Registerzeile	**F6** Index auf/zu
-	Vorige Registerzeile	**F7** Schlüssel des aktuellen Satzes
<	Vorige interne Satznummer	**F8** Funktionen zum aktuellen Satz
>	Nächste interne Satznummer	**F9** Neusatz erfassen
(	Voriger Datensatz bzw. Hilfeseite	**F10** Menü "Datei"
)	Nächster Datensatz bzw. Hilfeseite (in Reihenfolge des Aufrufs)	**F11** (Anzeigefeld breit<->schmal)
		F12 (Dieser Hilfetext)
		F2,**F3**,**F4**,**F8** : `onf2.flx` usw. auslösen

1 Flip1
2 (Die Flip-Funktionen können frei belegt werden,
... dazu dient der FLEX-Befehl flip)
9 Erklärung: `h xflip`
0 Flip10

`AltGr+2`	▼	`(Teilfeldcode)`
`AltGr+3`	ˇ	`(Haček)`
`AltGr+e`	€	`Euro`

Sonderzeichen ansonsten: → Anh. E / Online: Alt+m

Hinweis: Die Alt-Belegungen werden durch die "Hotkeys" (unterstrichene Buchstaben) der Buttons bzw. Menüpunkte bestimmt. Die englische Version hat daher andere Zuordnungen.

2.2.3 Bunte Daten? (Online und Offline)

Mehr dazu: **`h off`**

Hintergründe

Es wäre doch prima und alles ganz einfach, wenn man die Datenbank auf dem Bildschirm unmittelbar sehen könnte, am besten wie eine Tabelle, und wenn jede Änderung ganz direkt passieren würde! Wie auf einem Blatt Papier oder auf Karteikarten - oder wie bei einem Text- oder Tabellenprogramm. So ist es aber nicht und so *kann* es technisch gar nicht sein. Das schafft ein Potential für Verwirrung. Hiermit wollen wir ein wenig Aufklärungsarbeit leisten. (Bei Textprogrammen, nebenbei bemerkt, sieht man auch nicht unmittelbar den gespeicherten Text, man ist aber viel "dichter dran".)
Eins vorweg : Nicht jeder *braucht* alles zu kennen und *kann* alles nutzen, und nicht jeder *will* alles wissen, das ist auch den Entwicklern klar. Also zuerst die Frage:

Was *muß* denn jeder wissen? (Damit man keine groben Denkfehler macht)
Die Daten**bank** besteht aus Daten**sätzen** und jeder Satz aus Daten**feldern** oder "Kategorien". Das Konzept der Tabelle (wie bei relationalen Datenbanken) ist zu unpraktisch, weil es zu viele Felder gibt (die Tabelle würde viel zu breit) und weil viele Felder mehrfach belegbar sein sollen (das geht in einer Tabelle nicht gut).
Jeder Datensatz entspricht, klassisch-bibliothekarisch besehen, einer "Titelaufnahme".
Um alle Komplikationen zu vermeiden, *kann* man sich darauf beschränken, immer nur an *einem* Datensatz zu arbeiten und konsequent diesen erst fertig zu machen und *abzuspeichern*, bevor man irgendwelche anderen Aktionen macht, außer daß man kurz mal irgend etwas im Index nachschaut. Also immer schön "eins nach dem andern". Das ist dasselbe Verhalten wie beim DOS-Programm, bei dem es gar nicht anders ging.
Als Hilfe gibt es die Hintergrundfarbe! Wer es ganz einfach haben will, merkt sich nur:

> **Wenn Hintergrund gelb, dann:**
> **erst `[Speichern]`, bevor man sich den nächsten Satz vornimmt,**
> **oder `[Wechseln]`, um die Änderungen ungültig zu machen**
> **bzw. `[Deaktivieren]` (bei NEUen Sätzen), wenn man diesen Satz so *nicht* speichern will.**

Das genügt! Wie gesagt, so *kann* man es machen, und wer weiter nichts will, braucht nicht weiterzulesen. (Man nutzt ja ohnehin bei jeder Software nur einen kleinen Teil der Möglichkeiten, das ist ganz normal.)
Aber wenn man etwas mehr Farbe ins Leben bringen will:
Dieses Programm soll ein flexibleres Arbeiten zulassen als die alten **DOS-Programme**. Nicht mehr immer "eins nach dem andern", sondern auch mal was zwischendurch und nebenbei! Außerdem soll es auch bei Bedarf mit Daten umgehen können, die nicht zur Datenbank gehören. Es muß den Nutzer deshalb dabei unterstützen, die Übersicht zu behalten. Und das tut es mit Farben:

Bunte Daten : Grüne, gelbe, rote, blaue Sätze

Schauen wir doch mal, wie es sich mit den möglichen Zuständen eines Datensatzes genau verhält.
Es ist nicht schwer: an der **Hintergrundfarbe** kann man erkennen, was mit dem gerade sichtbaren Satz los ist. Wir nennen die Farben **grün, gelb, rot und blau**, obwohl es sich um sehr helle Farbtöne handelt.
Statt auf die Farben kann man auch auf die Status-Angaben achten, die man ganz unten links sieht. Aber die Farben nimmt man viel schneller wahr:

grün: Man erblickt gerade den Satz genau so, wie er in der Datenbank gespeichert ist. "Im grünen Bereich", wie man so sagt, auf Nummer Sicher. Macht man aber die geringste Änderung, dann wird der Hintergrund **gelb**:
(*Status:* **ONL** = Online = sichtbar für alle, die gerade an der Datenbank tätig sind)

gelb: Das bedeutet: *Aufpassen!* Man erblickt einen veränderten Datensatz, der aber noch nicht abgespeichert wurde - in der Datenbank steht noch der grüne (den man vor der ersten Änderung sah). Andere Nutzer im Netz sehen die Änderungen noch nicht, und es steht davon auch noch nichts in den Registern!
Am Sitzungsende wird der gelbe gespeichert. Aber mit dem Button [Wechseln] kann man das verhindern: dann ist der grüne wieder da und nichts passiert (der gelbe ist nicht weg, aber inaktiv. Nochmal [Wechseln] gedrückt, und er kommt zurück.).
Mit dem Button [Speichern] kann man schon vor dem Sitzungsende den gelben speichern lassen. Natürlich wird er dann **grün**. Der vorher grüne, den gibt's auch jetzt immer noch! Nur ist der jetzt gelb. Mit [Wechseln] kann man erneut zwischen beiden umschalten.
(*Status:* **EDT** = Edited = Bearbeitet - aber noch nicht gespeichert)

rot: Dieser Satz wurde gelöscht (Merkvers: "rot ist tot"). In den Registern ist er nicht mehr zu finden. Während der Sitzung kann man ihn mit dem Button [Aktivieren] wieder in die Datenbank speichern lassen, also vor dem endgültigen Aus retten. Dann wird er wieder **grün** und steht auch wieder im Index.
(*Status:* **DEL** = Deleted = Gelöscht)

blau: Davon gibt es zwei verschiedene Sorten: **neue** und **externe** Sätze. Die zweite Sorte sieht der Normalanwender nie! In den Registern steht von solchen Sätzen noch gar nichts.
(*Status:* **NEU** *bzw.* **EXT**; Merkvers: "blue is new"). Bei der ersten Änderung wird so ein Satz sofort **gelb**, der Status ist aber immer noch "NEU" bzw. "EXT", daran sieht man, daß er noch immer nicht zur Datenbank gehört. "NEU" bedeutet: er wird am Sitzungsende in die Datenbank gespeichert, "EXT" heißt: er wird in die Externe Datei zurückgespeichert. Das Speichern kann man verhindern mit dem Button [Satz lösch.] oder [Deaktivieren]. Dann wird er **rot**. Die externe Datei ist dann am Ende also kleiner.

Blaue Sätze mit Status NEU kommen durch Eingeben oder Einlesen während der Sitzung zustande. Sie werden immer am Ende der Offline-Liste angehängt. Mit dem Button [Daten in Bearb.] sieht man diese Liste. Aber auch die gelben (geänderten) Online-Sätze sammeln sich dort an! Mit [Voriger] / [Nächster] kann man darin blättern, unabhängig von der Ergebnismenge.
Und EXT? *Nur* wenn eine Offline-Datei in der INI-Datei angegeben ist (normalerweise also nicht), *oder* wenn man eine im Start-Aufruf angegeben hat (siehe unten) hat man überhaupt (gleich nach dem Start) eine Anzahl blauer Sätze mit dem Status "EXT". Mit [Daten in Bearb.] kann man sie sofort sehen.

Gelbe Sätze (EDT), können im Status ONL, NEU oder EXT sein - völlig klar.
Merken muß man sich trotzdem **nur**:

> **Gelb** bedeutet "**Achtung! So wird der Satz am Ende gespeichert**".

Keine Sorge, auch wenn Sie extrem fleißig sind und enorm viele Sätze während einer Sitzung eingeben und bearbeiten: das Programm verliert nicht die Übersicht, welche davon noch gespeichert werden müssen! Das kann man ihm völlig überlassen, aber man kann es auch jederzeit abfragen; es gibt dafür das **Menü "Extras"**.
Wichtig sind dort nur diese zwei Punkte:

IN ARBEIT befindl. Sätze
> Das sind gelbe Sätze mit dem Status **ONL**
> (Diese Menge kann mit dem Befehl **`find edt`** auch in einem FLEX ermittelt werden)

Neu, noch nicht gespeichert
> Alle Sätze mit dem Status **NEU**, gelbe und blaue. (Die roten - ungültig gemachte neue - fallen dabei unter den Tisch.)
> (Diese Menge kann mit dem Befehl **`find neu`** ebenfalls per FLEX ermittelt werden.)

Man bekommt die betr. Sätze als Ergebnismengen und kann sie sich anschauen. Über das Menü
> **Datei | Ergebnismenge -> Datenbank**

kann man jederzeit die Speicherung aller dieser Sätze veranlassen.
(Nebenbei: die FLEX-Befehle `fin edit` und `fin new` tun es auch. Die Statusbezeichnungen, ONL, EDT usw., stehen in der Datei UIFEGER, Zeilen 390-395, und man darf sie ändern.)

Am Sitzungsende prüft das Programm, ob diese Mengen noch etwas enthalten. Wenn ja, kommt die Frage
"Sollen die neuen und bearbeiteten (noch nicht gespeicherten) Sätze gespeichert werden?"
An dieser Stelle stutzen Uneingeweihte regelmäßig, denkend, sie hätten doch alles schon gespeichert. Aber das Gedächtnis des Programms ist in diesem Punkt denn doch besser...
Zur Vorsicht sollte man dann folgendes tun: Button [Abbrechen] drücken und sich diese Mengen nochmals zeigen lassen, da wird man's dann ja sehen. Vielleicht ist es nur ein einzelner Satz, wo man versehentlich nicht [Speichern] gedrückt hat, vielleicht ist es ein NEU-Satz, den man eigentlich wieder ungültig machen wollte - beides läßt sich sofort aus der Welt schaffen.
Per FLEX kann man das Ende noch sanfter gestalten, indem man sich einen **`_ENDFLX.FLX`** schreibt. Als Beispiel wird die Datei `ENDEFLX.FLX` mitgeliefert, die man auf diesen Namen umkopieren kann, um sie zu nutzen.
Man muß drei Situationen unterscheiden, die nachfolgend beschrieben werden. Dabei sind jeweils alle Aspekte und möglichen Fälle der Bearbeitung zu durchdenken
Schauen wir uns aber zuerst an, wie eine Sitzung schematisch abläuft:

So läuft eine Sitzung ab

Normalerweise hat man in der INI-Datei keinen Befehl "`Offline=`*dateiname*". Dann entfällt alles, was zur "Externen Datei" gesagt wird.

1. Start des Programms

1. Einlesen der INI-Datei
2. Öffnen der Datenbank, Ausführen des `_start.flx` (falls vorhanden)
3. (Einlesen der Externen Datei)
4. (Einlesen der von der letzten Sitzung übriggebliebenen Daten und/oder Ergebnismengen)

2. Arbeit an der Datenbank

1. **Bearbeiten von Datensätzen**
2. **Eingeben neuer Sätze** (Per Tastatur oder durch Einlesen externer Daten)
3. **Löschen von Datensätzen**
4. (Bearbeiten und Löschen von Sätzen in der Externen Datei)

3. Ende der Sitzung

Wenn im Menü "Option" die Zeile "Ergeb. aufbewahren" markiert ist:
Sichern des Zustands der gesamten Sitzung - dann entfallen nachfolgende Fragen und Schritte:

a) `Sitzung später fortsetzen?` (Ja: alle Ergebnismengen und bearbeiteten Sätze bleiben erhalten)

b) `Sollen die neuen und bearbeiteten Daten gespeichert werden?`
(Ja: die bearbeiteten Sätze werden noch gespeichert;
Nein: sie werden wenigstens aufbewahrt, wenn a) mit "Ja" beantwortet wurde)

Schließen der Datenbank. Wenn eine Offline-Datei geladen wurde:

`c) Offline-Datei aktualisieren, d.h. Änderungen sichern? Nein: Offline-Datei unverändert lassen`
Ja: Schreiben einer neuen Fassung der Externen Datei; die alte bleibt unter name.BAK erhalten)

Hat man eine Externe Datei bearbeitet, kann man sie anschließend beliebig weiterverwenden. Sie gehört nach wie vor nicht zur Datenbank. Es sei denn, man hat sie abspeichern lassen! (Über das Menü "Datei | Offline-Datei -> Datenbank".) Dann ist sie anschließend nicht mehr extern, sondern intern, verschluckt von der Datenbank.

Noch etwas mehr zu den Hintergründen : `h off` eingeben.

2.2.4 Export

Die Windows-Programme bieten mehr Möglichkeiten als die DOS-Programme, Daten in unterschiedlichster Form auszugeben. Beiden gemeinsam ist aber, daß Exporte parametriert werden können. Zu diesem umfangreichen Thema informieren die Kapitel 6 und 10. Hier geht es nur um den Überblick der Komfort-Exportmethoden, die auf einem eigenen Menü zusammengestellt sind (siehe **Komfort-Methoden** auf dem Menü "Export", dahinter steckt die Datei `exprtger.rtf`):

Komfortable Funktionen	Grundfunktionen
Tabellen erstellen	Aktuellen Satz ausgeben
(zur Verwendung in anderen Programmen)	
View-Technik	Aktuelle Erg.menge ausgeben
(Andere Ansicht einer Ergebnismenge)	...
Zählungen	Ganze Datenbank exportieren
(Statistik von Feldbelegungen)	
Summierung	Ganze Offline-Datei
(Ergebnismenge aufaddieren)	
Registerabschnitt	Kurzliste der Erg.menge
(als Textliste, ViewListe, Erg.Menge)	(in Ausgabedatei schreiben und anzeigen)
Volltextsuche / (per DOS)	Andere Exportparameter nehmen
(in der gesamten Datenbank)	(zur Zeit: e-w)
Sortierte Listen	Andere Exportdatei einstellen
(in verschiedener Aufbereitung)	(zur Zeit: output.adt)
XML-Ausgabe - MARC21	Ausgabedatei löschen
(Datei alg.xml produzieren)	(momentane Größe: 1475 Byte)
Neuerwerbungsliste	Handbuch
(Ergebnismenge dafür erstellen)	(Kap. 6 des Systemhandbuchs)
CockPit	Grunddateien durchsuchen, sortieren, exportieren
(Das Classico-System)	(Mit FLEX-Methoden)

Was die Export-Komfort-Funktionen im einzelnen bedeuten:

Jede Ergebnismenge, egal wie man sie angelegt hat, kann mit diesen Methoden behandelt werden. Die meisten dieser Punkte rufen ein eigenes Menü auf, das dann wiederum Hilfetexte besitzt.

Tabellen erstellen Geeignet zur direkten **Übernahme in Office-Programme** und andere Software (als "Delimiter"-Dateien) : sortierte Ergebnismengen (alphabetisch oder numerisch geordnet), und zwar wahlweise auch zur Übergabe an SQL-Systeme mit der Möglichkeit, anschließend von dort eine Erg.Menge auch wieder zurückzunehmen.
Man erhält ein eigenes Menü, zu dem es auch eine ausführliche Version mit Erklärung der Methodik gibt.

View-Technik Für **einfache sortierte Listen**, die man sofort am Bildschirm sehen kann, dann aber gleichfalls in Tabellenform exportiert werden können. Eine ***ViewListe*** ist eine vom Nutzer definierte Anzeige der Ergebnismenge. Mehr dazu: `h viewlist`
Dieser Punkt ruft ein eigenes Menü hervor, von dem es wiederum eine ausführliche Version gibt. Diese erklärt dann das Konzept und das Vorgehen.

Zählungen **Statistische Auswertung** von Feldinhalten. Z.B.: Welche Schlagwörter wurden in der Datenbank verwendet und wie oft? *Sonderfunktion:* Statistik der belegten Datenfelder und wahlweise auch der Teilfelder.

Summierung **Feldinhalte zusammenrechnen**, und zwar über eine Ergebnismenge oder die gesamte Datenbank. Außer der Summe wird auch der Durchschnittswert errechnet.

Registerabschnitt Das ist eine *Spezialfunktion:* Bringt jeden gewünschten Abschnitt aus den Registern in die Anzeige, wobei man verlangen kann, daß nur Zeilen mit mehr als x Treffern gezeigt werden. Statt einer Textliste kann auch eine ViewListe oder eine Ergebnismenge erstellt werden, die dann in derselben Reihenfolge wie im Register geordnet ist.

Volltextsuche Wenn man die Ergebnisse **nicht über die Register** oder mit Find-Menü zusammenstellen kann. Anschließend kann man damit umgehen wie mit jeder anderen Ergebnismenge, also exportieren, summieren, zählen, Tabellen und Views anlegen.
Die Variante **per FLEX** findet nicht in einem DOS-Fenster statt, dafür ist sie langsamer. Wer DOS allzu irritierend findet, kann sich hiermit behelfen.

CockPit **Bewährte DOS-Methoden** der Listenerstellung im *Classico*-System: Menüpunkte "Volltextsuche, Listen" und "Auswertungen" im Menü Routinen. *Mehr dazu:* → Kap. 6

Sortierte Listen **Dasselbe ohne DOS** : Die Parameterdateien können auch ohne CockPit ausgewählt werden; ein FLEX führt dann dieselbe Arbeit aus. Es entsteht eine Datei mit dem Namen LISTE.

XML-Ausgabe **Datensatz oder Erg.Menge mit XML-Tags ausgeben** : Mit den Feldbezeichnungen aus der CFG-Datei und/oder den Kategorienummern kann in vier verschiedenen Formen von XML-Tags ein Export produziert werden.

MARC21 Export im MARC-Format : Dabei stehen zwei Varianten zur Wahl: XML-Ausgabe und Textausgabe. Letztere kann z.B. mit MarcMaker in echtes MARC gewandelt werden.

Neuerwerbungsliste **Erg.Menge der neuen Datensätze erstellen** : Es ist dann nur das Anfangsdatum auszuwählen und die Erg.Menge dazu wird erstellt. (Die Arbeit macht `nel.flx`)

Andere Exportparameter **Wahl einer anderen Ausgabeform** : Es erscheint eine Dateiauswahl-Box, in der man sich die Parameterdatei aussuchen kann, die zum Ausgeben der Daten genutzt werden soll.

Andere Exportdatei **Name einer Datei** : Man kann frei wählen, in welche Datei die Daten auszugeben sind. Wenn die Datei schon existiert, kann man auch wählen, neue Daten anzuhängen.

Aktuellen Satz / Erg.Menge ausgeben Ausgabe erfolgt mit den eingestellten Parametern in die eingestellte Datei.

Ganze Datenbank exportieren Mehrere Möglichkeiten werden geboten, auch MARC21 und ein XML-Export.

Ganze Offline-Datei Als wäre es eine Ergebnismenge.

Kurzliste der Erg.Menge ausgeben Sie wird in die eingestellte Exportdatei geschrieben.

Exportdatei löschen **Datei beseitigen** : Wenn man sichergehen will, daß noch keine Daten in der Exportdatei stehen.

Grunddateien durchsuchen Externe Datendateien der Typen .xLG und .xDT (s. Kap. 0.2.2).

2.3 Das FLEX-Konzept

Online-Dokumentation : h flex

siehe auch: **http://www.allegro-c.de/flex/**

Vollständig und aktuell ist nur die Online-Dokumentation in ***a99***. Dieses Kapitel konzentriert sich auf das Wesentliche.
Die FLEX-Befehle bilden eine **Skript**sprache, speziell für das *allegro*-System. Fast alle Vorgänge, die man von Hand auslösen kann, sind auch über einen FLEX-Befehl ausführbar. Es gibt zwei Varianten: eine für das Dialogprogramm ***a99*** und eine andere, **Job-Sprache** genannt, für das Konsolprogramm **acon.exe**, das auch die Aufträge für den avanti-Server ausführt.
Sehr wichtig: es gibt eine **interne Variable** (kurz **iV**). Das ist ein temporärer Zwischenspeicher, der nur so lange „lebt", bis er durch einen weiteren Befehl überschrieben wird. Das geschieht z.B. durch die Befehle **`ask`**, **`date`** und vor allem **`var`**.
Die **iV** hat und braucht keinen Namen, d.h. man *sieht* sie in den Befehlen nicht, doch das vereinfacht und verkürzt viele Befehle. Man verwendet die interne Variable in der Regel sofort z.B. in einem "insert"-Befehl (z.B. `insert #98` , um den Inhalt der iV in die Kategorie #98 zu kopieren). Will man die iV etwas länger aufbewahren: mit **`ins $IV`** in eine Variable `$IV` kopieren. (Und von dort bei Bedarf wieder zurück in die iV mit dem Befehl **`var $IV.`**)
Zum Thema Variablen in FLEX: `h xcstring` eingeben. Interessant: die freien und die persistenten Variablen (ab V34)

2.3.1 So erstellt man FLEXe (Am wichtigsten ist Abschnitt D.)

A. Interaktiv (vor allem zum Testen!) : Der einfachste Weg, einen FLEX zu erstellen und auszuführen ist dieser:

1. **Eingeben und sofort ausführen lassen:** Befehlsfolge im Schreibfeld eintippen, \ als Trennzeichen, am Anfang ein x
   Beispiel: **`x message Index 1 wird gezeigt\index |1 shakesp`**
2. **Speichern und mehrfach ausführen:** Statt **`x`** kann man auch `#uXi` geben (*i*=0...9). Die Befehlsfolge wird dann in der Variable **`#uXi`** gespeichert. Ausführung mit Alt+**`i`** (z.B. **Alt+3**, um `#uX3` zu aktivieren.)
3. **Gespeicherten FLEX ändern:** Alt+r um den Reservespeicher zu sehen, Zeile `#uX3` auswählen; dann <Enter>, Befehlsfolge ändern, nochmals <Enter> zum Zurückspeichern. Mit Alt+3 wieder ausführen.

Auf diese Weise kann man sich bis zu 10 FLEX-Makros einrichten, die zusammen mit den Phrasen abgespeichert werden und somit für nachfolgende Sitzungen erhalten bleiben. Die 8 "Flip-Buttons" unten links (unter dem Auswahlfeld) lösen diese FLEXe aus. (Meistens setzt man ihren Inhalt aber in `_start.flx` oder verändert ihn inerhalb anderer FLEXe.)
Nur die FLEXe in den Variablen **`#uXi`** kann man mit der Alt-Taste auslösen, sie haben also eine Sonderstellung.
Längere FLEXe schreibt man in Dateien vom Typ `.flx` (ein Befehl pro Zeile). In einer `#uXi` stünde dann X *dateiname,* oder man startet den FLEX durch direkte Eingabe von X *dateiname.* Z.B. gibt es für die Erwerbungsfunktionen (Modell ORDA) mehrere solche FLEXe: o-bestel.flx etc.

B. Einbettung in Parameter oder Hilfetexte (→ Kap.10.2.7)
Andere Wege, FLEXe bereitzustellen, sind das Einbetten in Hilfedateien oder in die Anzeigeparameter. Das jedoch sind Aufgaben für *AllegrologInnen.* Es gibt etliche vorbereitete Parameterdateien (D-WRTF.APR) und Hilfedateien wie z.B. `input.rtf`, `catger.rtf` , `org.rtf` oder `doku.rtf`, die man als Vorlage heranziehen kann:
Geben Sie im Schreibfeld ein: `h doku.rtf` statt nur `h doku`, dann sieht man auch die darin eingebetteten FLEXe . Davon gibt es zwei Sorten:

- **Flipzeilen**, die mit '?' anfangen. Bei diesen FLEXen muß es immer zu einem `#uYi` ein `#uZi` geben, und der Inhalt des `#uYi` muß einer Zeichenfolge im Anzeigefeld genau entsprechen, und diese Zeichenfolge muß links und rechts mit dem Zeichen ` (Accent grave) markiert sein. Wird dieses angeklickt (doppelt), sucht das Programm das gleichlautende `#uYi` und führt das zugehörige `#uZi` aus.
- **Direkte FLEXe** : Mitten im RTF-Text können Strukturen vorkommen, die so aussehen:
  %=x *FLEX-Befehlskette***%** oder **%=X** *FLEX-Dateiname***%**
  Sofort beim Einlesen der RTF-Datei werden dann die FLEX-Befehle ausgeführt, und zwar an Ort und Stelle, bevor das Einlesen weitergeht. Wenn der letzte Befehl im FLEX **`var "END"`** lautet, wird das Einlesen der Parameterdatei abgebrochen und nichts wird angezeigt.

Einbetten in den RTF-Text kann man auch Kategorien und Variablen. Dazu schreibt man **`%#nnn`**, also z.B. `Titel = %#20` mitten in einem RTF-Text. In einem vorangehenden, direkten FLEX kann man diejenigen Variablen vorher präparieren, die dann im Text erscheinen sollen!

Tip zum Testen: Mit Alt+r gleich nach dem Einlesen einer RTF-Datei kann man sich anschauen, welche #uYi und #uZi im Hintergrundspeicher stehen.

C. Phrasenspeicher
Zum Dritten kann man FLEXe im Phrasenspeicher unterbringen! Wird dann die Phrase angefordert, wird statt dessen der FLEX ausgeführt.
Beispiel: wenn in der Datei `phrase.a99` diese 4 Zeilen stehen:

```
6   (steht für Strg+f; 1 ist Strg+a, 2 ist Strg+b usw. bis 26=Strg+z)
x var #uSW(1,0)\ins #uSW\var "="\ask +Suchwort?=#uSW\help\ins #uSW
7   (steht für Strg+g)
x var ">" #uSW(1,0)\ins #uSW\help
```

Dann wird mit **Strg+f** die Phrase 6 (Suchen im Anzeigefeld) und mit **Strg+g** die Phrase 7 (Weitersuchen) ausgeführt. Und zwar braucht dabei der Cursor nicht im Schreibfeld zu sein, das klappt auch, wenn er woanders ist. (s. Befehl **`help`**)
Achtung: die Werte 8, 9, 10 und 13 (Strg+h, i, j, m) dürfen nicht belegt werden, das sind die Funktionen "Backspace", "Tabulator", "Enter" und "Return". Die Werte 1, 3, 22, 24, 26 (Strg+a, c, v, x, z) braucht Windows für die Copy+Paste-Funktionen, deshalb muß man diese auch vermeiden. Phrasen können statt mit der Hand oder durch Eintragung in die Datei `phrase.a99` auch mit dem Befehl **`phrase`** belegt werden (mehr dazu: **`h phr`**)

D. FLEX- bzw. JOB-Dateien (siehe auch Übersicht der mitgelieferten FLEXe in der Hilfedatei, **`h flex`**)

Jede Folge von FLEX-Befehlen kann man in eine Textdatei schreiben, z.B. mit Notepad, Typ muß dann `.flx` sein.
Ausgeführt wird eine solche Datei namens `abc.flx` mit dem Befehl `X abc` , einzugeben im Schreibfeld.
Eine größere Zahl von FLEX-Dateien wird mitgeliefert und gehört zum Kernsystem. Diese FLEXe werden bei der Installation auf dem Unterverzeichnis FLEX des Programmverzeichnisses gelagert.
Eine FLEX-Datei wird meistens aus einer RTF-Datei heraus gestartet, siehe z.B. **`adm.rtf`**
Es ist auch möglich, eine FLEX-Datei von Hand zu starten, z.B. `kalender.flx`: Geben Sie dazu ein: **`X kalender`**.
JOB-Dateien für das Konsolprogramm `acon` sind dagegen vom Typ `.job`, z.B. `srch.job` und `update.job`. Man läßt eine solche Datei ausführen mit dem Befehl `acon -j` *jobdateiname –weitereOptionen*

E. ExFLEXe

Eine FLEX-Datei (die also eine Folge von Befehlen enthält), kann man dem Programm *a99* bzw. *alcarta* von außen übergeben. Ist `liste.flx` eine solche Datei, gibt man auf DOS-Ebene, meistens in einer Stapeldatei, den Befehl **`flex liste`**, dann wird das laufende ***a99*** oder *alcarta* die Datei ausführen. `flex.exe` ist ein Hilfsprogramm, das eigens zu diesem Zweck geschrieben wurde. Es gehört zum a99-Paket. Unter anderem konnte mit dieser Methode eine ganze WWW-Schnittstelle namens "RuckZuck" erstellt werden, die ohne *avanti*-Server und ohne Skriptsprache (Perl o.a.) auskommt. Mehr dazu erfährt man über **`h ruckzuck`** oder auch über die Web-Adresse http://www.allegro-c.de/. "RuckZuck" ist inzwischen aber nicht mehr sinnvoll.

F. AutoFLEXe

Etwas für Export-Experten:
Eine Sonderstellung haben die Variablen **`#uX:`** und **`#uX;`** : wenn man sie aus den Anzeigeparametern heraus mit einem FLEX belegt (Beginn mit **`x`** oder **`X`**), wird dieser sofort nach Fertigstellung der Satzanzeige ausgeführt und gleich danach wieder gelöscht. Im Falle von **`#uX:`** erfolgt die Ausführung nur dann, wenn gerade kein Index- und kein Ergebnisfenster geöffnet ist, im Falle **`#uX;`** aber ohne Einschränkung sofort.
Nochmal mit etwas anderen Worten: Die Anzeigeparameter haben bekanntlich die Aufgabe, einen Datensatz für das Anzeigefeld aufzubereiten. Wenn während dieser Aufbereitung nebenbei eine **`#uX:`** oder **`#uX;`** erzeugt wird, dann werden diese unmittelbar nach dem Ende der Parameter-Abarbeitung ausgeführt, d.h. sobald der Datensatz sichtbar wurde.

Tips für Entwickler

- Starten Sie immer zusätzlich die Demo-Datenbank. Sie haben dann ein eigenes Fenster, in dem Sie ständig die gerade gebrauchte Dokumentation abrufen können. Mit **`h flex`** kommt der FLEX-Einstiegstext.
- Mit **`h xa`** kommt eine alphabetische Liste aller Befehle. Jeden davon kann man anklicken, um die genaue Beschreibung zu sehen.
- Noch besser: Mit **`v flex`** kommt eine Übersicht nach Funktionsbereichen, als ViewListe eingerichtet. Wählt man eine Zeile an, erscheint im Anzeigefeld sofort die Beschreibung dazu.
- Mit den Hilfetexten auf dem HELP-Verzeichnis und den FLEXen auf dem FLEX-Verzeichnis hat man reichlich Anschauungsmaterial und wandlungsfähige Vorlagen für eigene Vorhaben. *Tip:* Wenn man einen Text sieht, dann steht im Schreibfeld dessen Name! Man ruft denselben Namen mit angehängtem ".rtf" auf, dann sieht man, was für FLEXe in welcher Weise eingebettet sind. *Beispiel*: man sieht im Schreibfeld **`h adm`**, dann ruft man **`h adm.rtf`** auf.

2.3.2 FLEX Befehle : Kleine Übersicht

Online in a99: `v flex`

Diese Liste zeigt nicht die gesamte Information zu jedem einzelnen Befehl! Sie dient vorwiegend dazu, den für eine Aufgabe passenden Befehl erst einmal zu finden oder wiederzufinden. Aktuell und vollständig ist nur die Dokumenation in a99, daher die *Empfehlung:* Um die vollständige Beschreibung eines Befehls zu sehen, nimmt man das Befehlswort, wie z.B. **`if`**, und gibt dann ein: **`h xif`** . *Oder* : **`h xa`** und das Befehlswort in der Liste anklicken, die dann kommt.
Kursive Angaben, z.B. *dateiname*, stehen für variable Befehlsteile. Die Abk. **iV** steht für die namenlose „interne Variable"
Ein *cstring* ist eine bel. Folge von Text- und Steuerzeichen sowie Feld- und Variableninhalten (mehr: `h xcstring` eingeben).
Mit `acon:` wird hingewiesen auf Varianten, die für das Konsolprogramm `acon` gelten.

	x:	**auch mit *acon* anwendbar** (insbes. f. Web-Skripte)
Abfrage an den Nutzer stellen		`ask`
Abfrage von Name und Paßwort		`ask @`
Abfrage mit Auswahlliste		`select`
Abfrage yes/no oder no/yes		`yesno` *frage*`,` `noyes` *frage*
Abfrageliste im Auswahlfeld zeigen		`show prompt`
Ablaufverfolgung (z.B. Fehlerdiagnose)		`flow 0/1/2`
Adresse des Datensatzes N feststellen		`get adr N`
Aktuellen Satz in den Hintergrund kopieren		`copy reserve`
Aktuellen Satz zur History List ergänzen		`set h`
Aktuellen Satz zu den Lesezeichen ergänzen		`set b`
Aktuellen Satz zur Erg.Menge hinzufügen		`find /`
Aktuellen Satz aus Erg.Menge entfernen		`find -`
ANSI<-->ASCII Codewandlung des iV-Inhalts	x	`ansi / ascii`
Anzeigebefehle		
Anzeige ausdrucken		`print`
Anzeige einer Dateiauswahlbox		`fnam` *bez|typ*
Anzeige im Anzeigefeld aktualisieren		`disp`
Anzeige im Auswahlfeld wechseln		`show`
Anzeige Inhalt speichern als txt oder rtf		`file` *dateiname*
Anzeige scrollen zu einer best. Zeile		`display f`*num*
Anzeigefeld zum Editieren benutzen		`file x\read rx`
Anzeigefeld auf "big" setzen		`set db`
normal setzen		`set dn`
umschalten in den anderen Zustand		`set d`
Anzeigeparam./Abschnitt durchlaufen		`display, deposit`
Anzeigeparameter wechseln		`display p` *name*
Arbeitsverzeichnis-Name		`var W`
ändern des aktuellen Arb.Verz.		`set W` *pfadname*
ASCII- oder ANSI-Textdatei anzeigen lassen		`help` *name*
Arbeitstage weiterrechnen (z.B. f. Ausleihfrist)	x	`Wday +` *tage*
aresqa-Fenster öffnen, Liste anzeigen		`aresqa` *dateiname*
Überschrift setzen		`set a` *überschrift*
Artikelprüfung (iV beginnt mit Artikel?)	x	`if Art ...`
Aufruf anderer Datenbank (FLEX senden)		`flex` *name*
Aufruf eines anderen FLEX		`exec X` *dateiname*
Aufruf FLEX über eigenen Menüpunkt		`menu`
Ausführen ab `#-k` in Anzeigeparametern		`deposit k`
Auswahlmenü anbieten mit aresqa		`Aresqa`
mit ViewListe		`View`
mit select (Pulldown-Liste)		`select`
Bedingungen prüfen	x	`if` *bedingung befehl*
Beenden des FLEXes	x	`end`
Beenden des Programms (ohne Rückfrage)		`STOP`
Blättern rückwärts (Back-Button)		`show <`
Blättern vorwärts (Forward-Button)		`show >`
Bookmarks anzeigen		`show bookm`
Bookmark für aktuellen Satz setzen		`set b`
Button [Wechseln]		`undo`
Clipboard --> iV / iV --> Clipboard		`ccopy / cpaste`
Codeumwandlung ASCII <-> ANSI der iV	x	`ansi / ascii`
Codeumwandlung UTF-8 -> interner Code	x	`xcode u`
cstring = codierte Zeichenfolge	x	`var / wri` *cstring*
Datei-Befehle		
Datei: eine Anzahl Bytes lesen		`fetch` *number* `/ ^`*code*
Datei: eine Zeile lesen	x	`get / get #xyz` (acon: nur `get`)
Datei: Größe in Bytes	x	`fsize` *dateiname*

Datei: Letztes Aktualisierungsdatum		ftime *dateiname*
Datei : Datensatz einlesen (aus .ALG, .ADT)	x	fetch record
Datei e.adt lesen (Erfassungshilfe)		read
Datei Existenz prüfen	x	fsize *name*\if no ...
Datei extern.dat lesen		read extern
Datei ins Anzeigefeld holen		help
Datei kopieren / löschen		fcopy *quell ziel* / delete *dateiname*
Datei öffnen zum Lesen mit get / fetch	x	open *dateiname*
Datei öffnen zum Schreiben mit write	x	open x *dateiname bzw. exp f name*
Datei schliessen	x	close / close x
Datei umbenennen		rename *altname neuname*
Dateiauswahlbox bringen		fnam *bez|typ*
Dateien eines Typs/Namensmusters auflisten	(x)	fnam |*typ* / fnam >*typ* / dir *namensmuster*
Dateiinhalt in die Ausgabedatei kopieren	x	write F*filename*
Dateiinhalt in die interne Var. Kopieren	x	var F*filename*
Dateinummer für Eingabe wechseln	x	input *n bzw.* acon*: set nnummer*
Daten in Bearb. – anzeigen / löschen		show off / erase off
Datenbank erster / nächster / letzter Satz	x	first # / next # / last #
Datenbank gesperrt? Dann ...	x	if tbl ...
Datenbank in Benutzung? Dann ...	x	if usr ...
Datenbank insgesamt satzweise durcharbeiten	x	first # / next #
Datenbank insgesamt exportieren		export database
Datenbankname	x	var B
Datenfeld --> siehe Kategorie		
DatenFont und Zeichensatz ändern		set D*dataFont=CharSet*
Datensatz in der Auswahlliste neu zeigen		show rec
Datensatz laden ohne Erg.Menge zu bilden	x	f1nd *suchbefehl* / f1nd #*interneNr*
Datensatz sperren/freigeben	x	set lock/unlock
Datenverzeichnis: Name	x	var D
Datenverzeichnis: Dateinamen auflisten	x	dir *namensmuster*
Datum des Tages -> iV	x	date b *bzw.* var #dts
Datumsrechnungen	x	day / Day / Wday (*acon: nur Wday)*
DOS-Fenster öffnen, Programm ausführen		Dos / dos *Befehl*
Druck des Anzeigefeldes		print
Druckform des Satzes zeigen		show print
Druckform der Erg.Menge zeigen		show Print
Druckseite einrichten		set P
Editor-Fehlermeldungen aus-/einschalten		set e0 / set e1
Eigenschaften feststellen/auswerten	x	var … (*cstring*)
Einbettung eines FLEX in Hilfetexte		%=x *flexbefehlsfolge*% oder %=X *flexdateiname*%
Eingabe vom Nutzer abfragen		ask …
Eingabe yes/no oder no/yes abfragen		yesno, noyes
Einstellungen aller Art -> set-Befehle	x	set oder switch
Environment-Variable lesen	x	get env *name*
Ergebnismengen-Befehle		
Aktuellen Satz zur Erg.Menge hinzufügen		find /
Aktuellen Satz aus Erg.Menge entfernen		find -
Ergebnismenge bilden ohne Anzeige	x	find *suchbefehl*
Erg.menge bilden, anzeigen, FLEX beenden		Find *suchbefehl*
Erg.menge bilden, anzeigen, iV belegen		choose *suchbefehl*
Erg.menge bilden, nur ersten Satz laden	x	f1nd *suchbefehl*
Ergebnismenge durcharbeiten	x	next / prev
Ergebnismenge expandieren (SR-Methode)	x	find &
Ergebnismenge: Druckform anzeigen		show Print
Ergebnismenge: erster/letzter Satz	x	first / last
Ergebnismenge: bestimmten Satz laden	x	find r*num*
Ergebnismenge: den Namen ändern		set R*name*
Ergebnismenge: Kurzliste in Anzeige kop.		display list
Ergebnismenge: Kurzliste exportieren		export brief (acon: list)
Ergebnismenge reaktivieren		find s*num* / :*name*
Ergebnismenge schließen/löschen		close res / r*num*
Ergebnismenge sortieren	x	order MP
Ergebnismenge speichern (geänderte S.)		save res
Ergebnismenge verknüpfen		find /.. -.. *..
Ergebnismenge wieder öffnen		find s*num* / :*name*
Ergebnismenge zeigen, im Erg.Fenster		show list
Ergebnismenge zeigen, in Druckform		show List

Ergebnismenge zur Familie bilden/anzeig.	x	`family / Family` (acon: nur `family`)
Satznummern der aktuellen E-Menge ausg.		`wri ixk-j`
Ergebnismengen, Auswahlliste zeigen		`show sets`
Liste der aktuellen E. in die Anzeige		`display list`
Liste aller Erg.Mengen in Anzeigefeld		`display sets`
Ergibtanweisung für iV	x	`var`
Feld aus Hintergrundsp. holen		`transfer #nnn`
iV-Inhalt in #xyz kopieren	x	`insert #xyz`
Ersetzungen im aktuellen Datensatz	x	`_abc_xyz_`
in einem Datenfeld	x	`*#nnn_abc_xyz_`
in der internen Variablen	x	`insert _abc_XYZ_`
Export: Ausgabedatei festlegen	x	`export f` *name*
Export: Ausgabedatei schließen		`close x` (acon: `open f` *name*)
Export: Parameterdatei laden	x	`export p` *name*
Exportieren Ergebnismenge	x	`export set`
Exportieren gesamte Datenbank		`export database`
Exportieren aktueller Satz	x	`export`
via Exportparameter		`Export`
via PrintParameter	x	`export fam`
samt verknüpften Untersätzen	x	`export Head/Foot`
Exportieren: Kopf-/Fußabschnitt ausführen		`export view`
Exportieren der ViewListe	x	`export -x` [*nur acon*]
Exportparameter, Abschnitt ausführen		Hilfsprogramm `flex.exe`
Extern (von außen) einen FLEX starten (ExFLEX)		`extern`
Externes Editieren des aktuellen Satzes	x	`family / Family`
Famile verknüpfter Sätze als Erg.Menge	x	`var Err / wri Err`
Fehlermeldungstext ausgeben		`set Err`
Fehlermeldungstext und -zustand rücksetzen	x	`if …`
Fehlerprüfungen a99	x	`if error=…`
Fehlerprüfungen in acon		`activate`
Fenster in den Vordergrund bringen		`set wn / wm`
Fenster auf Normal- / Maximalgröße setzen	x	`f1nd` *suchbefehl*
Find-Befehl ohne Erg.Menge zu bilden	x	`end`
FLEX bzw. Job beenden		`flex` *dateiname*
FLEX senden an anderes a99		`flip` *ixyz*
Flip-Buttons beschriften		`set f1 / set f2`
Flips mit Einzel- bzw. Doppelklick aktiv.		`activate i (1,2,3,4)`
Fokus in ein bestimmtes Feld setzen		`form i / form` *name*
Formular anzeigen		`addform` *dateiname*
Formulare hinzuladen		`set F1 / set F0`
Formularmodus direkt / indirekt	x	`var JR0` (vorher `z=i` Anzahl der Stellen setzen)
Führende Nullen (iV links auffüllen)		`onf2.flx` usw.
Funktionstasten mit FLEXen belegen		`if diff …`
Geänderter Satz? Dann …	x	`if deleted …`
Gelöschter Satz? Dann …	x	`if Lock …`
Gesperrter Satz? Dann …	x	`if hiera …`
Hierarchischer Satz? Dann …	x	`erase sub`
Hierarchischen Untersatz löschen		`help` *dateiname*
Hilfetext anzeigen lassen		`show reserve`
Hintergrundspeicher im Auswahlfeld zeigen		`display reserve`
Hintergrundspeicher in die Anzeige kopier.	x	*einzelne:* `var ""\ins #uxy` */alle:* `var ""\ins #ux~`
Hintergrundvariable #uxy löschen		`transfer #nnn`
in Arb.speicher kopieren		`show hist`
History List anzeigen (Erg.Mengen)		`janas` *dateiname*
HTML-Datei zeigen (Zusatzprogr. JanaS)	x	`if` *bedingung befehl*
if … : Bedingungsprüfungen	x	`if not` *bedingung befehl*
if not … : negative Bedingungsprüfung	x	`include` *dateiname*
include: andere Datei einfügen	x	`ixadd / ixdel` *satznr eintrag*
Indexeintrag einfügen / löschen		`button i`
Indexfenster öffnen		`index p` *name*
Indexparameterdatei laden (zum Testen)		`var V`
INI-Datei: Name	x	`var` *cstring*
Interne Variable (iV) anlegen	x	`insert #nnn`
iV in #nnn kopieren		`show IV/show +IV`
iV ins Anzeigefeld schreiben/anhängen		`show iV`
iV im Schreibfeld anzeigen		`show Bin`
iV binär anzeigen (Steuerzeichen rot)		

Inhalt manipulieren	x	`var` (*M-Befehle*)
Sondervariablen, Liste		`h xcstring / X cstring`
Interne Zahl (iZ) für Berechnungen	x	`Z= + - / * / var Z`
Interner Zähler (Ganzzahl), z.B. für Schleifen	x	`z= + - / * / var z`
Internet-Daten besorgen (in die iV kopieren)	x	`get I`*`url`*
Ja/Nein-Frage an den Nutzer		`yesno, noyes`
JanaS-Zusatzprogramm starten		`janas` *`url`*
Kategorie bearbeiten (lokale Ersetzung)	x	`*#nnn_abc_xyz_`
Kategorie einfügen	x	`ins #nnn`
Kategorie löschen	x	`var ""\ins #nnn`
Kategorie aus Hintergrund holen		`transfer #nnn`
Kategorieliste (CFG) erstellen		`katlist` (ergibt Datei `katlist.asy`)
Kategorieliste (erlaubte Felder) zeigen		`show cfg`
Kommentar (in FLEX-Datei)		Leerzeichen am Zeilenanfang bzw. // bei acon
Kurzliste Erg.Menge ins Anzeigefenster		`show list`
Kurzliste Erg.Menge in die Ausgabedatei	x	`exp brief / acon: list`
Kurzzeile des nächsten Satzes holen	x	`next r`
Leerzeichen, mehrfache in der iV löschen	x	`spaces`
Lesezeichen alle anzeigen		`show bookm`
Lesezeichen für aktuellen Satz setzen		`set b`
Liste aller Erg.Mengen in Anzeige kopieren		`display sets`
Listendatei anzeigen+bearbeiten		`aresqa` *`dateiname`*
LOG-Datei: letzte nnn Bytes als Erg.Menge		`find $nnn\show list`
Löschen des aktuellen Satzes	x	`erase`
Löschen einer Datei		`del name`
Löschen der "Daten in Bearb."		`erase off`
Löschen einer Ergebnismenge	x	`close res / r`*`num`* ` / `*`acon:`*` find *new`
Manipulationsbefehle für die int.Variable	x	`var #xyz(...) / h mbtest` (Beispiele)
Mitteilung anzeigen (mit [OK]-Button)		`message` *`botschaft`*
mit Warndreieck		`message !` *`botschaft`*
fuer n Sekunden (beendet sich ohne OK)		`Message n,` *`botschaft`*
Nummernvergabe		`h nummer` [Hilfetext]
Offline-Datei anzeigen		`show offline`
Offline-Datei: erster/letzter Satz		`first off/last off`
Offline-Datei löschen		`erase off`
On-FLEXe (Auslösung bei bestimmten Buttons		`... und Funktionen)`
Paßwort abfragen (Eingabe nicht sichtbar)		`ask @`
Periodische Aktion programmieren		`slice n=m`*`FLEXdatei`*
Persistente Variable einfügen / verwenden	x	`insert &`*`name`* ` / var &`*`name`*
Phrase belegen bzw. verwenden		`phrase i text`
Phrasendatei einlesen		`get phr` *`name`*
Phrasendatei speichern		`save phr` *`name`*
Phrasendatei: Name		`var v`
Phrasenliste anzeigen		`show phras`
Pipe öffnen	x	`pipe` *`befehl`* (dann mit "get" daraus lesen)
Primärschlüssel für update ändern	x	`set pX`
Programm starten (anderes)	x	`call, cAll, Call, Call` (acon: nur `call`)
`Call`: unabhängiges Fenster		`auch: Dos`
`cAll`: läuft nur in Taskleiste (nicht sichtbar)		`auch: dos`
Programm stoppen (abrupt, ohne Frage)		`STOP`
Programmverzeichnis: Name	x	`var P`
Rechenbefehle (Formelauswertung)	x	`eval` *`formel`* bzw. `Z=+-/* / z=+-/*`
Register aufblättern		`index` *`key value`*
Registerabschnitt anzeigen/exportieren	x	`qrix, Qrix`
Registereinträge d.aktuellen Satzes zeigen		`show keys`
-- in die iV kopieren	x	`var sK / var sk`
Reservespeicher: siehe -> Hintergrundspeicher		
Satz anlegen (ganz neuen Satz)	x	`new`
Satz kopieren (aktueller Satz -> Neusatz)	x	`copy`
Satz über interne Satznr laden	x	`find #N`
Satz mit der letzten Satznummer laden	x	`find #0` oder `var "#" t\find`
Satz (nächste/vorige Satznummer) laden	x	`next #, prev #`
Satz löschen	x	`erase`
Satz laden (wechseln) nach choose		`load`
Satznummern der akt. Erg.Menge ausgeben	x	`wri ixk-j`
Satznummernliste aus Datei einlesen	x	`read set` *`name`*

Schreiben in Export-Ausgabedatei	x	write *cstring*
Schreiben in die Fehlerausgabe (stderr)	x	Write *cstring* *NUR* acon
Schreiben in iV (interne Variable)	x	var *cstring*
Schleife mit Zähler	x	if z<…
Schleife abbrechen durch Taste <Esc>		keychk \ if yes …
Schlüssel d. akt. Satzes zeigen / schreiben	x	show keys / var sK, var sK
Set-Befehle (Setzen von Eigenschaften)	x	set …
Eingabedaten sind ASCII/ANSI	x	set c0/c1
Schriftgrößenänderung im Anzeigefeld	x	set d+/d-
DatenFont und Zeichensatz ändern		set D*dataFont=CharSet*
Environmentvariable setzen		set env name=value
Kopiermodus Index->Eingabe (klein/groß)		set L0/L1
Sicherungskopie: Verzeichnisname setzen		set C
Sondervariablen für 'var' und 'write'		*Siehe* h xcstring
Sortiermodus und -position setzen	x	set oMP
Sortierung Ergebnismenge	x	order a/d*pos*
Sortierung Textzeilen in der iV	x	sort a/d*pos*
Speichern mit Rückfrage		Put, Put new
Speichern ohne Rückfrage	x	put, put new
Speichern, auch wenn Satz gesperrt		put unlock
Speichern, dann Satz gesperrt lassen	x	put lock (nur acon)
Speichern, auch wenn .TBL gesperrt		put free
Speichern aller neuen/geänderten Sätze		save edit
Speichern aller geänd.Sätze der Erg.Menge		save result
Speichern aller Offline-Sätze		save off
Sperren der ges. Datenb. gegen Schreiben	x	set tbl loc/fre
Sprung zu einer Marke :label	x	jump *label*
Stammsatznummern durch Klartext ersetzen	x	a99: exp R, acon: set ai
Stopwortprüfung (ist iV-Inhalt ein Stopwort?)	x	if Stop …
Suchen im Anzeigefeld		help =abc / help >abc
Suchen mit „Regulären Ausdrücken“	x	srx
Tabellendatei anzeigen+bearbeiten		aresqa *dateiname*
Taste gedrückt? Esc-Taste		keychk \ if yes …
Taste gedrückt? (irgendein Buchstabe)		keychk x \ if "a" …
Text einlesen ohne Steuerzeichen		fetch ^T / ^A
Timer setzen (regelmäßige autom. Aktion)		slice n=m*FLEXname*
Tastatur abfragen: wurde <Esc> gedrückt?		keychk \ if yes …
Tastatur abfragen: wurde a/b/c… gedr.?		keychk x\if "a" …
Teilfeld einfügen / löschen	x	insert $a-#xyz
TEMP-Verzeichnis (Name)		var M
Überschriften der Kurzanzeige setzen		set iZtext
UIF-Zeile ausgeben		var ci (0<i<500)
UIF-Zeile verändern		set BiText (0<i<500)
Umcodieren des iV-Inhalts	x	xcode ab (a=i,d,x,u b=p,q)
Umschalten der Ausgabe-Codierung (avanti)	x	switch coding 0 / 1 / 2
Umschalten zw. Erg.Menge/Familie (avanti)	x	switch fam / res
Umschalten zw. Exp. 1 und Exp. 2	x	switch dow 1 / 2
Umschalten zw. Satz 1 und Satz 2	x	switch object 1 / 2
Unicode-Modus für insert ein/aus	x	set U1 / U2 / U0
Unicode-Modus für write ein/aus	x	exp wX / exp w0
Unterbrechung der Bearbeitung für 1 Sek.	x	sleep 1000
Unterprogramm aufrufen / beenden	x	perform *label* / return
Untersatz (hierarchisch) einschalten	x	sub #01 nnn
Upload (neue Daten einspeisen)	x	upload *dateiname*
Update: Modus setzen	x	set u*xyz*
Primärschlüssel bei #-X bilden	x	set pX
Vorgang starten	x	update *dateiname*
Variable, interne		h xiv / h xiz
V14-Ersetzungen: siehe Stammsatznummern	x	set ai / exp Ref
V14-Ersetzungen: hat die Datenbank welche?	x	if v14 ...
Verknüpfter Satz? dann ...	x	if fam ...
... Hauptsatz? dann ...	x	if main...
... Untersatz? dann ...	x	if sub ...
Verschlüsseln des iV-Inhalts	x	crypt
Version anzeigen / als Variable nehmen	x	show ab (*nur a99*) / var m
Verzeichnis (Pfadname) neu anlegen		mkdir *pfadname*
Verzeichnis setzen für Datenbank-Kopie		set C

Verzeichnisliste erzeugen		`fnam >.`*typ*
Verzeichnisse zur Auswahl anbieten		`fnam |.`*typ*
Viewliste öffnen / +anzeigen		`view / View`
Viewliste schließen		`close view`
Viewliste: Löschen von Zeilen erlauben		`set v1`
Viewlisten-Überschriftszeilen setzen		`set i`*Ztext*
Volltextvergleich im aktuellen Datensatz	x	`if _xyz_ ...`
Volltextsuche im akt. Satz mit Umcodierung		`find _xyz_`
Volltextsuche in der akt. Ergebnismenge		`find +xyz`
Von/Bis-Suche mit find-Befehl	x	`find `*key*` A---B`
Vorher angezeigte Daten neu anzeigen		`show hist`
Warten 1 sec	x	`sleep 1000`
Wertzuweisung für interne Variable (iV)	x	`var `*cstring*
aus Hintergrundspeicher		`transfer #nnn`
für Feld oder Variable #xyz	x	`insert #xyz`
Wichtigste Befehle zum Ausprobieren		`h flexw`
XML-Ausgabe eines Datensatzes	x	`xml 0/1/2/3`
Zahl setzen (iV -> interne Zahl)	x	`=`
Zahl berechnen (iZ +-/* iV)	x	`+ - / X`
Zeile aus der "open"-Datei lesen	x	`get #xyz / get iV`
Zeitüberwachung ein/aus		`slice n=m`*FLEXname*
Zwischenablage-Funktionen		`ccopy / cpaste ...`

2.4 Fremddaten mit *a99*: Nur ein Knopfdruck

Empfehlung: : in *a99* **`h fremd`** eingeben

Zu den beliebtesten Eigenschaften von PRESTO gehörte es, daß man mit Alt+a auf eine zweite (und dritte) Datenbank umschalten konnte und daß in dieser sofort dieselbe Registerstelle aufgeschlagen wurde. Mit *a99*/*alcarta* erreicht man mit der sog. "Schnellkopplung" einen ähnlichen Komfort.
Viel wichtiger aber: Es gibt auf dem `fremd`-Menü schnelle Direktimporte von DNB und GBV ohne Umwege.

2.5 Einrichtung des Windows-Systems

Online-Hilfe: `h inst`

Neueste Version besorgen: **http://www.allegro-c.de/newvers.htm**

Wir stellen hier alle evtl. zu beachtenden Einzelheiten zur Installation des gesamten *allegro*-Pakets zusammen.
Wichtig: Eine vollständige, kommentierte Checkliste aller zugehörigen Dateien ist unter dem Namen `filelist.rtf` im *a99*-Paket enthalten. Diese Checkliste abrufen: in a99 `h filelist` eingeben.
Man installiert am besten mit **`inst-all.exe`** in einen neuen Ordner oder in das vorhandene ***allegro***-Programmverzeichnis. Dieses Programm findet man auf der CD (Schlüsselzahl nötig), aber auch an der oben angegebenen Web-Adresse).

Empfehlung
Auf jedem PC, auf dem ***alcarta*** oder ***a99*** laufen sollen, die Schriften (.TTF Dateien) installieren (über das Windows-Menü "Start | Einstellungen | Systemsteuerung | Schriften". Die gesamte *Software* kann zentral auf einem Fileserver liegen und von dort gestartet werden, nur die *Schriften* müssen auf jedem PC installiert werden, sonst sieht man die Sonderzeichen nicht. Durch die Schriftdateien wird vollständige Kompatibilität zum DOS-System hergestellt (Zeichensatz OSTWEST.FON).
Jetzt aber Punkt für Punkt! Als **Beispiel** nehmen wir folgendes an: (setzen Sie hier Ihre eigenen Werte ein)

Konfiguration	**`k`**	d.h. Ihre Datenbank arbeitet mit k.cfg oder `$k.cfg` (Standard: `a`)
Datenbank	**`xyz`**	d.h. es gibt eine Datei `xyz.kpi` auf Ihrem Datenbankverzeichnis (Standard: `cat`)
Verzeichnis	**`d:\data`**	hier liegt Ihre Datenbank (Standard: `c:\allegro\katalog`)

0. INI-Datei

Vorlage: Datei **`a99.ini`**

Jede Datenbank braucht eine solche, es kann aber mehrere mit verschiedenen Einstellungen für dieselbe Datenbank geben. Man kopiert dazu die Vorlage **`a99.ini`** auf `xyz.ini` und ändert darin mindestens folgende Befehle (und mehr als diese vier braucht man nicht unbedingt!):

```
[General]
Konfiguration=k      (Wenn diese Zeile fehlt: default a)
DbName=xyz
DbDir=d:\data
access=9             (volle Schreibberechtigung in a99. Nur Katalogisieren: 3)
```

Bei Mehrplatzbetrieb muß jeder User auf einem eigenen Verzeichnis starten, eine INI-Datei kann aber, wie alle Dateien, zentral liegen. Auf dem Userverzeichnis werden einige temporäre Dateien und die Phrasendatei PHRASE.A99 untergebracht.

1. IndexParameter (Standard: Datei CAT.API)

1.1 Symbolische Registernamen (Kap.10.2.1.3)

Vorhanden sein müssen Zeilen für die symbolischen Registernamen und Restriktionen (wenn vorh.) nach diesem Schema:

```
I PER 1 "Personennamen"
I DIS 1D. "Dissertationen: Ort,Jahr"
...
R ERJ r1 "Erscheinungsjahr"
```

Es kann mehr symbolische als wirkliche Register geben! Siehe Register DIS: die Einträge stehen im Register 1 mit dem Präfix "`D `" vor der Angabe `Ort,Jahr`, also z.B. "`D berlin,1991`". Gebraucht werden diese Bezeichnungen für die Find-Befehle und für das Find-Menü, die man mit dem [Fernglas]-Button erscheinen läßt. (*Achtung:* für Register 10 '`:`' statt 10)

1.2 Überschriften der Register, der Titelanzeige und der Kurzanzeige (Kap.10.2.1.3)

Das sind die Zeilen mit | am Anfang, z.B.

```
|1="1 : Namensregister (Literatur von und über ...)"
```

Solche Zeilen waren schon für PRESTO wichtig und werden dort über den Registern angezeigt.
Zeile `|0="..."` ist für die Ergebnisliste (Kurzanzeige), Zeile `|a="..."` ist die Überschrift des Katalogs.

1.3 Tabellen für die Umcodierung ASCII←→ANSI und Unicode->ASCII

An das Ende der Indexparameter `xyz.kpi` schreibt man diese Befehle:

`to` zum Laden der Tabelle **`o.apt`** (s.a. Anh.E: Zeichensatz)

`tucodes` zum Laden der Tabelle **`ucodes.apt`** (Wandlung UTF-8 -> ASCII)

Man muß aber eine eigene `o.kpt` und `ucodes.kpt` schaffen, wenn man nicht OSTWEST.FON verwendet oder nicht den PC-Satz 437 (siehe Kommentar in `o.apt`). Wenn die Datenbank selbst ANSI-codiert ist, oder bei den Sinologischen Anwendungen, verwendet man eine leere `o.apt` und legt sie aufs Datenverzeichnis.

2. Hilfs-Parameterdateien (werden mitgeliefert)

`ad-aw.kpt`	ASCII → ANSI Umcodierung für Titelanzeige (normalerweise Kopie von **`ad-aw.apt`**)
`d-rtff.kpt`	RTF-Attribute für die Anzeige (incl. Farben)
`ucodes.kpt`	Tabelle f. Unicode-Wandlung in ASCII (Kopie von **`ucodes.apt`**), einbinden in Indexparameter
`e-w.kpr`	Export mit Kategorienummern (am einfachsten: Kopie von **`e-w0.apr`**)
`p-w.kpr`	Druck in Listenform (am einfachsten: Kopie von **`p-w0.apr`**)

(In den Dateien `ad-aw.apt` etc. stehen Kommentare). Die ersten drei muß man modifizieren, wenn man nicht mit dem OSTWEST-Zeichensatz arbeitet, sondern z.B. mit Pica oder ANSI (dem Windows-Originalzeichensatz).

3. Anzeigeparameter (Standard: Datei `d-wrtf`, Alternative: `d-krtf.apr`

... für die Darstellung der Datensätze im Anzeigefeld.
Gebraucht wird dafür eine Datei `d-wrtf.kpr` (Wenn sie fehlt, wird `d-w0.apr` genommen)
Meistens genügen wenige Änderungen in den DOS-Anzeigeparametern, die normalerweise unter `d-1.kpr` vorliegen.
Man macht davon eine Kopie namens `d-wrtf.kpr`. Die wichtigen Stellen sind dann folgende:

3.1 Grundwerte

```
zl=80     Zeilenlänge (mehr oder weniger möglich, je nach Schriftart)
zm=0      kein Seitenumbruch
fl=0      kein Kartenumbruch
dx=1      Zeichen-Umcodierung einschalten
 ke=""     diese Angabe vermeiden (unter DOS sowieso ohne Wirkung)
```

3.2 Einbindung vonTabellen

Am Ende der Datei `d-wrtf.kpr` sollten diese 3 Zeilen stehen (siehe 2.):

`td-rtff`	`d-rtff.kpt` laden (enthält Farbbefehle (Zwischenteile 71-78) und Befehle für Schriftattribute)
`tad-aw`	`ad-aw.kpt` laden (enthält Zeichencodes ASCII → ANSI für die Anzeige)
`to`	`o.kpt` laden (enthält Hilfstabelle ASCII ←→ ANSI für Auswahlfeld, Kurzanzeige etc.)

3.3 Abschnitt für Alternativanzeige mit F5, also Anzeige im Internformat:
Folgenden Abschnitt baut man in seine Anzeigeparameter `d-wrtf.kpr` ein:

```
#-(
#t{ s0 &0 #4 }    hier 5 statt 4 wenn 3stelliges Schema!
##                oder
    #L              alternativ: Anzeige mit den Labels aus der CFG
#t{ C }
#+#
```

In `d-wrtf.apr` kann man sehen, wie dies noch zu erweitern ist, wenn man in der Alternativanzeige keine V14-Ersetzungen sehen will. Man kopiert sich dann am besten den Abschnitt von dort.

3.4 Zwischenteile
Wenn darin oder in **indirekten** Prä-/Postfixen (`p{...}` oder `P{...}`) Umlaute vorkommen, erscheinen diese falsch.
Abhilfe: solche Umlaute in der `d-wrtf.kpr` durch ANSI-Codes ersetzen (z.B. mit NotePad oder WordPad).

3.5 Schriftart
Wenn eine spaltenrichtige (nicht-proportionale) Anzeige gewünscht wird, baue man diese Zeile ein:

```
as=" { \f1 "
```

dann wird `allegro Lucida` als Schriftart genommen (= a-lucida.ttf). Die zu verwendenden Schriftarten lassen sich in `disphead.rtf` aber individuell vorgeben und im Text jeweils mit Befehlen wie **`#t{ "\fn " }`** einstellen und danach mit **`#t{ "}" }`** abschalten, wobei **n** die Nummer eines der in `disphead.rtf` definierten Fonts ist.

4. Druckparameter [nur nötig, wenn der Druck anders als die normale Anzeige aussehen soll]
Als Default wird nach einer Datei `p-w.kpr` gesucht (Befehl `PrintParameter=` in der .INI). Die Druckparameter müssen eine Ausgabe mit RTF-Attributen machen, denn es erfolgt zuerst die Anzeige im Anzeigefenster, das Drucken dann mit dem [Printer]-Button. Im Anzeigefenster kann man schreiben, deshalb sind dort vor dem Druck noch Änderungen möglich! Als Druckparameter kann man seine Anzeigeparameter nehmen, minus Flips. Einzubinden: `p-d.kpt` und `p-rtf.kpt` (Kopien von p-d.apt und p-rtf.apt).

5. Hilfetexte

Dazu gibt es eine eigene Anleitung: **`h help`** eingeben im Schreibfeld, dann kommt sie. Für die eigene Datenbank empfiehlt sich, mindestens folgende Dateien anzulegen (obwohl die Programme ohne diese funktionieren):
DbName`eger.rtf` mit WORDPAD anzulegen (Vorbild: `catger.rtf` bei der Demodatenbank)
DbName`eng.rtf`
`ha_inger` ASCII-Dateien für die einzelnen Register, n=1..9

Mit `h help` erfahren Sie auch, wie man Flips, Variablen und FLEXe in Hilfedateien einbaut. Einführung: Fortbildung 13.4. Damit hat man umfangreiche Möglichkeiten zum Erstellen von Nutzerunterstützungen.

6. Start von a99 und alcarta

Es gibt mehrere Möglichkeiten, die Programme aufzurufen, hier gezeigt für ***a99***. In welchem Ordner *`ProgDir`* die Programme `a99.exe` und `alcarta.exe` liegen, ist dabei nicht wichtig.

`ProgDir``\`**`a99`**	Man kann nach dem Start eine INI-Datei suchen und auswählen
`ProgDir``\`**`a99`** *`name`*	*`name`*`.ini` wird geladen (Diesen Befehl kann man in eine Desktop-Verknüpfung einbauen)

Mit **`a99a`** statt **`a99`** startet man eine mehr Windows-typische Variante des Programms. Diese hat kleinere Buttons, alle in einer Leiste oben unter dem Menü, aber alle Funktionen können mit der Alt-Taste genauso ausgelöst werden wie im Programm ***a99***.

7. Noch mehr zum Thema Hilfe

Schon für PRESTO und APAC kann man zahlreiche Hilfedateien anlegen, die dann abhängig vom Kontext bei Druck auf F1 zum Vorschein kommen (Kap.0.3). Jetzt geht es noch weiter: die Hilfetexte können wiederum Flips (darunter auch FLEXe) enthalten, die z.B. blau unterstrichen erscheinen und wie bei HTML-Dokumenten, allerdings mit Doppelklick, eine weitere Datei hervorrufen – oder aber eine Ergebnismenge oder einen Registerabschnitt ansteuern.
Standort der Hilfedateien: Es besteht die Möglichkeit, alle Hilfedateien auf ein eigenes Verzeichnis zu legen. Dieses muß HELP heißen und am Programmverzeichnis hängen. Die Verzeichnisreihenfolge für das Suchen der Hilfedateien ist dann
`Lokal / DbDir / ProgDir\HELP \ ProgDir\FLEX / ProgDir`, also in sinnvoller Weise anders als bei Parametern: zuerst Lokal, dann die anderen Verzeichnisse, d.h. nutzerspezifische Hilfe hat Vorrang.
Mit den Buttons [Back] und [Vorward] (links oben) blättert man übrigens in den vorher aufgeschlagenen Hilfetexten vor und zurück, wie man es mit Browsern gewöhnt ist.
Noch eine Möglichkeit: eigene Datensätze mit Hilfetexten als Teil der Datenbank. Auch solche können dann wieder Flips zu anderen Hilfethemen enthalten!

2.6 Database Publishing mit *alcarta*

Wie schon das "alte" Programm APAC, so kann man auch, das ist ausdrücklich freigestellt, das OPAC-Programm ***alcarta*** auf CD-ROM mitliefern, wenn man eine Datenbank veröffentlichen möchte. Dafür gibt es schon mehrere Beispiele, u.a. den OPAC der UB Braunschweig, der auf der CD enthalten ist, auf welcher die ***allegro***-Software ausgeliefert wird. Solche CD-Produkte kann man als Vorlage nehmen und die Dateien modifizieren.
Das Kap. 0.8 informiert darüber, welche Dateien zu einer Datenbank gehören, also auch mitgeliefert werden müssen, wenn man sie veröffentlichen will. Diese Angaben gelten auch für den Fall, daß die Benutzung mit ***alcarta*** erfolgen soll. Es ist zu empfehlen, daß man eines der existierenden Beispiele studiert, und die Liste der Dateien, die dazugehören. Zusätzlich gibt die Datei FILELIST.RTF Auskunft über jede einzelne Datei, die vorkommen kann oder muß.

2.7 *a99* als Server [veraltete Methode]

Online-Doku in ***a99 :*** `h ruckzuck`

FLEXe können, wie oben schon erwähnt, auch zur Fernsteuerung des Programms ***a99*** eingesetzt werden, d.h. von außen und aus anderen Programmen heraus kann man ***a99*** dazu veranlassen, Daten auszugeben oder auch in die Datenbank einzuspeisen. Davon wurde Gebrauch gemacht in der ersten Web-Schnittstelle namens ***RuckZuck***, die es möglich machte, jede ***allegro***-Datenbank mit sehr wenig Arbeit ins Netz zu bringen. Das Verfahren entstand im Jahr 2000 und ist seit 2003 nur noch von akademischem Interesse. Alles zum Thema Web-Anbindung und ***avanti*** : **`www.allegro-c.de/avanti`**

2.8 Weitere Themen

Zu den hier aufgezähltenThemen gibt es ausführliche Dokumentationen, die aktuell gehalten werden, auf dem Webserver und in der Online-Dokumentation der Windows-Programme. Das Handbuch kann sich deshalb mit einigen Hinweisen begnügen.

JanaS : Genaue Beschreibung: Geben Sie ein: `x janas janas.htm`
Seit V24 gibt es ein Zusatzprogramm namens JanaS, das aus nichts anderem als einem Browser-Fenster besteht. Es kann drei Dinge:

- Im Internet surfen, also u.a. HTML-Dateien anzeigen.
  Die Schaltfläche mit der Tür startet JanaS und zeigt die allegro-Homepage
- a99 wieder aktivieren und ihm Befehle und Daten zusenden.
  (Geben Sie ein: `janas doku.htm` - hinter jedem Link steckt ein Befehl für a99)
- Auch DOC- und PDF-Dateien anzeigen, diese jedoch ohne eingebettete Befehle für a99.

Unicode : Genaue Beschreibung siehe `http://www.allegro-c.de/unicode/`

Es gibt zwei Varianten der Unicode-Anwendung in *allegro*-Datenbanken (ab V23.2)
Verfahren 1 : Interne Codes unverändert wie bisher, zusätzliche Zeichen als "HTML4-Entitäten" (s.a. → 10.2.4.1)
Verfahren 2 : Intern alles in UTF-8 codiert. Dies erfordert gute Kenntnisse der Export-Parametrierung.
Das *Eingeben* und *Bearbeiten* von Daten mit Unicode ist in bequemer Weise nur mit der neuen Browser-Oberfläche **a35** möglich (auch ***allegro-B*** genannt), die ab 2013/14 entwickelt wurde.

XML : `http://www.allegro-c.de/axml.htm`
Es gibt den FLEX-Befehl `xml`, der ohne weitere Vorbereitungen allegro-Daten in XML-codierte Daten verwandelt. Mehr dazu findet man hier: `h xxml` eingeben. Für die Produktion von MARCXML-Daten gibt es die Parameter `marcxml.apr`.

SQL : `http://www.allegro-c.de/aresqa.htm` oder: `h aresqa` eingeben
Länger schon konnte man mit a99 Tabellen erstellen zur Weiterverwendung in Word oder SQL-Datenbanken. Mit V25 wurde das "aresqa"-Konzept eingeführt, mit dem man Tabellen an externe Systeme übergeben, dort bearbeiten und dann mit a99 wieder einlesen kann.

RIA-Programm a30 (mit Adobe Flex/Flash) / Web-Oberfläche **a35** mit **HTML5** auch für Tablet und Smartphone
2009 wurde eine neue Art der Web-Anbindung entwickelt: eine *Rich Internet Application* namens a30. Ausführliche Dokumentation und offene Quellen unter `http://www.allegro-c.de/doku/a30/`. Verwendet wird Adobe Flash. Ab 2013 wird a30 abgelöst durch a35: `http://www.allegro-c.de/doku/a35/`, und zwar ohne Flash mit HTML5.

OAI
Für a99 und avanti+acon wurde auch eine OAI-Methodik entwickelt: `h oai` eingeben.
Mehr dazu: `http://www.allegro-c.de/oai.htm` und `http://www.openarchives.org/`

3 Datenerfassung

a99: Button **[Neusatz]**

3.0 Was ist dabei zu bedenken?

Datenerfassung, also die Eingabe neuer Daten, ist bei jedem Datenbankbetrieb eine zentrale Tätigkeit. Es ist die zeitaufwendigste und deshalb kostspieligste Aktivität, es ist echte Arbeit. Beim Eintippen macht es wenig Unterschied, ob man unter DOS oder Windows arbeitet, nur wenn man viel mit Copy+Paste arbeiten kann.
Je größer die Datenmenge und je höher die Erwartungen an die Qualität der Datenbank, umso weniger kann man schnelle Ergebnisse erwarten! Die folgenden Fragen sollte man sich deshalb bei jedem größeren Datenbankprojekt am Anfang stellen. Es geht um Entscheidungen, die Ihnen das System nicht abnehmen kann!

- **Welche Objekte** müssen wirklich erfaßt werden? Natürlich kann man alles irgendwann auch wieder löschen, aber ob man sich erst die Mühe der Eingabe macht, das ist die Frage!
- **Welche Datenelemente** werden gebraucht? Zu leicht übersieht man wichtige Elemente, die einem später fehlen, oder man gibt Dinge ein, die eigentlich überflüssig oder redundant sind.
  Bevor Sie mit der Erfassung echter Daten beginnen, studieren Sie die Möglichkeiten, die das Standard-Kategorienschema bietet (→ Anhang B). Erweiterungen des Schemas (Einführung neuer Kategorienummern) sind auch nachträglich möglich (→ Anhang A.1).
- Wie, d.h. **in welcher Form**, geben wir diese Elemente ein? Wenn man sich nicht an strenge Regeln hält (für den Bibliotheksbereich gibt es die bekanntlich), wird jede größere Datenbank im Chaos der "Inkonsistenz" enden. Die Eingabeform entscheidet oft auch darüber, ob man bestimmte Ausgabeformate oder brauchbare Registereinträge erzeugen kann.
  Nicht überall wird es nötig oder möglich sein, großkalibrige Regelwerke wie RAK oder AACR (ab 2016 RDA) anzuwenden. Man bedenke aber: Der intellektuelle Aufwand, verbunden immer auch mit hohem Zeitaufwand, den man in eine Datenbasis investiert, wird umso eher Bestand haben und zukunftssicher sein, je enger man sich an nun einmal existierende Normen hält. So bleibt auch die Chance offen, Fremddaten mit geringem Aufwand zu übernehmen bzw. Daten mit anderen auszutauschen und zusammenzuführen:
- Können wir Daten aus anderen Quellen, sog. **Fremddaten**, beziehen? Für viele Zwecke sind nationalbibliographische Datenbanken (zunehmend auf CD-ROM zugänglich) die idealen Quellen. Mögliche Lieferanten sind auch die Datenbanken der Bibliotheksverbünde, falls man deren Ressourcen "anzapfen" kann. Für Dokumentationen kommt auch "Downloading" aus einschlägigen Fachdatenbanken in Betracht. Der Gewinn ist nicht nur in der Arbeitsersparnis zu sehen: man erhält bei geringerem Eigenaufwand Daten von höherer Qualität, als man sie selber überhaupt produzieren könnte, und man findet Anschluß an die wichtigen Normen, z.B. was Schlagwörter und Namen (Personen und Körperschaften) angeht. Es wurde auch schon hier und da praktiziert, fachliche Teilbestände aus großen Datenbanken zu selektieren und in Form einer ***allegro***-Datenbank dann als lokale Fremddatenbasis zu nutzen.

  In a99 und auch in a35 kann man Fremddaten von DNB und GBV sehr leicht und schnell nutzen. Dazu mehr in dem Text, den man mit `h fremd` in a99 aufrufen kann. Ein etwas weniger direkter Weg zu noch mehr Quellen ist das Portal ZACK: http://opus.tu-bs.de/zack/ (dort ausführliche eigene Hilfetexte)
- **Wer** soll es machen? Problembewußtsein, Regelbeherrschung, Sachkompetenz und Geschick im Umgang mit Computern sind unverzichtbar. Es zeigt sich regelmäßig, daß die Schwierigkeiten und der Aufwand unterschätzt werden, daß man dies aber zu spät merkt. Personelle Kontinuität ist sehr wichtig, Fluktuation bringt zusätzliche Probleme, denn man muß jeweils beträchtliche Einarbeitungszeit veranschlagen.

Die Eingabearbeit selbst muß dann so effizient wie möglich gestaltet werden. Das Datenbanksystem muß als integrierten Bestandteil einen sogenannten **Editor** besitzen, der ein schnelles und bequemes Eingeben und Bearbeiten ermöglicht. Neben den üblichen und geläufigen Editorfunktionen braucht man zusätzliche Eingabeerleichterungen, wie z.B. Formal- oder Plausibilitätsprüfung sofort bei der Eingabe, Phrasenspeicherung, Kopieren von Datenfeldern oder ganzen Datensätzen, Wiederverwendung von früher erfaßten Datenelementen, u.a.m. Alles das gibt es auch bei ***allegro***. Der **integrierte Editor** ist nicht mit einem **Texteditor** zu vergleichen; er ist zwar auch bildschirmorientiert, aber zugleich kategorie- und befehlsorientiert. Sein Gebrauch erfordert, wie jeder Editor, ein wenig Übung, vor allem bei komplexen Datenstrukturen. Hilfe dazu für a99: `h inp`
Die folgenden Abschnitte sollen zur Arbeit mit dem integrierten ***allegro***-Editor hinführen. Das beste Verfahren der Einarbeitung ist die Benutzung der Beispieldatenbank, denn darin können Sie bedenkenlos und nach Belieben alles ausprobieren.

Abschnitt 3.5 gibt einen systematischen und einen alphabetischen Überblick über alle DOS-Editorbefehle.

3.1.1 Erfassen und Bearbeiten - Kompakter Überblick (DOS)

Windows: Online-Hilfe zur Dateneingabe mit **`h inp`**

Das Wichtigste dazu ist auf den folgenden zwei Seiten zusammengestellt. Besonders als Merkzettel beim Einarbeiten kann die Liste nützlich sein. Das Kap. 3.5 dagegen bietet eine vollständige, alphabetische Liste der Editorfunktionen.

Im Gegensatz zu mancher anderen Software gibt es bei ***allegro*** nicht sehr viele verschiedene Bildschirme. Diese werden auch als "4 Gesichter einer ***allegro***-Datenbank" bezeichnet. Man könnte auch statt dessen von den "Standardsituationen" sprechen, die man kennen und sicher beherrschen sollte.

Also: welche Standardsituationen gibt es in **PRESTO**, wie erkennt man sie am Bildschirm, und was kann man in diesen Situationen tun?
(Die Fenster-Bearbeitung ist kein eigener Bildschirm, sondern Teil des Editor-Bildschirms, hat aber doch eigene Funktionen.)

Situation / Aufgabe	*erkennbar an*	*Was kann man tun (Tastenbefehle) ?* *Sprung nach:*
ANZEIGE-Bildschirm Präsentation eines Datensatzes (Kap. 1.5)	Oben links steht: Datenbank ... Datei# ... Zu sehen ist ein Datensatz (eine Titelaufnahme)	**E** Satz bearbeiten → EDITOR-Bildschirm **C** Satz kopieren → EDITOR-Bildschirm **I** Neuen Satz eingeben → ABFRAGELISTE **F5** Darstellung umschalten auf Kategorieanzeige **<Shift>+F5** Datensatz in Hintergrundspeicher kopieren **<Alt>+***Ziffer* anderes Register **<Alt>+a** andere Datenbank (falls eine vorhanden) **L** Ergebnismenge der vorher bearbeiteten Sätze **F7** Registereinträge kontrollieren **F8** Abbruch → ***CockPit***
ABFRAGELISTE Eingabe eines neuen Datensatzes (Beispiel: Kap. 3.2)	Der Cursor steht in einer Abfragezeile: `#nn " Frage: "_` (Veränderung der Abfragen: in der CFG-Datei)	*Eingabe* (Text des Feldes eingeben) **F6** Register benutzen (mit <Enter> zurück) → REGISTER **<Strg>+<Enter>** Registerzeile übernehmen (und zwar die, wo zuletzt der Zeiger stand) **<Esc>** Textbausteine (Phrasen) **.** (Punkt) die momentan abgefragte Kategorie aus dem Hintergrundspeicher kopieren **x** Abfrageliste verlassen → EDITOR-Bildschirm **F9** zurück zur vorigen Abfrage (wenn leer weggedrückt) **F8** Abbruch der Erfassung → REGISTER-Bildschirm
EDITOR-Bildschirm Bearbeitung eines Datensatzes (Kap. 3)	Datensatz mit Kategorienummern ist zu sehen, darunter steht `#_`	*Korrektur* mit Cursor (jede korrigierte Kategorie mit <Enter> bestätigen) *Neueingabe* einer Kategorie (Nummer Text eingeben) *Editor-Befehle* (nächste Seite und Handbuch: Kap. 3) **b** (Editor-Befehl) → Fenster-Bearbeitung **F6** Register benutzen (mit <Enter> zurück) → REGISTER **<Strg>+<Enter>** Registerzeile übernehmen **<Esc>** Textbausteine (Phrasen) **<Einfg>** setzt nur ein Leerzeichen ein **F10** Speichern → ANZEIGE-Bildschirm **F8** Abbruch → ANZEIGE-Bildschirm
Fenster-Bearbeitung Bearbeitung eines Datenfeldes (Kap. 3, #b)	Rahmen mit Kategorietext in der Mitte des Bildschirms nachdem Befehl #b gegeben wurde	**<Einfg>** schaltet Modus um auf "Einfügen" **<Esc>** Buchstabe Phrase abrufen **F6** Register benutzen (mit <Enter> zurück **<Strg>+<Enter>** Registerzeile übernehmen **F8** Abbruch → EDITOR-Bildschirm
REGISTER-Schirm Alphabetisches Suchen (Kap. 1.4)	Alphabetische Liste mit einem Zeiger: **==>**	*Eingabe* eines Suchbegriffs **<Enter>** Zugriff auf den Datensatz bzw. Rückkehr zur Bearbeitung (wenn mit F6 aus EDITOR gekommen) **<Alt>+***Ziffer* Register wechseln **<Shift>+F8** Erweitertes Register einschalten / **<Esc>** abschalten **<Shift>+F9** Kurzanzeige der Erggebnismenge / **<Esc>** abschalten **<Alt>+a** andere Datenbank (wenn vorhanden) **F8** Abbruch → ***CockPit*** **F10** Trunkierung **<Backspace>** Normalzustand

---===>>> **In allen Situationen: F1 für Hilfe.** <<<===---

Als Einsteiger probiert man am besten erst einmal alles an der Demo-Datenbank durch und fragt sich dabei immer, ob und wofür man die Funktionen selber einsetzen kann.

3.1.2 Editor-Befehle : Die wichtigsten für Einsteiger

Man schreibt den Befehl immer direkt hinter das "Promptzeichen", das ist das **#**, welches automatisch unter dem Datensatz erscheint.
Wenn also gesagt wird, "**Befehl #a**", bedeutet dies, daß man nur das **a** eingeben muß, weil das **#** automatisch vorhanden ist.
Ausführliche und vollständige Beschreibungen der Befehle stehen, alphabetisch geordnet, im Kap. 3.5. Schlagen Sie dort nach, wenn Sie mehr wissen wollen. Dort steht auch, welche Funktionen in ***a99*** den DOS-Befehlen entsprechen!

#a **Hintergrundspeicher einsehen**
In anderer Farbe werden die Datenfelder gezeigt, die im Hintergrundspeicher stehen. Jedes davon kann man in den aktuellen Datensatz übernehmen: Cursor hinfahren und <Enter> drücken
z.B. **#a7** Nur die Felder der Gruppe #7 aus dem Hintergrundspeicher anzeigen

#b **Fensterbearbeitung** (bequeme Bearbeitung)
Die Felder des aktuellen Satzes werden einzeln in einem Bearbeitungsfenster gezeigt und können darin editiert werden. Insbesondere funktioniert der Einfügemodus (ein und aus mit <Einfg>).
z.B. **#b20** Nur Kategorie #20 im Fenster bearbeiten

#l **Liste der aktuellen Kategorien** anzeigen
Der aktuelle Datensatz wird in Bearbeitungsform neu angezeigt. Nützlich, um zu sehen, wie der Gesamtzustand ist, nachdem man mehrere Änderungen gemacht hat.
z.B. **#l2** Nur die Kategorien der Gruppe #2 anzeigen. Nützlich, wenn diese wegen der Länge des Datensatzes oben aus dem Bildschirm verschwunden sind - weil man nicht nach oben blättern kann!

#p **Phrasen anzeigen**; anschließend 'a' und 'p' drücken, damit die Phrasen erscheinen. Mehr über Phrasen siehe unten.

#t **Transfer** (= Übernahme) aus dem Hintergrundspeicher
z.B. **#t75** Kategorie #75 wird aus dem Hintergrundspeicher in den aktuellen Datensatz kopiert.
Statt 't' kann auch der Punkt benutzt werden, also `#.75.`

#v **Vergessen** (= Löschen) von einzelnen Kategorien (mit Überschreiben am Bildschirm geht es nicht!)
z.B. **#v90** Lösche #90

#X **Globale Ersetzungen definieren**. Durchgeführt werden die Ersetzungen dann aus dem ANZEIGE-Bildschirm heraus mit F10. Wenn zu dem Zeitpunkt eine Ergebnismenge besteht, können die Ersetzungen in der gesamten Ergebnismenge ausgeführt werden.
z.B. **#X_alt_neu_** Die Zeichenfolge "**alt**" soll durch "**neu**" ersetzt werden. (Mehr Beispiele: Kap. 3.5)
Wenn in `alt` oder `neu` ein `'_'` vorkommt, dann die drei `'_'` alle durch `','` ersetzen!

Phrasenfunktionen [siehe auch Befehl #p im Kap. 3] ***a99:*** **`h phras`** eingeben

Eine **Phrase** ist ein Wort, eine Wortgruppe, oder allgemein eine beliebige Zeichenfolge. ***allegro-C*** ermöglicht das Speichern von Phrasen zwecks beliebiger Verwendung im Eingabetext.
Als Funktionstaste für das Definieren, Verwenden und Löschen von Phrasen dient die **<Esc>**-Taste:

neue Phrase definieren: (siehe auch Kap. 1.4.2: Indexeintragungen als Phrase)
- man setzt hinter das Prompt '#' einen Kennbuchstaben oder auch ein beliebiges anderes Zeichen,
- schreibt den Phrasentext (Länge beliebig) und
- drückt dann **zweimal <Esc>.** Dann wird bestätigt, daß unter dem angegebenen Kennbuchstaben nun diese Phrase gespeichert ist.

z.B. `#tTechnische Universität <Esc> <Esc>` erzeugt eine Phrase mit dem Kennbuchstaben t.

Phrase verwenden: (auch im ABFRAGE-Zustand und im Bearbeitungsfenster möglich)
An beliebiger Stelle während einer Eingabe tippt man **<Esc> [Kennbuchstabe]** : sofort erscheint diese, der Cursor steht dahinter und die Eingabe kann weitergehen. **Achtung**: Taste <Esc> dabei nicht festhalten, nur kurz antippen.
z.B. bei **<Esc> t** erscheint `Technische Universität.`

Phrase löschen:
- Man gibt nur das Kennzeichen direkt hinter dem '#' ein
- und dann **<Esc> <Esc>** .

z.B. **#t <Esc> <Esc>** um die oben definierte Phrase zu löschen

Jedes Zeichen, das per Tastatur eingebbar ist, kann als Phrasen-Kennzeichen benutzt werden, insbesondere können Groß- und Kleinbuchstaben mit unterschiedlichen Phrasen belegt werden!
Stets benutzbar sind zwei **Sonderphrasen**: mit **<Esc> 0** erhält man Datum/Uhrzeit, mit **<Esc> [Leertaste]** nur das Datum, wenn man in der .CFG den Befehl D8 setzt (→ Anh.A.1.3).
Die angesammelten Phrasen werden am Ende der Sitzung als Datei PHRASE.APH gespeichert und bei der nächsten Sitzung automatisch wieder geladen, d.h. sie sind sofort wieder verfügbar. (Siehe auch Kap.3.5, Befehl `#p`)
Auch die Windows-Programme haben eine Phrasenfunktion; die Datei dazu heißt aber PHRASE.A99. Mehr: `h phras` eingeben.

3.2 Abfrageliste (zur Dateneingabe)

Sehr wichtig im DOS-Programm

a99: 2mal **Alt+y**

Die Abfrageliste erscheint auch im Prog.1 bei Funktion **I** = Erfassung, insbesondere beginnt so der Neuaufbau einer Datenbank, den man von ***CockPit***-Menüpunkt "Datenbank anlegen" aus startet.
Die übliche Bezeichnung "Eingabemaske" ist hier nicht ganz treffend! Es handelt sich um ein Abfragesystem (engl. "prompting"), in das nach Wunsch des Benutzers mehrere Satztypen mit unterschiedlichen Abfragen und Verzweigungen eingebaut sein können (→ Anh.A.2 für individuelle Anpassungen).
Bei der Standardkonfiguration A.CFG kommt als erstes eine **Typ-Abfrage**. Die darauf folgenden Fragen sind je nach Typ unterschiedlich:

```
Typ 1 = selbständig (Buch)   2 = unselbständig (Aufsatz)      3 = Stammdaten
    z = Zeitschr.-Stammsatz  r = siehe- und siehe-auch-Verw.  7 = Sachdaten
    9 = sonstige Kategorien  u = Einzelband   v = verknüpfte Bandaufnahme
 1 <┘      (d.h. ein Buch soll erfaßt werden)
#40  Verfasser: Riedel, Rupert <┘
#20  Sachtitel: Biologie der Erkenntnis : die stammesgeschichtlichen Grundlage
n der Vernunft <┘
#74        Ort: München  <┘
#75     Verlag: Deutscher Taschenbuch Verlag  <┘
#76       Jahr: 1988  <┘
#77     Umfang: 230 S.  <┘
#87       ISBN: 3-423-10858-5  <┘
#85      Serie: dtv ; 10858  <┘          oder <┘ , wenn's kein Serienstück ist
#90   Signatur: xxx-123  <┘              hier geben Sie Ihre Signatur ein oder <┘ , wenn Sie keine haben
```

Folgende Punkte sind für den Anfang wichtig:

- Ein Rücksprung mit Hilfe der Cursortasten zwecks Korrektur von bereits eingegebenen Daten ist bei diesem Vorgang möglich, nach ausgeführter Korrektur dann <┘ drücken.
- Hat man eine Abfrage versehentlich leer "weggedrückt": mit **F9** kommt sie zurück.
- Nach dem Durchlaufen der Abfrageliste, wenn man im **Befehls- und Eingabemodus** des Editors ist, kann auch noch korrigiert werden: Zuerst erscheint links das Zeichen **'#'** mit dem Cursor rechts daneben. Dies zeigt an, daß das Programm nun auf weitere Daten- und Befehlseingaben wartet. Nach jeder Eingabe oder Ausführung eines Befehls erscheint wieder **'#_'** an der ersten Stelle der nächsten Zeile, d.h. Sie geben bei den hier gezeigten Beispielen nur das ein, was jeweils hinter dem '#' steht. Das Zeichen '#' wird als sog. "Promptzeichen" immer bereits vorgegeben.
- **Grundprinzip**: Nach jeder Korrektur an einer auf dem Bildschirm stehenden Kategorie drückt man <┘, um die korrigierte Kategorie in den Arbeitsspeicher zu übertragen. Mit anderen Worten: nicht korrigieren und zurückfahren, sondern sofort nach dem Korrigieren, während der Cursor noch in der korrigierten Kategorie steht, <┘ drücken. Der Cursor geht dann von selbst wieder zurück.
- Die wichtigsten Befehle des Editors sind in Kap.3.5 mit *WICHTIG* gekennzeichnet.
- Mit **F6** kann man jederzeit (egal, wo der Cursor gerade steht) den Index "aufklappen", dort etwas suchen, in eine Phrase kopieren (z.B. mit **<Esc> a**), mit <┘ zum Editor zurückkehren und die Phrase einkopieren (erneut **<Esc> a** geben). (Gilt nur für online-, nicht offline-Bearbeitung.)
- Punkt eingeben: Feld übernehmen aus dem Datensatz, der vorher in der Anzeige stand.
- **F8** bricht die Abfrageliste ab. Man landet im Editor oder, wenn noch nichts eingegeben wurde, wieder im Register. Mit #E kann man die Abfrage erneut starten, um die noch fehlenden Dinge einzugeben.

Nicht relevante Abfragen übergeht man mit <┘ . Wenn man in der Abfrageliste aus Versehen ein Feld leer weggedrückt hat, gibt es drei Möglichkeiten:

- mit **F9** holt man die Abfrage zurück (geht auch mehrmals)
- man gibt es nachträglich ein: wenn Sie z.B. den Serientitel übersprungen hatten, geben Sie jetzt
  `#85 dtv ; 10858  <┘`
  und das Programm sortiert #85 intern sofort richtig vor #87 ein.
- man gibt den Befehl #E , dann kommt wieder die Typabfrage, man wählt denselben Typ wie vorher, und es werden die zuvor nicht eingegebenen Elemente **nochmals** abgefragt.

Nehmen Sie das Kategorienschema (→ Anh.B) zu Hilfe, um Datenelemente einzugeben, die in der Abfrageliste nicht vorkommen. Jede erlaubte Kategorie kann, wie oben bei #85 gezeigt, im Befehlsmodus direkt eingegeben werden.

3.3 Zeichen mit Sonderbedeutung

***a99*:h codes**

Nur wenige Zeichen haben bei ***allegro*** eine Sonderbedeutung:
(In der .CFG-Datei kann man diese Aufgaben auf andere Zeichen übertragen (→ Anh.A.2))

**Promptzeichen des Editors** (Eingabe- und Befehlszeichen)
es erscheint nach einer Eingabe immer automatisch am Anfang der letzten Bildschirmzeile und fordert praktisch zur nächsten Eingabe (Befehl oder Datenfeld) auf. Wenn man dahinter eine Kategorienummer eingibt, wird die Eingabe als Datenfeld genommen, wenn es eines der Befehlszeichen ist, als Befehl interpretiert. Das Zeichen '#' darf auch im Text einer Eingabe vorkommen, nur am Zeilenanfang hat es Steuerfunktion. (In den Dateien wird es nicht gespeichert, das spart etwas Platz.)

¬ **Nichtsortierzeichen**. Es wird in den Kat. #20 #22 #23 #24 #60 #61 und #85 automatisch eingesetzt, wenn das erste Wort ein Artikel ist. Wenn es in Anzeige oder Druck nicht erscheinen soll, macht man es in der Parameterdatei D-1.APR mit den Befehlen `p ¬ 1 und q ¬ 1` ungültig. Es dient dazu, für die Sortierung und das Drucken den Artikel kenntlich zu machen und evtl. auszublenden. Das Zeichen hat den Code 170. Von Hand eingeben kann man es nur über die Taste <Alt>: man drückt <Alt>, hält es fest, gibt 170 ein (und zwar auf dem Zifferntastenblock!) und läßt <Alt> wieder los. ***a99***: Strg+k
Auch innerhalb von Titeln können Wörter hiermit markiert werden. Benutzt man bei den Exportparametern dann den Bearbeitungsbefehl 'u' oder 'U', so werden in der betreffenden Kategorie alle mit diesem Zeichen gekennzeichneten Wörter bei der Ausgabe weggelassen (→ 10.2.6.3). Die Funktion des Nichtsortierzeichens kann auch einem anderen Zeichen übertragen werden (→ Anh.A, Befehle N und n).

@ **Nicht-Stopzeichen**, auch "Entstoppungszeichen" genannt. Dieses setzt direkt man vor Wörter, die in der Stopwortliste (SWL1.cPT) vorkommen, aber in einem Einzelfall trotzdem ins Wortregister sollen. Eingabe mit **<AltGr>+q**.

¶ a) **Zeilenvorschub-Steuerung, Absatz-Endezeichen** : Es kann vorkommen, daß man beim Abdruck innnerhalb eines Datenfeldes einen Zeilenvorschub (Absatz) auslösen möchte. Dafür ist das Zeichen mit Code 20 vorgesehen; Eingabe (auch in ***a99***) mit **Strg+t**. Beim Druckbefehl wird dann an dieser Stelle abgebrochen und eine neue Zeile begonnen. Man kann jedem anderen Zeichen diese Funktion zuweisen (→ 10.2.4.4), wenn man '¶' nur als Trennzeichen verwenden will:

b) **Trennzeichen für Mehrfach-Einträge** : Wenn mehrere Namen oder Schlagwörter etc. in einer Kategorie untergebracht werden sollen, empfiehlt sich als Trennung dieses Zeichen, wenn man nicht "; " benutzen will. (Der einzige Vorteil von '¶' ist, daß die Einträge dann leicht auf getrennten Zeilen untereinander erscheinen können, → 10.2.4.4). Für Index- und Registerproduktionen sowie Kartenköpfe kann man die Kategorie anhand dieser Zeichen wieder zerlegen lassen. [Beim N-Format übernimmt | diese Funktion]

▉ **Füllzeichen** (Code 191). Man gibt es nicht mit der Hand ein und bekommt es auch nicht zu sehen, sondern es dient beim Speichern dazu, am Ende eines Satzes noch Füllraum zum Verlängern zu reservieren, und zwar nur in .ALD-Dateien, nicht in Grunddateien. Dies bedeutet nicht, daß ***allegro*** Platz vergeudet. Das Konzept des Füllzeichens wird im Anhang A erklärt. Es ist durch ein anderes ersetzbar, aber nur, bevor die Datenbank neu aufgebaut oder generiert wird.

▼ **Teilfeld-Trennzeichen** (Code 31). Man gibt es mit **Strg-** ("Control-Minus") ein. Mit diesem Zeichen kann man innerhalb von Kategorien Untergliederungen in Teilfelder vornehmen. Die ***allegro***-Programme für Ausleihe und Erwerbung nutzen diese Technik intensiv. Hinter das Zeichen kommt ein Buchstabe oder eine Ziffer, und beides zusammen stellt so etwas wie eine "Kategorie innerhalb der Kategorie" dar. Jedes Teilfeld ist dadurch eindeutig bestimmt und beim Export leicht zu isolieren (→ 10.2.6.3, Befehl ▼).

a99/alcarta: Eingabe mit AltGr+2 (die Ziffer 2 oben links)

Akzente: werden **vor** die Buchstaben gesetzt (→ 10.2.4.3 "Backspace-Liste")

DIN 31628: Dieser Zeichensatz existiert für DOS und Windows. Hilfssoftware dazu wurde an der UB Braunschweig erarbeitet: DOS-Aktivierung mit **`fontload <ostwest.fon`** (→ Anhang E). Unter Windows müssen die TTF-Schriften installiert sein, die mitgeliefert werden. Das passiert normalerweise automatisch durch das "Gesamtpaket".

Wenn man die Schrift A-DOS.FON installiert, kann man die Sonderzeichen auch im Windows-Fenstermodus sehen (Schriftgröße 8x13 und 10x19), sonst nur im Vollbild.

3.4 Besondere Tastenfunktionen im DOS-Editor

Übersicht für ***a99***: Alt+c

4. **Enter** und **Return** sind (für ***allegro***) gleichwertig. Auf **Return** ist meist nur das Symbol **<┘** drauf.
5. **Esc** hat im Editor und Index die Bedeutung einer Phrasen-Steuertaste
   (→ Kap.1.4 und Kap.3.5/Befehl #p.)
   Man "entkommt" aus dem Editor mit **F8**, nicht mit **Esc** .

6. Die **Cursortasten** haben die normale, von Textprogrammen bekannte Wirkung. Ausnahme: die **Einfg**-Taste (englisch **Ins**) fügt an der Cursorstelle ein Leerzeichen ein und verschiebt den Rest der Kategorie um ein Zeichen. Den "Einfügemodus" zum Ein- und Ausschalten gibt es nur bei dem Befehl #b (→ Kap.3.5) Die **Entf**-Taste (englisch: **Del**-Taste) beseitigt das Zeichen, auf dem der Cursor steht und zieht die rechts davon stehenden bis zum Ende der Kategorie nach vorn. Auf den Menüs des Schnellzugriff-Programms hat **Entf** andere Löschfunktionen (→ 1.4 und 1.5). Eine ganze Kategorie löschen kann man nicht mit **Entf** oder durch Überschreiben, dafür gibt es den Befehl **#v** (→ 3.5).

7. Die **Rücktaste** oberhalb **<┘** (mit dem Pfeil nach links, auch **BACKSPACE** genannt) wirkt so: der Cursor geht eine Position zurück, löscht das dort stehende Zeichen und zieht die hinter ihm stehenden Zeichen bis zum Kategorieende nach (auch wenn es mehrere Zeilen sind).

8. **Funktionstastenbelegung** im Editor:

   F1, F6 und F8 gehen immer, bei den anderen muß man evtl. vorher nochmal <┘ drücken.

   F1 : HILFE-Taste : Wenn der Cursor gerade z.B. in Kategorie #245 ist, kommt Hilfetext H245; wenn H245 nicht existiert, wird H24 gezeigt; wird dieses ebenfalls nicht gefunden, sucht das Programm noch nach H2. Im Prinzip kann man individuell zu jeder Kategorie #xyz eine eigene Hilfeseite HXYZ anlegen, die auch direkt mit dem Befehl #hxyz abrufbar ist.
   Andere Hilfeseiten kommen mit **Shift+F1, Alt+F1** und **Ctrl+F1**. (Übersicht → 0.3)

   F2 : Druckmenü (wie bei Befehl #d oder #D)

   F3 : formatierte Anzeige des aktuellen Satzes; wenn z.B. (standardmäßig) die Parameterdatei D-1.APR geladen ist, erfolgt Anzeige aller geköpften Karten

   F4 : Druck aller Karten, die in D-1.APR oder P-KARTE vorgeschrieben sind (→ 10.2.2)

   F5 : Anzeige des aktuellen Satzes in geordneter, kategorisierter Form

   F6 : (nur im Programm 1) Index "aufklappen", um dort etwas nachzusehen und evtl. etwas als Phrase für die Erfassung zu übernehmen (→ 1.4, **Esc**), Rückkehr zur Erfassung mit <┘ oder "Cursor links". Anschließend kann man mit **<Strg>+<┘** die Registerzeile, auf der zuletzt der Zeiger stand, in die Eingabe kopieren.

   F8 : Aussprung aus der Bearbeitung/Erfassung ohne Speicherung ("Stornieren")

   F9 : Während der *Abfrage*: eine vorher leer weggedrückte Abfrage kommt zurück,
   Beim *Bearbeiten*: Abschluß des aktuellen Satzes und Ablage im Arbeitsspeicher. Man bleibt im Editor und kann anschließend hierarchische Untersätze in beliebiger Zahl und Stufung (Kategorien #01...#06) eingeben. Für eine Bandeingabe gibt man bei der Typabfrage (→ 3.2) den Buchstaben u und wird dann zuerst nach Kategorie #01 (Bandnummer) gefragt. Wenn kein weiterer Band mehr einzugeben ist: mit **x** <┘ aus der Abfrage aussteigen. Mit **Pos1** (engl. **Home**) kann man zum Hauptsatz zurück, Mit <**Bild↑**> und <**Bild↓**> blättert man zwischen den Untersätzen. Überall kann man noch alles korrigieren, bevor man mit F10 am Ende alles abspeichert:

   F10 : Abschluß des aktuellen Satzes. Im Programm 1: sofortige Abspeicherung in der Datei und im Index. Sollen hierarchische Untersätze eines mehrbändigen Werkes erfaßt werden: statt dessen **F9** geben und dann die Banddaten eingeben (Eingabetyp u). Erst nach dem letzten Band **F10**.

3.5 Editor : Befehle (DOS-Programm PRESTO)

Direkteinstieg aus **a99**: Menü "Datei | DOS-Programm"

3.5.0 Vorbemerkungen

Dieses Kapitel gibt einen vollständigen Überblick über den **integrierten Editor** des ***allegro***-DOS-Systems, mit dem man Daten eingibt und bearbeitet. Im alphabetischen Nachschlageteil 3.5.3 werden alle Befehle genau beschrieben, mit denen man bei der Dateneingabe und Korrektur arbeiten kann. Kurz: alles, was man tun kann, wenn man das Promptzeichen `#_` sieht. Denn dieses bedeutet: das Programm wartet auf Daten und/oder Befehle.
Der Editor ist nichts sehr viel anderes als ein Text-Editor mit einigen Sonderfunktionen. Eine formularbasierte Bearbeitung gibt es in PRESTO nicht, als Ersatz kann man die Abfrageliste ansehen. In ***a99*** dagegen gibt es ein Formularsystem (`h form` eingeben).

Die Abschnitte 3.0 bis 3.4 sind dagegen als Einführung in die Datenerfassung und -bearbeitung gedacht und soll dem Erlernen der allerwichtigsten Funktionen dienen. Einsteiger sollten damit beginnen und erst bei Bedarf (wenn man mehr wissen will) dann den Abschnitt 3.5 heranziehen. Dort werden auch Hinweise auf die entsprechenden Befehle in ***a99*** gegeben.

Der schnelle Einstieg:
(Falls Sie noch nicht zur Lektüre von 3.0 bis 3.4 gekommen sind, hier nur ganz knapp die Schritte)

- Demodatenbank vom ***CockPit*** aus mit Menüpunkt **`Benutzen`** starten
- Vom Anzeigebildschirm (nicht vom Register, → Kap.1.5) bei einem Datensatz, den Sie ändern wollen, ein großes **`E`** (für Edit) geben: dann erscheinen die Datenfelder, Sie befinden sich im **Editor** und können das Bearbeiten ausprobieren (ziehen Sie Kap.3.1 hinzu, da stehen die wichtigsten Befehle)
- **`F10`** zum Speichern (es wird gefragt: `Speichern? j/n )`
- Das große **`I`** eingeben (für Input): jetzt können Sie eine Neuaufnahme machen (→ Kap.3.2: "Abfrageliste")
- Nachdem alles eingegeben ist: mit **`F10`** speichern
- Ein großes **`C`** (für Copy) macht Ihnen eine Kopie des aktuellen (gerade angezeigten) Satzes. Diese Kopie können Sie nun bearbeiten. Erst wenn Sie dann **`F10`** geben, wird daraus ein neuer Datensatz.
- Mit **`F8 j`** oder **`F10 n`** kann jede dieser Aktionen ohne Auswirkung abgebrochen werden.

Anhang B beschreibt, welche Datenfelder (**Kategorien**) es standardmäßig gibt. Vielleicht haben Sie aber ein ganz anderes Schema, wenn Ihnen die Software von anderer Seite installiert wurde. Wenn man Abwandlungen wünscht oder ein ganz eigenes Schema realisieren will, muß man Anhang A studieren.

Sie finden in diesem Kapitel
- eine genaue Beschreibung der Form ("Syntax") der Befehls- und Dateneingabe,
- eine systematische Kurzübersicht über alle Befehle,
- in alphabetischer Folge die genauen Befehlsbeschreibungen mit folgenden Kennzeichnungen:
  WICHTIG = diesen Befehl müssen Sie unbedingt kennen,
  wichtig = den können Sie auch oft gebrauchen,
  nützlich = auch diese Befehle sollten Sie sich mal aneignen.
  (WICHTIG) = nur in den Programmen 2 bis 5 wichtig, nicht im Normalbetrieb (also nicht in PRESTO)

Die Beschreibungen dieses Kapitels sind für die Programme 1 - 5 gültig, denn in jedes dieser Programme ist der interne Editor (Eingabe- und Bearbeitungsmodus) integriert. Im Schnellzugriff kommt man über die Funktionen E, C und I nur vom Anzeigebildschirm aus hinein (→ 1.5, nicht vom Register!), in den Programmen 4 und 5 durch Unterbrechung des Programmlaufs mit Druck auf **`x`**.
Bei den Befehlen wird jeweils vermerkt, ob es bei einzelnen Programmen Besonderheiten oder Einschränkungen gibt.

Anmerkung: Der integrierte ***allegro***-Editor ist speziell nur für ***allegro*-Datensätze** zuständig. Für ASCII-Dateien, wie z.B. die Parameter- oder Konfigurationsdateien, benötigt man einen anderen, "normalen" Texteditor (→ Anh.D). Mit einem solchen wiederum lassen sich ***allegro***-Grunddateien und erst recht Datenbankdateien nicht bearbeiten. In ***a99*** dagegen kann man Datensätze auch in einem externen Texteditor wie etwa Notepad bearbeiten (Alt+t drücken).

3.5.1 Dateneingabe und -bearbeitung (DOS)

Wenn nach Durchlaufen der Abfrageliste (→ Kap.3.2) noch Kategorien fehlen (oder wenn man überhaupt ohne Abfrageliste arbeitet), gibt man sie in beliebiger Reihenfolge einzeln ein, jeweils mit abschließendem **Enter (<↵).**
Das Zeichen '#' als **Kategorie-Anfangszeichen** und **Befehls-Anfangszeichen** gibt der Rechner auf dem Bildschirm vor, Sie geben es also **nicht** mit ein. Bei anderen Programmen wird so etwas auch **Promptzeichen** oder **Bereitschaftszeichen** genannt. '#' ist also das **Promptzeichen** des ***allegro***-Editors.
Hinter dem Promptzeichen gibt man entweder **Daten** (d.h. eine Kategorie) ein oder aber einen **Befehl** (→3.1). Nach dem Druck auf <Enter> untersucht das Programm Ihre Eingabe, stellt fest, um was es sich handelt und reagiert dementsprechend: ein Befehl wird ausgeführt, eine Kategorie untersucht und abgespeichert.

Dies ist die allgemeine Form ("Syntax") der **Dateneingabe**:

```
#xxfabc... <↵
```

dabei ist	**#**	das automatisch vorgegebene Kategorie-Anfangszeichen oder Promptzeichen
	xx	die **Kategorie-Nummer**
	f	normalerweise ein Leerzeichen oder ein **Folgezeichen** (bei Mehrfachbesetzung oder dreistelligen Kategorien)
	abc	der Kategorie-Text (die eigentlichen Daten), der sich auch über mehrere Zeilen erstrecken kann. Bei Formaten wie MAB (D.CFG) oder MARC (U.CFG) beginnt der Kategorietext mit einem oder zwei Indikatoren (s. Anhang A).

Man schreibt längere Kategorien fortlaufend, ohne sich um das Zeilenende zu kümmern. Denn: Zeilenumbruch ist eine Exportaufgabe, findet also nur automatisch beim Drucken, bei der Bildschirmanzeige oder bei sonstiger Ausgabe statt - wenn und wie es gebraucht wird.

Mehrfachbesetzung (Wiederholbare Felder)
Es gibt zwei Methoden, gleichartige Kategorien innerhalb eines Datensatzes zu wiederholen:

a) man arbeitet mit **Folgezeichen** oder **Wiederholungszeichen** hinter der Kategorienummer:
```
#40 Bergmann, Ludwig      (1. Verf.)
#402Schäfer, Clemens      (2. Verf.)
```
Jedes Zeichen kann als Folgezeichen benutzt werden (d.h. jede Kategorie ist theoretisch so oft besetzbar wie es Zeichen gibt.) Man verwendet z.B. meistens #402 bzw. #403 für 2. bzw. 3. Verfasser, #40a und #40b sind genauso möglich und üblich; es ist dabei natürlich sinnvoll, konsequent zu sein. Empfohlen wird für das konsolidierte Format: man verwende Buchstaben, wenn die Kategorien inhaltlich eine unterschiedliche Bedeutung haben, und Ziffern, wenn es sich um Mehrfachkategorien handelt.
Hinter das Folgezeichen setzt man kein Leerzeichen; das Programm beseitigt es andernfalls automatisch. Der Kategorietext beginnt also immer auf Position 4.
Bei einem drei- oder vierstelligen Kategoriesystem (z.B. U.CFG, P.CFG) gibt erst ab Version 13 ein Folgezeichen, d.h. wiederholbare dreistellige Kategorienummern (→ Anh.A, Befehl **t**).

b) Innerhalb der Kategorie trennt man mehrere gleichartige Elemente durch ein festgelegtes Zeichen oder auch eine Zeichenkombination. Empfohlen wird die Kombination "; ", möglich ist auch das Zeichen ¶ (Strg+t). Diese Methode der Mehrfachbesetzung ist für alle ***allegro***-Funktionen der anderen gleichwertig: insbesondere lassen sich die einzelnen Bestandteile indexieren oder als Kartenköpfe verwenden.

Methode a) ist datentechnisch sauberer und bei Exportaufgaben günstiger, wenn die Kategorie eine innere Struktur mit Teilfeldern hat, wie z.B. Schlagwortketten, Erwerbungsdaten oder Exemplarangaben.

Unmittelbar nach <↵ führt der Rechner eine **Formalprüfung** der eingegebenen Kategorie durch und gibt, falls nötig, eine Fehlermeldung aus. Bei Fehlern wird entweder

a) die Kategorie zurückgewiesen (z.B. bei nicht erlaubter Nummer, falschen Teilfeldern, ...), oder

b) eine ja/nein-Frage gestellt, die mit 'j' oder 'n' zu beantworten ist.

Korrigieren

Wenn man irgendeinen Fehler erst nach der Eingabe bemerkt, hat man folgende Möglichkeiten (hier aufgeführt in der Reihenfolge ihrer Wichtigkeit):

- Korrektur **mit dem Cursor** vornehmen, dann <↵ drücken (während der Cursor noch innerhalb der betreffenden Kategorie ist!), oder aber
- **Befehl #b** anwenden: wenn man #bxxf gibt, wird Kategorie #xxf in einem "Bearbeitungsfenster" angeboten und kann mit allen von Textprogrammen gewohnten Cursorfunktionen bearbeitet werden. Das Fenster ist genau eine Zeile lang, d.h. es wird immer ein 80-Zeichen-Ausschnitt aus dem Kategorietext gezeigt. Dieser Ausschnitt bewegt sich dann nach Bedarf nach links und rechts. Günstig ist diese Bearbeitungsmethode für lange Kategorien (z.B. Abstracts), die dann auch länger als der Bildschirm werden können (bis 6000 Zeichen).
- die fehlerhafte Kategorie einfach **neu eingeben**. Wenn es sich um eine kurze Angabe handelt, wie etwa das Erscheinungsjahr, kann dies die schnellste Methode sein. Der Rechner ersetzt dann die fehlerhafte durch die neu eingegebene Kategorie.
- **Such- und Ersetzungsbefehl** (`#_X_Y_, Beschreibung am Ende von 3.3`)

Aus dem Eingabetext werden vor der Übertragung in den Arbeitsspeicher alle überflüssigen Leerzeichen am Anfang, am Ende und in der Mitte (wenn zwei oder mehr nebeneinander stehen) beseitigt. Hinter dem '#' darf kein Leerzeichen stehen, sonst gibt es eine Fehlermeldung.

Aber wie ändert man eine Kategorienummer?

Wenn man eine Kategorienummer ändert, z.B. #40 in #41, hat man anschließend beide Kategorien in der Aufnahme, denn das Programm merkt nicht, wenn man auf dem Bildschirm die Änderung durchgeführt hat, daß die Kategorie vorher anders hieß. #41 wird als neue Kategorie einsortiert, #40 bleibt erhalten.

Und wie kriegt man die falsche Kategorie wieder weg?

Mit dem Befehl **`#v`**, also z.B. **`#v40`**, beseitigt man eine falsche oder überflüssige Kategorie.
Sonderfall: Man hat fälschlich so etwas sie `#74Hamburg` eingegeben, also das Leerzeichen hinter der Kategorienummer vergessen. Das Programm "denkt", #74H sei eine eigene Kategorie. Nur mit dem Befehl **`#v74H`** läßt sie sich beseitigen.

Aufnahme-Abschluß

Eine fertige Aufnahme wird abgeschlossen, indem man

F10 **`j`** bzw. **`#rj`** (Prog.1) [oder `#`**`#<↵ <↵`** `(nur Prog.2/3)`]
eingibt. Danach kann man sofort mit der nächsten beginnen.

Stellt man aber sofort nach dem Speichern einen Fehler fest:

`'E'` zum erneuten Bearbeiten (Prog.1) [oder `#`**`- <↵`** `(nur Prog.2,3,4,5)`]
Probieren Sie alles selbst aus und beobachten Sie, was passiert; so lernen Sie am besten die Funktionen kennen. Es wird dringend empfohlen, hierfür etwas Zeit zu investieren - umso mehr Zeit spart man später bei der Erfassung.

3.5.2 DOS-Editor-Befehle : Systematische Übersicht

Wozu dieser Abschnitt?

Hier steht alles, was man machen kann, nachdem man im DOS-Programm PRESTO einen Datensatz angezeigt bekommen und dann E gedrückt hat:

Wenn mit einzelnen Kategorien oder dem ganzen Datensatz etwas Bestimmtes geschehen soll, muß dazu ein **Befehl** gegeben werden. Ein Editorbefehl hat die folgende allgemeine

Form: #**bxxf** **<↵**

Dabei ist # das automatisch vorgegebene **Promptzeichen**

b das **Befehlszeichen**, siehe unten

xx die **Kategorie-Nummer** (bei einigen Bef. nur **x** oder keine Nummer)

f **Leerzeichen** (Leertaste) oder ein **Folgezeichen** (bei mehrfach besetzbaren Kategorien, s.o., oder **3. Stelle der Kat.Nr.**

Ein Befehl dieser Form wirkt nur auf die Kategorie #xxf. Bei Befehlen, die sich auf den ganzen Datensatz beziehen, ist nur das Befehlszeichen (ohne xxf) einzugeben.

Folgende Befehle gibt es : (knappste Übersicht, Abruf im Programm mit Befehl **#hb**)

```
  l? : aktuellen Datensatz am Bildschirm anzeigen (= F5)
  a? : alte Kategorien  (= Alt+F6)        t$ : alte Kategorie übernehmen
  z1 : erste Aufn.   zm : markierte Aufn. zl : letzte Aufn. zeigen
  -  : eine Aufn. zurückblättern: <Bild↑> +  : eine weiterblättern: <Bild↓>
  r  : Rückkehr zum Menü                  i  : Aufn. vor aktuelle einfügen
  v$ : Kategorie vergessen                V  : aktuelle Aufnahme löschen
  fX : X im Datensatz+Untersätzen suchen _X_Y_ oder ,X,Y,: X durch Y ersetzen
       (ab dem aktuellen Satz)             (im aktuellen Datensatz)
  X  : Ersetzungsbefehle definieren (für automat. Suchen u. Ersetzen)
  M  : diese Aufnahme markieren           m  : mark. A. hinter akt. einfügen
  b? : Bearbeitung im Fenster             k  : mark. A. hinter akt. kopieren
  d  : Druckfunktionen (Menü) = F2        D  : wie d, aber Bildschirmausgabe
  p  : Phrasen/Param. laden etc. (Menü)   x  : Sonderbefehle (Menü)
  E  : Erfassungs-Abfrageliste aufrufen   #  : Aufn. abschl./speichern = F9
  e  : Ende der Erfassung/Bearbeitung     s  : Speicherzustand anzeigen
 Esc*: Phrase * abrufen              Esc Esc: Phrase speichern
   (* = beliebiges Zeichen = Zeichen hinter # bei ESC ESC, siehe Befehl #p)
für ? kann eine Kat.Nr. xx oder deren erste Ziffer x gesetzt werden
       wenn nicht, werden alle Kategorien angesprochen
für $ muß eine vollständige Kat.Nr. (evtl. mit Folgezeichen) stehen
für X bzw. Y darf jede beliebige Zeichenfolge stehen, Y kann auch entfallen
Die Befehle   + - f k m   werden durch  <↵  wiederholt.
```

Einige Befehle (m,M,f,z) sind im Schnellzugriff kaum relevant, weil man dort jeweils nur eine Aufnahme(gruppe) im Arbeitsspeicher hat, bei den anderen Programmen jedoch u.U. hundert und mehr. Wenn man allerdings umfangreiche, hierarchisch gegliederte mehrbändige Werke hat, also Aufnahmegruppen mit vielen Untersätzen, sind die genannten Befehle auch im Schnellzugriff nützlich.

Hinweis: Die Begriffe "Aufnahme" und "Datensatz" oder "Satz" werden gleichwertig verwendet.

Die Editor-Befehle sind auch benutzbar, wenn man in den Programmen 4 - 6 einen laufenden Vorgang mit 'x' unterbricht und dann in den Editor gelangt. Die zu dem Zeitpunkt im Arbeitsspeicher stehenden Daten sind dann genauso bearbeitbar wie bei einer Erfassungssitzung. Mit Befehl #r wird der unterbrochene Vorgang dann wieder fortgesetzt.
Die **aktuelle Aufnahme** ist der Datensatz, der gerade auf dem Bildschirm steht bzw. an der man gerade arbeitet. Bei Befehl `#l` oder F5 bekommt man sie in der korrekten Reihenfolge der Kategorien angezeigt.
Alle Fragen, die das Programm stellt, sowie alle laufenden Vorgänge (Einlesen von Daten, Ausdrucken etc.) können durch Druck auf 'x' (in SRCH oder IMPORT) unterbrochen oder abgebrochen werden. Man ist also weitgehend gegen irrtümliche Befehlsausführung gesichert.
Zu jedem Befehl finden Sie im Teil 3 dieses Kapitels eine eigene Seite mit ausführlicher Beschreibung, alphabetisch geordnet. Zuvor aber eine

Systematische Befehlsübersicht: (einige Befehle liegen zusätzlich auf Funktionstasten)

Befehlsgruppe	Bef.	F-Taste	Bedeutung
1. **Anzeige**befehle	**#l**	**F5**	Satz ganz oder teilweise anzeigen
	#a		Hintergrundspeicher anzeigen
2. **Blätter**befehle	**#+**	**<Bild↓>**	nächste Aufnahme/Unteraufnahme anzeigen
	#-	**<Bild↑>**	vorige Aufn./Unteraufn. anzeigen
	#z		bestimmte Aufn./Unteraufn. anzeigen
	#f		bestimmte Aufn./Unteraufn. suchen
3. **Export**befehle	**#d #D**	**F2/F3/F4**	aufbereitete Ausgabe Drucker/Bildschirm
4. **Korrektur**befehle	**#_ #,**		Suchen und Ersetzen
	#b		Bequem-Bearbeitung in einem Fenster
	#X		Ersetzungsbefehle definieren für Automatik
5. **Kopier**befehle	**#t**		einzelne Kategorien kopieren (Transfer)
	#k		ganze Aufnahme kopieren (vorher #M)
6. **Bewegung**sbefehle	**#M**		aktuelle Aufn. markieren
	#m		#M-Aufn. hinter die aktuelle verschieben
	#i		neue Aufn. vor die aktuelle einfügen
7. **Lösch**befehle	**#v**		einzelne Kategorie löschen (vergessen)
	#V		ganze Aufn. (nur im Arbeitsspeicher!)
8. **Phrasen**- und	**#p**		Phrasen oder Parameter laden/speichern
Parameterbefehle	**<Esc>**		Phrasen definieren/abrufen/löschen
9. **Speicher**befehle	**F9**		Aufn./Unteraufnahme abspeichern
	F10		Rückkehr zum Menü mit/ohne Speicherung
	F8		Abbruch der Bearbeitung oder Eingabe
10. Sonstiges	**#E**		Erfassungs-Abfrageliste aufrufen
	#q	**F6**	Index aufschlagen (nur Progr.1)
	#s	**Alt+F7**	Speicherzustand anzeigen
	#h	**F1**	HELP
	Esc 0 / Esc Leer		Datum/Uhrzeit in Aufn. speichern

3.5.3 DOS-Editor-Befehle : Alphabetische Übersicht

Wozu dieser Abschnitt? Jeder DOS-Editor-Befehl wird genau erklärt. Anwendbar in PRESTO und SRCH16.
(In Abschnitt 3.5.0 findet man, was die Angaben *WICHTIG*, *nützlich* usw. bedeuten)

Befehl #a **Befehl #a**
nützlich *nützlich*

a99 : **Alt+r** (Auswahlfeld) bzw. **F5** (Anzeigefeld)

Bedeutung: alte Kategorien (sog. Hintergrundspeicher) anzeigen

Eingabe	bewirkt	siehe auch →
a	Anzeige der "alten" Kategorien. Das sind diejenigen, die beim letzten Speicherbefehl besetzt waren oder vom Funktiosnmenü des Schnellzugriffs mit Shift+F5 oder Alt+F5 in den **Hintergrundspeicher** kopiert wurden. Sie stehen dort bereit zur Übernahme mit dem Befehl #t (z.B. #t40 zur Übernahme von #40). Auch bei #V wird dieses Kopieren ausgeführt, d.h. man kann mit #t alles zurückholen. Mit #a kann man sich also jederzeit überzeugen, ob etwas Verwendbares im Hintergrundspeicher steht. Die angezeigten Datenelemente können auch mit Cursor bearbeitet und dann mit <┘ vom Bildschirm übernommen werden.	## F10 #t #v
ax	Anzeige der Kategorien mit der Anfangsziffer x. Manchmal möchte man sich z.B. nur die Erscheinungsangaben ansehen. Das sind die #70-er Kategorien. Gibt man a7, so werden nur diese angezeigt.	#lx
au	*Sonderfall:* Gibt man `#au` , bekommt man die Anwendervariablen `#uxy` zu sehen.	
axx	Gibt man z.B. a20, so wird nur die Kategorie #20 aus dem Hintergrundspeicher angezeigt. Ist eine Kategorie mehrfach besetzt, was z.B. bei #40 vorkommen kann, so werden alle vorhandenen Eintragungen ausgegeben also z.B. #40, #402, #403, usw. *Sonderfall:* Mit `#aux` kommen die Anwendervariablen, die mit `#ux` anfangen.	#lxx

Hinweise:
Der Zweck dieses Befehls ist die Erleichterung der Erfassung von Materialien mit einem hohen Anteil von identischen Elementen in einzelnen Kategorien.
Im Prinzip sind immer die letzten 3 Aufnahmen im Hintergrundspeicher verfügbar. Die Kategorien der "älteren" Aufnahmen erhalten andere Anhängeziffern. Wenn man z.B. unter #74b eine brauchbare Eintragung sieht, fährt man entweder hin, setzt ein Leerzeichen auf das 'b' und drückt Enter, oder man gibt den Befehl `#t74b,74`.
Auf Farbbildschirmen werden die Hintergrundkategorien in der Farbkombination wm (normal grün auf schwarz) angezeigt, so daß sie sofort von den Daten der aktuellen Aufnahme (Farbe wd, normal hellblau auf schwarz) zu unterscheiden sind (→ Anh.A.3).

Befehl #b gleichwertig: **w** **Befehl #b**
nützlich *nützlich*

Bedeutung: bequemes Bearbeiten in einem Fenster

Mit diesem Befehl gelangt man in die **Fenster-Bearbeitung**. Sinnvoll ist dieser Befehl für sehr lange Kategorien (über 2.000 Zeichen), die man sonst nur mit dem Such- und Ersetzungsbefehl (#_) bearbeiten könnte, weil sie nicht als Ganzes auf den Bildschirm passen. Solche Kategorien können z.B. beim Import von "download"-Daten entstehen (abstracts!).

Außerdem ermöglicht es der Befehl, wenn ohne Kategorienummer eingegeben, alle Kategorien einer Aufnahme hintereinander zu bearbeiten, ohne jeweils mit dem Cursor immer wieder nach oben fahren zu müssen, wie es sonst nötig ist.

Eingabe	bewirkt	→

bxxf Kategorie #xxf erscheint in einem Fenster in der Mitte des Bildschirms, der Cursor steht hinter dem letzten Textzeichen. Wenn der Text nicht vollständig ins Fenster paßt, erscheint nur der letzte Abschnitt. Mit den Cursortasten kann man sich problemlos im gesamten Kategorietext bewegen, mit **F8** kann man die Bearbeitung abbrechen (Kategorie bleibt dann unverändert).
Die Tasten **Pos1, Entf, Einfg, Ende** haben die bei Textprogrammen übliche Wirkung.
Insbesondere schaltet **Einfg** den Einfüge-Modus ein und aus, was am rechten oberen Rand auch angezeigt wird. (Im normalen Editormodus setzt **Einfg** ein Leerzeichen an der Cursorstelle ein.)

Mit 2maligem **<Esc>** entkommt man aus diesem Modus. Ansonsten hat **<Esc>** auch während einer horizontalen Bearbeitung die Wirkung einer Phrasen-Taste. #p

b Alle Kategorien der aktuellen Aufnahme werden hintereinander im Fenstermodus zum Bearbeiten vorgelegt. Bei jedem <Enter> wird die Kategorie in den Speicher übernommen und die nächste angezeigt. Mit **F9** wird zur vorangehenden Kategorie zurückgesprungen.
Mit **F8** bricht man diese Bearbeitung ab.

Hinweise:

- Im Programm 1 ist es möglich, mit **F6** einen "Ausflug" in den Index zu machen. Man sucht dort ganz normal, übernimmt evtl. mit **<Esc>**+Buchstabe Registerzeilen in den Phrasenspeicher (→ Befehl #p) und kehrt mit <Enter> zurück. Wenn man innerhalb des Textes <Strg>+<Enter> drückt, wird die Registerzeile, auf der der Cursor zuletzt stand, an der Cursorposition eingefügt.
- Mit **F1** erhält man eine Übersicht aller Tastenfunktionen, die es in der Fenster-Bearbeitung gibt.
- *Achtung:* Das Löschen einer Kategorie ist **nicht** dadurch möglich, daß man einfach das ganze Fenster leer macht. Das hat keine Wirkung. Man muß die Kategorienummer stehenlassen, oder den Befehl **#v** benutzen. #v

Befehl #E — Befehl #E

nützlich — *nützlich*

a99 : **Alt+y 2mal**

Bedeutung: Erfassungs-Abfrageliste aufrufen

Wenn die Abfrageliste zur Erfassung einer neuen Aufnahme durchlaufen ist, kann man alle noch fehlenden Kategorien mit Nummer eingeben. Es kann aber auch sein, daß man verfrüht mit 'x' aus der Abfrageliste ausgestiegen ist oder einzelne Fragen mit <Enter> leer weggedrückt hat. Wenn man den Befehl #E gibt, werden die in der aktuellen Aufnahme noch fehlenden Kategorien nochmals abgefragt.

Eingabe	bewirkt	→

E Die Abfrageliste wird erneut durchgearbeitet. Es werden aber nur diejenigen Kategorien abgefragt, die in der aktuellen Aufnahme noch nicht belegt sind.

Hinweise:

- Die schon vorher eingegebenen Kategorien korrigiert man entweder mittels Cursor auf dem Bildschirm, oder man gibt den Befehl #b, worauf man sie einzeln für die "horizontale Bearbeitung" vorgelegt bekommt. #b

- Innerhalb der Abfrageliste kann man einen Abschnitt für seltener vorkommende Kategorien einrichten.

Wenn dieser z.B. in der Typenliste die Ziffer 9 hat, wählt man nach Eingabe von #E zunächst die 9, dann wird diese Sonderliste durchgearbeitet (→ Anhang A).

- Die abgefragten und eingegebenen Kategorien gelangen alle in die aktuelle Aufnahme, auch wenn man #E mehrfach aufruft. Nur der Typ u = unselbständiger Band bewirkt, daß eine neue Bandaufnahme begonnen wird. Sie kommt ans Ende. Sonst muß man Befehl #I benutzen. #I
- Wenn man die aktuelle Aufnahme **abschließen** und eine **neue** beginnen will, muß man zuerst **F10** geben: #r
  - in PRESTO wird dann die aktuelle Aufnahme abgespeichert und der Index aktualisiert; mit Funktion 'I' startet man die nächste Erfassung (→ Kap.1.5);
  - in den anderen Programmen kommt dann automatisch die Abfragliste für die nächste Aufnahme (2 und 3) oder das Programm läuft weiter (4 und 5).

Befehl #f Befehl #f

***a99*: Strg+f**

Bedeutung: Finde im Arbeitsspeicher eine vorgegebene Zeichenfolge

In den Programmen 3, 4 und 5 beobachtet man im Editor folgendes:
Wenn eine größere Datenmenge im Arbeitsspeicher versammelt ist, wird es langweilig, mit den Befehlen #+ und #- oder <Bild↓> / <Bild↑> zu einer gesuchten Aufnahme vor- oder zurückzustoßen. Sehr viel schneller arbeitet dann der Befehl #f:

Eingabe	bewirkt	→
fX	Beginnend bei der aktuellen Aufnahme wird im Arbeitsspeicher nach der Zeichenfolge X gesucht. X kann beliebig lang sein und beliebige Zeichen enthalten. Es kann also ohne weiteres auch ein Wortbruchstück oder nur ein einzelnes Zeichen sein. Sobald das Programm irgendwo auf X stößt, zeigt es die betreffende Aufnahme an und macht sie zur aktuellen Aufnahme, so daß man gleich daran arbeiten kann. Ist es aber nicht die richtige, weil X evtl. mehrfach vorkommt: <↵ drücken und die Suche geht weiter. Das ist wiederholbar, bis man schließlich am Ziel ist. Oder es kommt die Meldung: "nichts gefunden".	

Hinweise:

- Wie beim Befehl _X_Y_ dient auch hier das Fragezeichen '?' zur Maskierung: #_
  mit **`#fMa?er`** findet man `Mayer, Maier, Maler` etc.
- Groß- und Kleinschreibung sowie Umlaute müssen korrekt eingegeben werden: mit **`#fmueller`** würde man `"Müller"` nicht finden, wohl aber `"Oelmueller"`.
- Dieser Befehl ist **im Schnellzugriff nur dann sinnvoll**, wenn man eine größere Aufnahme(gruppe), d.h. ein umfangreiches mehrbändiges Werk, im Arbeitsspeicher hat. #f sucht nicht in der Datenbank, sondern nur im Arbeitsspeicher!
- Mit **`#s`** sieht man, wieviele Aufnahmen im Arbeitsspeicher sind. #s

Befehl #h gleichwertig: **F1** **Befehl #h**
nützlich *nützlich*

***a99*: F1, Alt+h** und Button **[?]**

Bedeutung: Hilfe

Eingabe bewirkt →

F1 oder
h Es erscheint die Hilfeseite H mit den wichtigsten Befehlen
hb die vollständige Befehlsübersicht (Hilfeseite HB, → 3.2)
hxyz Der Hilfetext mit Namen HXYZ wird angezeigt, z.B. HF mit der Beschreibung der Funktionstasten.

Hinweise:

- **F1** ist abhängig von der Kategorie, in der sich gerade der Cursor befindet: ist er z.B. in der Kategorie #331, erscheint Hilfeseite H331; wenn es die nicht gibt, sucht das Programm nach H33, dann nach H3. So kann man detaillierte Hilfeseiten für einzelne Kategorien anlegen, aber auch sich auf Hilfeseiten für die Kategoriegruppen beschränken. Steht dagegen hinter dem '#' ein Befehlsbuchstabe, wird die zu diesem gehörige Hilfeseite gezeigt, z.B. HV wenn es #v ist.

- Sie können beliebige Hilfetexte selbst anlegen:
  - ein Hilfetext muß eine ASCII-Datei sein, wie man sie mit jedem Texteditor erstellen kann,
  - der Name einer Hilfedatei muß mit 'h' beginnen und darf keine Typkennung ("extension") haben. An den Namen kann man die Sprachbezeichnung (normalerweise GER hängen. Zuerst sucht das Programm nach dem Namen MIT, dann OHNE Sprachbezeichnung.
  - eine Hilfedatei sollte nicht länger als 23 Zeilen sein
  - die Datei muß im aktuellen Datenverzeichnis oder im Programmverzeichnis \ALLEGRO liegen. Gesucht wird zuerst auf dem Datenverzeichnis, dann auf dem Programmverzeichnis.

  Jede Datei, die diesen Bedingungen genügt, kann mit dem Befehl #h als Hilfetext aufgerufen werden.

- Eine Hilfedatei "hk" enthält eine Kurzübersicht aller verwendeten Kategorien. Ändern Sie diese bei Bedarf ab. Ändern Sie diese Datei, wenn Sie ein anderes Schema benutzen.

- Für alle Befehle und Kategoriegruppen werden Hilfedateien mitgeliefert. Geben Sie also im Erfassungsmodus die Befehle #hv bzw. #h6 ein, dann sehen Sie den Effekt.

- Wird die angeforderte Hilfeseite nicht gefunden, zeigt das Programm als Ersatz die Hilfeseite H1C. (Mehr zu den Hilfeseiten → Kap.0.3)

Befehl #i,#I **Befehl #i,#I**
(wichtig) *(wichtig)*

a99 : **F9 "hierarchischer Untersatz"**

Bedeutung: Einfügen einer neuen Aufnahme oder Unteraufnahme (insert)

Eingabe bewirkt →

---

i **vor** die gerade angezeigte (aktuelle) Aufnahme kann nach diesem Befehl eine neue Aufnahme eingefügt werden. Es wird zunächst automatisch eine leere Kategorie #00 vorgegeben, d.h. man hat eine aktuelle Aufnahme, die aus nichts weiter als #00 besteht. Alle benötigten Daten sind nun ganz normal einzugeben, Abschluß mit Befehl ##.

I dasselbe, jedoch wird die Abfrageliste benutzt, während mit 'i' die Eingabe mit der Hand gemacht werden muß, d.h. jede Kategorie mit Nummer einzugeben ist.

Hinweise:

- Im Arbeitsspeicher werden die vorher aktuelle Aufnahme und alle nachfolgenden während der Eingabe automatisch nach "hinten" geschoben, deren Reihenfolge verändert sich also nicht.

- Um die Reihenfolge der bereits eingegebenen Aufnahmen zu ändern benutzt man die Befehle #M und #m. #M #m

- Jede (Unter-)Aufnahme läßt sich mit dem Befehl #V auch wieder völlig aus dem Arbeitsspeicher entfernen. #V

- Einfügen eines neuen Bandes, insbes. im Schnellzugriff:
  Im Schnellzugriff ist Befehl #i/#I **nur** zu benutzen, um im Arbeitsspeicher innerhalb eines mehrbändigen Werkes, (das immer als Ganzes, als Aufnahmegruppe eingelesen wird) einen neuen Band zwischen mehrere schon vorhandene einzufügen. Keinesfalls kann man im Schnellzugriff eine neue Hauptaufnahme zwischen zwei schon vorhandene mitten in eine Datei einfügen, das geht nur bei .ALG-Dateien im Redigierzugriff! Man "fährt" mit den Befehlen #+ und #- auf den Band, **vor** welchen der neue einzufügen ist, gibt #i und dann die Kategorie #01 mit der Nummer des neuen Bandes, sodann die weiteren Angaben zu dem betr. Band, genauso wie bei einer selbständigen Aufnahme.

  Die neuen Möglichkeiten der Satzverknüpfungen, erstmals mit Version 11.2 eingeführt, werden es bei konsequenter Anwendung ermöglichen, auf hierarchische Sätze mit Unteraufnahmen völlig zu verzichten. Einige Parameterdateien für Anzeige und Druck gibt es ab Version 12.2.

- In ***a99*** muß man nach Eingabe eines neuen hierarchischen Untersatzes diesen evtl. noch verschieben, denn er wird am Ende angehängt, wenn es schon mehrere Untersätze gibt. Das Verschieben geht so:
  1. Balken auf die #01 des zu verschiebenden Untersatzes setzen
  2. Alt+m zum Markieren
  3. Balken in denjenigen Untersatz (nicht auf seine #01) setzen, hinter den der markierte Untersatz zu verschieben ist
  4. Nochmals Alt+m

Befehl #k Befehl #k

a99 : **Alt+k**

(vorher mit Alt+m den zu kopierenden Untersatz markieren)

Bedeutung: Kopieren (Verdoppeln) von (Unter-)Aufnahmen

Eingabe	bewirkt	→
k	Ein Duplikat der mit Befehl #M markierten Aufnahme wird im Arbeitsspeicher hinter die aktuelle Aufnahme kopiert.	#M

Hinweise:

- In **PRESTO** ist #k **nur** bei **mehrbändigen** Werken anzuwenden, und zwar um Kopien von Bandaufführungen (Unteraufnahmen) zu machen, nicht von der Hauptaufnahme, denn:
  Man erzeugt sich Kopien von einzelnen Aufnahmen nicht im Editor, sondern vom Anzeigebildschirm aus mit **'C'** (→ 1.5).
  #k ist also ein hauptsächlich für die (nicht mehr relevante) "offline"-Erfassung konzipierter Befehl, nützlich aber auch für das Kopieren von Bandaufführungen.

- Der Befehl #k ist durch mehrmaliges <↵ beliebig wiederholbar. Dadurch lassen sich mehrere identische Unter-)Aufnahmen erzeugen, die man anschließend bearbeiten, also durch individuelle Daten ergänzen kann, indem man mit den Befehlen #+ (<Bild↓>) und #- (<Bild↑>) von einer zur anderen fährt und dann unterschiedliche Eingaben macht.

Sonderfälle:

- Die aktuelle Aufnahme ist zugleich die markierte: hinter diese Aufnahme wird eine Kopie ihrer selbst geschrieben.
- Wenn die markierte Aufnahme hinter der aktuellen steht, passiert nichts.

Befehl #L gleichwertig: **F5** **Befehl #L**
WICHTIG *WICHTIG*

***a99*: F5**

Bedeutung: aktuelle Aufnahme auflisten (anzeigen)

Eingabe bewirkt →

L die aktuelle Aufnahme wird am Bildschirm angezeigt,alle Kategorien in der richtigen Reihenfolge.
l

Wenn man 'L' benutzt, werden Kategorien mit mehr als 76 Zeichen verkürzt ausgegeben, und zwar auf 76 Stellen mit drei blinkenden Punkten dahinter. Der Sinn ist, daß man auch längere Aufnahmen auf diese Weise noch als ganzes überblicken kann. Wenn man eine abgeschnittene Kategorie dann vollständig sehen will, gibt man z.B. #l20, um die gesamte Kategorie #20 zu Gesicht zu bekommen.

Lx Anzeige der Kategoriegruppe x, d.h. aller Kategorien mit der Anfangsziffer x. Manchmal möchte man sich z.B. nur die Erscheinungsangaben ansehen. Das sind die #70-er Kategorien.
Gibt man L7, so werden nur diese angezeigt. #ax
Oder die Personennamen: dann gibt man L4 und L5.

Lxx Gibt man z.B. L20, so wird nur die Kategorie #20 der aktuellen Aufnahme angezeigt. Ist eine Kategorie mehrfach besetzt, was z.B. bei #40 vorkommen kann, so werden alle #axx
vorhandenen Exemplare ausgegeben, bei #l40 also z.B. #40, #402, #403, #40a usw.

Ld Anzeige der aktuellen Aufnahme und aller im Arbeitsspeicher #dd
folgenden mit Zeilenumbruch. Abbruch mit 'x' .

Hinweise:

- Man gewöhnt sich bei der Erfassungsarbeit sinnvollerweise an, den Befehl #L oder F5 immer vor dem Abschlußbefehl ## noch einmal zu geben, um zu sehen, ob man nichts vergessen hat. Nützlich ist das dann, wenn man die Kategorien nicht in der richtigen Reihenfolge eingibt und wenn man die Befehle #t und #v benutzt hat. Das Bild auf dem Schirm ist dann evtl. etwas unübersichtlich geworden, obwohl im Arbeitsspeicher alles seine Ordnung hat.

- Wenn die Aufnahme **übermäßig lang** ist, so daß sie nicht mehr auf den Schirm paßt, sind Lx und Lxx sehr brauchbar.

- Der Befehl #L wirkt nur auf die aktuelle Aufnahme. Um eine andere anzeigen zu lassen, benutzt man die Befehle #z #+ #- #f <Bild↓> <Bild↑> #z
(in den Programmen 3 bis 6, wenn man mehrere Aufnahmen im Arbeitsspeicher hat; im Schnellzugriff nur bei mehrbändigen Werken). #f
Die dann angezeigte Aufnahme wird dadurch aber sofort zur aktuellen
Aufnahme, auf die sich dann alle Eingaben und Befehle beziehen!
Mit den gerade genannten Befehlen #f, #+ und #- muß man zuerst zu der vorher aktuellen Aufnahme zurückkehren, um daran weiterzuarbeiten.

- Mit #Ld kann man, wenn vorher #z1 gegeben wurde, den gesamten Arbeitsspeicher am Bildschirm durchlaufen lassen. Zu jedem Zeitpunkt kann der Durchlauf mit Druck auf x abgebrochen, mit <Leertaste> unterbrochen werden. Dann wird gefragt, ob man zur aktuellen Aufnahme zurückkehren will (<↵ drücken) oder die zuletzt angezeigte nun die aktuelle sein soll (# drücken).

- `#L und #l` sind fast gleichwertig (s.o.), man wird sogar meistens der Bequemlichkeit wegen das kleine 'l' nehmen. In dieser Beschreibung wurde 'L' geschrieben, weil sich das kleine 'l' nicht deutlich genug von der Ziffer '1' und dem großen Buchstaben 'I' unterscheidet.

Befehle #M,#m Befehle #M,#m

a99 : **Alt+m**

(siehe auch Befehl #i)

Bedeutung: Markieren und verschieben (move) von Aufnahmen

Eingabe bewirkt →

M die gerade angezeigte (aktuelle) Aufnahme wird **markiert** d.h. das Programm "merkt sich" deren Nummer, damit sie danach mit den Befehlen #m, #k und #zm #k
verwendet werden kann. #z

m die markierte Aufnahme wird hinter die gerade aktuelle eingesetzt und an ihrem alten Platz gelöscht. Die Markierung geht dann automatisch auf die Aufnahme über, die vorher auf die markierte folgte und nun auf deren Platz gerückt ist.

Hinweise:

- Im Schnellzugriff, wo zu jedem Zeitpunkt nur eine Aufnahme(gruppe) im Arbeitsspeicher steht, ist der Befehl nur zum Vertauschen von Bandaufführungen anwendbar: man greift mit **'E'** auf das mehrbändige Werk zu, fährt mit #+ zu dem zu verschiebenden Band, gibt #M, fährt zu dem Band, hinter den der markierte einzufügen ist, und gibt #m.

- Es kann stets nur eine Marke gesetzt werden. Zum Wiederfinden von Aufnahmen im Arbeitsspeicher setzt man ansonsten den Suchbefehl #f ein. #f

- Dieser Befehl ist wiederholbar, indem man sofort danach <┘ drückt. Daher kann man eine Gruppe von Aufnahmen als Ganzes im Arbeitsspeicher an eine anderen Stelle bringen: man markiert die erste der Gruppe, fährt dann zu der Aufnahme, hinter die diese Gruppe gebracht werden soll, gibt einmal den Befehl #m und dann noch entsprechend oft <┘ .

- Die Markierung gilt nur im Arbeitsspeicher und wird nicht in die Datei übernommen. Bei erneutem Redigieren ist also eine gesetzte Markierung nicht mehr vorhanden.

Sonderfälle:

- Die markierte Aufnahme ist identisch mit der aktuellen: dann wird sie mit der vorangehenden vertauscht. Dadurch kann man sehr leicht zwei aufeinanderfolgende Aufnahmen miteinander vertauschen: man steuert die zweite der beiden an und gibt hintereinander die Befehle #M und #m .

- Die markierte Aufnahme steht bereits hinter der aktuellen, wo sie hin soll: dann passiert nichts.

- Die markierte Aufnahme ist die letzte im Arbeitsspeicher: nach der Verschiebung geht die Markierung auf die vorletzte über, die ja dadurch zur letzten wird.

Befehl #p **Befehl #p**
wichtig *wichtig*

a99 : **p in der Befehlszeile eingeben**

Hilfe zu den Phrasenfunktionen: `h phras`

Bedeutung: Phrasen ausdrucken, speichern, laden;
und: Export-Parameter laden und ändern

Eine **Phrase** ist ein Wort, eine Wortgruppe, oder allgemein eine beliebige Zeichenfolge. ***allegro-C*** ermöglicht das Speichern von Phrasen zwecks beliebiger Verwendung im Eingabetext. Als Funktionstaste für das Definieren, Verwenden und Löschen von Phrasen dient die **<Esc>**-Taste:

a) **neue Phrase definieren:** (siehe auch 1.4: Indexeintragungen als Phrase)
- man setzt hinter das Prompt '#' einen Kennbuchstaben oder auch ein beliebiges anderes Zeichen,
- schreibt den Phrasentext (Länge beliebig) und
- drückt dann **zweimal <Esc>.** Dann wird bestätigt, daß unter dem angegebenen Kennbuchstaben nun diese Phrase gespeichert ist.

Beispiel: #TTechnische Universität <Esc> <Esc>
erzeugt eine Phrase mit dem Kennbuchstaben T.

b) **Phrase verwenden:**
An beliebiger Stelle während einer Eingabe tippt man **<Esc> <Kennbuchstabe>** : sofort erscheint diese, der Cursor steht dahinter und die Eingabe kann weitergehen.
Beispiel: bei **<Esc> T** erscheint Technische Universität.

c) **Phrase löschen:**
- Man gibt nur das Kennzeichen direkt hinter dem # ein
- und dann **`<Esc> <Esc>`** .

Beispiel: **`#T <Esc> <Esc>`** löscht die Phrase T, die oben definiert wurde

Jedes Zeichen, das per Tastatur eingebbar ist, kann als Phrasen-Kennzeichen benutzt werden, insbesondere können Groß- und Kleinbuchstaben mit unterschiedlichen Phrasen belegt werden!
Stets benutzbar sind zwei **Sonderphrasen**: mit **`<Esc> 0`** erhält man Datum/Uhrzeit, mit **<Esc> <Leertaste>** nur das Datum, wenn man in der .CFG den Befehl D8 setzt (→ Anh.A.1.3).

Die gespeicherten Phrasen können als **Phrasendatei** auf Diskette/Platte gespeichert und dann immer wieder verwendet werden. Man kann sich die Phrasen auch auf dem Bildschirm als Liste anzeigen oder auf dem Drucker ausgeben lassen. Alles dies leistet der Befehl #p. Jedoch:

Den momentanen Inhalt des Phrasenspeichers sichert das Programm am Ende in einer Datei mit Namen **`PHRASE.APH`** auf dem aktuellen Verzeichnis. Bei Beginn einer neuen Sitzung wird sie automatisch geladen, so daß die Phrasen permanent sind. Gesucht wird eine Phrasendatei immer zuerst auf dem Aufrufverzeichnis, dann auf dem Datenbank, dann auf dem Programmverzeichnis. Natürlich kann man die Datei löschen, um die Phrasen loszuwerden.

Parameter für Datenausgabe (Export) werden detailliert in Kap.10 erläutert. Man benötigt sie, um Aufnahmen z.B. in Karten- oder Listenform auszugeben. Solche Parameter stehen in eigenen Dateien, z.B. P-KARTE.APR für die Produktion von Katalogkarten nach internationalem Standard. Dabei ist 'P-KARTE' der Name der Parameterdatei 'A' der Kennbuchstabe des Kategoriesystems A.CFG.

p Sie haben die Wahl zwischen 6 Funktionen:

```
a = Phrasen/Parameter anzeigen      d = drucken
s = Phrasen speichern               l = Phrasen laden
e = Export-/Druckparameter laden    i = Dr.init.
x = Irrtum
```

diese bewirken im einzelnen folgendes

a Sie können sich verschiedene Dinge anzeigen lassen: Phrasen, Zwischenteile (→ 10.2.0), Ersatzdarstellungen (→ 10.2.4) und die momentan eingestellten Ersetzungsbefehle. #X
Wenn mehrere Parameter-Sets geladen sind, erscheinen zusätzlich auch die Set-Nummern.

d Die Ausgabe erfolgt in gleicher Weise auf dem Drucker

s Die gesamten Phrasen werden als Datei auf Diskette/Platte abgespeichert. Das Programm fragt dann zunächst nach Laufwerk und Name für die Phrasendatei. Wenn Sie z.B. "`phrs`" als Name angeben, heißt die Datei anschließend `PHRS.APH`
Sie können eine Phrasendatei auch mit einem Textprogramm bearbeiten. Es ist leicht durchschaubar, wie eine solche Datei strukturiert ist. (***CockPit*** **`pd <Phrasendateien>`**)

l Eine mit s vorher abgespeicherte Phrasendatei kann geladen und dann wiederverwendet werden. Es können mehrere Phrasendateien hintereinander geladen werden. Tritt ein Kennbuchstabe darin mehrfach auf, dann gilt immer die Phrase der zuletzt geladenen Datei. Beim Einlesen werden die Phrasen mit ihren Kennzeichen am Bildschirm angezeigt und ob evtl. schon existierende Phrasen dabei ersetzt (überschrieben) werden.

i **Drucker-Initialisierung:** Der Parameter **`di`** aus der aktuellen Parameterdatei wird an den Drucker ausgegeben (→ 10.2.4).
Tip für Experten: um vom Editor aus (online) einen Befehl an den Drucker abzugeben, verfahre man so: zuerst den Druckerbefehl definieren mit
`#pdi=CS` (für CS setzt man die gewünschte Steuerzeichenfolge, → 10.2.0)
dann:
`#p <Enter>` **`i`** zum Absetzen des Befehls
Dieselbe Funktion ist im Druckmenü (Befehl #d) enthalten. #d

Hinweise: Mit der Option **`-t`** veranlaßt man das Laden einer Phrasendatei schon beim Programmaufruf. Wenn z.B. **`presto ... -t phrs`** gegeben wird, dann sind anschließend die Phrasen der Datei `PHRS.APH` sofort benutzbar. (→ Kap.12)
Man benötigt ab Version 12.2 die Unterfunktionen p und l nur noch, wenn man für besondere Zwecke eigene Phrasensammlungen anlegen will, denn ansonsten sind ja die einmal definierten Phrasen permanent verfügbar (s.o.).

`e` Es erscheint eine Auswahl-Liste (→ 0.5) der wählbaren Parameterdateien. Man fährt mit dem Cursor zu der gewünschten hin, z.B. **P-NORMAL**, setzt ein '+' davor und drückt <Enter>. Die Parameterdatei wird sofort in den Arbeitsspeicher geladen. Danach haben Sie z.B. P-NORMAL zusätzlich zu D-1 verfügbar. Über das Druckmenü schalten Sie zwischen beiden um. #d
Benutzt man diesen Befehl mehrmals, so wird jeweils gefragt, ob die Parameter zusätzlich geladen werden sollen. Es können nämlich bis zu 4 Sätze (Sets) von Parametern gleichzeitig geladen werden. Die Umschaltung zwischen diesen Sets erfolgt über den Befehl #d. (→ Kap.10.2.11.6)
Nach dem Laden kommt die Frage:

```
Ausgabe auf (Laufwerk:)(Pfad\)Datei:
(x = keine Ausgabe gewünscht)
```

Man gibt hier x, wenn man nur drucken oder das Ergebnis auf dem Bildschirm betrachten will. Ansonsten gibt man einen Dateinamen. In diese Datei wird dann jeweils die Ausgabe beim Befehl #d geschrieben (**Export**).

x die Funktion #p wird ohne Aktion verlassen.

Hinweise:

- Gibt man hinter dem **#p** eine Parameterangabe ein (→ 10.2.1-5 und 10.3.1) so wird der betreffende Parameter im aktuellen Set sofort geändert. Z.B. würde **#pzl=70** bewirken, daß die Zeilenlänge auf 70 Zeichen geändert wird. Alle Parameter bis auf die Angaben zu den einzelnen Kategorien können mit diesem Befehl geändert werden. Siehe dazu auch oben die Bemerkung zum Unterbefehl **i** .
  Ein wichtiger Fall: mit **#pdx=1** schaltet man die Druckercodierung auch für die Bildschirmanzeige ein. Wenn man Einzelbände von mehrbändigen Werken drucken will, muß man so verfahren, falls Druckersteuerungen Verwendung finden (Fettschrift u.ä.). Man gibt dann diesen Befehl im Editor, kann anschließend mit **F3** vom Anzeigebildschirm aus (→ 1.5) alle gewünschten Bände drucken, und schaltet dann evtl. mit **#pdx=0** die Codierung wieder ab.

- Für Exportzwecke kann jedem geladenen Set eine andere Datei zugeordnet werden. Man kann also in einem Zuge mehrere ganz verschiedene Ausgabedateien herstellen. (→ Kap.4 und 6/7; Kap.12, Option -e)

Befehl #r gleichwertig: **F10** **Befehl #r**
WICHTIG *WICHTIG*

Bedeutung: abhängig vom Programm, in dem man sich befindet:

Programm 1:	Rücksprung in den Anzeigebildschirm mit oder ohne Speichern
Programme 4-7:	Fortsetzen (Resume) des Programmlaufs, wenn man mit 'x' unterbrochen hatte und so in den Editor gelangt war.

Allgemein also: Rücksprung aus dem Editor in die Verarbeitungsfunktion des jeweiligen Programms, in dem man sich gerade befindet.

Im Schnellzugriff bewirkt **#r** dasselbe wie **F10**: Speicherung der bearbeiteten Aufnahme und Rücksprung ins Menü. Nach Eingabe von **#r** wird im Schnellzugriff gefragt:

```
Speichern? j/n
```

worauf folgendes passiert:

j : Aufnahme wird abgespeichert, alle Indexmanipulationen werden ausgeführt, die das Programm durch Untersuchung der Aufnahme als notwendig feststellen kann.

n : Rücksprung in den Anzeigebildschirm (→ 1.5) ohne Speicherung; alles bleibt unverändert.

Statt dessen kann man auch gleich **#rj** bzw. **#rn** geben.

Befehl #t gleichwertig: **'•'** (Punkt) **Befehl #t**
wichtig *wichtig*

a99 : **Alt+r** (Hintergrundspeicher wird angezeigt)

Bedeutung: Transfer aus dem Hintergrundspeicher

Eingabe bewirkt	→
txxf Wenn im Hintergrundspeicher eine Kategorie #xx mit Folgezeichen f	
oder steht (Kontrolle mit #a), wird sie in die aktuelle Aufnahme übernommen.	#a
.xxf War darin bereits eine Kategorie mit derselben Bezeichnung vorhanden,	
wird sie überschrieben.	
Im Hintergrundspeicher bleibt die betreffende Kategorie so lange stehen, bis	##
sie durch einen Befehl #v, #V oder F10 (Editor) oder Shift+F5	#v
(Anzeigebildschirm, → 1.5) automatisch ersetzt wird.	#V
txxf,yyg Wenn im Hintergrundspeicher eine Kategorie #xx mit Folgezeichen f vorkommt,	
wird sie in die aktuelle Aufnahme übernommen, jedoch in #yyg umgewandelt. Für 'f'	

muß man evtl. ein Leerzeichen eingeben.
Z.B. würde man mit #t41 ,40 den Hrsg. zum Verfasser machen.

Der Nutzen des Befehls liegt darin, daß man identische Elemente leicht von einer Aufnahme zur nächsten übernehmen kann. Statt also bei aufeinanderfolgenden Büchern desselben Verlages jedesmal dessen Namen (#75) und den Ort (#74) einzugeben, braucht man bei der zweiten und allen weiteren Aufnahmen nur jeweils #t74 und #t75 zu geben bzw. in der Abfrageliste einen Punkt einzugeben, wenn #74 und #75 abgefragt werden.

Hinweise:

- Für einzelne Wörter oder Wortgruppen (Phrasen), die sich mehrfach wiederholen, benutzt man besser den **Phrasenspeicher**. #p
  In den Programmen 4 und 5 gilt:
- Will man etwas aus einer weiter zurückliegenden Aufnahme übernehmen, was im Hintergrundspeicher schon wieder durch andere Inhalte ersetzt wurde, kann man wie folgt verfahren: man fährt mit #- oder #z oder #f zu der betreffenden Aufnahme zurück, gibt den Befehl ## (dann gehen deren Kategorien in den Hintergrundspeicher!) und geht wieder zu der noch zu bearbeitenden Aufnahme, wo man jetzt den #t-Befehl anwenden kann. War diese Aufnahme die letzte im Arbeitsspeicher und hatte man sie noch nicht mit ## abgeschlossen, so erscheint sie automatisch wieder.
- Statt #**t** kann man auch **#.** geben, das finden viele bequemer.
- Im Schnellzugriff hat man die Möglichkeit, eine vorhandene Aufnahme, die man über den Index findet, mit dem Befehlszeichen 'C' zu kopieren. Man erhält eine Kopie zum Editieren vorgelegt, die man nach Belieben ändern und dann als neue Aufnahme speichern kann (→ 1.5).
  Vor allem bei **Normdaten** geht man am besten so vor: man kopiert sich bis zu drei Aufnahmen (z.B. Normdatensätze) mit **Alt+F5** in den Hintergrundspeicher und geht dann mit 'I' in die Erfassung und benutzt #a und #t, um die benötigten Angaben aus dem Hintergrund zu übernehmen.
- In der Abfrageroutine (→ Kap.3.2) kann '.' statt eines Textes eingegeben werden, wenn die betreffende Kategorie, die gerade abgefragt wird, mit derjenigen aus der vorangehenden Aufnahme identisch ist. Dann wird gleichfalls der Inhalt des Hintergrundspeichers herangezogen.

Befehle #v,#V Befehle #v,#V
WICHTIG *WICHTIG*

a99: Kategorie anwählen, dann **Taste <Entf>**

Bedeutung: Vergessen (Löschen) einer Kategorie bzw. Aufnahme

Eingabe	bewirkt	→
vxxf	Die (irrtümlich eingegebene) Kategorie #xx mit Folgezeichen f wird aus der aktuellen Aufnahme gelöscht. Vorher wird sie aber in den Hintergrundspeicher kopiert und kann mit #t wieder aktiviert werden.	#t
v	Der gesamte aktuelle Datensatz wird in den Hintergrundspeicher kopiert und aus dem Arbeitsspeicher gelöscht. Die nachfolgende (Unter-)Aufnahme wird zur aktuellen und wird angezeigt. War die aktuelle Aufnahme die letzte im Arbeitsspeicher ist der Zustand derselbe wie nach dem Befehl ##, d.h. man hat eine völlig leere Aufnahme vor sich, und die Abfrageroutine beginnt automatisch mit der Abfrage.	#a ## ##

Hinweise:

- Hat man eine Kategorienummer falsch eingegeben, z.B. #47 statt #74, so muß man nicht mit v diese Kategorie löschen und sie dann mit richtiger Nummer neu eingeben. Man kann auch mit dem Befehl #**_#47_#74_** die Nummer ändern.
  Der Hintergrundspeicher wird beim Verlassen des Programms gelöscht. Nur während der laufenden Sitzung kann man auf den Inhalt des Hintergrundspeichers zurückgreifen. #a

- #V löscht **nur im Arbeitsspeicher**, **nicht in der Datei**. Aus einer Datenbankdatei und damit auch aus dem Index bekommt man eine Aufnahme **nur** dadurch weg, indem man auf dem **Anzeigebildschirm** (→ 1.5), nicht im Editor, die Taste **Entf** (= **Del**) benutzt und die dann folgende Frage "löschen? j/n" mit 'j' beantwortet.

- Bei Version 14 konnte man eine Kategorie löschen, indem man sie leer eingibt. Wenn man z.B.
  #512 <Enter>
  eingab, also ohne Text, wurde die #512 gelöscht. Wenn allerdings diese Kategorie nicht besetzt war, hatte man anschließend eine leere Kategorie - was nicht sein soll. Dieser Fehler ist in V15 beseitigt. Leereingabe hat keine Folge mehr, nur noch innerhalb der #b-Bearbeitung. #b

Befehl #X Befehl #X

wichtig ***a99:*** **Alt+g**

Bedeutung: Ersetzungsbefehl (eXchange) definieren

Erläuterung: **Nur** in PRESTO hat man die Möglichkeit, auf einer Datenbank globale Veränderungen (innerhalb einer Ergebnismenge) vorzunehmen. Dazu gibt es auf dem Anzeigebildschirm des Programms 1 die Funktionstaste F10. Damit wird eine globale Veränderung **ausgeführt.** Zunächst muß jedoch einmal **definiert** werden, welche Veränderung überhaupt vorzunehmen ist. Das macht man mit eben diesem Befehl 'X' im Editor.
Man geht zuerst mit 'E' in den Bearbeitungszustand (egal, bei welchem Datensatz!), gibt dann die gewünschten #X-Befehle (s.u.), dann mit **F8** wieder in den Anzeigeschirm, dann **F10**.
Testen Sie immer zuerst die korrekte Wirkung an einem Einzeldatensatz!
Dieser Befehl ist wiederholbar: mehrere Veränderungen können zum selben Zeitpunkt definiert sein. Sie werden, soweit anwendbar, dann in jeder betroffenen Aufnahme ausgeführt.

Eingabe bewirkt →

---

1. Normale Ersetzung **#_**
#X_*alt_neu*_ oder
#X,*alt,neu*, die Zeichenfolge ***alt*** wird durch die Zeichenfolge ***neu*** ersetzt.
Sonderfall: wenn ***neu*** leer ist (**#X_*alt*__**), dann wird ***alt*** beseitigt.

2. Endabschnitt einer Kategorie ersetzen bzw. abschneiden
#X_*alt*$_*neu*_ bzw.
#X_*alt*$__ Der mit ***alt*** beginnende Teil einer Kategorie wird bis zum Ende ersetzt. (*Achtung:* das Dollarzeichen als Textzeichen läßt sich deswegen nicht ersetzen!)

Diese zwei Fälle wirken jeweils auf die gesamte Aufnahme. Wenn ***alt*** an mehreren Stellen vorkommt, wird jede ersetzt. Will man die Ersetzung auf eine bestimmte Kategorie(gruppe) beschränken, müssen die Befehle so lauten:
#X*#*nnn_alt_neu*_ bzw.
#X*#*nnn_alt*$_*neu*_ Wenn man z.B. **#X*#3_Computer_Rechner_** gibt, wird in jeder Kategorie der Gruppe #3 **Computer** durch **Rechner** ersetzt.

3. Kategorienummer ändern
#X_#*nnn*_#*kkk*_ Die Kategorie #*nnn* wird in #*kkk* umgewandelt; sie wird dann auch an die richtige Position innerhalb der Aufnahme gebracht, also umsortiert.

4. Kategorie löschen
#X_#*nnn*__ (D.h. **#*kkk*** fehlt) Kategorie #*nnn* wird entfernt. (*Achtung:* **2** Unterstriche hinter nnn)

5. Kategorie einfügen
#X__#*nn abc*_ Die Kategorie #*nn* mit dem Inhalt *abc* wird in die Aufnahme eingefügt. Sollte sie schon vorhanden sein, wird sie ersetzt durch diesen neuen Inhalt. (*Achtung:* **2** Unterstriche hinter dem 'X')

Spezialfall: Man will "**#99 leer**" einfügen, aber nur, wenn **#99** nicht besetzt ist. Dann definiert man, in dieser Reihenfolge, drei Ersetzungen gleichzeitig:
1. #X_#99 _#99x_ 2. #X__#99 leer_ 3. #X_#99x_#99 _
Der dritte Befehl ersetzt **#99 leer** wieder durch den alten Inhalt, der in **#99x** zwischengelagert wurde. (Man kann im Editor bis zu 10 **#X**-Befehle geben, die dann bei Benutzung von **F10** jeweils alle hintereinander auf jedes Mitglied der Ergebnismenge angewendet werden.)

X_ Alle zu diesem Zeitpunkt definierten Ersetzungen werden wieder aufgehoben.
Wenn man dies nicht macht, bleiben die vorher definierten Veränderungen aktiv und werden zusätzlich zu jeder neu definierten Veränderung ausgeführt. Darauf sollte man achten, wenn man mehrere unterschiedliche Änderungsaktionen innerhalb einer Sitzung macht.

Hinweise:
3 DOS-Editor

- Schreibt man **#X*xx_abc_zyx_** statt **#X*#xx_abc_zyx_**, vergißt also das '#', so wird die Veränderung in allen Kategorien ausgefürt, in denen die Zeichen **xx** vorkommen, und zwar in dem Teil hinter diesen Zeichen. Unerwünschte Ersetzungen sind dann denkbar, wenn auch wenig wahrscheinlich.
- Das '?' kann (mit aller Vorsicht) als Maskierungszeichen in der Zeichenfolge "abc" benutzt werden.
- Beim Verlassen des Programms 1 werden alle definierten Ersetzungsbefehle vergessen.
- Die momentan eingestellten Ersetzungen sieht man mit dem Befehl #p <Enter> a e `#p`

Befehle #+ #- #z Befehle #+ #- #z

(wichtig) *(wichtig)*

gleichwertig: **<Bild↓>** **<Bild↑>** **<Pos1>/<Ende>**

Bedeutung: zeige eine bestimmte Aufnahme, zugleich:
Wechsel der aktuellen Aufnahme

sehr wichtig in den Funktionen 4 und 5, in PRESTO nur bei der Bearbeitung umfangreicher **hierarchischer** Aufnahmen, also vielbändiger Werke.

Eingabe bewirkt →

---

z wenn hinter dem z kein weiteres Zeichen steht, wird die laufende Nummer der aktuellen Aufnahme innerhalb des Arbeitsspeichers gezeigt. Ansonsten gibt es drei andere Möglichkeiten:

z1 Die **erste** Aufnahme im Arbeitsspeicher wird angezeigt

zm Die markierte Aufnahme wird gezeigt `#M`

zl Die **letzte** Aufnahme im Arbeitsspeicher wird gezeigt.

+ es wird im Arbeitsspeicher zur nächsten Aufnahme
oder
<Bild↓> übergegangen, d.h. die der aktuellen folgende wird angezeigt. Ist die aktuelle zugleich die letzte, so kommt die erste zum Vorschein (zyklisches Verfahren).

- es wird die Aufnahme angezeigt, die im Arbeitsspeicher
oder
<Bild↑> vor der aktuellen steht. Ist die aktuelle zugleich die erste, wird zur letzten übergegangen (zyklisch).

Die angezeigte Aufnahme wird jeweils zur aktuellen Aufnahme gemacht, so daß man daran sofort Änderungen vornehmen kann.

Hinweise:

- Hat man viele Aufnahmen im Speicher (in den Programmen 2-5) und arbeitet gerade an einer ganz bestimmten, so setzt man zweckmäßigerweise die Markierung mit #M, wenn man diese Aufnahme kurzfristig verlassen will, um an anderen Stellen Bearbeitungen vorzunehmen. Mit #zm kommt man sofort zu der markierten Aufnahme zurück.
- Wenn man viele Aufnahmen im Arbeitsspeicher hat, benutzt man zur gezielten Suche nach einer ganz bestimmten besser den Befehl `#f.` `#f`

Befehl #_ gleichwertig: **','** (Komma) **Befehl #_**
nützlich *nützlich*

***a99* : Alt+g**

Bedeutung: Suchen und Ersetzen von Zeichenfolgen im aktuellen Datensatz

Dieser Befehl dient dazu, Daten zu korrigieren oder zu ergänzen. Er bietet eine zusätzliche Möglichkeit neben der direkten Korrektur mit dem Cursor. Der Befehl **#_** wirkt, im Gegensatz zu **#f**, nur auf die aktuelle Aufnahme.
Statt **#_** kann man auch **#,** (Komma) geben, jedoch nicht Unterstrich und Komma im selben Befehl. Wenn man einen Unterstrich ersetzen oder einfügen will, muß man das Komma als Befehlszeichen nehmen und umgekehrt.

Eingabe bewirkt →

_A_B_ oder **,A,B,** — In der aktuellen Aufnahme wird die Zeichenfolge A gesucht und durch B ersetzt. A kann, wie bei #f, beliebig lang sein und beliebige Zeichen enthalten. Dasselbe gilt für B, welches statt A eingesetzt werden soll. (Hinweis: _A,B bzw. ,A_B bewirkt nichts) #f

_A__ — Wenn A ersatzlos entfallen soll, läßt man B weg (d.h. 2 Unterstriche hinter dem A).

Nach Eingabe des Befehls wird zunächst angezeigt, an welcher Stelle A vorkommt: es erscheint die betreffende Kategorie (evtl. verkürzt) mit **** anstelle von A.
Dann gibt es drei Möglichkeiten:
<Enter> = ersetzen Leertaste = nicht ersetzen x = Ende
diese Tasten bewirken folgendes:

<↵ : an der angezeigten Stelle wird A durch B ersetzt. Danach sucht das Programm in der aktuellen Aufnahme weiter, ob A nochmals vorkommt, und wiederholt dann die Abfrage.

Leertaste: A wird an der angezeigten Stelle nicht ersetzt. Die nächste Stelle wird gesucht und angezeigt.

x: beendet den Vorgang ohne Aktion.

Hinweise:

- In den Zeichenfolgen A und B kann jedes Zeichen außer 0 auch durch eine Codierung \nnn (nnn = dezimaler ASCII-Wert) dargestellt werden, z.B. \64 für den Klammeraffen. Nützlich ist das hauptsächlich, wenn man Such- und Ersetzbefehle in eine Export-Parameterdatei einbaut: dabei kann man z.B. Sonderzeichenfolgen durch "Escape-Sequenzen" für den Drucker ersetzen. (→ 10.2.4.5.)
- Soll eine Ersetzung nicht nur in einer Aufnahme, sondern in vielen stattfinden, so arbeitet man im Schnellzugriff mit der **globalen Ersetzung** (→ 1.5). #X
  Dort sind weitere Möglichkeiten mit zusätzlichen Finessen beschrieben, die man allerdings als Direktbefehle kaum brauchen dürfte. Man gibt die Befehle genau wie dort gezeigt, nur ohne das **X**, dann wirken sie direkt auf die aktuelle Aufnahme.
- Wie bei **#f** kann auch hier **'?'** benutzt werden, um einzelne Zeichen zu maskieren, jedoch ist das selten sinnvoll.
- **$** als letztes Zeichen von A hat eine besondere Wirkung: Gibt man z.B. _xyz$ ein, so wird der gesamte Rest derjenigen Kategorie, in der "xyz" vorkommt, abgeschnitten, beginnend mit dieser Zeichenfolge "xyz".

  Dieser Sonderfall ermöglicht es, von einer längeren Kategorie den hinteren Teil an beliebiger Stelle abzuschneiden, wenn man die Aufnahme aus irgendwelchen Gründen verkürzen will. (Man müßte sonst für A diesen gesamten Teil eingeben!)
- Will man, umgekehrt, eine Kategorie verlängern, so gibt man für A z.B. die letzten 5 Zeichen der Kategorie ein, für B nochmals diese Zeichen und dazu den zu ergänzenden Rest.
- Wenn A und B Kategorienummern sind, wird die geänderte Kategorie, falls notwendig, im Arbeitsspeicher automatisch umgeordnet. *Beispiel:* wenn man **#_#25_#81** gibt, wird die Kategorie #25 nicht nur in #81 umbenannt, sondern auch entsprechend ihrer Reihenfolge innerhalb der Aufnahme umgeordnet.
- Man wird vorwiegend auf die direkte Art (mit dem Cursor) korrigieren. Nach einiger Einarbeitung wird jedoch dieser Befehl ebenfalls geschätzt. Übermäßig lange Kategorien (mehr als 1998 Zeichen, also länger als ein Bildschirm), lassen sich per Cursor nicht korrigieren. Dann muß man mit "Suchen und Ersetzen" oder Befehl #b arbeiten.

ACHTUNG: Programm SRCH.EXE ist erst ab V32 ein 32bit-Programm. Ältere Versionen laufen nicht unter Win‘7/64

4 Volltext-Suche und Selektion

a99: Menü **Finden/**Volltextsuche
Suchen mit regulären Ausdrücken

4.0 Aufgabenstellung und Konzept

Die sonst so bequemen Register (und damit auch die Fernglas-Suche mit a99) sind nutzlos, wenn man

- Wort**bestandteile** sucht, oder aber
- Inhalte von **nicht indexierten Datenfeldern**.

Dafür gibt es ein ganz anders arbeitendes Programm: SRCH für die "Volltext-Suche", im Windows-Programm eine entsprechende Funktion im Menü "Find". (Nicht die Texte der Dokumente sind gemeint, sondern natürlich nur die Texte der Datensätze!)
Das Programm SRCH hat allerdings zwei Aufgaben, und wichtiger dürfte die zweite sein:

1. **Volltext-Suche:** im Gegensatz zum Schnellzugriff findet es einen Datensatz sogar dann, wenn das gesuchte Wort darin nur als Wortbestandteil vorkommt (z.B. "energie" als Teil von "Atomenergie"), also eben auch, wenn es nicht im Index steht. Außerdem kann es nicht nur Datenbank-Dateien durchsuchen, sondern auch Grunddateien, die im Schnellzugriff (mit PRESTO) unzugänglich sind! Gleichwohl: der Indexzugriff ist erheblich schneller und für den Benutzer viel komfortabler. Man wird deshalb nicht die Volltextsuche als OPAC-Variante anbieten wollen. Es ist ein Werkzeug für Auswertungen, die anders nicht durchführbar sind. [Anm.: ab V21 kann man aber im Index auch Wortbestandteile haben!]
2. **Formatierung:** SRCH kann gefundene Datensätze in jedem gewünschten Format, das sich mittels der Export-Sprache beschreiben läßt (→ Kap.6 und 10), ausgeben. Es leistet also **Selektion** und Export in einem Durchgang, und zwar sogar bis zu vier Exportvorgänge gleichzeitig. Insbesondere kann man Ergebnisse als Grunddatei (Typ .cLG) ausgeben lassen und anschließend mit Programm 7 oder 9 (→ Kap. 7 und 9) in eine andere Datenbank überführen.

Was mit SRCH **nicht** geht: gefundene Aufnahmen sofort in der Datenbank korrigieren. Dazu benötigt man:

- DOS: PRESTO (für Dateien des Typs .cLD per Schnellzugriff → Kap.1). Das ist der Normalfall.
- Windows: ***a99***

Wichtige Voraussetzung: die Datei **S1.ASP** muß vorhanden sein. Sie enthält eine Tabelle von Zeichenumwandlungen, z.B. Umlautauflösungen. Ohne diese Datei funktioniert SRCH zwar auch, aber man muß alle Suchbegriffe in exakter Schreibweise (Groß- und Kleinbuchstaben, Umlaute etc.) eingeben.
S1.ASP enthält eine Zeile für jedes umzuwandelnde Zeichen. Kommentare sind darin nicht erlaubt. Man schreibt an erster Stelle das umzuwandelnde Zeichen, nach einem Leerzeichen folgt das statt seiner zu benutzende Zeichen, oder eine Zeichenkombination (bei Umlauten) oder nichts, wenn das Zeichen zu ignorieren ist:
Anmerkung zum Thema "Linkstrunkierung": durch eine kleine Erweiterung der Parametrierung (Beispiel in CAT.API) kann man erreichen, daß neben jedem einzelnen Titelwort auch jeder Teil eines Wortes indexiert wird. Außerdem kann man in ***a99*** durch das Eingabefeld "Nur Einträge mit" die Registeranzeige auf solche Einträge beschränken, die bestimmte Bestandteile enthalten (→ Kap.2.1).

Beispiele für Einträge in `s1.asp`

`A/Z a`	A...Z in a...z wandeln (dies ist das Muster für die Behandlung einer **Folge** von Zeichen)
`ä ae`	ä in ae auflösen
`. P`	damit man einen Punkt finden kann: das Zeichen '.' in 'P' umwandeln
`@`	den Klammeraffen ignorieren
`!/-`	die Zeichen `'!'` bis `'-'` ignorieren
`é e`	Akzent beseitigen
`X`	Leerzeichen durch X ersetzen (Leerzeichen innerhalb von Suchbegriffen nicht zulässig!)

`...`

Hinweis: Die Umwandlungen gelten nur für den Vergleich mit dem eingegebenen Suchbegriff, es werden keine Daten dabei verändert.

Ab Version 14: Wenn mit Stammsätzen gearbeitet wird, z.B. für Personen oder Schlagwörter, kann man mit der Volltextsuche die Namen und Schlagwörter auch dann finden, wenn in den Datensätzen nur die Identnummern der Stammsätze stehen. Jedoch auch die Nummern können gesucht werden. Intern wird jeder eingelesene Datensatz, bevor die Suche beginnt, vorbearbeitet: die Zeichenersetzungen werden ausgeführt, und zu den Identnummern werden über das Ersetzungsregister die Ansetzungsformen herangeholt. (→ 10.2.6.8)

4.1 Programmablauf (mit dem Konsolprogramm `srch.exe`)

Achtung ab V33: Das 32bit-Programm `srch.exe` bzw. `srch` unter UNIX hat nicht mehr die hier beschriebene interaktive Arbeitsweise, sondern ist ein reines Konsolprogramm. Wie man es nutzt, ist in 4.2 beschrieben.

Die hier folgende Beschreibung ist für Systemverwalter gedacht, die auch einmal einen Vorgang "von Hand" starten wollen. Der "Normalverbraucher" startet die Volltextsuche am bequemsten über ***CockPit***, und zwar

- ohne Export: Menü "**R**outinen / Volltextsuche/Listen / 3" **`µr e 3`**
- mit Export: Menü "**R**outinen / Volltextsuche/Listen / 2" oder Menü "**F**unktionen / 4 = SRCH"

Dann läuft bis auf die Eingabe des Suchbegriffs (siehe unten Punkt 8. und Beispiele in Abschnitt 4.3) alles automatisch bzw. menügeführt ab. Die Exportfragen sind beschrieben im Kap. 6.2.
Statt in dieser Weise interaktiv zu arbeiten, kann eine Volltextsuche auch als Konsolprogramm laufen: siehe Abschnitt 4.2.

Nehmen wir an, es sollen aus den Dateien, die auf dem Unterverzeichnis KATALOG des Programmverzeichnisses stehen, alle Sätze selektiert werden, in denen die Namen Gödel, Escher oder Bach irgendwo vorkommen. Die Ergebnisse sollen auf das Verzeichnis F:\TEXTE mit dem Parametersatz P-NORMAL.APR ausgegeben werden, und zwar in eine Datei namens LISTE. Wenn die schon existiert, wollen wir die neuen Daten hinten anhängen.
Die Fragen und Meldungen des Programms sehen Sie im folgenden in `Courier-Schrift`, die Antworten **`fett`**.
Das Suchprogramm wird gestartet:

a) mit dem Befehl **`SRCH -f4`** (den Ablauf sehen Sie gleich anschließend), oder

b) vom ***CockPit*** aus über **`µf 4`** (Menü "Funktionen" mit bequemen Eingabeformularen), oder

c) aus einer Stapeldatei heraus, indem man SRCH mit den geeigneten Optionen aufruft. Damit kann man Routineabläufe vollständig automatisieren (→ 4.3). Dann wird SRCH aus einem Batchfile u.U. mehrmals aufgerufen. Die Erstellung **sortierter Listen** ist der wichtigste Anwendungsbereich, wo SRCH routinemäßig im Batch aufgerufen werden muß (→ Kap.6). Das ***CockPit***-Menü "Routinen / Volltextsuche/Listen" startet solche Vorgänge. Endanwender, die sich auf diese Standardanwendungen beschränken, brauchen vom Programm SRCH fast nichts zu wissen, nur wie man einen Suchbegriff eingibt.

Im ersten Fall läuft folgender Dialog ab:

1. `Bitte Dateien zum Durchsuchen auswählen:`

   `(Laufw.:)(Pfad\)  (x = Irrtum)` **`katalog`**

   Sie geben je nach Situation nur den Laufwerksbuchstaben oder auch den Namen eines Unterverzeichnisses ein. In diesem Fall reicht `katalog`, weil das Verzeichnis direkt am Programmverzeichnis hängt.

2. Es erscheint zur Auswahl die sortierte Liste (→ 0.5.) der .ALG- und .ALD-Dateien, die auf dem gewünschten Verzeichnis stehen. Man markiert die zu durchsuchenden Dateien mit '+' . Wenn alle durchsucht werden sollen, gibt man nur 'a'. Dann: <Enter>

3. `Export gewünscht? (= Ausgabe auf Drucker/Datei) ? j/n` **`j`**

   Man antwortet mit j oder n, was zur Folge hat:

   n: Ausgabe der Ergebnisse nur am Bildschirm

   j: es erscheint die Auswahlliste der verfügbaren Export-Parameterdateien (Typ .cPR), aus der man wie üblich eine auswählt. Mit F5 kann man sich jeweils die erste Zeile einer Parameterdatei anzeigen lassen, um zu sehen, um was es sich handelt.

Wenn die Ergebnisse ausgegeben (und nicht nur am Bildschirm angezeigt) werden sollen, wird die Ausgabe durch frei wählbare Parameter gesteuert. Daher wird an dieser Stelle eine Export-Parameterdatei benötigt (→ Kap.10).
Also: Man setzt ein '+' vor den Namen der .APR-Datei, die für die Ausgabe zu benutzen ist z.B. P-NORMAL oder I-1 oder E-1. Diese mitgelieferten Standardparameter haben folgende, oft benötigte Funktionen:

`p-normal` Ausgabe als Literaturliste mit Seitenaufbereitung. Die Datei `p-normal.apr` ist kommentiert; Sie können darin z.B. die Ausgabe der Seitenzählung so ändern, daß alternierend ungerade Zahlen rechts und gerade Zahlen links erscheinen, wie man es für einen zweiseitigen Druck braucht. Die dafür nötigen Änderungen sind in `p-normal.apr` erläutert.

`i-1` Ausgabe im ***allegro***-Grundformat (zwecks Nachbearbeitung oder Einmischen in eine andere Datenbank) (ganz früher: `pa.apr`)

`e-1` Ausgabe mit Kategorienummern als ASCII-Textdatei. (früher: `pk.apr`)

Die mit `i-1` erstellten Ausgabedateien sollten den den Typ .ALG erhalten, dann können sie aufgrund ihrer Struktur mit ***a99*** als "externe Datei" eingelesen oder auch mit 7 = INDEX oder mit 9 = UPD in eine andere Datenbank eingespeist werden. Alle diese Parameterdateien lassen sich beliebig abwandeln und somit speziellen Wünschen anpassen. Die Kommentare in den Dateien sagen Ihnen, was nicht verändert werden darf und wo welche Anpassungen möglich sind (→ Kap.10).

4. `Ausgabe auf (Laufwerk:)(Pfad\)Datei:`
`(x = keine Ausgabe gewünscht)`

Antwort: **`f:\texte\liste`**

Die Frage ist also mit dem Namen (mit Laufwerk und/oder Unterverzeichnis, falls es nicht das aktuelle Verzeichnis ist) der Datei zu beantworten, in welche die Ergebnisse zu schreiben sind.
Antwort **`x`** hat keine direkte Wirkung; wenn man während der Suche mit Druck auf 'x' in den Bearbeitungsmodus geht, kann man aber jede Aufnahme mit Befehl #d ausgeben lassen.

Wenn **`LISTE`** schon auf **`F:\TEXTE`** existiert, kommt noch die Entscheidung:

```
Datei f:\texte\liste existiert schon
 1 = neue Daten anhängen   0 = Datei löschen   Esc = Irrtum  1
```

Man antwortet mit '1' oder '0' oder <Esc>.
Wenn man mit <Esc> antwortet, wird ein neuer Name abgefragt.

Will man ohne Umschweife die Ergebnisse sofort gedruckt bekommen:
`prn` oder **`lpt1`** statt eines Dateinamens eingeben. (Nicht sinnvoll, wenn man `I-1` gewählt hatte)

5. `noch weitere Ausgabe? j/n` **`n`**
Es können mehrere Exportvorgänge parallel abgewickelt werden, d.h. die Prozedur unter 3./4. ist bis zu 3mal wiederholbar.

6. `manueller Eingriff? j/n` **`j`**
normalerweise mit 'j' zu beantworten; längere Auswertungen gehen bei 'n' etwas schneller. Wenn man die Ergebnisse sofort bearbeiten will, bevor der Export erfolgt, antwortet man 'j' (siehe 10.)

7. `Anzeige gewünscht? j/n` **`j`**

   (Frage kommt nur, wenn 3. mit 'j' beantwortet wurde)
   Wenn man neben der automatischen Produktion noch eine Bildschirmanzeige der Ergebnisse wünscht, um das Geschehen zu verfolgen und gegebenenfalls mit 'x' einzugreifen, wählt man hier 'j'.

8. Jetzt erst kommt die Frage nach dem **Suchbegriff**. Dieser kann aus einem simplen Wort oder einer beliebigen Zeichenfolge (Buchstaben und Ziffern, keine Sonderzeichen außer ",-;:/()><" , siehe unten) bestehen. Es kann aber auch ein kompliziert zusammengesetzter Ausdruck sein.

   Das Programm informiert Sie kurz über seine Modalitäten und stellt dann die Frage nach dem Suchbegriff:

```
Suchbegriff = jede Zeichenfolge aus Buchstaben und Ziffern
Eingabe in Kleinbuchstabenä = ae usw.              . = Maskierung
logische Operatoren:            + = UND    / = ODER      - = NICHT
Vergleichsoperatoren:           < = KLEINER              > = GROESSER
Einschränkungs-Operator:  X,Y = suche Y im Feld X
Beispiel:     (goethe/lessing)-schiller
              = alles von und über Goethe oder Lessing, ohne Schiller
Beispiel:     #40,goethe
              = Goethe nur als Verfasser

Suchbegriff?  goedel/escher/bach
```

Der Suchbegriff kann bis zu 300 Zeichen lang sein. Die Bearbeitung des Suchbegriffs geht genauso vor sich wie die Fenster-Bearbeitung langer Datenfelder mit dem Editorbefehl #b (→ 3.3).
Beispiele für Suchbegriffe und Bemerkungen zu Sonderfällen finden Sie in Abschnitt 4.3.

9. Wurde bei 3. 'j' gesagt, läuft alles automatisch ab. Am Ende hat man die Ergebnisse in der gewünschten Datei stehen bzw. man bekommt sie formatiert ausgedruckt, den gewählten Parametern entsprechend.
   Wurde bei 3. 'n' gesagt, hält das Programm nach jedem Treffer an, damit man den gefundenen Datensatz in Ruhe betrachten kann. Auch das Blättern in den bereits gefundenen Aufnahmen ist möglich: mit '+' und '-' geht es vor- und rückwärts. Fortsetzung der Suche durch 'Leertaste'.

10. **Unterbrechung:** Der Suchvorgang kann jederzeit mit Druck auf 'x' unterbrochen werden. Man erhält dann die zuletzt gefundene Aufnahme im Eingabeformat angezeigt und befindet sich automatisch im Editor (→ Kap.3). Das Suchprogramm reiht die gefundenen Aufnahmen im Arbeitsspeicher sozusagen wie in einer Warteschlange hintereinander, wenn man 6. mit 'j' beantwortet hatte. Erst dann, wenn der Platz knapp wird, erfolgt jeweils die Ausgabe der ersten (ältesten) Aufnahme in der Schlange auf dem Drucker bzw. in die Datei. Was noch im Arbeitsspeicher steht, kann also vor der Ausgabe noch bearbeitet werden. Mit <Bild↓> und <Bild↑> blättert man die Aufnahmen im Arbeitsspeicher durch.

    Im Editor gibt es zwei Befehle mit besonderer Wirkung:

    #r oder F10: **Weitersuchen** ("resume")

    #e oder F8: **Ende des Suchlaufs** (Abbruch), (Rückkehr zum ***CockPit***, wenn von dort gestartet)

Experimentieren Sie mit den Beispieldaten, um die Möglichkeiten gründlich kennenzulernen.

Derselbe Vorgang wird vom ***CockPit*** aus dem "Funktionen"-Menü so gestartet:

Sie sehen hier die Momentaufnahme zu dem Zeitpunkt, wo der Name für die Ausgabedatei eingegeben wird. Das abgebildete Menü wird so benutzt:

- man bewegt den Leuchtbalken in der linken Hälfte auf die zu ändernden Optionen und drückt jeweils <Enter>
- die betreffenden Elemente werden, ihrer Art entsprechend, über ein Eingabefeld oder eine Auswahlliste abgefragt
- Ist alles richtig ausgefüllt, startet man das Programm durch Aktivierung von **GO (START)**.

CockPit produziert und startet dann eine Batchdatei namens CCC.BAT. Dieses enthält den korrekten Aufruf von SRCH, wie ihn der nächste Abschnitt zeigt:

4.2 Aufruf des Programms aus Batchdateien (auch Win´7/64)

Das Programm SRCH kann, wie die anderen, auch als Modul in Stapeldateien benutzt werden. Sämtliche oben beschriebenen Angaben, die das Programm abfragt, kann man ihm als Optionen "von außen" mitgeben, dann läuft es automatisch ab. Für die ältere DOS-Variante galt: Diejenigen Angaben, die es nicht über Optionen erhält, fragt es noch ab. Man kann es daher so einrichten, daß immer nur die wirklich veränderlichen Angaben im Dialog einzugeben sind.
Aufrufe in Batchdateien werden ausführlich in **Kap.12** behandelt. SRCH wird häufig in dieser Weise für Exportvorgänge eingesetzt, die mehr oder weniger regelmäßig als Routineproduktionen ablaufen. So z.B. in SR-LIST.BAT und PR-LIST.BAT.
Der in 4.1 als Beispiel beschriebene Vorgang wäre mit folgendem Befehl zu starten:

```
srch -f4 -dkatalog/cat*.ald -ep-normal+f:\texte\liste -m1 -v1 -sgoedel/escher/bach
```

Wenn innerhalb P-NORMAL.APR Sätze nachzuladen sind (→ 10.2.6.7) und wenn mit Stammsätzen gearbeitet wird (V14), muß man noch die Option **`-b katalog\kat`** hinzufügen, damit SRCH weiß, welche Datenbank dafür zu benutzen ist. Wenn Sie die Befehlszeile mit dem Dialog und der ***CockPit***-Anzeige vergleichen, erkennen Sie leicht die Bedeutung der Optionen, und wie sie anzuwenden sind.
TIP: Option **–F** hinzufügen, wenn bei dem betr. Export keine V14-Ersetzungen notwendig sind. Dann geht es schneller.

Ab V30: Alternative : Konsolprogramm acon.exe + srch.job. Den Anfang der Befehlszeile jeweils so verändern:
`acon -j srch -f...`
Dann aber die Optionen vollständig angeben, nicht etwa z.B. `-s...` weglassen. Dies Verfahren ist langsamer und nicht mehr nötig, seit es srch als 32bit-Programm gibt:

Ab V32: Reguläre Ausdrücke : Die neue Option **`-r`** statt `-s` ermöglicht das Volltextsuchen mit regulären Ausdrücken.
In ***a99*** eingeben: `h rtf`, um dazu die genauen Regeln zu erfahren.

4.3 Beispiele für Volltext-Suchbegriffe (Option `-s`)

Achtung: Bei Start über ***CockPit*** (bis V32) oder Batch: 'G' statt '>' und 'S' statt '<' , sonst Fehlfunktion!!!
Die Eingabe in Kleinbuchstaben funktioniert nur, wenn die Datei `s1.asp` vorhanden ist (Zeichen-Umwandlungstabelle).
Die Suche mit "regulären Ausdrücken" (wie in ***a99***, mit Option **`-r`**) folgt anderen Regeln! Dazu `h ftr` eingeben in a99.

`kunst+philosophie`	beide Wörter sollen irgendwo im Datens*atz* vorkommen (Reihenfolge gleichgültig)
`kunst,philosophie`	"Kunst" und "Philosophie" sollen innerhalb eines Daten*felds* in dieser Reihenfolge vorkommen
`#77G999`	in #77 soll eine Zahl größer als 999 stehen
`#99nG20010631`	alle ab 1.7.2001 erfaßten Datensätze (wenn #99n das Erfassungsdatum enthält; G für "größer")
`"#40.<m"`	einer der Verfassernamen (#40, #40a oder #40b) soll mit A...L anfangen
`"#20.>jzzz + #20.<q"`	Der Titelanfang soll zwischen k und p liegen (einschließlich; denn es ist k>jzzz und pz<q)
`#15`	alle Sätze selektieren, in denen ein Feld #15 vorkommt.
`#3,goethe`	"Goethe" muß in einer Kategorie der Gruppe #3 vorkommen (Sacherschließungsdaten)
`ma.er`	"Mayer" und "Maier" werden gefunden - aber auch "Maler" und "mager" etc. Sucht man ausdrücklich nach "Mayer" oder "Maier", dann "mayer/maier" geben.
`(goethe/lessing)-(schiller/fichte)`	"Goethe" oder "Lessing" sollen vorkommen, "Schiller" oder „Fichte“ aber nicht.
`(goethe/lessing)-#4,schiller`	"Goethe" oder "Lessing" sollen vorkommen, jedoch keine Werke von Schiller. ("Schiller" darf hierbei im Titel oder sonstwo vorkommen, nur nicht in einer mit #4 beginnenden Kategorie.)

Jedes dieser Beispiele kann man in Klammern einschließen und mit anderen geklammerten Suchbegriffen verknüpfen.
Die zwei Teile links und rechts vom Komma oder Semikolon dürfen keine mehrteiligen Ausdrücke sein, also z.B. `(#40/#41),goethe` wäre nicht möglich: man müßte statt dessen `(#40,goethe)/(#41,goethe)` schreiben. Komma-Ausdrücke müssen in Klammern eingeschlossen sein, wenn sie mit anderen Ausdrücken kombiniert werden sollen!
Besonderheit: die Operatoren '<' und '>' reagieren automatisch "richtig": wenn eine Zahl dahinter steht, wird der numerische Wert, wenn ein Buchstabe, dann der lexikalische Wert verglichen. Vor dem Vergleichsoperator muß eine Kategorienummer stehen, sonst ergibt sich kein Sinn. **Sonderfälle:**

- Gibt man '0' (eine Null) als Suchbegriff ein, so wird **jeder Satz als "Treffer"** behandelt. Sinnvoll ist das, wenn man Dateien **komplett** (ohne Selektion) **exportieren** will (→ Kap.6). Dies ist der einzige Unterschied, wenn man SRCH als Exportprogramm (d.h. mit Startoption `-f6`) und nicht als Suchprogramm (mit `-f4`) benutzt.
- **Leerzeichen in Suchbegriffen** sind durch 'X' zu ersetzen: wenn man z.B. "Künstliche Intelligenz" sucht, gibt man "kuenstlicheXintelligenz" ein. (Der Grund für diese Besonderheit liegt im Betriebssystem)
- **Punkte in Suchbegriffen** sind durch 'P' zu ersetzen. Dieser Trick ist notwendig, weil der Punkt '.' als "Jokerzeichen" oder Maskierungszeichen fungiert.
- Die **Funktionszeichen** `/+,;)><` müssen, falls sie in einem Suchbegriff vorkommen sollen, mit vorgesetztem `'/'` eingegeben werden, also z.B. `1987//88` statt `1987/88`. Die Zeichen `'-'` und `'('` , die auch als Funktionszeichen direkt hinter `'/'` vorkommen können, müssen in der Form `"--"` bzw. `"(("` eingegeben werden, wenn man sie als Zeichen sucht. Keine Leerzeichen rechts und links der Funktionszeichen!
- Wenn man vom ***CockPit*** aus startet oder SRCH in einem Batch aufruft, muß man die Zeichen '>' und '<' ersetzen durch 'G' (für "greater") und 'S' (für "smaller"), denn die Spitzklammern haben leider innerhalb von Batchfiles eine Sonderbedeutung (sog. Umlenkzeichen), die das korrekte Funktionieren verhindert. Also (auch hier wieder X statt Leerzeichen):
  `SRCH -s #76XG1980` statt `SRCH -s #76 >1980` um Titel nach 1980 zu selektieren.
  Möglich ist auch : `-s"#76.>1980"`
- **Klammern** können mehrfach geschachtelt werden. Man braucht sie hauptsächlich, um ODER-Verbindungen zusammenzufassen. (Keine Leerzeichen vor und hinter den Operatoren!)
- Das **Komma** wirkt so: Zuerst wird das gesucht, was *vor* dem Komma steht. Das dann gefundene Datenfeld wird von der Stelle ab zum Ende hin nach der Zeichenfolge durchsucht, die *hinter* dem Komma steht. Wenn man also z.B. die Zeichenfolge `#3,goethe` eingegeben hat, bedeutet das: Zuerst wird nach `#3` gesucht. Damit werden dann Feldnummern wie `#30, #31, ...` gefunden. Jedes dieser Felder wird durchsucht, und zwar exakt ab dem Zeichen, das hinter der Zeichenfolge `#3` steht. Das Komma kann somit auch als "Mitteltrunkierung" gelten:
  Die Eingabe `atom,energie`
  würde auch das Wort Atomkernenergie oder die Phrase "Atom- und Fusionsenergie" finden. Nichts jedoch, wenn "energie" nicht im selben Feld steht.

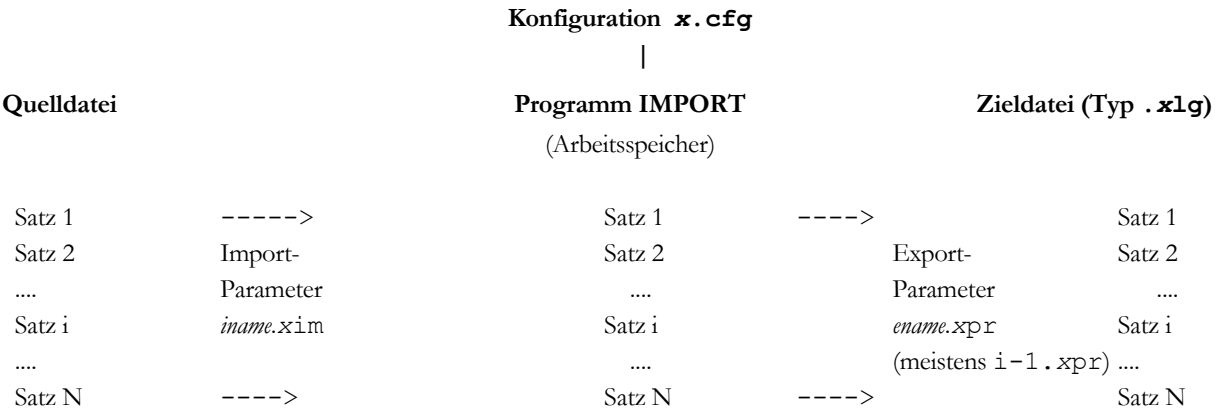
5 Fremddaten konvertieren (Import)

***a99*:h fremd**

5.0 Aufgabenstellung und Konzept

Immer wieder stellt sich das Problem, vorhandene Daten so umzuwandeln, daß ***allegro*** etwas damit anfangen kann. Woher die Daten kommen, spielt keine Rolle. Sie können vorher mit anderer Software auf demselben Rechner produziert worden sein. Oftmals werden sie aus anderen Systemen abgezapft, z.B. per "download" aus Verbunddatenbanken. Früher fielen Daten aus CD-ROMs oder anderen Diensten an, wie z.B. VLB, DNB, EKZ, etc. Auch ein WWW-Browser kann mit der Funktion "File / Save as..." Download-Daten liefern oder mittels "Copy&Paste". Die Fähigkeit zum Konvertieren wird ansonsten oft als *Schnittstelle* oder *Interface* bezeichnet. Das **Konvertieren** ist immer nur ein **erster Schritt**, das **Einspeisen** in die eigene Datenbank ist der **zweite Schritt;** siehe dazu die Kap. 7 und 9.

Woher die Daten auch stammen, sie haben alle eines, aber leider nur dieses gemeinsam: jede Datenquelle hat ihre eigene, unverwechselbare Struktur. Also kann es unglücklicherweise kein Programm geben, das von allein mit allen Daten fertig wird. Das ***allegro***-DOS-Programm Nr. 5 (es heißt IMPORT.EXE, macht aber nur den ersten Schritt, die Konvertierung!) nähert sich diesem hohen Ziel folgendermaßen: es stellt eine eigens entwickelte "Datenmanipulationssprache" bereit - ein Werkzeug, mit dem man fremde Datenstrukturen und ihre Überführung in ***allegro***-Strukturen beschreiben kann. Vom Anwender wird dabei keine Kenntnis irgendeiner echten Programmiersprache verlangt, es handelt sich um eine sog. "höhere Sprache". In Kap.11 wird diese **Importsprache** genau beschrieben.
In das Importprogramm ist aber zugleich auch die Exportfunktion eingebaut, wie in die anderen Programme auch. Außerdem kann man, weil auch der Editor eingebaut ist, in den Programmablauf eingreifen und Korrekturen anbringen, bevor die umgewandelten Daten abgespeichert werden. (Dies gilt nicht für UNIX.)
Folgendes Diagramm veranschaulicht den Importvorgang:



Grundlage für einen Importvorgang ist, wie bei den anderen Programmen auch, eine Konfigurationsdatei `x.cfg`, die neben anderen Dingen das Ziel-Kategoriesystem enthält (→ Anh. A). Wenn also der Import in das Format `u.cfg` hinein gehen soll, muß der Start mit **`import -ku`** erfolgen.
Die Beschreibung der Quelldaten steht in einer Import-Parameterdatei *iname*.`xim` (→ 0.3 und 11). Diese kann nur in Verbindung mit c.CFG arbeiten, daher ist auch hier der Konfigurations-Kennbuchstabe Bestandteil des Dateityps (`.xim`). Wer z.B. mit U.CFG arbeitet, braucht eine darauf abgestimmte Datei des Typs .uim .
Nochmals der Hinweis: Die "Zieldatei" ist noch nicht die Datenbank! Es entsteht nur eine Datei vom Typ `.x`LG, die man mit UPD in eine Datenbank einspeisen muß (s. Kap. 9).

Ab V33:
Für a99 gibt es in manchen Fällen viel bequemere Methoden, z.B. ein Sofort-Direktimport von DNB-Daten (`X dnb` eingeben) oder GBV-Daten (`X srugbv`) und zwei Verfahren zur Nutzung von Z39.50-Datenquellen: Zack und zDirect (`X zc` eingeben).

Das Importprogramm liest mit Hilfe einer `.xim`-Datei die Quelldaten Satz für Satz ein. Die Importparameter sagen dem Programm unter anderem, wie das Satzende aussieht oder wie die Satzlänge zu bestimmen ist. Für das Standardformat $a.cfg werden viele Importparameterdateien (Typ `.aim`) mitgeliefert, z.B. `marc21.aim`.

Der eingelesene Fremdsatz wird zuerst in einen internen Satz umgewandelt. Wie das zu machen ist, beschreiben eben gerade die Import-Parameter. Der interne Satz kommt zunächst in den Arbeitsspeicher, als hätte man ihn mittels Editor von Hand eingegeben. Wenn noch genügend Platz frei ist, liest das Programm den nächsten Fremdsatz und wandelt ihn um.
Der Arbeitsspeicher faßt etwa 40K Daten. Bevor er überläuft, beginnt das Programm mit dem Satz 1, die Daten in die Zieldatei zu schreiben. Beim weiteren Einlesen der Fremddaten wird immer dann wieder ein Satz abgespeichert, wenn der Arbeitsspeicher voll zu werden droht. So ist ständig eine Datenmenge von bis zu 35K im Arbeitsspeicher. Wenn sehr lange Sätze vorkommen können: besser mit Option **`-m0`** starten. Dann wird immer nur ein Satz eingelesen und sofort wieder herausgeschrieben.

Zum Testen:
Die 16bit-DOS-Version (bis V32) ermöglicht den **Eingriff während des Programmlaufs**.
Während ein Import läuft, kann man diese Tasten drücken:

`a` **Anzeige** wird umgeschaltet: die durchlaufenden Sätze werden angezeigt bzw. diese Anzeige wieder abgeschaltet.

`Leertaste` Durchlauf wird **angehalten / fortgesetzt**. Mit geschickten Fingern kann man so die Sätze einzeln durchlaufen lassen und jeden ausführlich betrachten.

`x` Sprung in den **Editor**. Der im Arbeitsspeicher vorhandene, gerade umgewandelte Satz kann genauso bearbeitet werden, wie man es von PRESTO gewohnt ist (→ Kap.3). Man kann sogar neue Sätze hinzufügen oder vorhandene löschen. Mit <Bild↑> und <Bild↓> kann man die bereits umgewandelten Sätze durchblättern!

Zwei Befehle haben dann (nach 'x') im Editor eine besondere Wirkung:

- #r (für "resume") oder **F10** läßt den unterbrochenen Importvorgang weiterlaufen
- #e oder **F8** beendet ihn vorzeitig oder am Ende des Durchlaufs endgültig.

Sobald der letzte Satz eingelesen ist, werden schließlich alle noch im Arbeitsspeicher stehenden Sätze ausgegeben.
Die Ausgabe erfolgt über eine Parameterdatei *ename*.cPR. Auch diese muß auf das Kategoriesystem c.CFG abgestimmt sein. Jedoch kann *ename*.cPR jede beliebige, für c.CFG geschriebene Export-Parameterdatei sein. Die Zieldatei kann daher eine völlig andere Struktur haben als die Interndaten im Arbeitsspeicher! Z.B. könnte man aus OCLC-Daten in einem Durchlauf Katalogkarten gemäß ISBD machen, wobei als Zwischenstruktur sowohl das Standardformat A.CFG als auch das USMARC-Format U.CFG dienen kann. Dabei hätte man als Import-Parameterdatei OCLC.AIM bzw. OCLC.UIM zu wählen, als Export-Parameterdatei P-KARTE.APR bzw. D-1.UPR.
Ein wichtiger **Sonderfall**, in der Praxis aber eher der Normalfall, ist die Export-Parameterdatei `I-1.cPR` (ganz früher einmal hieß sie `PA.cPR`). Sie schreibt das jeweilige Internformat `c.CFG` unverändert in die Zieldatei, stellt also eine Original-***allegro***-Datei im Grundformat her. Dabei muß die Zieldatei die Typkennung .cLG erhalten, um von den Programmen anerkannt zu werden. Also .ALG, wenn man mit der Konfiguration A.CFG oder $A.CFG arbeitet.
Zur internen Struktur von ***allegro***-Grunddateien: Kap. 0.2.2. Das ist die Struktur, die mit den Parametern `I-1.APR` erzeugt wird.

Das Importprogramm kann, wie erwähnt, unter anderem und vor allem Grunddateien (.ALG) herstellen, indem es **`I-1.cPR`** als Exportparameter benutzt. Mit dem Indexprogramm (→ Kap.7) oder mit UPD (→ Kap.9) lassen sich solche Grunddateien in vorhandene Datenbanken einspeisen. Das Importieren von Fremddaten in eine Datenbank hinein ist deshalb ein **zweistufiger Vorgang**. Schematisch läuft das so ab (hier für den Fall des a-Formats (`$a.cfg`) :

	Konvertierung			Datenbankaufbau	
	Prog. **IMPORT**			Prog. **INDEX** oder **UPD**	
Fremddaten	`---->`	**Grunddatei**	`---->`	**Datenbank**	
iname`.aim`	`i-1.apr`	Typ `.alg`	`.api`-Param.	`.ald/.adx`	
Importparam	Exportparam.			`.tbl/.stl/.res`	

Um korrekt zu sein, müßte man IMPORT.EXE als "Transferprogramm" oder "Konvertierprogramm" bezeichnen und es CONVERT.EXE nennen, da es nur die erste Phase durchführt und nicht das Einmischen. Da jedoch die Überführung in eine ***allegro***-Umgebung hinein der häufigste Einsatzzweck ist, wird der Name "Import-Programm" beibehalten.
Wie entscheidet man, ob UPD oder INDEX das Einmischen übernehmen soll? Dazu muß man eines wissen: UPD kann vorhandene Sätze durch neue ersetzen (→ Kap.9), es braucht dazu aber einen eindeutigen **Primärschlüssel**, über den die Identifikation läuft. INDEX kann das nicht, es fügt jeden Datensatz als neuen Satz in die Datenbank ein. Daraus ergibt sich:
Man verwendet **vorzugsweise** UPD zum Einmischen, wenn kleine Datenmengen (wenige hundert Sätze) in große Datenbanken (mehrere -zigtausend Sätze) einzufügen sind und ein Primärschlüssel existiert.
Aber: man verwendet INDEX **oder** UPD, wenn es keinen zuverlässigen Primärschlüssel gibt (→ Kap.9), aber UPD muß dann mit Option `-fm01` gestartet werden, um unerwünschte Ersetzungen zu vermeiden.
Man verwendet **unbedingt** UPD, wenn sich unter den neu importierten Sätzen solche befinden, die vorhandene Sätze ersetzen sollen. (Wie gesagt, es muß dazu einen zuverlässigen Primärschlüssel geben.)
Man verwendet **nie** INDEX im Mehrplatzbetrieb, wenn die Datenbank zugleich von anderen benutzt wird! INDEX ist nicht mehrplatzfähig, nur UPD.
Bis V15 galt:: Man startet UPD, wie PRESTO, im Einzelplatzbetrieb mit Option **`-S`**, im Mehrplatzbetrieb ohne diese Option.

Die Arbeit mit dem Importprogramm setzt immer voraus, daß man für einen konkreten Einsatzfall zunächst die Import-Parameterdatei konstruiert. Die vorhandenen Beispiele geben dabei Hilfestellung. Eine systematische Einführung in die "Importsprache" finden Sie in Kap.11.

Einige wichtige Eigenschaften des Importprogramms sollen hier noch einmal knapp zusammengefaßt sein:

- Es können beliebig viele gleichartige Dateien in einem Durchlauf importiert und in eine Zieldatei überführt werden.
- Das Importprogramm kann in einem Durchlauf bis zu 3 verschiedene Zieldateien erzeugen, d.h. es können 3 Export-Parameterdateien gleichzeitig geladen werden.
- Beliebige Import- und Exportparameter können gekoppelt werden, wodurch letztlich zwischen zwei Externformaten in einem Durchlauf konvertiert werden kann.
- Der Importvorgang ist am Bildschirm beobachtbar, wobei man hintereinander den Originaldatensatz (incl. Steuerzeichen!) und den umgewandelten Satz zu sehen bekommt.
- Ein laufender Importvorgang kann jederzeit zum manuellen Redigieren der bereits umgewandelten Daten unterbrochen oder auch abgebrochen werden.
- Wenn die Zieldatei eine ***allegro***-Grunddatei ist, kann sie sofort anschließend oder später mit UPD oder INDEX in eine ***allegro***-Datenbank umgewandelt oder in eine solche eingemischt werden. Mit ***a99*** kann man solche Dateien über das Menü "Datei" direkt einlesen (als "Offline-Ergebnismenge") und dann betrachten und die Sätze einzeln oder insgesamt abspeichern lassen. Mit dem FLEX-Befehl "update" kann ***a99*** auch die Funktionen des DOS-Programms UPD ausführen.

5.1 Programmablauf (DOS interaktiv bis V32)

Nachdem die Grundlagen und Konzepte erklärt sind, kann der eigentliche Programmablauf nun relativ knapp dargestellt werden. Zuerst in der Weise, daß man IMPORT **von Hand** startet und alle Angaben dann vom Programm abfragen läßt. Danach wird gezeigt, wie dasselbe per ***CockPit*** gemacht wird, am Ende noch als **Batchaufruf** (unter Win'7/64 und UNIX geht es nur so!).

0. Nehmen wir als Beispiel an, wir wollen 3 Dateien gleicher Struktur importieren, die auf dem Verzeichnis **`C:\TEMP`** liegen. Die Struktur sei das `MAB2`-Format. Die Import-Parameterdatei dazu ist **`MAB2.AIM`**, die Export-Parameterdatei **`I-1.APR`**. Die Zieldatei soll **`NEUDAT.ALG`** heißen und auf **`k:\opac`** liegen. Wenn sie schon existiert, soll sie überschrieben werden.

1. Start durch Wahl von '5' auf dem Menü "Funktionen" des ***CockPit*** (→ 0.11.1, dort sehen Sie das Funktions-Untermenü mit der Darstellung genau dieses Vorgangs)

   oder mit dem Befehl **`import -f5`** vom Programmverzeichnis aus.
   Im zweiten Fall geht es so weiter:

2. `Bitte Dateien zur Import-Bearbeitung auswählen`

   `Laufwerk(:Pfad\)` **`c:\temp\`**
   Man gibt i.a. den Namen des Datenpfades an, auf dem sich die zu importierenden Daten befinden. Wenn es sich um ein Unterverzeichnis des ***allegro***-Programmverzeichnisses handelt, z.B. ALLEGRO\DATA, braucht man nur den Namen dieses Verzeichnisses anzugeben, also in diesem Fall `data\`. Es empfiehlt sich, den Namen mit einem "backslash" (umgekehrten Schrägstrich) abzuschließen, da sonst u.U. nicht das Verzeichnis, sondern Dateien mit zufälligerweise gleichem Namensanfang auf dem Programmverzeichnis angewählt werden.
   Man kann auch beliebige Unterverzeichnisse auf anderen Laufwerken anwählen. Dann ist die komplette Pfadbezeichnung samt Laufwerk anzugeben, z.B. `a:\` oder `e:\fremd\`.

3. Das Programm zeigt die auf dem betreffenden Pfad liegenden Dateinamen, und zwar alle, auch z.B. Programmdateien. Man setzt jeweils ein '+' vor die Namen der umzuwandelnden Dateien und drückt dann <Enter> (→ 0.5 "Auswahlliste")

4. Man erhält nun die Auswahlliste der Import-Parameterdateien (Typ .cIM). Daraus muß die passende ausgewählt werden. Man fährt mit dem Cursor zu dem Namen, setzt '+' davor und drückt Enter.
   Das Programm meldet z.B. `MAB2.AIM wird geladen`

5. In derselben Form erscheint die Liste der Export-Parameterdateien, aus der man gleichfalls die richtige auswählt. Das Programm meldet z.B.
   `I-1.APR wird geladen...`

   Anmerkung
   Die Auswahllisten zeigen immer zuerst diejenigen Parameterdateien, die auf dem in 2. gewählten Datenverzeichnis stehen, wenn es dort welche findet. Wenn die gewünschte nicht dort steht, schaltet man mit **F2** um auf das Startverzeichnis, und man bekommt die dort vorhandenen Dateien. Der Sinn ist, daß man datenspezifische Parameterdateien im selben Verzeichnis unterbringen kann, wo die Daten stehen.

6. `Ausgabe auf (Laufwerk (Pfad\)Datei:  (x = keine Ausgabe gewünscht)`

   Antwort: **`k:\opac\neudat.alg`**
   Laufwerk und/oder Pfad sind nur anzugeben, wenn die Zieldatei nicht im Programmverzeichnis liegen soll.

Handelt es sich um ein Unterverzeichnis des aktuellen Verzeichnisses, so genügt dessen Name zusammen mit einem Dateinamen.
Wenn die Zieldatei existiert, kann das Programm diese entweder überschreiben oder die neuen Daten hinten dranhängen. Es fragt Sie, was es tun soll.

7. `noch zusätzliche Ausgabedatei? j/n`
   Wenn im selben Durchlauf eine zweite (und evtl. dritte) Zieldatei entstehen soll, muß man hier 'j' antworten, dann wiederholen sich 5-7.
8. `Originaldaten-Anzeige gewünscht? j/n`
   Wenn man 'j' antwortet, werden beim Durchlauf jeweils zuerst die Originaldaten, dann die umgewandelten Daten gezeigt. Sinnvoll ist das zum Testen neu erstellter Parameterdateien, weil man dann das Ergebnis sofort sieht. Das Programm läuft schneller, wenn man hier 'n' antwortet. Dann gibt es jeweils nur die Satznummer und -länge aus. *Wichtiger Hinweis:* Jederzeit kann man während des Programmlaufs 'a' drücken, um die Anzeige einzuschalten. Nochmals 'a' schaltet sie wieder ab.

9. `Selektion gewünscht?  j/n`
   Wenn man mit 'j' antwortet, kann man einen Suchbegriff angeben, nach dem dann jeder eingelesene Satz durchsucht wird. Die Umwandlung erfolgt dann nur, wenn der Suchbegriff vorkommt. Die Suchlogik ist dieselbe wie beim Suchprogramm (siehe Kap. 4). Einziger Unterschied: es können nur exakte Zeichenfolgen in exakt vorgegebener Schreibweise gefunden werden. Umlautauflösung und Egalisierung der Groß-/Kleinschreibung finden nicht statt, auch keine Ersetzung von Identnummern durch Stammsatzdaten wie beim Programm SRCH.

10. `manueller Eingriff gewünscht? j/n` **`j`**
    Es bringt einen geringen Geschwindigkeitsvorteil, hier mit 'n' zu antworten. Sinnvoll ist dies außerdem in fertigen Anwendungsumgebungen, wenn man IMPORT in Prozeduren ("batch files") einbaut, wo mehrere Dinge hintereinander ohne Eingriff ablaufen sollen.
    Man sollte immer mit 'n' antworten, wenn man sehr lange Datensätze (ab etwa 15K Größe) erwartet.
    Man sollte beim Testen immer mit 'j' antworten, dann ist es möglich, mit Druck auf 'x' den Programmlauf jederzeit zu unterbrechen; man gelangt in den Editor und kann mit <Bild↑> und <Bild↓> die bereits umgewandelten Sätze durchblättern, um die Ergebnisse sofort zu begutachten.
    Nun verrichtet das Programm sein Werk. Wenn man nicht vorher mit 'x' unterbricht (und dann vom Editor aus mit Befehl #e abbricht), arbeitet es die Quelldatei bis zum Ende durch. Wenn man unterbrochen hat, wird der Programmlauf mit F10 wieder aufgenommen (mit Befehl #r desgleichen).

11. Wenn man bei 10. 'j' gesagt hatte, zeigt das Programm am Ende den Speicherzustand an, so daß man sieht, wieviele Sätze noch im Arbeitsspeicher sind. Diese sind nun bearbeitbar wie im Redigiermodus (→ Kap. 3.5). Der Befehl #e bewirkt dann, daß die restlichen Sätze noch in die Zieldatei geschrieben werden. Die Abschlußmeldung teilt mit, wieviele Sätze eingelesen bzw. umgewandelt wurden.

Nach gelungener Umwandlung von Fremddaten in eine .ALG-Datei kann diese sofort mit dem Programm INDEX (→ Kap.7) oder UPD (→ Kap.9) in eine existierende Datenbank eingespeist werden, oder man macht, gleichfalls mit INDEX, eine neue daraus (→ Kap.7). Einen Gesamtüberblick über alle Vorgänge gibt das Diagramm in Anhang C.

Beachten Sie: Das Programm kann nicht prüfen, ob Fremddatei und Parameterdatei korrekt zusammenpassen. Wenn das nicht der Fall ist, wenn Sie z.B. eine .EXE-Datei importieren wollen oder wenn Sie dBase-Daten mit MARC-Parametern importieren wollen, kann es zum Absturz kommen. Irgendwelche Schäden sind dabei jedoch nicht zu befürchten.

Derselbe Vorgang wird vom ***CockPit*** aus so gestartet: (siehe auch → 0.11.1)

Sie sehen hier die Momentaufnahme zu dem Zeitpunkt, wo der Name für die Ausgabedatei eingegeben wird. Wenn Sie diese Abbildung mit der entsprechenden im vorigen Kapitel vergleichen, bemerken Sie eine große Ähnlichkeit. Der wichtigste Unterschied ist, daß das Importprogramm natürlich eine Import-Parameterdatei braucht und nur bis zu drei statt vier Exporte zugleich machen kann - was man aber kaum jemals ausnutzen wird. Als Datenverzeichnis ist hier dasjenige anzugeben, wo die Ausgangsdaten liegen, nicht dasjenige, wo die Zieldatenbank liegt - mit der letzteren hat IMPORT nichts zu tun, denn es wandelt ja die Daten nur um, das Einmischen muß dann UPD machen (→ Kap. 9).
Die Auswahl der tatsächlich zu importierenden Dateien findet erst nach dem Start des Programms (über Menüpunkt "GO (START)" statt. Dann erscheint die Auswahlliste aller auf C:\TEMP stehenden Dateien, und man setzt ein '+' vor jede gewünschte Datei. Wie schon erwähnt, müssen diese Dateien in ihrer Struktur zur der gewählten Import-Parameterdatei passen, sonst geht mit Sicherheit etwas schief.

5.2 Aufruf des Programms aus Batchdateien (ab V33 auch für Win‘7/64)

Das Programm IMPORT kann, wie die anderen, auch als Modul benutzt werden. Sämtliche oben beschriebenen Angaben, die das Programm abfragt, kann man ihm als Optionen "von außen" mitgeben, dann läuft es automatisch ab. Diejenigen Angaben, die es nicht über Optionen erhält, fragt es aber noch ab. Man kann es daher so einrichten, daß immer nur die wirklich veränderlichen Angaben im Dialog einzugeben sind. Makros werden ausführlich in Kap.12 behandelt. Der in 5.1 als Beispiel beschriebene Vorgang wäre mit folgendem Befehl zu starten:

`import -f5 -d c:\temp\`*`dateiname`* **`-i mab2 -e I-1/k:\opac\neudat.alg -v1 -s0 -m1`**

Genau diesen Befehl würde auch das ***CockPit*** produzieren, wenn man es auf diese Aufgabe ansetzt. Hinter **`-i`** kann ein Dateiname mit vollständigem Pfad stehen, wenn die Importparameterdatei weder auf dem Daten- noch auf dem Programmpfad steht (nur auf diesen zwei Verzeichnissen sucht das Programm danach).
Achtung: `import` erfordert in der Option -d eine der folgenden Bedingungen:
- Angabe eines absoluten Pfadnamens mit Laufwerk in der Option –d, (Namensmuster mit *, falls mehrere Dateien)
- Angabe einer Datei relativ zum aktuellen Verzeichnis (Angabe jedoch *ohne* .\), also darin oder darunter
Kann man eine solche Bedingung nicht herstellen, dann: vorher die Datei z.B. auf das aktuelle Verzeichnis kopieren.
Für **Routineaufgaben**, die sich regelmäßig wiederholen, wird man dafür ein Skript anlegen, womöglich sofort kombiniert mit einem nachfolgenden Aufruf von INDEX oder UPD, um die frisch umgewandelten Daten sofort in eine Datenbank einzuspeisen. Beispiele dafür sind die mitgelieferten Skripte DBDISK.BAT, OCLC.BAT, PICA.BAT und VLBKONV.BAT. Diese sind in der Datei CP.OPT verankert (in R- oder S-Zeilen, → 0.11.6), so daß sie vom ***CockPit*** aus gestartet werden können. Sie beziehen sich dann jeweils auf die gerade eingestellte Datenbank und erwarten die Fremddaten im Startverzeichnis unter den Namen MAB2.DAT, MARC.DAT, PICA.DAT bzw. VLB.DAT. Verwenden Sie diese Beispiele als Strickmuster für eigene derartige Prozeduren. Letztlich braucht der Anwender dann diese Kapitel nicht zu lesen - ein einziger Tastendruck im ***CockPit*** veranlaßt den automatischen Ablauf aller Vorgänge.

6 Export

Exportfunktionen in ***a99***: **`h exprt`**

6.0 Aufgabenstellung und Konzept

Export bedeutet: Daten in eine Datei oder auf ein Gerät ausgeben. Fast immer müssen dabei die Daten in irgendeiner Form verändert, man sagt auch **formatiert** werden, damit z.B. ein anderes Programm damit arbeiten kann oder damit überhaupt ein lesbarer Abdruck zustandekommt, denn in einer Datei sind die Daten computergerecht, nicht menschengerecht aufgezeichnet. Exportergebnisse müssen, je nach Verwendungszweck, den unterschiedlichsten Anforderungen genügen. Deshalb sollte ein Exportprogramm kein starres, festgelegtes Programm mit einem abgegrenzten Repertoire sein, sondern der Datenbank-anwender braucht Möglichkeiten, selbst zu definieren, wie das Programm in einem konkreten Fall Daten formatieren soll.

Heißer Tip:
Die Windows-Programme ***a99/alcarta*** verfügen über eine Methode, Daten in Tabellenform auszugeben, die man anschließend sofort mit Office-Programmen nutzen kann. Diese Methode findet man auf dem Export-Komfortmenü.
Sofortzugriff: geben Sie im Schreibfeld **`h table`** ein.

Das ***allegro***-System hat kein eigenes Export**programm**, das *nur* für diese Aufgabe da wäre, sondern es gibt eine Export**funktion**, die in alle Programme (außer ASORT) eingebaut ist, auch in die der "neuen Generation", ***acon, a99***, ***alcarta***. Alle Programme können also, zusätzlich zu ihren sonstigen Aufgaben, **auch** Daten exportieren, und alle in derselben Weise.

Unter den Begriff **Export** fallen bei ***allegro*** auch solche Funktionen wie Bildschirmanzeige, Druckausgabe, und sogar die Generierung von Registereinträgen für den Index. Das heißt: die genannten Programme nutzen die Exportfunktion auch für einige andere Zwecke. Die Vorschrift für die Programme, in welcher Form genau die Daten auszugeben sind, wird in jedem Fall mit Hilfe einer eigenen Sprache geschrieben, der **Exportsprache**. Die Vorschrift wird gespeichert in einer **Export-Parameterdatei** (Dateityp `.cPR`). Wer sich diese Sprache aneignet, kann Daten für unterschiedlichste Zwecke beliebig manipulieren, kann mit derselben Sprache aber auch die Bildschirmanzeige und die Registergestaltung programmieren. In Kap. 10 ist diese Exportsprache dokumentiert. Die Exportfunktion übernimmt somit auch solche Aufgaben, für die es bei anderen Datenbanksystemen einen sog. "Reportgenerator" gibt. ***allegro*** hat die DOS-Pakete QUEX und QUANT, mit denen bestimmte Typen von Exporten in einem Dialog erstellt werden können (→ 6.3/4), unter Windows gibt es das Exportmenü mit noch mehr Spezialitäten.
Die Exportfunktion im Sinne der **formatierten Ausgabe** von Daten wird vor allem vom Programm 4, dem Volltext-Suchprogramm SRCH miterledigt: dieses leistet **Selektion und Export** in einem Durchgang, und zwar bis zu vier Exportvorgänge in unterschiedlicher Form gleichzeitig.

Wenn man den Befehl **`srch -f6`**
eingibt, startet man de facto einen Exportvorgang, denn die Option **`-f6`** bewirkt, daß keine echte Suche stattfindet, sondern **jeder** Datensatz als "Treffer" behandelt wird. Hinter "**F**unktionen / **6**" vom ***CockPit*** verbirgt sich nichts anderes als dieser Befehl. Man erhält die Liste der Export-Parameterdateien zur Auswahl.
Insbesondere kann man Ergebnisse als Grunddatei (Typ *.d*LG) ausgeben lassen (dazu wählt man `I-1.APR`) und anschließend mit ASORT sortieren und/oder mit INDEX oder UPD in eine andere Datenbank überführen. Wählt man zum Exportieren die `E-W.APR`, kann man die entstandene Datei dem ***avanti***-Server oder auch ***a99*** zum Einmischen in eine andere Datenbank übergeben. Das ist eine ganz neue Methode, die noch sehr wichtig werden wird.

Eine **Produktion** findet statt, wenn aus einer größeren Datenmenge, im allgemeinen mehreren Dateien, automatisch ein neues Produkt erstellt wird. Das kann eine Liste, ein Stapel Karteikarten, aber auch eine neue Datei sein, die man u.U. mit anderen Programmen weiterverwenden oder -bearbeiten will.
Alle Produktionsaufträge folgen demselben Schema: Nimm das vorhandene Datenmaterial (***allegro***-Dateien der Typen *.d*LG und *.d*LD) und verarbeite dieses nach bestimmten Vorschriften (= Parameterdatei) zu einer Ausgabe (= Datei oder Druck).
Ein Beispiel ist die Ausgabe einer Datei in Form einer Liste mit Zeilen- und Seitenumbruch. Dazu eignet sich die Parameterdatei `P-NORMAL.APR`.
Ein zweites Beispiel wäre die Ausgabe von Daten als sog. ASCII-Datei, mit allen Kategorienummern und darin vorhandenen Angaben, um diese Daten mit einem anderen System auswerten zu können. Dazu gibt es die Standard-Parameterdatei namens `E-1.APR`. Diese ist sogar unabhängig vom Kategoriesystem. Wenn man Sie z.B. auf `E-1.PPR` kopiert, ist sie sofort für `P.CFG` verwendbar.

6.1 Sortierte Listen produzieren

Funktionen in ***a99*** *:* **`h exprt`**

Vorbemerkungen
Im **Windows-System** geht vieles einfacher: Man kann eine Ergebnismenge schon vor dem Export online sortieren lassen. Entweder direkt im Kurzanzeige-Fenster mit dem Button [Sort] (→ Kap.2.1.1) oder mit Hilfe der View-Technik (`h viewh` eingeben).
Es geht in diesem Kapitel um das DOS-System, wo die Dinge anders liegen.
Wer sofort wissen will, was man im **Normalfall** tut, kann gleich zu **6.2 bis 6.3** weiterblättern. Hier folgt zuerst etwas Theorie. Das folgende gilt sinngemäß für jede Konfiguration. Die geeigneten Parameterdateien werden jedoch nur für die Konfiguration A.CFG mitgeliefert. Für andere wären sie vom Anwender zu erstellen.
Wenn eine ***allegro***-Datei sortierbar sein soll, muß das Sortierkriterium, z.B. der Verfassername, das Schlagwort, die Signatur etc., zunächst an den Anfang jedes Datensatzes gestellt werden. Die Sätze müssen dabei ansonsten in ihrer Struktur unverändert bleiben. Nur: aus .ALD-Dateien werden .ALG-Dateien gemacht, denn das Programm **ASORT kann nur Grunddateien sortieren**! Für diesen Vorgang können als Muster die Parameterdateien **S-*.APR** dienen, die für eine Anzahl unterschiedlicher Sortierfolgen mitgeliefert werden. In diesen Dateien können Sie Veränderungen vornehmen, z.B. andere Sortierfelder festlegen und/oder eine andere Sortieraufbereitung programmieren. Nach dem Sortieren muß dann ein weiterer Produktionslauf stattfinden, z.B. mit einer der Parameterdateien P-*.APR, um das endgültige Produkt zu erstellen.

Drei Vorgänge müssen folglich hintereinander ablaufen, wenn eine sortierte Liste entstehen soll:

Selektieren : Auswahl der zu druckenden Titel und Vorbereitung als sortierfähige Datei
Dazu gibt es zwei Möglichkeiten:
a) Bildung einer Ergebnismenge im Schnellzugriff und Export mit F4 (Prog. PRESTO, → Kap.1.5)
b) Heraussuchen und Exportieren mit Hilfe der Volltextsuche (Programm SRCH, → Kap.4)
In beiden Fällen entsteht eine noch ungeordnete Datei, die aber schon **sortierfähig** ist.
Und das heißt: die Datensätze haben noch die kategorisierte Form, aber der gewünschte Sortierbegriff (z.B. Sachgruppe+Name+Erscheinungsjahr, ...) ist in Kategorie `#u1` an den Anfang gesetzt.
Das selektierende Programm (PRESTO oder SRCH) muß aber vorher wissen:
1. welche **Sortierfolge** gewünscht wird, SORT.APR
2. ob alle **Datenelemente** für die Liste gebraucht werden oder nur eine Auswahl SELECT.APT
Die nötigen Anweisungen entnimmt das Programm aus den rechts angegebenen Parameterdateien.

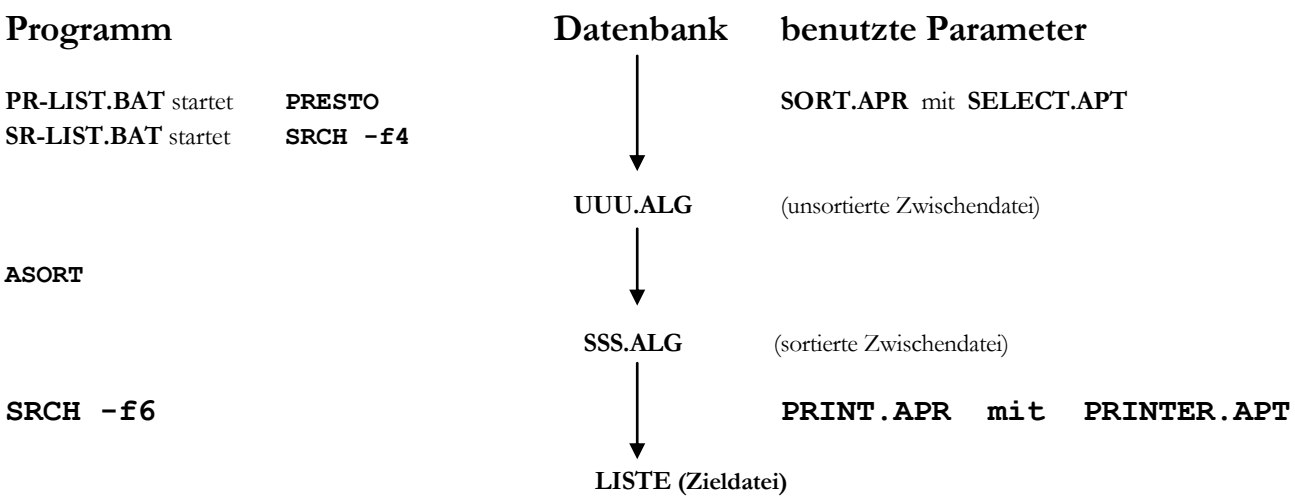
Sortieren : das Programm ASORT ordnet die entstandene sortierfähige Datei. Hierbei bleibt die interne Struktur der Daten unverändert (Typ .ALG), d.h. die Datei ist noch nicht **druckfähig**. Deshalb:

Formatieren : Produktion der druckfähigen Form und sofortige Ausgabe auf dem Drucker.
Oder in eine Datei, die dann evtl. noch mit einem Textprogramm nachbearbeitet werden kann.
Diesen Schritt führt **immer** das Programm SRCH aus, und es muß dazu zwei Dinge wissen:
3. welche **Druckgestaltung** gewünscht wird PRINT.APR
4. welcher **Drucker** benutzt werden soll. PRINTER.APT
Die Anweisungen dazu findet das Programm in den rechts angegebenen Parameterdateien.

Ein grundlegend wichtiger Gedanke für die **Rationalisierung** der Produktion ist nun dieser: Wenn man für jede der Teilaufgaben 1. bis 4. eine Anzahl von Parameterdateien hat, die jeweils ihre Teilaufgabe in sinnvoller Weise lösen, kann man je eine aus jeder Gruppe auswählen und dadurch beliebige Kombinationen zusammenstellen. Dadurch gewinnt man erheblich mehr Flexibilität, als wenn man alle Einstellungen für eine sortierte Liste in nur einer Parameterdatei unterbringen müßte.
Nur zwei Batchdateien werden für die Listenproduktionen benötigt: **PR-LIST.BAT** startet PRESTO und ermöglicht den Export von Ergebnismengen, **SR-LIST.BAT** startet SRCH und erzeugt die Liste mittels Volltextsuche.
Im wesentlichen enthalten diese Batchdateien die folgenden drei Anweisungen für die drei Vorgänge: (die nötigen Angaben zur Datenbank hat ***CockPit*** vorher mit SET-Befehlen im Environment untergebracht!)

```
presto -e SORT/uuu.alg ...        bzw.        srch -e SORT/uuu.alg -f4 -m0 -v0 ...
asort uuu.alg sss.alg
srch -dsss -m0 -e PRINT/liste -f6 -v0 -b%-D%\%-B%
```

Beide Batchdateien machen also im Prinzip dasselbe:



Für jede der oben genannten vier Aufgaben 1. - 4. werden einige Parameterdateien mitgeliefert. ***CockPit*** ermöglicht die Auswahl unter diesen Dateien (im Menü **Routinen / Volltextsuche** oder **Optionen**) und kopiert die gewählte jeweils um auf den Namen SORT.APR usw., wie oben angegeben. So wird erreicht, daß man mit den zwei Batchdateien auskommt und nicht für jede Art von Listenproduktion eine eigene machen muß. Die Produktion selbst wird über das Menü **Routinen** gestartet: der Menüpunkt **exportieren** enthält die nötigen Unterfunktionen (s.u.).
Die 4 Arten von Parameterdateien haben festgelegte Namensformen: (an der Stelle des '*' steht der individuelle Name der jeweiligen Parameterdatei. Zu den Exporttypen siehe → Kap.10.0):

1. **`S-*.APR`** Diese Parameterdateien (Exporttyp SORT) erledigen die Aufbereitung der Sortierkriterien. Sie produzieren eine Kategorie **`#u1`**, die im Exportsatz am Anfang steht und und die den sortierfähigen Text enthält. Dahinter kommt eine Kategorie **`#u2`**, die denselben Inhalt hat, jedoch in Originalform. #u2 kann dann am Ende für den Druck benutzt werden (siehe 3.). Eine Datei dieser Gruppe ist z.B. S-PJ.APR; sie erzeugt Sortierköpfe zur Ordnung nach **P**erson und **J**ahr.

2. **`S-*.APT`** Diese Dateien enthalten die Anweisungen für den von der Sortierung unabhängigen Hauptteil des Datensatzes. In dieser Gruppe befinden sich die Dateien S-KURZ.APT, S-MITTEL.APT, S-LANG.APT und S-VOLL.APT. Die Namen lassen schon erkennen, daß die Datensätze dadurch in unterschiedlicher Vollständigkeit ausgegeben werden.

Je eine Datei aus diesen beiden Gruppen wird vom ***CockPit*** umkopiert auf den Namen SORT.APR bzw. SELECT.APT, und die zweite wird von der ersten nachgeladen (mit dem Befehl tSELECT). Auf diese Weise ist erreicht, daß jede Sortierform mit jeder Stufe der Vollständigkeit kombiniert werden kann. Produkt dieser Kombination ist in jedem Fall eine Datei UUU.ALG, die die selektierten Sätze mit den gewünschten Sortierköpfen und der Auswahl von Feldern enthält. ASORT erledigt die Sortierung und liefert die Datei SSS.ALG. Diese nimmt SRCH sich vor und bearbeitet sie mit einer Kombination der Typen 3 und 4:

3. **`P-*.APR`** Diese Dateien (Exporttyp PRINT) steuern das Aussehen der gedruckten Liste, also deren Anordnung, Interpunktion und Zeilenumbruch. Die Kopfelemente, die einem Ausgabesatz vorangehen, werden dabei jeweils aus der Kategorie #u2 entnommen (siehe oben)! Daher brauchen diese Dateien nicht zu "wissen", ob sie nun eine nach Namen, Sachgruppen, Signaturen oder sonstwie geordnete Liste ausdrucken. Der Sortierbegriff in druckfähiger Form steht in jedem Fall in #u2. Wenn es mehr als eine Ordnungsgruppe gibt (S-VJ.APR erzeugt z.B. zwei: Name und Jahr), dann sind sie durch **`"¶"`** (`<Strg>+t`) getrennt, so daß sie mit den Befehlen b**"¶"** und e**"¶"** separiert werden können.

Die Druckaufbereitung erfordert, wenn man mit Stammsätzen und mit selbständiger Speicherung von Banddaten arbeitet, das Nachladen aus der Datenbank. Auch deshalb ist dieser Teil der komplizierteste der gesamten Aufgabe. Aufsätze aus Zeitschriften und Sammelbänden, hierarchisch gestufte Teile mehrbändiger Werke, Serienstücke - alles dies muß korrekt ausgedruckt werden.
Ferner drucken diese Parameterdateien kurze wie lange Datensätze gleichermaßen aus. Für unterschiedlich ausführliche Listen werden also an dieser Stelle nicht verschiedene Parameterdateien benötigt: schon bei der Selektion werden ja die nicht gewünschten Kategorien weggelassen.
Mitgeliefert werden unter anderen: P-NORMAL.APR, P-EINZEL, P-WORDP, P-1ZEIL und P-2ZEIL. Was machen sie? Die erste ersetzt einen sich wiederholenden Kopf durch "---", die zweite druckt jeden einzelnen Kopf jedesmal aus, die dritte liefert eine WordPerfect-Rohdatei (sie enthält Kommentare zu ihrer Benutzung, bitte lesen). P-1ZEIL und P-2ZEIL schließlich stellen Listen her, die spaltenförmig angeordnet sind, mit 1 bzw. 2 Zeilen je Titel.

4. **`P-*.APT`** Das sind die Druckertreiber. Sie enthalten die Codes für die verschiedenen Schriftattribute (Zwischenteile 79-99) sowie, falls nötig, p- und q-Befehle zur Umcodierung von Zeichen. Sie können auch Angaben für Zeilenlängen, Satzende, Druckerinitialisierung etc. enthalten, die dann die Standardangaben in P-*.APR ersetzen. Bis Version 11.2 wurden solche Parameterdateien unter einer anderen Namensform geliefert: _*.APT. Diese wurde aufgegeben. Der aktuelle Druckertreiber wird von PRINT.APR aus mit dem Befehl `tPRINTER` geladen. Dieser Befehl steht jeweils am Ende der Datei `P-*.APR`.

Alle vier Gruppen sind vom Nutzer beliebig erweiterbar. Hält man sich an die angegebene Namensform (S-* bzw. P-*), wird ***CockPit*** jede neu geschaffene Datei bei der Auswahl mit anbieten.

6.2 Und so produziert man eine sortierte Liste

Windows: **`h exprt: Sortierte Listen`**

Nochmals: Die Windows-Programme stellen zur Produktion sortierter Listen mehrere Methoden bereit. Darunter ist eine, genannt "Registerausschnitt", mit der man die Liste unmittelbar in der Reihenfolge eines Registers erstellen lassen kann. Alle Methoden sind sofort zugänglich auf dem Exportmenü: geben Sie ein **`h exprt`**
Die hier beschriebene Methode wurde unter DOS entwickelt, ist auf dem Windows-Exportmenü aber gleichfalls noch benutzbar.

1. Einstellungen vornehmen **`µr v 1-4`**

CockPit-Menü **`Routinen / Volltextsuche + Listen`** ansteuern, dann nacheinander die Punkte 1 - 4 unter **`Export-Optionen`** anwählen:

`1 Sortierung`: Auswahl zwischen unterschiedlichen Ordnungsverfahren, z.B. `S-PT.APR`
`2 Vollständigkeit`: Auswahl zwischen Kurz, Mittel, Lang und Voll, z.B. `S-KURZ.APT`
`3 Listengestaltung`: Wahl des Ausgabeformats, z.B. `P-EINZEL.APR`
`4 Drucker`: Wahl des "Druckertreibers", z.B. `P-DSKJET.APT`

Wenn man einen dieser Punkte anwählt, erscheint jeweils die Liste der verfügbaren Parameterdateien. Wenn man eine davon mit <Enter> auswählt, sieht man deren Kopfzeilen, die einen erklärenden Text enthalten.
An dieser Stelle kann man die auszuwählende Parameterdatei auch vorher noch betrachten, bearbeiten, oder eine neue Version davon anlegen. Man beantwortet dann die Frage `"ist das die richtige?"` mit 'v' (für **v**ielleicht) und bekommt die Möglichkeit zu den genannten Aktionen.
CockPit kopiert die gewählte Datei auf den jeweiligen Namen (SORT.APR usw.). Dadurch bleiben diese Einstellungen über die aktuelle Sitzung hinaus erhalten. Man braucht also nur dann wieder dieses Menü aufzusuchen, wenn eine der Einstellungen zu ändern ist. Damit sind dann die oben unter 1. - 4. erwähnten Fragen beantwortet.

2. Vorgang starten **`pr e`**

Menü **`Routinen`**, Punkt **`Volltextsuche/Listen`** ansteuern. Man sieht fünf Unterpunkte, die ersten zwei starten die Produktion: (das Programm zeigt dann zur Kontrolle noch einmal die aktuellen Einstellungen an)

`1` **`Selektieren per Schnellzugriff`** : die Datenbank wird zur Benutzung geöffnet wie sonst auch. Man stellt beliebige Ergebnismengen zusammen und exportiert sie mit **F4**. Soll etwa eine Neuerwerbungsliste entstehen, wird man die Ergebnismenge über Register 9 (Erfassungsdatum oder Zugangsnummer) zusammenstellen. Sobald man mit F8 die Datenbank verläßt, laufen die anderen Vorgänge automatisch ab. Es entsteht eine Datei namens LISTE, die das Endprodukt enthält, zum Abschluß bekommt man sie zu sehen. Mit Menüpunkt 4 bzw. 5 kann man sie auch später noch betrachten bzw. drucken (siehe unten).
Anm.: Aufgerufen wird die Stapeldatei PR-LIST.BAT.

`2` **`Selektieren per Volltextsuche`** : das Programm SRCH wird aufgerufen.
Es fragt nach einem Suchbegriff (→ Kap.4.3), sucht die gesamte Datenbank als Volltext danach ab, exportiert die gefundenen Sätze in derselben Art wie PRESTO. Danach läuft alles genauso ab wie unter 1, d.h. das Ergebnis ist ebenfalls eine Datei mit Namen LISTE.
Anm.: Aufgerufen wird die Stapeldatei SR-LIST.BAT.

Die anderen Punkte unter "Volltextsuche/Listen" sind diese:

`3` **`nur Volltextsuche`** : SRCH wird gestartet und fragt nach einem Suchbegriff (→ Kap.4). Es wird aber nichts exportiert und keine Liste erstellt. Diese Funktion ist nur eine **Hilfe,** um schnell eine Volltextsuche in der Datenbank durchzuführen, die Ergebnisse aber nur am Bildschirm zu betrachten.

`4` **`Produzierte Liste betrachten`** : am Bildschirm wird die vorher erstellte Datei LISTE angezeigt. Mit den Cursortasten und <Bild↑> und <Bild↓> kann man darin blättern.

`5` **`Produzierte Liste drucken`** : Die Datei LISTE wird an den Drucker übergeben.
Achtung: Wenn man vorhat, diese Funktion zu nutzen, gibt man vor dem Start des ***CockPit*** (also vor dem Startbefehl **`cp`**) den MS-DOS-Befehl **`print`** , denn sonst könnte das Drucken schiefgehen. Der Grund ist, daß der PRINT-Befehl von MS-DOS einen sog. "residenten" Teil hat, der geladen sein muß, bevor ein Anwendungsprogramm drucken kann.

Wenn man aber nicht mit A.CFG arbeitet?

Dann muß ein vergleichbares Arrangement (für eine Konfiguration c.CFG) erst geschaffen werden:

- Parameterdateien für Sortieraufbereitung und Kategorieauswahl (Typen 1. und 2.) anlegen, wobei die mitgelieferten als Muster dienen können. Die Namensformen `S-*.cPR` und `S-*.cPT` sind verbindlich.
- Zunächst eine vorhandene Druckparameterdatei auf P-NORMAL.cPR umkopieren. Die Kopfbehandlung (Auswertung der Kategorie #u2) analog zu dem Muster P-NORMAL.APR anpassen.
- Die eigenen Druckertreiber auf die Namensform P-*.cPT bringen. Die mitgelieferten P-*.APT können ohne weiteres in .cPT umbenannt werden, denn sie sind unabhängig vom Kategoriesystem.

Damit sollte die Sache in den Grundzügen funktionieren und läßt sich dann beliebig erweitern und verfeinern. Die Batch-Dateien PR-LIST.BAT und SR-LIST.BAT sind unverändert verwendbar. In der Regel wird es ein Haufen Arbeit sein, die Parameterdateien in diesem Sinne einzurichten, aber es lohnt sich!
Die Anwender anderer Konfigurationen haben in der Regel bereits eigene Batchsysteme für die Produktion von Listen aller Art. Diese sind unverändert weiter benutzbar. Die hier vorgestellte Methodik mag jedoch eine Anregung sein, neue Entwicklungen in ähnlicher Weise zu systematisieren.

Auch wer nicht mit `$a.cfg` arbeitet, kann die Komfortfunktionen des Windows-Systems, z.B. die Tabellenfunktionen, ohne weitere Voraussetzungen nutzen!

6.3 QUEX : Listenproduktion ganz einfach (DOS)

a99: **h exprt**, "Registerabschnitt" und "Tabellen"

QUEX (= QUick and Easy eXport) ist eine stark vereinfachte DOS-Exportmethode, die ohne Parametrierung auskommt. Für schnelle ad-hoc Listen ohne Anspruch auf Perfektion oder spezielle Finessen braucht man nicht zu parametrieren, besonders wenn ohnehin eine Nachbearbeitung mittels WORD, dBase o.a. vorgesehen ist.

Was leistet QUEX?

- Die Methode ist weitestgehend unabhängig vom Kategoriesystem, d.h. sie funktioniert für jede Konfiguration, ohne daß man an den Parameterdateien etwas ändern muß.
- Dialoggeführte Erstellung einfacher sortierter Exportdateien ohne jede Parametrierung, wobei besonders die Eignung der Datei für die Übernahme durch andere Software im Vordergrund steht. (Wenn man etwa Daten an eine zeitgenössische Textsoftware übergibt, hat man heute enorme Möglichkeiten der Umstrukturierung mittels Makros, daher muß nicht immer schon eine diffizile Struktur durch "allegro" geliefert werden.)
- Wahl einer Kategorie für die Sortierung, mit Alternativen für den Fall, daß die Kategorie nicht besetzt ist.
- Für den Export vorgebbar sind u.a.: die Sortierweise (z.B. auch numerisch), die Zeilenlänge sowie Zeichenkombinationen für den Satzanfang, die Feldtrennung und das Satzende, ferner Prä- und Postfixe für jedes auszugebende Feld.
- Hohe Geschwindigkeit und geringer Platzbedarf, damit das Verfahren auch für große Eportmengen anwendbar ist.

In die ***CockPit***-Vorgabendatei CP.OPT ist folgende Zeile unter "Eigene Routinen" eingebaut:

```
R QUick & Easy eXport, QUEX.BAT, Einfache Listen und Dateiausgabe
```

damit man mit einem Knopfdruck starten kann. QUEX.BAT kann man auch von Hand starten, dann teilt es mit, welche Environment-Variablen man vorher setzen soll.

Ergebnis: Heraus kommt am Ende eine alphabetisch geordnete Datei namens LISTE mit den selektierten Datensätzen.

Es folgt eine knappe Übersicht des Ablaufs. (Vollständige Beschreibung: siehe Datei QUEX.TXT)

Zuerst muß man eine Reihe von Fragen beantworten: (Fragen erscheinen grün, Antworten gelb)

0. Wollen Sie

```
1 : Alles mit der Hand eingeben?
2 : Frühere Einstellungen übernehmen?
```
(Das geht natürlich beim ersten Mal noch nicht)

Wenn man '1' wählt, geht es mit 1. weiter, bei '2' kann man gespeicherte Einstellungen früherer QUEX-Produktionen erneut aktivieren. Routineproduktionen lassen sich damit weitestgehend automatisieren.

1. Nach welcher Kategorie soll sortiert werden?

```
(Geben Sie eine oder mehrere Kategorien an, getrennt durch Kommas:
z.B.    40,60,20
(Das bedeutet: wenn #40 nicht belegt, wird #60 zum Sortieren genommen, usw.)
```

Wenn der Sortierbegriff ein Teilfeld ist: z.B. 245$a eingeben, dann wird Teilfeld ▼a von #245 genommen.

2. Wenn keine der Sortierkategorien vorkommt, was dann?

```
0 : Satz nicht ausgeben
a : Satz am Anfang der Liste plazieren
e : Satz an das Ende der Liste setzen
```

3. Sortiermodus:

```
1 = Normal (Umlaute auflösen, Nichtsortierwörter beseitigen)
2 = Pica-Namensmodus (Vorname@Nachname)
8 = Numerisch sortieren
9 = Keine Umcodierung - unverändert sortieren
```

Hier können unterschiedliche Sortiermethoden eingestellt werden. Je nach Wahl wird die Sortierkategorie dann entsprechend manipuliert. Beim numerischen Sortieren werden auch Negativzahlen korrekt behandelt!

Für Sortiermodus 1 oder 2 wird die Tabelle S.APT benutzt.

4. Welches Zeichen ist das Trenn- oder Endezeichen des Sortierfeldes?

```
( ;  ¶  - / . , ( * ' : =  oder  b=Leerzeichen, $=Teilfeldzeichen )
(Wenn Eingabe nicht klappt, Eingabe mit t davor, z.B. t; )
```

Mit Hilfe des Trennzeichens wird die Sortierkategorie zerteilt.

5. Welche Teile des Sortierfeldes sind auszuwerten:

```
Wählen Sie:     1 = nur der Anfangsteil  (Ende beim Trennzeichen)
                2 = jeder Teil    (auch Teile zwischen den Trennzeichen)
```

Wenn man 2 wählt, können natürlich mehrere Ausgabesätze je Datensatz entstehen.

6. Wieviele Ausgabesätze maximal je Datensatz?

```
(Geben Sie eine positive Zahl ein, oder 0 für unbegrenzt)
(wenn Sie <Enter> drücken, wird 1 angenommen)
```

Hier kann man ein Maximum setzen, wenn durch die Mehrfacheinträge zu viele Ausgabesätze entstehen könnten.

7. Welche Art der Ausgabe-Aufbereitung ?

```
1 = nach Art von dBase (comma-delimited)          (Dann weiter bei ──> 8a)
9 = nach eigenen Vorgaben                         (Dann weiter bei ──> 8b)
```

Wenn man mit 1 antwortet, wird die Datei quexdb.apt auf export.apt kopiert und es geht bei 8a weiter.

Wenn man hier 9 eingibt, muß man noch einige Fragen beantworten –> 8b:

(An dieser Stelle können *Allegrologen* Erweiterungen mit eigenen quexXY.apt-Dateien anbringen, um bestimmte Aufbereitungen vorzuprogrammieren und diese hier zur Auswahl anzubieten)

8a. Welche Kategorien sollen exportiert werden, außer der Sortierkategorie?

```
(Geben Sie eine oder mehrere Kategorien an, getrennt durch Kommas:
      z.B.    20,74,75,76,77,90
```

Es kann leider nicht eine Gruppe von Kategorien hier mit einer einzigen Angabe wie z.B. "#40." zusammengefaßt werden. Teilfelder können aber vorgegeben werden: 100$a,260$c usw.

Daraus wird zunächst %C9% gemacht und später verwendet (s.15.).....──> Weiter geht es bei 10.

Man beachte: die ersten zwei werden mit zum Sortieren herangezogen, wenn der Sortierbegriff nicht eindeutig ist.

8b. Wiederholungszeichen:

```
<Enter>, wenn Sortierbegriff jedesmal mit auszugeben ist
0      , wenn der Sortierbegriff nicht mit auszugeben ist
```

Sonst geben Sie hier die Zeichen ein, z.B. **** oder ---, die gesetzt
werden sollen, wenn der Sortierbegriff sich wiederholt
(Leerzeichen gehen nicht: statt dessen den Unterstrich _ benutzen, wird dann automatisch durch Leerzeichen ersetzt)

Danach sind für jedes auszugebende Feld (n = 1..19) folgende Fragen zu beantworten: (Die erste Kategorie ist beim Druck die zweite, denn zuerst kommt die Sortierkategorie, intern #u2. Es sei denn, man hat 8b mit 0 beantwortet)

- **Die 1. Kategorie bitte:**
  Man gibt die Nummer ein, z.B. 20
- **Welches Textelement soll VOR diese Kategorie? (C = Neue Zeile)**

```
0=keins       1='. '        2=', '        3='; '        4=C
5=' : '       6=C C         7=' = '       8='. - '      9=' / '
10=' ('       11=')'        12=' ['       13=']'        14='   '
Geben Sie nur die Nummer ein!
```

  Man gibt z.B. 8, wenn Nummer 8 gewünscht ist
  (Diese Liste von Zwischenteilen ist änder- und erweiterbar: sie steht in QUEX2.BAT, QUEX3.BAT und Q-EXPORT.APT. An allen drei Stellen muß man dieselben Werte eintragen.)
- **Das Textelement soll erscheinen:**

```
1 = In jedem Fall     2 = Nur wenn die Kategorie belegt ist
```

  Wenn Nachverarbeitung durch Text- oder Datenbanksoftware vorgesehen ist, wird hier meistens 1 die richtige Antwort sein. Man muß dann überlegen, wie man hinterher mit leeren Feldern umgeht.
- **... und welches soll dahinter?**

```
(Nummer eingeben, <Enter> wenn nichts)
```

  Wenn die Kategorie belegt ist, wird sie mit diesem Element beendet.

9. Die 2. Kategorie bitte: (Ende: x)

...

Es wiederholen sich die vier Fragen unter 8b. bis zu 19mal, d.h. man kann bis zu 20 Kategorien vorgeben, jeweils mit eigenen Prä- und Postfixen. Eingabe von 'x' statt einer Kategorienummer beendet die Schleife.

10. Zeilenlänge? (0 = ohne Umbruch)

Will man den Zeilenumbruch einem Textsystem überlassen, gibt man hier 0 ein, ansonsten eine Zahl, die für den benutzten Drucker und die eingesetzte Schriftart brauchbar ist, z.B. 72.

11. Satz-Anfangszeichen? (in "..." , oder C = Neue Zeile)

Am Anfang eines Satzes braucht man in der Regel kein Steuerzeichen, jedoch kann man hier etwas einsetzen. Wenn man C eingibt, erhält man eine Leerzeile zwischen zwei Datensätzen. Drucker-Escape-Sequenzen können hier zusätzlich als Folge von Dezimalzahlen eingegeben werden, z.B. **27 40 115 49 54 72 C** (ohne Anführungszeichen!).

12. Satz-Endzeichen? (in "..." , oder C = Neue Zeile)

Und am Ende des Satzes könnte ebenfalls eine beliebige Kombination verlangt werden.

Das Normale ist, hier C einzugeben (nicht "C"), um eine neue Zeile am Ende des Satzes zu erzeugen, oder C C, wenn es zwei sein sollen. Jedoch ist auch jede andere Kombination möglich, etwa " ENDE" C , um das Wort " ENDE" an den Schluß jedes Ausgabesatzes zu setzen, und dann einen Zeilenvorschub auslösen.

Hintergrund für Allegrologen: Die Datei Q-EXPORT.APT muß vorhanden sein. Sie wird kopiert auf EXPORT.cPT, die dann automatisch ergänzt wird: die Antworten auf 8. bis 12. werden an die Datei EXPORT.cPT angehängt, diese wird per `"tEXPORT"` in P-QUEX.cPR eingebunden. Die Antwort auf Frage 8a. wird dagegen als `-Uc9%C9%` an das Programm `srch` übergeben (s. 15). So kommt man auch bei der Dateiausgabe ohne Eingriff in die Parameterdatei aus.

Jetzt kommt eine Kontroll-Anzeige, wo man nochmals die wichtigsten Einstellungen sieht. Wenn irgendwas nicht stimmt, antwortet man mit 'n', und es geht wieder von vorn bei 1. los.

Wenn man aber 'j' sagt:

13. Wie wollen Sie die Daten herausziehen?

```
1 : mit Volltextsuche
2 : als Ergebnismenge (Export mit F4)
```

Bei '1' wird SRCH aufgerufen, dann muß man einen Volltext-Suchbegriff eingeben,

bei '2' wird PRESTO aufgerufen, dann muß man eine Ergebnismenge bilden und sie per F4 exportieren.

Hintergrund: Beim Aufruf wird `-Uc1%C1% -Uc2%C2%` mitgegeben, d.h. es entstehen die Anwendervariablen `#uc1` und `#uc2`, diese werden in S-QUEX.APR verwertet.

14. Es entsteht die Datei QX.cLG, mit ASORT wird sie sortiert. Das Ergebnis ist dann QUEX.cLG; QX.cLG wird gelöscht.

15. SRCH verarbeitet QUEX.cLG zu der Ergebnisdatei LISTE.

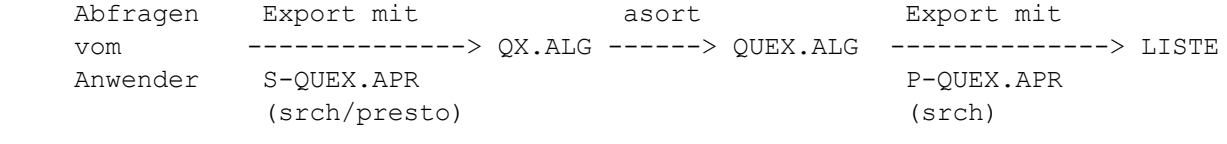
Dabei wird mit `-Uc9%C9%` die Liste der auszugebenden Kategorien dem Programm übergeben. Zur Aufbereitung wird EXPORT.cPT herangezogen, das oben unter Punkt 4 bis 8 erzeugt wurde. (Wenn man bei 7. mit 1 geantwortet hat, ist EXPORT.cPT eine Kopie von QUEXDB.APT).

Zur Umcodierung der Ausgabe wird PRINTER.cPT herangezogen, d.h. man sollte vorher per CockPit oder von Hand die geeignete Druckertreiberdatei auf PRINTER.APT kopieren (bzw. z.B. PRINTER.PPT, wenn P.CFG benutzt wird)

QUEX.cLG wird am Ende gelöscht.

Hintergrund: Das Verfahren ist auch bei größeren Datenmengen effizient, weil die Sortierdatei relativ klein ist, denn die Datensätze selbst werden erst im letzten Schritt (15.) nachgeladen. V14-Ersetzungen werden ausgeführt.

Schematischer Ablauf von QUEX.BAT: (für Konfiguration A)



6.4 QUANT für Statistiken

a99: **`h ct`**

Neben dem Aufgabenbereich der rechnerischen Auswertungen (Exporttyp R) gibt es die Häufigkeits-Auswertungen von Datenfeldeinhalten. Nur um diese soll es hier gehen. Unter DOS gibt es dafür QUANT, die universelle QUANTitative Auswertungsmethodik. Wie mit QUEX kann man auch mit QUANT sofort arbeiten, weil es in einem Dialog abläuft.
Beliebige *Datenfeldinhalte* können ausgewertet werden. Das Windows-Programm hat noch eine andere Statistik-Methode, die das Auswerten von *Vorgängen* ermöglicht, etwa bei der Ausleihe: **`h statist`** eingeben.

Was leistet QUANT?

- Auch diese Methode ist unabhängig vom Kategoriesystem, d.h. funktioniert ohne weitere Vorbereitung mit jeder Datenbank. V14-Ersetzungen werden automatisch ausgeführt.
- Beliebige Kategorien sind auswertbar, auch mehrere verschiedene in einem Durchlauf; außerdem:
- Mehrfachbesetzungen innerhalb einer Kategorie können zerlegt werden, um alle Einträge zu zählen,
- oder es wird wahlweise nur der Anfangs- oder Endteil einer Kategorie berücksichtigt werden, bis zu oder ab einem bestimmten Trennzeichen.
- Es gibt mehrere Sortierweisen zur Wahl, u.a. auch numerisches Sortieren (für Felder, die Zahlen enthalten)
- Schwellenwerte sind vorgebbar ("nur Fälle mit mindestens M oder höchstens N Einträgen sollen in der Liste stehen"), wenn man nur die ganz großen oder die ganz kleinen Werte sehen will.
- Als Ergebnis entsteht eine nach Häufigkeit geordnete Liste, wahlweise auf- oder absteigend.
- Als Sonderleistungen gibt es statistische Auflistungen der vorkommenden Kategorienummern bzw. Teilfeldkennungen.

In die ***CockPit***-Vorgabendatei CP.OPT ist folgende Zeile unter "Eigene Routinen" eingebaut:

```
R Quantitative Auswertung, QUANT.BAT, Häufigkeitslisten von Kategorieinhalten
```

`QUANT.BAT` kann man auch von Hand starten, dann teilt es mit, welche Environment-Variablen man vorher setzen soll.

Ergebnis: eine Datei QUANT bzw. QUANTS mit den ausgewerteten Daten, alphabetisch bzw. nach Größe geordnet.

Es folgt eine knappe Übersicht des Ablaufs. (Vollständige Beschreibung: siehe Datei QUANT.TXT).
Zuerst sind verschiedene Fragen zu beantworten:
(Die Fragen erscheinen immer in grün, die Antworten in gelb)

`0. Wollen Sie`

```
1 : Alles mit der Hand eingeben?
2 : Frühere Einstellungen übernehmen?
3 : Sonderfunktion: statistische Belegung der Kategorien
4 : Sonderfunktion: statistische Belegung der Teilfelder
x : Vorgang beenden
```

Wenn man '1' wählt, geht es mit 1. weiter, bei '2' kann man gespeicherte Einstellungen früherer QUANT-Produktionen erneut ausführen lassen. Routineproduktionen lassen sich damit weitestgehend automatisieren.

`1. Welche Kategorien sollen ausgewertet werden?`

`(Geben Sie eine oder mehrere Kategorien an, getrennt durch Kommas:)`

Man gibt eine oder mehrere Kategorienummern, getrennt durch Komma, also z.B. (bei A.CFG)
#40,#402,#403
wenn man die Verfasser (bis zum dritten) auswerten will. Es kann hier leider nicht eine Gruppe von Kategorien mit einer einzigen Angabe wie etwa "#40." oder "#40?" zusammengefaßt werden. (Das Zeichen '#' kann entfallen, also auch 40,402,403 ist möglich.) Die Eingabe wird in %C1% gespeichert.
Will man Teilfelder auswerten, fügt man z.B. $c für Teilfeld c an die Kategorienummer an: #260$c

`2. Minimalwert?` Daraus wird %MIN%

`Maximalwert?` Daraus wird %MAX%

Diese zwei Fragen können mit <Enter> übergangen werden, dann gelten die Werte 0 bzw. 1000000, d.h. keine Schwellen. Es muß MIN < MAX sein, sonst kommt nichts heraus. Die Fragen beziehen sich darauf, wie häufig ein Feldinhalt vorkommt, nicht auf den Inhalt selbst - wenn die Felder Zahlen enthalten, könnte man das verwechseln!

3. Sind Stammsatz-Ersetzungen notwendig (Methode V14)? j/n
(Wenn nicht, geht es bis zu 50% schneller!)
(Wenn Sie nicht sicher sind: Antwort j)

Antwort 'j' ist nur dann zu geben, wenn die auszuwertenden Kategorien Stammsatznummern enthalten können (z.B. Namen oder Schlagwörter), die mit V14-Methode ersetzt werden müssen. Das braucht etwas mehr Zeit, deshalb ist 'n' zu empfehlen, wenn es nicht der Fall ist. Die Antwort wird in %C6% hinterlegt.

Hintergrund: Bei Antwort 'n' wird S.APT, bei 'j' wird Q4.APT umkopiert auf QUANT.cPT (wobei `c = %-K1%` ist). Der Unterschied ist nur der Befehl `i4=1`, der nur in Q4.APT vorkommt.

4. Welches Zeichen ist das Trennzeichen?

```
Möglichkeiten: eines der Zeichen ¶ ; - / . , ( * ' : $ =
       oder  b = Leerzeichen
       oder  x = keine Trennung
     (Wenn Eingabe nicht klappt, t davorsetzen, z.B. t; )
```

Eines der dreizehn Zeichen ¶ ; b - / . , (* ' : $ = kann man eingeben. Wenn ein anderes gewünscht wird, muß man in `C-FI.APR` eingreifen. Die Stellen sind kommentiert.

'$' ersetzt das Teilfeldzeichen '▼', weil man dieses nicht eingeben kann.

Wenn die Eingabe nicht funktioniert: ein 't' davorsetzen.

Am gewählten Zeichen werden die Kategorien zerlegt. Nur **ein** Zeichen ist möglich, das für alle Kategorien benutzt wird. Daraus wird die Variable %C2% gemacht.

5. Auswertungsmodus:

```
Wählen Sie: 1 = nur den Anfangsteil auswerten (Ende beim Trennzeichen)
            2 = jeden Teil auswerten (auch Teile zwischen den Trennzeichen)
            3 = nur Teil hinter dem Trennzeichen
```

Dies läßt sich nur mit 1, 2, 3 beantworten, was in %C3% gespeichert wird.

6. Sortiermodus:

```
1 = Normal (Umlaute auflösen, Nichtsortierwörter beseitigen)
2 = Pica-Namensmodus (Vorname@Nachname)
8 = Numerisch sortieren
9 = Keine Umcodierung - unverändert sortieren
```

Man gibt eine der möglichen Ziffern; sie wird in %C5% gespeichert. Bei 1 oder 2 wird QUANT.APT als Sortiertabelle benutzt. Bei 8 werden auch negative und Dezimalzahlen korrekt sortiert.

Sonderservice: Wenn man 8 wählt, wird am Ende zusätzlich die Gesamtsumme und der Durchschnittswert ausgegeben. (das ist in COUNT2.APR eingebaut).

Jetzt kommt eine Kontroll-Anzeige, wo man nochmal sieht, was jetzt als Vorgabe benutzt wird. Wenn irgendwas nicht stimmt, antwortet man mit 'n', und es geht von neuem bei 1. los.

Wenn man 'j' gesagt hat:

7. Sollen diese Einstellungen aufbewahrt werden? j/n

Wenn man 'j' antwortet, muß man einen Namen xyz eingeben (bis zu 6 Zeichen).

Es wird dann eine Datei C-xyz.BAT erstellt. Darin stehen die Einstellungen als SET-Befehle. Später kann man sie erneut benutzen (siehe oben 0.)

8. Wie soll die Auswertung erfolgen?

```
1 : mit Volltextsuche
2 : als Ergebnismenge (Export mit F4)
```

Bei '1' wird SRCH aufgerufen, dann muß man einen Volltext-Suchbegeiff eingeben, bei '2' wird PRESTO aufgerufen, dann muß man eine Ergebnismenge bilden und sie per F4 exportieren. Wenn man hintereinander mehrere Ergebnismengen exportiert, braucht man keine Duplikate zu befürchten.

Hintergrund: Beim Aufruf wird `-Uc1%C1% -Uc2%C2% -Uc3%C3% -Uc5%C5%` mitgegeben, d.h. es entstehen die Anwendervariablen `#uc1, #uc2 , #uc3, #uc5`. Diese werden in C-FI.APR verwertet.

9. Es entsteht die Zwischendatei QUANT, mit ASORT wird sie sortiert.

Das Ergebnis ist dann QUANT.ALG, und QUANT wird gelöscht.

10. SRCH verarbeitet QUANT.ALG zu der Ergebnisliste QUANT.

Verwendet wird dazu immer COUNT2.APR. Darin werden Duplikate eliminiert (#u1 enthält zusätzlich die Satznummer, sonst ginge das nicht!) Dabei werden mit -Umi%min% -Umx%max% die Schwellenwerte dem Programm übergeben.

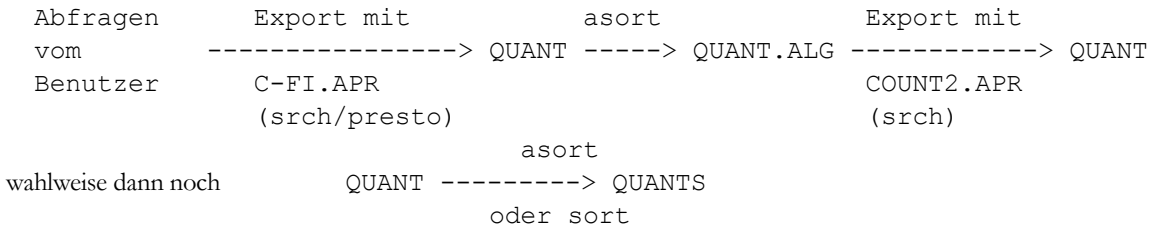
11. Nach Anzahl sortieren?

```
a = aufsteigend
b = absteigend
u = unsortierte Liste nochmals anzeigen
x = Ende
```

Antwort 'a' oder 'b' bewirkt, daß die Datei QUANT nach Häufigkeit sortiert wird. Die Anordnung ist so, daß dann eine auf-/absteigend nach Zahlen sortierte Liste QUANTS entsteht, während QUANT alphabetisch geordnet ist. Die nach Zahlen sortierte Liste wird dann auch noch angezeigt, und die Frage 11. wird wiederholt, damit man evtl. nochmals andersrum sortieren kann. (Es kommt zum absteigenden Sortieren MSDOS-SORT zum Einsatz)

12. QUANT und QUANTS sind ASCII-Textdateien, die man beliebig weiterverwerten kann.

Schematischer Ablauf von QUANT.BAT :



6.5 EXPEX : Exportieren lernen mit MARC

Die **Exportsprache** ist eine tragende Säule des ***allegro***-Systems und begründet seine Flexibilität und Offenheit für alle Formate. Bildschirmanzeigen, Karten, Listen, Ausgabedateien, Indexregister werden alle damit programmiert. Mächtige und vielseitige Werkzeuge sind jedoch nicht leicht zu beherrschen, und so kann man leider nicht von einem Tag zum nächsten ***Allegrologe*** werden. Große Verdienste um die Verbreitung von Kenntnissen der Exportsprache hat sich Heinrich Allers mit seinem Lehrbuch erworben. Es bietet zu jedem einzelnen Befehl eine ausführliche Lektion mit jeweils genau passenden Beispielen. Dieses Lernsystem, organisiert mittels ***CockPit***, kann durchaus auch neben dem Handbuch als Referenz der Exportsprache dienen. Nicht jeder Befehl ist jedoch gleich wichtig, und so fühlt sich mancher am Anfang erschlagen von der Fülle des Materials. Wo soll man beginnen, was ist nicht so wichtig, was wird unbedingt gebraucht?
Das Lernpaket EXPEX hat zum Ziel, interaktiv von einfachsten Ansätzen bis zu einer recht anspruchsvollen Parameterdatei in die Programmierung einer Bildschirmanzeige einzuführen. Man lernt dabei eine Reihe von wichtigen Befehlen, man lernt aber auch gleich das Testen mit der Expressmethode der "Merseburger Testschleife" (Handbuch S.204), denn wer produktiv sein will, muß schnell arbeiten können.
EXPEX wurde in Zusammenarbeit mit David Helliwell von der Bodleian Library in Oxford entwickelt. Deshalb sind die Kommentare in den Dateien in Englisch, und es wird mit einem vereinfachten MARC-Format gearbeitet. Das mag als Nachteil gesehen werden, andererseits wird neuerdings sehr viel von internationaler Vereinheitlichung geredet und von einer notwendigen Orientierung an internationalen Standards, sprich MARC und den Anglo-American Cataloguing Rules (AACR). Damit kennt sich noch nicht jeder aus. So kann EXPEX nebenbei auch ein wenig helfen, das Einarbeiten in diese Standards zu erleichtern.
Wie arbeitet man mit EXPEX? Es erklärt sich weitgehend selbst, daher können wir uns hier kurz fassen. Nach der V15-Installation hat man ein Unterverzeichnis EXPEX, auf dem alle Dateien liegen.
Start: entweder mit der Hand, indem man von C:\ALLEGRO aus den Befehl **`expex\expex`** gibt, oder man baut sich in die Vorgabendatei CP.OPT diese Zeile ein (bei den "Eigenen Routinen"):

```
R Export-Experimente, EXPEX\EXPEX.BAT, Interaktiver Einstieg in die Exportsprache
```

dann braucht man nur im ***CockPit*** diesen Menüpunkt anzuwählen. Alles weitere erklären Ihnen dann die Menütexte und die Kommentare in den Dateien. Wenn man EXPEX nicht benötigt oder alles verstanden hat, braucht man nur das Unterverzeichnis EXPEX zu löschen, und das Ganze ist spurlos beseitigt, kein Ballast bleibt zurueck.
Wenn man anschließend auf den Geschmack gekommen ist, kann man sich sukzessive mit dem Lehrbuch weitere Feinheiten aneignen. Wenn man noch mehr über MARC wissen will, greift man zur Konfiguration $U.CFG und den Dateien MARC.UPI und D-1.UPR, um damit eine "richtige" MARC-Datenbank aufzubauen, oder man holt sich die Beispieldatenbank, die unter FORMINT.EXE auf dem Verzeichnis FORMATE des FTP-Servers liegt. Sie enthält die offiziellen Testdaten der LC mit allen verschiedenen Satztypen, die im Zuge der "format integration" geschaffen wurden.

6.6 Listen mit geordneten Registern

Komplizierter wird es, wenn nicht nur eine sortierte Liste, sondern dazu auch noch ein alphabetisches Register zu erstellen ist. Das Prinzip: man läßt bei der Druckaufbereitung gleichzeitig einen weiteren Export ablaufen, und zwar vom Typ ALPHA (Prototyp A-0.APR, → 10.0). Wenn z.B. mit dem oben beschriebenen Verfahren ein alphabetischer Verfasserkatalog produziert wird, kann bei der Druckformatierung zugleich ein Export mitlaufen, bei dem eine sortierfähige Schlagwortdatei herauskommt, und wobei die Seitenzahl des Katalogs (Export-Sonderkategorie #pa, → 10.2.6.2) mit ausgegeben wird. Anschließend kann diese Schlagwortdatei sortiert und in einem weiteren Exportlauf in Form eines Registers ausgegeben werden; dabei können die Seitenzahlen des Hauptteils dann erscheinen.
Alternativ könnte man während Schritt 3. statt der Seitenzahlen eine laufende Nummer produzieren und in die Schlagwortliste übernehmen (Sonderkategorie #nr) und diese dann im Schlagwortregister mit ausdrucken.
Es gibt für die Erstellung von Listen mit Registern noch keine Einstellmöglichkeiten per ***CockPit***. Man muß also vorerst "nach alter Art" Batchdateien machen, die den gesamten Ablauf enthalten.

Hier eine Batchdatei (ALPHA.BAT, wird mitgeliefert), die den Ablauf illustriert:

```
echo off
rem  1.Schritt: Export mit SORT.APR in sortierfähiger Form
rem             (SORT.APR ändern, wenn andere Sortierung gewünscht)
rem             srch durch presto ersetzen, wenn Anwendung auf Ergebnismengen
srch -f6 -e sort/hhh1.alg -m0 -v0 -ddatenpfad\*.?ld
rem  2.Schritt: Sortieren
asort hhh1.alg hhh2.alg
del hhh1.alg
rem  3. Schritt: a) Druckformatierung, und parallel dazu
rem              b) sortierfähiges Registerdatei
rem     (PRINT.APR und ALPHA.APR ändern, wenn lfd.Nr. statt Seitenzahl gewünscht)
srch -f6 -d hhh2.alg -e PRINT/liste -e ALPHA/hhh -m0 -b%-D%\%-B% -v0

rem  der Rest bleibt gleich (bei Seitenahl bzw. Nummern):
rem     und kann auch für andere Konfigurationen übernommen werden!
del hhh2.alg
rem  4.Schritt: Sortieren der Registerdatei
asort hhh hhh.alg
del hhh
rem  5.Schritt: Zusammenfassen der identischen Einträge (Export mit ALPHA2.APR)
srch -f6 -d hhh.alg -e alpha2/rng.alg -m0 -v0
del hhh.alg
rem  6.Schritt: Druckformatierung als Register mit Seitenumbruch (ALPHA3.APR)
srch -f6 -d rng.alg -e alpha3/REG.TXT -m0 -v0
del rng.alg
echo LISTE = alphabetisches Verzeichnis
echo REG.TXT   = Register
```

Die Dateien ALPHA2.APR und ALPHA3.APR sind von der Art des Registers unabhängig. Will man also andere Register, z.B. ein Schlagwortregister, produzieren, wäre nur A-0.APR zu modifizieren. ALPHA2 und ALPHA3 sind sogar vom Kategoriesystem unabhängig, d.h. nach Umbenennung sofort für andere Konfigurationen nutzbar.

Eine andere Art Register, ein **Häufigkeitsregister**, ist mit der dialoggeführten Methode QUANT erstellbar: z.B. ein Register, das alle vorkommenden Namen oder Schlagwörter enthält und wie oft sie vorkommen. Dieser Registertyp wird in Abschnitt 6.4 behandelt.

7 Indexieren

Online: `h org`

7.0 Datenbank-Generierung : Aufgabe und Konzept

Der direkte und schnelle Zugriff auf Datensätze über eine alphabetisch geordnete Liste von Suchbegriffen ist nur möglich, wenn es eine **Indexdatei** gibt, denn darin stecken diese Listen.
Brauchen Sie dieses Kapitel? Nur wenn Sie sich mit dem Neuaufbau oder der Reorganisation von Datenbanken beschäftigen, also mit Systemverwaltung und Parametrierung. Sonst genügt es, wenn Sie gelegentlich eine Neuindexierung (*a99*: Füllhorn / Reorganisieren oder ***CockPit*** → 0.11.2, Menüpunkt "Routinen / organisieren") durchführen, das sind nur wenige Klicks oder Tasten.
Wenn man eine völlig neue Datenbank anlegt (Füllhorn / Neue Datenbank anlegen), entsteht beim Abspeichern des ersten Datensatzes auch sofort die Indexdatei (→ 1.1). In diesem Fall wird das Programm INDEX nicht gebraucht! (Sondern erst später, wenn der Index einmal erneuert werden soll.)
Wenn aber schon Daten vorhanden sind, nur noch kein Index, muß ein Programm aus den Daten zunächst eine Datenbank machen. Ebenso, wenn die Indexdatei abhanden gekommen oder unbrauchbar geworden ist, oder wenn ihre Struktur (die Form der Zugriffsschlüssel) verändert werden soll. Das dazu nötige Programm ist die Nummer 7 und heißt INDEX. Es gibt den Datensätzen (wenn nötig) neue Nummern, schreibt sie evtl. in eine neue *.d*LD-Datei oder hängt sie an eine vorhandene an, und vor allem extrahiert es aus den Sätzen die Suchbegriffe und sortiert sie in die Indexdatei. Diesen gesamten Vorgang bezeichnet man auch als das **Generieren** der Datenbank. Das **Indexieren** ist also nur ein Teil davon, wenngleich der aufwendigste.

> **Grundsätzlich:** INDEX im **Mehrplatz**-Betrieb nur einsetzen, wenn gerade niemand anders an der Datenbank arbeitet. Das Programm UPD (→ Kap.9) kann dagegen auch Daten in eine "lebendige" Datenbank einmischen.

Es gibt 7 mögliche Ausgangssituationen, denen 7 Funktionen des Programms INDEX entsprechen. Diese Funktionen werden beim Aufruf des Programms INDEX mit der Option **`-f`** ausgelöst: (→ ***CockPit*** 0.11.2)
(Was hier über .ALG und .ALD gesagt wird, gilt natürlich genauso für .PLG etc.)

1 Es liegen ***allegro***-Daten zum Indexieren vor, und zwar
- .ALG-Dateien, die man aus eigenen Vorarbeiten oder anderen Quellen per Import erhalten hat,

und / oder
- .ALD-Dateien, die bereits zu einer Datenbank gehören oder gehört haben.

Dann genügt es, das Programm INDEX auf diese Daten loszulassen, und es generiert mit Hilfe einer Index-Parameterdatei eine völlig neue Datenbank bzw. fügt die Daten zu einer anderen hinzu.
Wichtig: alle Sätze erhalten neue **interne** Satznummern, es wird aber keine Kategorie verändert.
Option: **`-f7`** (**Standardfunktion** des Programms 7; ausführliche Beschreibung → 7.1)

2 Eine Datenbank soll einen neuen Index erhalten, weil der bisherige nicht mehr funktioniert oder man die Parameter verändert hat. Die .ALD-Dateien (samt Satznummern) sollen unverändert bleiben.
Option: **`-fi`** (Funktion "**Index erneuern**")

3 Der Index ist in Ordnung und kann so bleiben, aber die .ALD-Dateien enthalten durch viele Löschungen viel leeren Raum, den man wieder freimachen will. Dazu müssen nur die .ALD-Dateien "entlüftet" werden, wie ein salopper Ausdruck dafür lautet.
Option: **`-fr`** (Funktion "**Rückgewinnung" oder "Bereinigung"**, auch "Entlüftung")

4 Die Umstände 2 und 3 sind beide gegeben. In diesem Fall kann INDEX auch beide Aufgaben in einem Durchlauf erledigen. Wichtig: die internen Satznummern sollen erhalten bleiben.
Option: **`-fn`** (Funktion "**Nummerntreue Erneuerung** der Datenbank")

5 Nur die Satztabelle (.TBL) soll erneuert werden (wenn sie z.B. versehentlich gelöscht wurde).
Option: **`-ft`** (Funktion "**Satztabelle erneuern**")

6/7 Nur das Kurztitelregister (.STL) oder die Restriktionsdatei (.RES) soll erneuert oder erstmals angelegt werden (dann sind zuerst die Vorkehrungen in der .cPI zu treffen).
Option: **`-fs`** bzw. **`-fx`** ("**Kurzanzeige erneuern**" bzw. "**Restriktionsdatei erneuern**")

Alle diese Funktionen können, wie schon gesagt, mit einem Knopfdruck vom ***CockPit*** aus erledigt werden (Untermenü "organisieren", **`µr o`** : "Routinen / organisieren"). Dieses Kapitel erklärt, was dann abläuft.

Zwei weitere Situationen sind hiermit noch nicht erfaßt:

a) Man hat Daten aus anderen Quellen, die anders strukturiert sind als ***allegro***-Daten.
Dann muß zunächst mittels IMPORT das Material in ***allegro***-Grunddateien umgewandelt werden (Typ .ALG); dann liegt Situation 1 vor, und INDEX kann daraus eine Datenbank machen.
"Import" ist ein Kapitel für sich, oder sogar zwei: Kap. 5 und 11 zeigen, wie man aus anders strukturierten Fremddaten die benötigten .ALG-Dateien erzeugen kann.

b) Datensätze aus anderen, aber gleich strukturierten ***allegro***-Datenbanken sollen **eingemischt** werden, wobei vorhandene Sätze ganz oder teilweise durch neue zu **ersetzen** sind. Dafür ist das Programm UPD zuständig (→ Kap.9). INDEX kann Sätze nur hinzufügen, nicht austauschen.

Daß nach einem "Absturz" (z.B. bei Stromausfall) der Index nicht mehr angezeigt wird, also unbenutzbar geworden ist, das kommt selten vor (meistens überlebt er das), und wenn man gewissenhaft Datensicherung betrieben hat, braucht man in diesem Fall keine Index-Erneuerung (Funktion `-fi`) zu machen: bei einer großen Datenbank geht es schneller, dann ein "Playback" zu fahren (→ Kap.0.7, Kap.9).
Notwendig zur Generierung einer Datenbank ist eine **Index-Parameterdatei** (Typ .cPI, → Kap.0.3 und Kap.10). Diese Datei enthält die genaue Beschreibung, wie die Indexeinträge aus den Datensätzen zu erzeugen sind. Entweder man benutzt eine der mitgelieferten, ändert sie evtl. nach eigenen Wünschen ab, oder man legt selbst eine neue an (→ Kap.10.1).

7.1 Datenbank-Generierung : Ablauf

Diese Standardfunktion (Situation 1) des Programms `index` soll hier besonders ausführlich beschrieben werden.
Angenommen,

- man hat seine Ausgangsdaten auf einem Verzeichnis `d:\books` im Standardformat, d.h. es sind Dateien des Typs .ALG
- die Datenbasis soll `buch` heißen, d.h. man hat eine Parameterdatei `buch.api` für die Indexstruktur
- Die Datenbank soll dann
  a) auf dem Unterverzeichnis `buecher` des ***allegro***-Verzeichnisses auf Laufwerk C: stehen, oder
  b) dort steht bereits eine und soll durch die neuen Daten angereichert werden.
  (Vorsicht: letzteres im Mehrplatzbetrieb nur durchführen, wenn niemand dran arbeitet!)

Dann sieht das **Schema der Befehle** folgendermaßen aus, jeweils für die Situationen ohne bzw. mit V14-Schlüsseln:
(In rot die sinngemäß zu ändernden Angaben). Angenommen ist, daß die Befehle auf dem Programmverzeichnis ausgeführt werden; standardmäßig ist das `c:\allegro`).
Hinweis: Ab V34 geht auch unter Windows das = statt / in der Option `-e`.

a1) Neuaufbau ohne V14-Schlüssel

```
index -f70 -dd:\books\*.alg -ebuch/buecher -n1 -ka  (UNIX: ... -ebuch=buecher ...)
qrix -fq -dbuecher -ebuch/buecher -ka
```

a2) Neuaufbau mit V14-Schlüsseln (z.B. beim Standardsystem mit cat.api)

```
index -f70 -@1 -dd:\books\*.alg -ebuch/buecher -n1 -ka  (UNIX: ...-ebuch=buecher ...)
qrix -fq0 -dbuecher -ebuch/buecher -ka
index -fi1 -@2 -dbuecher\*.ald -ebuch/buecher -ka
qrix -fq1 -dbuecher -ebuch/buecher -ka
index -fa -dbuecher\*.ald -ebuch/buecher -ka
```

b1) Einspeisen in eine schon bestehende Datenbank, ohne V14

```
index -f71 -dd:\books\*.alg -ebuch/buecher -n1 -ka  (UNIX: ... -ebuch=buecher ...)
qrix -fq1 -dbuecher -ebuch/buecher -ka
```

b2) Einspeisen in eine schon bestehende Datenbank, mit V14

```
index -f71 -@1 -dd:\books\*.alg -ebuch/buecher -n1 -ka  (UNIX: ... -ebuch=buecher )
qrix -fq1 -dbuecher -ebuch/buecher -ka
index -fi1 -@2 -dbuecher\*.ald -ebuch/buecher -n1 -ka
qrix -fq1 -dbuecher -ebuch/buecher -ka
index -fa -dbuecher\*.ald -ebuch/buecher -ka
```

b3) Erneuerung einer Zusatz-Indexdatei (MultiX) : zum Beispiel die Datei `cat.agx`

```
index -fi0 -@2 -Zg -dbuecher\*.ald -ebuch/buecher -ka
qrix -fq0 -Zg -dbuecher -ebuch/buecher -ka
```

Hinweise: das Programm nennt bei `-f7` und bei `-fn` die Ausgangsdateien um: die Typen .ALG bzw. .ALD werden zu .A8G bzw. .A8D (früher .A1G bzw. .A1D). Das ist eine Vorsichtsmaßnahme, insbesondere, damit nicht eine .ALD-Datei an sich selbst hinten ankopiert wird. Wenn alles gut gelaufen ist, hat man hinterher neue .ALD-Dateien und die alten sind entbehrlich. Folgerung: normalerweise kann und **sollte** man die Ausgangsdateien, also Typ `.c8d`, löschen, denn alle Daten sind in neue .ALD-Dateien überführt worden. Will man aber aus irgendeinem Grund die Aktion wiederholen, oder passierte während des Durchlaufs ein Zusammenbruch: Typ .ALD löschen, `.A8D` in .ALD umbenennen, und das Ganze nochmal.
Das Löschen alter `.c8d` -Dateien geschieht beim nächsten Indexieren automatisch.

Weitere Hinweise

Dateigröße begrenzen
Die neu entstehenden Dateien des Typs .ALD werden normalerweise bis zur Größe von 16.000.000 Bytes aufgefüllt. Dann beginnt das Programm mit einer neuen Datei (auf `buch_1.ald` folgt `buch_2.ald` usw.). Will man die Größe der einzelnen Datei z.B. auf handliche 3 MB beschränken, gibt man beim Start zusätzlich die Option
`index ... -z3000000`
Sollen die Dateinummern nicht mit 1 anfangen, sondern z.B. mit 73, ergänzt man **`-n73`**.
Sollen größere Dateien als 16MB zulässig sein (sog. *Aufbohrung*): Wert `ii` in der Indexparameterdatei setzen, z.B. `ii=2` für das Doppelte usw. Dies geht nicht nachträglich, sondern anschließend ist ein Neuaufbau erforderlich (Menü `h org`).

Zwischenmeldungen
Das Indexprogramm gibt die Zwischenzeiten für die Bearbeitung von jeweils 100 Datensätzen bzw. bis zu 100.000 Schlüsseln an (abhängig von deren Länge). Am Ende werden Sie informiert über den Umfang Ihrer Indexdatei. Diese Angaben werden zusätzlich in eine Datei namens PROTOK geschrieben, die man sich ansehen oder ausdrucken kann. In PROTOK findet man auch am Ende die Befehlszeile, mit der das Programm QRIX anschließend aufgerufen wird, um die Zwischendateien zusammenzumischen.

Endergebnis
Am Ende erhält man die Indexdatei `buch.adx`, und sie liegt im Unterverzeichnis `allegro\buecher`. Noch einmal zu den Bestandteilen des Namens:

`buch` :	der von Ihnen vergebene Name der Datenbasis, stimmt überein mit dem Namen Ihrer Index-Parameterdatei (`buch.api`)
`.adx` :	Typkennung für "Indexdatei zum Kategoriesystem A" (wenn Sie z.B. unter `p.cfg` arbeiten, brauchen Sie `BUCH.PPI` und es entsteht `buch.pdx`).

Satznummern
Das Programm `INDEX` gibt bei Funktion **`-f7`** jedem Satz automatisch eine neue, interne **Satznummer**. Diese steht nicht in irgendeiner Kategorie! Bei Neuaufbau einer Datenbank beginnt die Numerierung mit 1. Die internen Satznummern gelten, solange man nicht mit Funktion **`-f7`** neu indexiert. Mit **`-fi, -fn und -fr`** bleiben sie erhalten.
Wenn Sie eigene **Identifikationsnummern** haben, können Sie diese z.B. in Kategorie #00 eingeben und mit in den Index einsortieren lassen, um darauf zugreifen zu können und um einen Primärschlüssel für `UPDATE` zu haben (➢ Kap.9.). Solche Nummern sind dann dauerhaft und von der Datenbank unabhängig. Die Nummernvergabe für neu hinzukommende Sätze kann automatisiert werden: dazu muß man die Befehle `cn` und `ce` in der `.CFG` in geeigneter Weise einrichten (➢ Anh. A.1.3).

Wollen Sie die **gesamte Datenbank wieder löschen**, dann geben Sie die folgenden Befehle, und zwar auf dem Unterverzeichnis `allegro\buecher:`

```
del buch.adx
del buch.tbl
del buch.stl
del buch.res
del buch.log
del buch*.ald   Diese Dateien aber nicht löschen, wenn Sie eine Datenbank daraus neu generieren wollen!
```

Datenbank neu generieren
Um eine Datenbank neu zu generieren, z.B. wenn man Änderungen an der Indexdefinition vorgenommen hat, braucht `index` nur die `.ald`-Dateien und die Indexparameter *dbn*`.api`, daher kann man *dbn*`.tbl`, *dbn*`.stl`, *dbn*`.res` und *dbn*`.a?x` unbedenklich löschen, denn `index` wird sie neu erstellen. Dadurch gehen keine Anwenderdaten verloren! Dann nach a1) bzw. a2) vorgehen, je nachdem, ob die Datenbank mit V14-Ersetzungen arbeitet oder nicht:

V14 : Indexierung in zwei Schritten
Ab Version 14 ist es möglich, in Datensätzen Stammsatznummern zu verwenden, z.B. `_123` statt eines Personen- oder Körperschaftsnamens, aber im Index trotzdem den Datensatz unter dem Klartext des Namens zu finden (-➢ 10.2.6.8). Das Programm `index` muß dann einen weiteren Durchlauf machen, weil es im ersten Durchlauf die Nummern noch nicht durch die Klartexte ersetzen kann. Das passiert nicht automatisch, sondern man ruft `index` für diesen Durchlauf nochmals auf und zwar mit der dafür vorgesehenen Option **`-fa1`**.
Alternative für Systemverwalter: man indexiert in zwei bzw. drei Schritten, und zwar mit diesen Befehlen: (s.o. 7.1 a2 und b2)

```
index -@1 -fi0...          nur Primärschlüssel indexieren
qrix -fq0...               die Zwischendateien mischen
index -@2 -fi1...          alle anderen Schlüssel indexieren.
qrix -fq1...               die Zwischendateien mischen
index -fa1...              Ersetzungsschlüssel (Klartexte) einsetzen.
```

Statt **`-fi0`** kann auch **`-f70`** oder **`-fn0`** vorkommen, aber wichtig ist: im zweiten Durchlauf muß es **`-fi1`** heißen. Diese Methode ist schneller, wenn es viele Stammsätze gibt.
CockPit macht das automatisch so (siehe dann die Stapeldatei `ccc.bat`, die es erstellt), wenn es eine V14-Datenbank ist (d.h. wenn **`i5`** in den Indexparametern gesetzt ist).
Ebenso ***a99***, wenn man auf dem Menü "Reorganisieren" die Funktion "Index erneuern" wählt. Es entsteht dann eine Datei **`org.bat`**, in der die genauen Befehle stehen. Daraus kann man sich dann auch eine eigene Stapeldatei machen, die man zu gegebener Zeit nutzt.

Mißglückter Versuch
Löschen muß man eine Datenbank z.B. nach einem mißglückten Datenbankaufbau, damit die dann vorhandenen, aber unbrauchbaren Dateien den nächsten Versuch nicht behindern oder scheitern lassen und der Platz wieder freigemacht wird. Dann sollte man auch die *.c*`ld`-Dateien löschen, weil man die Ausgangsdaten (unter `*.c8d`, früher .`c1d`) in der Regel noch hat und zur Generierung wieder auf diese zurückgreifen kann, nachdem man sie wieder umbenannt hat.

7.2 Datenbank ganz oder teilweise erneuern

Windows: **h org**

Empfehlung: Machen Sie dies alles per ***CockPit*** (→ 0.11.2, S.35f) oder mit *a99*, dabei braucht man sich um die Einzelheiten nicht zu kümmern. Andernfalls gehen Sie wie folgt vor:

- Man trifft, falls nötig, die Vorbereitungen in der Index-Parameterdatei: Definition neuer Schlüssel, Änderung oder Beseitigung existierender Definitionen.
- Man startet den Vorgang mit dem Befehl

index -f70 -n0	gesamte Datenbank erneuern	oder , viel bequemer per ***CockPit***	**µr o 7**
index -fi0	für den Index		**µr o i**
index -ft	für die .TBL-Datei		**µr o t**
index -fs	für die .STL-Datei		**µr o k**
index -fx	für die .RES-Datei		**µr o x**

  und beantwortet etwaige Fragen des Programms sinngemäß. Wenn die Datenbank mehrere .ALD-Dateien hat, kreuzt man sie auf der erscheinenden Auswahlliste **alle** an ('+' davorsetzen).
  Bei **-f7**: die Frage nach "Dateinummer für neue Daten?" mit **0** beantworten!

Anmerkung: Die Dateitypen .TBL, .STL und .RES werden bei **-fi** nebenbei mit erstellt, wenn sie nicht existieren. Wenn sie existiert, bleibt .TBL bei -fi unverändert.
Bei einer Erneuerung des Index verschwinden die Änderungen und Ergänzungen, die man manuell gemacht hatte (→ Kap.1.4, Inst/Del). Wenn nur *einzelne* Register zu erneuern sind, nicht der gesamte Index: → 7.6.

7.3 "Entlüften" von Datenbankdateien

CockPit **µr o R**

Windows: **h org**

Zweck: Rückgewinnung von Speicherplatz, Beschleunigung beim Speichern.
Auf dem Windows-Menü (**h org**) heißt es "**Bereinigen**".)
Bei der normalen Arbeit mit einer Datenbank kann es geschehen, daß sich immer mehr "Leersätze" ansammeln. Das sind Speicherplätze innerhalb von *.d*LD-Dateien, die nicht mehr belegt sind, weil die Datensätze, die dort standen, wegen Verlängerung "umgebettet" werden mußten. Diese Vorgänge spielen sich völlig automatisch ab, daher kann der Anwender das Phänomen nicht bemerken. Die "Leersätze" werden automatisch wieder benutzt: wenn ein neuer Datensatz gespeichert werden soll, der in einen dieser Plätze paßt, nimmt das Programm den am besten passenden Leersatz. Dadurch ist es kaum zu erwarten, selbst nach Jahren nicht, daß die Größe des ungenutzten Platzes ins Uferlose wächst. Handlungsbedarf entsteht nur dann, wenn man ständig Datensätze verlängert, indem man neue Angaben hinzufügt. Dann werden sich allmählich immer mehr kurze Leersätze ansammeln. Ein unangenehmes Symptom kann auftreten: das Abspeichern dauert manchmal ungewöhnlich lange. Nach dem Entlüften geht's wieder schneller. Der Grund ist: die Suche nach einem passenden gelöschten Satz zum Abspeichern eines neuen oder verlängerten Satzes dauert länger, wenn sehr viele Leersätze vorhanden sind.

Achtung: Grunddateien (Typ .ALG) kann man nicht entlüften. Insbesondere könnte man auf diesem Wege keine neuen Daten in eine Datenbank einbringen. Das geht nur mit **index -f71**. Um den Aufbau einer Datenbank aus mehreren Ausgangsdateien zu beschleunigen, könnte man nur so verfahren: zuerst die Index-Parameterdatei "abspecken": nur einen einzigen ak-Befehl aktivieren. Dann mit **index -f71** die einzelnen Dateien indexieren.
Dann alle ak-Befehle wieder aktivieren und per ***CockPit*** **µr o i** (Index erneuern) starten. Nötig wäre das alles nur, wenn aus den .ALG-Dateien mehrere *.d*LD-Dateien werden sollen, ansonsten kann man gleich alles auf einmal indexieren.

Wieviele Leersätze eine Datenbank hat, kann man so feststellen:
Auf dem Registerbildschirm (→ 1.4) gibt man **F6 // Enter** im Register 1 . Dann erscheinen Zahlen, vor denen jeweils "//" steht. Das sind die Zugriffsschlüssel der Leersätze, und zwar geben die Zahlen deren Länge an. Geben Sie jetzt noch **F10 2** <Enter> (d.h. Trunkierung auf 2 Stellen), dann sehen Sie `"// ..." und links daneben` die Anzahl der Leersätze. (Mit **F7** Trunkierung wieder aufheben.) Wenn Sie zugreifen auf einen dieser Sätze, werden Sie einen ganz normalen Datensatz sehen, es wird nur oben die Mitteilung erscheinen "Dieser Satz wurde getilgt". Der Inhalt des Satzes bleibt bei einer Löschung zunächst unverändert stehen; nur die Zugriffsschlüssel werden aus dem Index beseitigt und der Leerschlüssel wird eingeordnet. Erst bei einer erneuten Benutzung des Leersatzes wird der dort stehende Text tatsächlich überschrieben.

Einfacher und noch besser: Die Anzahl Leersätze und deren Umfang in Byte erfährt man mit ***CockPit*** **µr o u** (Funktion "ungenutzten Platz prüfen") oder in a99 über das Org-Menü (**h org**).
Wenn Sie also sehen, daß es sehr viele Leersätze gibt, können Sie die Datenbank "entlüften", um den ungenutzten Plattenraum wieder für das gesamte System (nicht nur für ***allegro***) verfügbar zu machen.
Ausgeführt wird diese Arbeit vom Programm INDEX. Es kopiert die *.d*LD-Dateien, wobei es die Leersätze wegläßt, beseitigt die //-Schlüssel aus dem Index und aktualisiert die .TBL-Datei. Die alten, löcherigen *.d*LD-Dateien haben anschließend den Typ .c8D und können gelöscht werden, nachdem alles korrekt gelaufen ist. Es empfiehlt sich sehr, sie dann zu löschen! Vor einer weiteren Neu-Indexierung besteht ***CockPit*** sogar darauf, a99 macht es automatisch.
Überlegen Sie vorher, wie groß die Anzahl Füllzeichen je Satz sein soll, denn diese wird beim Schreiben der neuen Datei an jeden Satz angehängt. Ändern Sie evtl. diesen Wert (dort steht z.B. **f 3**) in Ihrer .CFG-Datei.
Mit dem Befehl
index -fr -n0
starten Sie die Entlüftung. Wenn mehrere `.c`LD-Dateien existieren, werden diese als Auswahlliste angezeigt, und man wählt mit '+' diejenige aus, die es nötig hat. Wenn man mehrere ankreuzt, werden diese alle entlüftet. Die Ausgangsdateien werden umbenannt auf den Typ `.c`8D.
Vorsicht: Entlüftung im **Mehrplatz**betrieb nur, wenn niemand an der Datenbank arbeitet! Sonst drohen Schäden. Prüfen Sie anschließend, ob die Datenbank in Ordnung ist. Sie finden unter **/[0]** eine neue Eintragung im Index. Darunter sind die nicht mehr besetzten Satznummern der gelöschten Sätze versammelt. Beim Zugriff darauf kommt die Meldung `"nichts gefunden, Satznummer nicht besetzt"`, was aber nichts ausmacht. Wenn alles glatt gegangen ist, müssen die .c8D-Dateien gelöscht werden (mit dem MS-DOS-Befehl **del *.?8?**). Wenn nicht, löscht man die neuen Dateien (**del xxx.?ld, del *.?dx, del *.tbl**), benennt die alten wieder zurück (**ren *.?8? *.?L?**) und indexiert sie neu, was allerdings viel mehr Zeit kostet. Oder man holt eine Sicherungskopie hervor und fährt "Playback" (→ 0.7).
Die unbenutzten Nummern werden übrigens wieder verwendet, so daß die Einträge `/[0]` schließlich verschwinden.

7.4 Datenbank als Ganzes erneuern

Entlüftung und Index-Erneuerung können in einem Zuge erfolgen, wobei die Satznummern erhalten bleiben. Deshalb wird dieser Vorgang auch "nummerntreue Erneuerung" genannt. (Wenn es darauf nicht ankommt, könnte man genausogut nach 7.1 verfahren, wobei der Vorteil ist, daß es hinterher keine unbesetzten Satznummern gibt. Allerdings werden die, wie gesagt, im weiteren Verlauf wieder benutzt.)
Man startet die nummerntreue Erneuerung mit dem Befehl

index -fn ***CockPit*** **µr o n**

kreuzt dann alle zu bearbeitenden Daten auf der erscheinenden Auswahlliste an und beantwortet etwaige Fragen sinngemäß. Nachdem alles gelaufen ist, gibt es evtl. eine oder mehrere Dateien des Typs `.A8D`, die gelöscht werden können. Wenn was schief gegangen ist: die `.A8D`-Dateien evtl. wieder in `.ALD` umbenennen und die Sache wiederholen.
Diese Funktion ist auch sinnvoll, wenn man den Parameter **ii** für die maximale Dateigröße verändert hat (→ 10.2.1.3).
Also konkret: wenn eine `.ALD`-Datei nun eine maximale Größe von 48 MB haben soll, muß man zuerst in den Indexparametern einsetzen
ii=3 (denn 3*16 = 48)
und danach die Erneuerung starten mit

index -fn ...
und dabei weder `-n0` noch `-z...` setzen, damit wirklich neue `.ALD`-Dateien der gewünschten Größe entstehen.

7.5 QRIX - Werkzeug für Datenbank-Ingenieure

Registerabschnitte mit *a99*: **`X qrix`**

Wie arbeitet INDEX?

Eine Indexdatei ist ein recht kompliziertes Gebilde. Seit Version 10.2 gibt es mehrteilige Indexdateien mit bis zu 9 Registern, ab 12.1 sind es 11. Mit zunehmender Größe der Anwendungen wurde der Wunsch nach einer flexibleren Handhabung und schnellerem Aufbau der Indexdateien dringender. Ab Version 10.3 ist das Programm INDEX intern anders gestaltet, und zu seiner Unterstützung wurde ein neues Programm, QRIX, entwickelt. Beide Programme wurden mehrfach optimiert; gegenüber älteren Versionen braucht das Indexieren nur noch Bruchteile der Zeit. Heutige Rechner schaffen nicht selten mehr als 1000 Sätze pro Sekunde.

Hier soll zunächst die Arbeitsweise des Programms INDEX etwas näher erläutert werden, bevor Funktionen und Nutzung von QRIX erklärt werden.

Das Programm INDEX erledigt nicht die gesamte Arbeit der Indexerstellung, sondern es erzeugt zunächst einmal vorsortierte Zwischendateien mit den Namen `ii`*k* (*k*=1,2,...). Ist das erledigt, muß ein zweites Programm, eben QRIX (sprich "cricks", für "Quick Reorganization of IndeX") den Rest erledigen, d.h. die Zwischendateien vereinigen zu einer richtigen Indexdatei. QRIX schreibt protokollarische Meldungen in die Datei PROTOQ, INDEX dagegen in PROTOK.

Um es etwas genauer zu beschreiben, geht eine Indexierung (= Datenbank-Generierung) folgendermaßen vor sich (s.a. Kap.7). Man startet den Vorgang normalerweise aus a99, mit "Index erneuern" vom ORG-Menü (`h org` eingeben). Dabei entsteht eine Datei `org.bat`, in der man sich bei Interesse alle Programmaufrufe anschauen kann (siehe dazu auch Kap. 7.1).

1. INDEX liest `.c`ld-Dateien oder `.c`lg-Dateien ein und gibt eine neue Datenbankdatei (Typ `.c`lg) aus und eine Satztabelle (Typ `.tbl`). Dabei produziert es mit Hilfe der Index-Parameterdatei die Zugriffsschlüssel, die es aber zunächst im Arbeitsspeicher aufbewahrt.
2. Wenn der Arbeitsspeicher voll wird, sortiert INDEX die gesammelten Einträge, das sind bis zu 100.000 Stück, und schreibt sie in eine komprimierte Hilfsdatei mit dem Namen `ii`*k*.

   Dann läuft Vorgang 1.+2. weiter, bis die Ausgangsdaten zu Ende sind. Die Zahl *k* beginnt bei 1 und wird jedesmal um 1 hochgezählt.
3. Sind alle Ausgangsdaten durchgearbeitet, liegen die Zwischenergebnisse vor: die Hilfsdateien `ii1, ii2, ..., iiN`. Dann ist das Programm QRIX zu starten. In `org.bat` sieht man, wie es aufgerufen wird.
4. QRIX nimmt jeweils 10 Hilfsdateien und mischt sie zu einer neuen zusammen: aus den Nummern 1-10 wird Nummer 1, aus 11-20 wird Nummer 2 usw., bis nur noch höchstens 10 Dateien übrig sind.
5. Wenn schon eine `.c`DX-Datei vorhanden ist, sind 2 Situationen denkbar, und dem Programm wird mit Option `-fq` mitgeteilt, welche vorliegt:
   - **`-f0`** Die alte Indexdatei kann weg, denn es wurde eine Index-Erneuerung gestartet. Dann wird die `.c`DX -Datei zunächst gelöscht und aus den Hilfsdateien neu aufgebaut.
   - **`-f1`** INDEX war gestartet worden, um neue Daten zur Datenbank hinzuzufügen, oder um den zweiten Durchlauf zu erledigen: in diesem Fall muß QRIX die neuen Hilfsdateien mit der vorhandenen `.c`DX -Datei zu einer neuen `.c`DX-Datei zusammenmischen.
6. Wenn mit Stammsätzen gearbeitet wird (es kommt ein Befehl **`i5`** in der Index-Parameterdatei vor), dann ist ein **zweiter Durchlauf** (→ 10.2.6.8) nötig. Hierbei werden die Ersetzungen der Stammsatznummern durch die Klartexte der Namen, Schlagwörter etc. durchgeführt (→ 10.2.6.3 Index-Sonderbefehle). Dies gilt auch für die Kurzanzeige: sie wird automatisch mit bearbeitet. Im Falle der DemoBank ist dies so, und in der `org.bat` sieht man die Form des Aufrufs, wie auch im Kap. 7.1. Dem zweiten INDEX-Durchlauf folgt dann auch ein zweiter QRIX-Durchlauf.

Ab V27.2 kann man mehr als eine Indexdaei erzeugen lassen (MultiX-Konzept). Diese werden entweder alle in einem Arbeitsgang erzeugt oder wahlweise nur eine davon. (Beim Start aus dem ORG-Menü von a99 hat man die Wahl.)

Hinweis: Die Hilfsdateien `ii`*k* verschwinden am Ende automatisch. Falls der gesamte Vorgang ergebnislos abbricht und noch solche Dateien übrig sind, kann man sie untersuchen mit dem Hilfsprogramm `qr.exe`: z.B. mit dem Aufruf
`qr ii7 >ii7.txt` verwandelt man die Datei `ii7` in eine Textdatei `ii7.txt`. Die kann man sich anschauen, ob am Anfang oder Ende Zeilen stehen, die nicht mit | oder mit ~ anfangen. Das deutet dann auf einen Fehler in den Indexparametern.

Wiederanlauf nach Unterbrechung
Wenn QRIX z.B. durch Stromausfall abgebrochen wurde, stellt man zunächst fest, wie weit es gekommen war. Beispiel: es gibt die Dateien `ii1` bis `ii76` und `ii751` bis `ii1248`. Das bedeutet: es muß mit `ii751` weitergemacht werden (`ii76` ist sehr wahrscheinlich unvollständig). Dies veranlaßt man durch
`qrix -fq -n751 -d...`
Dann entsteht eine neue `ii76` aus `ii751` - `ii760`, usw.
Ist dann die letzte verarbeitet, geht es automatisch wieder vorn los: aus `ii1` bis `ii10` wird eine neue `ii1`, aus `ii11` bis `ii20` wird `ii2`, usw. Sind schließlich nur noch bis zu 10 `ii`-Dateien übrig, wird daraus die eigentliche Indexdatei erstellt.

Zum Zeitbedarf
Abhängig von der Leistung des Systems können je Stunde 100.000 bis mehr als 1.000.000 Datensätze indexiert werden. Mehrtägige Indexierungsläufe treten heute auch bei Millionenmengen nicht mehr auf: Das Indexieren der Daten des alten VK mit 15 Mio. Datensätzen dauerte z.B. nicht länger als 7 Stunden. Man kann mehrteilige Indexdateien auch etappenweise erstellen oder einzelne Indexteile herausnehmen oder getrennt erneuern, ohne den Gesamtindex neu generieren zu müssen (dazu gibt es die QRIX-Optionen -w und -W). Oder z.B. zeitweise einen Zusatzindex dranhängen und später wieder beseitigen, den man für eine Dublettenprüfung braucht (→ 7.6/7).

Aufruf
Mit folgenden Optionen ruft man QRIX auf: Zunächst die **Funktionen:**

`-fc` **Index kompaktieren** (insbes. Kombination mit -w und -W, s.u. !) ***CockPit*** **`µr o c`**
vorhandene .cDX-Datei komprimieren; .dLD und .TBL bleiben unberührt
Wenn man `-fc1` gibt, wird der alte Index nicht gelöscht, sondern in OLDIX umbenannt. Nach Rückbenennung ist er dann wieder benutzbar.
Nur bei dieser Funktion bleiben die manuellen Änderungen (→ Kap.1.4, Funktionen Inst/Del), die man im Index gemacht hat, erhalten.
Diese Funktion ist sinnvoll, wenn ein Index sich nach längerem Gebrauch stark vergrößert hat.
Die Kombination `QRIX -fc -W47` würde z.B. bewirken, daß beim Kompaktieren die Teil-Indizes 4 und 7 weggelassen werden (wozu das gut ist? → 7.6 und 7.8).

`-fd` **Index anzeigen** (default-Funktion)
zusammen mit den Optionen -s, -S, -w, -W und -x kann man sich verschiedene Teile einer Indexdatei am Bildschirm ansehen oder ausdrucken lassen. Stop/Abbruch mit **`x`** . Damit erstellt man beliebige Auszüge aus einem Index. Ganz ähnliche Leistungen ermöglicht der FLEX-Befehl `qrix`. (In a99 eingeben: `h xqrix` für die Doku und für eine Anwendung: `X qrix`)

`-fqi` **Hilfsdateien mischen und Index aufbauen** (i = 0 oder 1)
Dieser Aufruf wird von INDEX automatisch gemacht, sobald die ii-Dateien fertig sind. Man kann aber auch nach dem Ablauf von INDEX unterbrechen (Option `-m1`) und `QRIX -fq` später mit der Hand aufrufen. Die von INDEX erzeugten Dateien iik werden zusammengemischt. Wenn keine .cDX-Datei existiert, wird sie neu erzeugt. Wenn eine vorhanden und ist und `-fq1` gegeben wurde, wird sie mit den Hilfsdateien zu einem neuen Index vereinigt, bei `-fq0` wird der vorhandene Index gelöscht; wenn i fehlt, wird gefragt, ob das Vereinigen stattfinden soll oder nicht.
Wenn man **`QRIX -fq`** von Hand gibt, füge man die Option **`-Kn`** mit n=Schlüssellänge hinzu. (Wenn INDEX automatisch QRIX aufruft, passiert auch das automatisch.) Sonst fragt das Programm nach dieser Zahl, falls es die Index-Parameterdatei nicht findet (denn darin steht die Zahl im Befehl `il`).

Mit diesen drei Funktionen sind folgende Möglichkeiten kombinierbar:

Zusatz-Optionen: ***CockPit*** **`µf q`**

`-e`*iparam=dbdir*
Indexparameter: QRIX muß wissen, wie die Indexdatei heißt bzw. heißen soll, wo sie liegt bzw. liegen soll und was die maximale Schlüssellänge ist. Diese Option ermöglicht ihm, diese Dinge festzustellen.

`-E`*datei* **interne Satznummern ausgeben** : obsolet. Dies wurde in den ersten Web-Anwendungen eingesetzt, um eine Datei der internen Satznummern zu erhalten. Dies war dann zu kombinieren mit `-M` und `-t`:
[Damals gab es FLEX noch nicht. Heute wird für Web-Anwendungen der FLEX-Befehl `qrix` eingesetzt]

-Mn **Zeilenanzahl vorgeben** : es werden n Zeilen ausgegeben (nur bei `-fd` sinnvoll)

-tnnn,ppp **Kategorienummern** für die Ausgabe in *datei*. Default: `-t00,20`
Die Satznummer wird in Kategorie #nnn ausgegeben, in #ppp kommt die Kurzanzeige. Es entstehen Ausgabesätze im Grundformat, die nur aus #nnn und #ppp bestehen. Diese Grunddatei kann anschließend per SRCH verarbeitet werden. Dabei kann man die eigentlichen Sätze laden (per Nachladebefehl **`|01`**). Dies wurde für die frühe WWW-Methodik gebraucht, als es den *avanti*-Server noch nicht gab. Die entstandene *datei* mußte dann anschließend mit SRCH verarbeitet werden.

-sabc **Selektive Anzeige** (Ausschnitt aus dem Index, nur bei `-fd` sinnvoll)

-Sxyz Das Programm soll bei `abc` mit der Anzeige beginnen, bei `xyz` enden

-wijk **Register auswählen** (w = with; i j k = aufsteigende Ziffern zwischen 1 und 9, ':' für 10, ';' für 11)
nur die Register i, j und k sollen angezeigt werden; z.B. `-w135`

-Wijk **Register i, j und k weglassen** (W = Without)
z.B. mit -W123579 würde man nur die Register 4, 6 und 8 bekommen
(sind -w und -W gleichzeitig angegeben, wirkt nur -w)

-xn **Schwellenwert für die Anzeige**
nur Einträge mit mehr als n Datensätzen erscheinen in der Kontrollanzeige und in PROTOQ. Standardwert ist **`-x999`**. Wenn man den gesamten Index sehen will: **`-x0`** geben (nur bei `-fd` sinnvoll)

-T **Testbetrieb** : man erhält zusätzliche Meldungen über den Verlauf der Arbeit des Programms.

-Q*proto* **Protokolldatei**: Meldungen werden in die Datei PROTOQ geschrieben. Mit dieser Option kann man einen anderen Namen vorgeben.

>*filename* **Ausgabe in Datei** *filename* **umlenken** : für Option **`-fd`** ist es sinnvoll, die Ausgabe statt auf den Bildschirm in eine Datei schreiben zu lassen. Bei den ältesten Web-Schnittstellen wurde dies eingesetzt, um Registerauszüge zunächst in eine Datei schreiben zu lassen und diese dann für die Anzeige in Netscape aufzubereiten.

Beispiele:
```
qrix -fq1 -dbibl –ecat=bibl -x1000
```
`ii`-Dateien auf `bibl\` suchen und dort in `cat.adx` einmischen
```
qrix -fd -ddemo2 -ecat=demo2 -w8 >reg8.txt
```
Inhalt des Registers 8 von `demo2\cat.adx` ausgeben, und zwar in die Datei `reg8.txt`

7.6 Einzelne Register erneuern

Wer größere Datenmengen hat (d.h. mindestens mehrere zehntausend), wird es lästig finden, wegen einer Änderung an einem der Register gleich einen umfassenden Gesamtlauf zur Neu-Indexierung zu machen. Ist auch nicht notwendig. Man geht so vor: Nehmen wir an, die Register 2 und 5 sollen erneuert werden. Dann:

- Man führt die nötigen Änderungen an der Index-Parameterdatei aus und macht zunächst alle nicht für die Register 2 und 5 zuständigen ak-Zeilen unwirksam, indem man jeweils ein Leerzeichen davor setzt.
- Man beseitigt die Register 2 und 5 wie in Abschnitt 7.8 gezeigt (**`qrix -fc –W25 ...`**)
- **`index -fi1 -0`** erstellt die neuen Register 2 und 5 und arbeitet sie in die alte Indexdatei ein, evtl. unter Mithilfe von QRIX, welches aber automatisch aufgerufen wird. (Achtung: bei `index -fi0` hätte man anschließend **nur** noch die Register 2 und 5, die anderen wären weg!) Die nur für INDEX geltende Option **`-0`** (Null) verhindert übrigens, daß die Leerschlüssel, die ja ins Register 1 kommen, bei dem Durchlauf neu gebildet werden. Schaden könnte das zwar nicht, aber es würden lauter Meldungen "Ladefehler ..." kommen.
- Nicht vergessen: in der Index-Parameterdatei wieder alle ak-Zeilen wirksam machen (Leerzeichen vor den ak-Zeilen wieder wegnehmen). Sonst werden bei künftigen Änderungen und Neueingaben nur die Register 2 und 5 bedient.

7.7 Neues Register hinzufügen

Solange man noch weniger als 11 Register hat, kann man ohne weiteres eines oder mehrere neue hinzufügen. Das geschieht wie folgt: (wenn es um das Kurztitelregister geht, siehe 7.0, Situation 6)

- In der Index-Parameterdatei ergänzt man die nötigen ak-Zeilen und Prozeduren für die Schlüssel, die in dem neuen Register stehen sollen. Dann macht man zeitweilig die anderen ak-Zeilen alle unwirksam, indem man jeweils ein Leerzeichen davor setzt.
- Man startet **`index -fi1`** und läßt es durchlaufen. Das neue Register wird angelegt und in die vorhandene Indexdatei eingearbeitet.
- Nicht vergessen: in der Index-Parameterdatei wieder alle ak-Zeilen wirksam machen (Leerzeichen vor den ak-Zeilen wieder wegnehmen).

7.8 Ein Register ganz löschen

Ein Register **k**, das nur temporär gebraucht wird, kann auch mit Option **`-W`** wieder völlig beseitigt werden. Das leistet der Befehl

`qrix -fc -Wk -e`*param/datapath*

Man muß also zu diesem Zweck nicht die gesamte Indexproduktion neu durchlaufen lassen. Zugleich wird dabei der Index kompaktiert - was nie schaden kann.

7.9 Registerauszüge

Windows: **`h exprt`**, dann Menüpunkt "Registerabschnitt"

Hinweis: Viel leichter geht's mit den Windows-Programmen über den Button [List] im Indexfenster (→ Kap.2). Oder mit dem FLEX-Befehl `qrix` (Menü Export-Komfortfunktionen / Registerabschnitt).
Manchmal wird gewünscht, einzelne Register ganz oder teilweise in eine Textdatei kopieren zu können. Wie das geht, wurde oben unter 7.5 schon erwähnt. Wir zeigen es hier noch deutlicher und mit Beispielen, damit es jeder leicht durchführen kann, auch ohne sich mit den anderen Optionen zu befassen. Wenn man die Bildschirmausgabe mit dem MS-DOS-Steuerzeichen **>** in eine Datei oder auf den Drucker umlenkt, kann man Registerauszüge in Textdateien umwandeln oder ausdrucken. Das ***CockPit***-Menü **`µf q`** für QRIX enthält einen Punkt **d Dateiname**. Dort trägt man den gewünschten Namen oder PRN ein. Der QRIX-Aufruf wird dann ergänzt um die Option *>dateiname*. Diese kann man natürlich auch in eigene Aufrufe einbauen.

Beispiel 1: Auf KATALOG (Unterverzeichnis von C:\ALLEGRO) liegt die Datenbank CAT. Aus dem Register 1 soll ein Ausschnitt von 30 Zeilen Länge in die Datei EXTRACT geschrieben werden, beginnend bei "mozart". Das könnte man so erreichen:

```
qrix -dkatalog -fd -x0 -smozart -M30 -w1 -ecat=katalog >extract
```

Beispiel 2: Man will aus Register 1 die Zeilen mit den gelöschten Sätzen sehen. Die Schlüssel beginnen alle mit "//". Der Befehl kann so aussehen:

```
qrix -fd -x0 -dkatalog -s// -S0 -w1 -ecat=katalog
```

DOS: Anstatt diese Aufrufe mit der Hand einzugeben oder in Batchdateien zu installieren, kann man auch im ***CockPit*** das Menü "Funktionen / QRIX" aufrufen. Alle Einzelheiten des Aufrufs werden dort in einem Eingabeformular eingestellt.

`newinx ???` Eine Datei mit diesem Namen entsteht, wenn QRIX nicht weiß, wie die zu produzierende Indexdatei heißen soll, oder wenn die alte sich nicht löschen läßt. Normalerweise tritt dieser Fall nicht auf.
Abhilfe: einfach `newinx` umbenennen, also z.B. `ren newinx cat.adx`.

Ab 2008 32bit-Programm. Mehr: http://www.allegro-c.de/bluechyp/sort.htm

8 Sortieren

a99: **h ac8**

Man erwartet, daß die Software in immer neuer Auswahl und Variation geordnete Dateien und Listen hervorbringen kann. Die Produktion einer sortierten Datei oder Liste zerfällt in drei Vorgänge, damit wirklich jede Anforderung erfüllt werden kann. Alle drei Vorgänge läßt man normalerweise über eine Batchdatei automatisch hintereinander ablaufen (→ Kap.6).
Beispiele sind SR-LIST.BAT, PR-LIST.BAT, QUEX.BAT und RN.BAT. Darin ist das Sortieren enthalten, und man bemerkt es nur an den Zwischenmeldungen.
Hinweis: Bei den Windows-Programmen kann man Ergebnismengen unmittelbar am Bildschirm sortieren lassen, wobei zum Sortieren jeder Punkt der Kurzanzeige benutzt werden kann: der Titel, der Verfasser, das Jahr etc. Danach kann die Ergebnismenge gleich in der betr. Reihenfolge exportiert werden (Drucker-Button!). Unter DOS ist das nicht möglich.
Es folgt die Beschreibung, was man tun muß, wenn man ausnahmsweise alle drei Schritte einzeln durchführen will:

1. Zuerst wird das Volltext-Suchprogramm SRCH (→ Kap.4) oder das Schnellzugriffprogramm (→ Kap.1) mit Export-Option (-e...) benutzt, um aus den vorhandenen Daten eine "sortierfähige Datei" aufzubereiten.

Eine Datei ist **sortierfähig**, wenn jeder Datensatz
1. mit einem Sortierbegriff beginnt und
2. mit den Zeichen "Carriage Return" und "Line Feed" (Codes 13/10) endet.

Mit Hilfe der Parameterdateien des Typs S-*.APR können die Programme solche Dateien erstellen. Als Sortierbegriff wird dann z.B. die Kombination aus den Kategorien #40 und #20 genommen und in sortierbarer Form an den Anfang jedes Satzes geschrieben. Sehen Sie sich dazu die Datei S-PJ.APR an, die Kommentare darin ermöglichen Ihnen, eine andere Sortierung einzustellen: verändern Sie dazu die nötigen Teile (hauptsächlich die Kategorienummern) mittels Ihres Texteditors.
Die Tabelle S.APT, die von den S-*.APR-Dateien automatisch nachgeladen wird, enthält die Vorschriften für die Zeichenumwandlung (Klein-/Großbuchstaben, Umlautauflösung, Sonderzeichen weglassen etc.). Auch diese Tabelle können Sie verändern.

Hat man jedoch, als Sonderfall, in die Kategorie #00 jedes Datensatzes ein sortierfähiges Kriterium von vornherein selbst eingegeben, und will man nun dieses und kein anderes zum Sortieren benutzen, dann benutzt man die Parameterdatei I-1.APR zum Erstellen der Sortierdatei.

2. Vom ***CockPit***-Menü "Funktionen" aus ruft man mit Ziffer 8 das Sortierprogramm auf. Man wird gefragt
- auf welchem Laufwerk die zu sortierende Datei ist;
  es kommt dann die Liste der auf dem Laufwerk stehenden Dateien, so daß man auf die übliche Art (durch Setzen eines '+') die gewünschte Datei markieren kann,
- wohin die sortierte Datei zu schreiben ist und wie sie heißen soll. Wenn auf dem Ziellaufwerk nicht mindestens die doppelte Größe der Sortierdatei frei ist, sucht das Programm nach einem anderen Laufwerk mit genügend Platz. Wenn keines zu finden ist, kommt eine entsprechende Nachricht, und ASORT tut erst einmal nichts. Schaffen Sie Platz, und zwar mindestens doppelt so viel, wie die zu sortierende Datei beansprucht.

Nach diesen Vorbereitungen wird die eigentliche Sortierarbeit erledigt. Einige Zwischenmeldungen informieren Sie über den Fortgang des Prozesses.

3. Nach getaner Arbeit erscheint wieder das ***CockPit***. Wenn die sortierte Datei nun gedruckt oder mit anderen Parametern exportiert werden soll, wählt man Funktion 6 ("Export", **µf 6**) und dort die geeignete Parameterdatei für die Produktion.

Eine Anmerkung zum Sortieren von Zahlen: korrekte Sortierung von Zahlenwerten erfordert, daß die Ziffern spaltenrichtig mit führenden Nullen eingegeben wurden. Es wird Zeichen für Zeichen von links nach rechts sortiert, daher können Zahlen mit unterschiedlicher Stellenzahl nicht automatisch nach ihrem Zahlenwert geordnet werden. Mit einem Manipulationsbefehl wie z.B. `r7,0` kann man jedoch beim Exportieren erreichen, daß Zahlen für einen Sortierkopf korrekt aufbereitet werden (→ 10.2.6.3). Ziffern ordnen ansonsten vor Buchstaben.

In Kap. 10.4.2 ist der Prototyp einer Parameterdatei abgedruckt, die ***allegro***-Daten zum Sortieren vorbereitet. Sie können diese Datei, S-0.APR, als Muster für eigene Sortierproduktionen nehmen.

Typisch bei der Produktion sortierter Listen ist es, daß zuerst eine sortierfähige Grunddatei erstellt und diese mit ASORT sortiert wird, anschließend folgt dann noch ein SRCH-Durchlauf zur Erstellung des Endprodukts aus der sortierten Grunddatei.

Aufruf des Programms in Stapeldateien

Das Programm ASORT läßt sich, wie die anderen Programme, in einer Stapeldatei benutzen. Dies ist wichtig, wenn man das Programm in größere Abläufe einbauen will. In den schon genannten Batchdateien (SR-LIST.BAT usw.) kann man sich anschauen, wie so etwas gemacht wird.

Der einfachste Fall (der Normalfall) sieht so aus, daß eine Datei namens *xyz* vorliegt und durch Sortierung daraus eine Datei namens *abc* entstehen soll. Dann genügt der Befehl

`asort` *xyz abc*

wobei die Namen wie üblich mit Pfadangabe zu geben sind, wenn die Dateien nicht auf dem aktuellen Pfad liegen. Ein Beispiel:

`asort uuu.alg sss.alg`

Damit ASORT flexibel einsetzbar wird, gibt es noch ein paar Feinheiten. Es kann beim Aufruf die folgenden Optionen auswerten:

- zwei Dateinamen: Eingabe- und Ausgabedatei, in genau dieser Reihenfolge.
  Man kann die Namen ohne '`-`' angeben, jedoch sind auch **`-i`** *eingabedatei* und **`-o`** *ausgabedatei* möglich ('i' für Input, 'o' für Output).
- **`-d`** *muster* : [obsolet!] alle mit dem *muster* übereinstimmenden Dateinamen werden zum Sortieren angeboten, man wählt auf die übliche Art (Markieren mit einem '+') die zu sortierende Datei aus (nur eine).
- eine Positionsangabe für den Beginn des Sortierbegriffs. Diese wird mit
  `-c` *pos* ('c' steht für "column")
  angegeben (die Zählung beginnt mit 0 für das erste Zeichen). Default ist `-c 0`, d.h. ohne **`-c`** beginnt der Vergleich beim ersten Zeichen. Meistens ist das in Ordnung, auch wenn zuerst eine immer gleiche Kategorienummer kommt, denn die wirkt sich auf die Sortierung dann ohnehin nicht aus. **`-c`** muß also nur angegeben werden, wenn vor dem Sortierbegriff eine feste Anzahl von Zeichen steht, die unterschiedliche Werte haben können. Die Länge des Sortierbegriffs ist nicht begrenzt (kann auch länger als 256 Zeichen sein).
- **`-u1`** bzw. **`-u2`**: wenn die Kategorien #u1 bzw. #u1 *und* #u2 nach dem Sortieren zu beseitigen sind. Mit **`-u3`** wird zusätzlich auch noch der Code 01 am Satzanfang beseitigt.
- **`-r`** schaltet die Rückwärts-Sortierung ein

Allgemeine Form:

`asort` *`[laufw:][pfad\]eingabedatei [laufw:][pfad\]ausgabedatei`* [**`-c`** *`pos`*]

Ein solcher Befehl kann auch direkt mit der Hand eingegeben werden, um eine Datei ad hoc zu sortieren.

Anmerkung für DOS- und UNIX-Kenner

Mit ASORT kann man auch andere Dateien sortieren, wenn die zu ordnenden Einheiten (Datensätze) durch die Kombination 13 10 (0Dh 0Ah, auch "Carriage Return / Line Feed" genannt) getrennt sind. Das ist die einzige Voraussetzung - die Datensätze können ansonsten von beliebiger Länge sein. Das Programm bildet Portionen von etwa 64K Größe und sortiert diese zunächst vor. Dann werden jeweils 4 dieser Portionen zu einer Datei zusammengemischt. Das setzt sich fort, bis schließlich nur noch eine Datei übrig ist. Das ist dann die Ergebnisdatei. ASORT prüft auch, bevor es beginnt, ob genügend Platz auf dem aktuellen Laufwerk ist. Wenn nicht, sieht es sich die anderen Laufwerke an und nimmt bei Erfolg ein anderes für die Zwischendateien. (ASORT wurde geschrieben, weil das SORT von MS-DOS nur Dateien bis 64K und Sätze bis 255 Zeichen sortieren kann. ASORT schafft 640 MB und Sätze bis 16K. Oder anders ausgedrückt: 8 Millionen Sätze mit durchnittlich 80 Byte.)

Unter UNIX gibt es bessere Sortierprogramme, z.B. `gsort` (GNU-sort), daher wird ASORT nicht fuer UNIX gebraucht. Für SUN und LINUX wird es gleichwohl angeboten. Einzig die Option **`-u`** haben andere Programme nicht. Das Programm `gsort.exe` gibt es auch für DOS. Es ist auf dem FTP-Server und auf der CD-ROM von V15 (Verzeichnis `\programm\util`) zu finden. Man startet es mit `gsort` <*Eingabedatei* >*Ausgabedatei*.

`gsort` kennt noch weitere Optionen, die ASORT nicht bietet: `-f` zum Ignorieren des groß-/klein-Unterschiedes, Sortieren nach bestimmten Feldern, oder das Eliminieren von Sätzen mit gleichem Sortierfeld). Unter UNIX liefert der Befehl `man sort` oder `man gsort` die Dokumentation zum Programm. Probleme kann es mit dem Feldende-Code 0 geben.

Mit ***allegro*** können nur Export-Experten solcherlei Leistungen realisieren, indem sie geeignete Sortierfelder per Export produzieren.

Tip: Wenn man z.B. absteigend nach Erscheinungsjahr sortieren will, kann man auch im Sortierfeld das Jahr als komplementäre Zahl codieren: `#76 b3 x"*1" x"-10000" f"-" e"."` (b3, wenn `ks=1` ist). Die Jahreszahl wird also von 10000 subtrahiert, dadurch kehrt sich dann die Ordnung um.

9 Datenbank-Management, Update

a99: `h update` ; FLEX : **`h xupdate`**

9.0 Konzept

Das Programm Nummer 9, **`UPD`**, kann Datensätze aus einer Grund- oder Externdatei (Typ .ALG bzw. ADT) lesen, mit vorhandenen Sätzen in der Datenbank identifizieren und evtl. austauschen, zusammenmischen, oder auch löschen, und zwar auf Mehrplatzsystemen auch während des laufenden Betriebs. Wie PRESTO zeichnet es seine Aktionen auch in der .LOG-Datei auf. (Für das Programm INDEX trifft beides nicht zu, dafür arbeitet es viel schneller.)

Das Programm UPD hat die folgenden drei Funktionen:

`-fp` : **Playback** : eine Sicherungsdatei (Typ .LOG) wird in eine Backup-Kopie der Datenbank eingespeist, um den Zustand vor einem Zusammenbruch wiederherzustellen (→ 0.7). Die Identifizierung erfolgt über die interne Satznummer, daher wird kein eindeutiger Schlüssel benötigt. Das Playback funktioniert nur dann richtig, wenn man zum Zeitpunkt der Datensicherung die .LOG-Datei **löscht**. Danach entsteht dann eine neue .LOG-Datei, die exakt die nach der Sicherung erfolgenden Veränderungen aufzeichnet. (.LOG-Datei **keinesfalls** mit der Datenbank zusammen sichern, denn ihr Inhalt ist ja schon komplett in der Datenbank enthalten!)
Auf Mehrplatzsystemen sollte man erst nach Abschluß des Playback den Betrieb wieder anfahren.

`-fm` : **Mischen (Merge)** : eine Grunddatei (Typ .cLG) wird in eine Datenbank eingespeist, wobei auch Sätze gelöscht werden können. Das Einspeisen kann auf unterschiedliche Art geschehen. Identifiziert wird dabei über einen vom Benutzer definierten "Primärschlüssel", der eindeutig sein muß (mehr darüber später).
Die Funktion "Mischen" kann auch vom ***avanti***-Server ausgeführt werden (→ 9.5), ebenso aber das Windows-Programm ***a99***. Geben Sie **`h fremd`** im Schreibfeld ein.

`-fc` : **Check** : eine Grunddatei, die z.B. durch Import entstanden ist, wird an einer Datenbank auf Dubletten geprüft. Wenn eine Übereinstimmung bei vorher festzulegenden Zugriffsschlüsseln erkannt wird, gibt das Programm den neuen Datensatz und den oder die vermutlich identischen, vorhandenen Sätze aus (Parameterdatei und Dateiname sind mit Option **`-e`** anzugeben). Der Anwender kann die Liste kontrollieren, die Grunddatei redigieren und sie dann mit Funktion "Mischen" einarbeiten lassen.

Hinweise: In Wirklichkeit startet ab V32.1 `upd.exe` dann das Programm `acon` mit dem Jobskript `update.job`
Das Programm INDEX (→ Kap.7) kann mit seiner Funktion `-f7` ebenfalls neue Daten in eine Datenbank einbringen, jedoch kann es keine Sätze identifizieren und austauschen, sondern nur als neue Sätze hinzufügen. Außerdem darf man INDEX nicht im laufenden Mehrplatzbetrieb einsetzen, sonst gibt es Schäden an den Registern.

9.1 Daten einspeisen, Ablauf

Das Programm UPD wird vom ***CockPit***-Menü "Funktionen" über Ziffer 9 aufgerufen oder von MS-DOS aus mit seinem Namen UPD. Unter UNIX muß man es per Shellscript mit allen nötigen Optionen starten.
Die wichtigsten Optionen für den Start von UPD sind (→ Kap.12):

`-fp` Funktion "Playback" (also Einspeisen der LOG-Datei)

`-fm`*xy* Funktion "Mischen" (die zwei Ziffern *xy* bestimmen den Modus, siehe unten)

`-fc` Funktion "Check" (Vergleich mit Primärschlüssel), mit Kontrollausgabe
(dafür ist zusätzlich eine Option **`-e`** *param/file* nötig, → Kap.12)

`-d` *dbp**dbn* Angabe der Datenbank, an der das Programm arbeiten soll;
dbp = Name des Datenbank-Verzeichnisses, *dbn* = Datenbankname

`-u` *ufile* Name der einzuspeisenden Datei
Wenn die Datei *ufile* vom Typ .LOG ist, wird "Playback" gefahren, auch bei **`-fm`** oder **`-fc`**.

`-S` **Einzelplatz**modus (bis V14 **wichtig**, sonst funktioniert es auf Einzelplatzsystem nicht),
bei **Mehrplatz**betrieb dagegen **darf** diese Option nicht benutzt werden!

`-F` Beschleunigung auf Mehrplatzsystem, aber die anderen Plätze erhalten beim Speichern (F10) meistens die Meldung "bitte warten". (Wenn -S gesetzt wurde, ist -F automatisch aktiv)
Setzt man eine Ziffer hinter **F**, wird diese als Sekundenzahl gedeutet und das Programm dadurch künstlich verlangsamt, und zwar um die angegebene Zahl von Sekunden je Satz.

-I*name* andere Indexparameter benutzen (*name*.cPI statt *dbn*.cPI)
-m1 manuelle Unterbrechung möglich (durch Druck auf 'x')
-n *k* Neue Datensätze sollen in Datei *k* gespeichert werden
-N *j* für neue Datensätze gilt: wenn *j*=0: Leersätze nicht zum Speichern benutzen, *j*=1: Alle Leersätze zum Speichern zugelassen (d.h. nicht nur die in der mit `-n`*k* angegebenen Datei)
Für *j*=2 (default) gilt: nur in die mit Option `-n` gewählte Datei wird gespeichert.
-e*param*/*outfile* Kontroll-Liste mit Parameterdatei *param*.ALG in Datei *outfile* ausgeben
-x*proto* Die Protokollmeldungen sollen in die Datei *proto*; sonst gehen sie in `upro`.
-R Aufzeichnung: alle Bildschirmmeldungen werden auch in die Protokolldatei geschrieben.

Mit DOS:
Wenn man auf dem ***CockPit***-Menü "Funktionen" die **9** wählt, kann man alle UPD-Optionen in ein übersichtliches Formular eintragen. Das sieht etwa so aus:

Hier soll folgendes passieren: die Datenbank auf C:\ALLEGRO\KATALOG ist zu aktualisieren, und zwar mit der Mischfunktion **-fm41** (mehr dazu später). Die Aktualisierungsdaten stehen in der Datei NEWDATA.ALG auf dem Laufwerk A: (die Momentaufnahme zeigt den Punkt, wo gerade dieser Name eingegeben wurde). Wenn man das Eingabefeld "Update-File" anwählt, werden zuerst zur Auswahl die .ALG-Dateien angeboten, die auf dem Daten- bzw. Programmverzeichnis stehen. Gibt es keine solche oder wählt man davon keine aus sondern verläßt die Liste mit <Esc>, kommt man in das unten sichtbare Eingabefeld, wo man "freihändig" eine Datei vorgeben kann, die auf einem ganz anderen Pfad steht. Menüpunkt **GO** löst dann den Start aus. Dabei entsteht eine Batchdatei CCC.BAT, die hauptsächlich diese Zeile enthält:

```
upd -fm41 -kA -n1 -dC:\ALLEGRO\KATALOG -PC:\ALLEGRO -a3 -ua:\newdata.alg -lger
```

Dies also ist ein spezielles Beispiel für einen Aufruf des Programms.
Die allgemeine Form des **Aufrufs** von UPD für einen automatischen Ablauf sieht so aus:

für Playback: **upd -fp -d** *sub**dbn* **-u** dbn**.log [-k***c***]**

für Mischen: **upd -fm***xy* **-d** *sub**dbn* **-u** *filename* **-n1 [-k***c***] [-F] [-e...] [-N***j***]**

für Check: **upd -fc -d** *sub**dbn* **-u** *filename* **-e** *param*/*outfile* **[-k***c***]**

wobei *filename* der vollständige Name (mit Laufwerk und Pfad) einer .cLG-Datei ist und die Datenbank *dbn* auf dem Unterverzeichnis \ALLEGRO*sub* liegt. Die Optionen in [...] sind nicht immer notwendig.

Nochmals der Hinweis: Ab V32 wird durch UPD automatisch das Konsolprogramm acon gestartet mit update.job:

`acon -j update -f...`
Dann aber müssen die Optionen vollständig angegeben sein, nicht etwa z.B. `-u...` weglassen. Denn das Programm arbeitet nicht interaktiv, fragt also fehlende Angaben nicht vom Nutzer ab.

Für das alte DOS-Programm galt: Wenn man die Option **-u** wegläßt, fragt das Programm selbst nach dem Namen der einzumischenden Datei. Es genügte bei der DOS-Version sogar, beim Aufruf nur die Optionen **-k** und **-f** anzugeben, alles andere wurde dann abgefragt.
Für den "Check" muß die Export-Option -e*param*/*outfile* gesetzt werden; dabei entsteht mit Hilfe von *param*.dPR (z.B. mit `e-1.apr`) eine Ausgabedatei *outfile*, in der am Ende die identifizierten Datensätze, die vorhandenen und dazu die neuen, im Format *param* stehen sollen.
Für DOS gilt: fehlt **-d***sub*, wird der Pfadname genommen, der in der Umgebungsvariablen -D (Datenbankpfad) eingetragen ist, wenn diese nicht existiert, wird das aktuelle Verzeichnis genommen.
Auch bei Funktion "Mischen" kann man mit einer Option **-e** verlangen, daß ein Export erstellt wird.
Die Option **-k***x* muß nur gesetzt werden, wie immer, wenn man nicht `a.cfg` verwendet. DOS: Alternativ kann man auch die Umgebungsvariable **-K** vorher setzen (mit **set -k=***x*).

Zu erklären bleiben die Variablen *x* und *y* der Option -fmxy:
x wirkt sich in solchen Fällen aus, wenn zu dem neuen Datensatz ein alter mit gleichem Primärschlüssel gefunden wird:

x = **0** : Es findet **keine Ersetzung** statt; jeder Satz wird ohne Vergleich des Primärschlüssels wie ein neuer Satz eingemischt

1 : der alte Satz wird durch den neuen **vollständig ersetzt**, mitsamt aller Untersätze

2 : der alte Satz bleibt **unverändert**, d.h. der neue wird ignoriert,

3 : der alte Satz wird **kategorieweise in den neuen eingemischt** (d.h. vorhandene Kategorien des alten Datensatzes bleiben erhalten, vom neuen werden nur die zusätzlichen Kategorien übernommen!). Die Untersätze des alten Satzes bleiben komplett erhalten, wenn der neue welche hat, fallen sie weg.

4 : der neue Satz wird **kategorieweise in den alten eingemischt** (d.h. eine im neuen Satz vorhandene Kategorie ersetzt eine entsprechende, im alten Datensatz vorhandene; vorhandene Kategorien bleiben nur erhalten, wenn sie im neuen Satz nicht vorkommen.) Untersätze des alten Satzes verschwinden vollständig und werden durch die des neuen ersetzt!

y gilt für den anderen Fall, wo kein Datensatz mit gleichem Primärschlüssel existiert:

y **= 0** : neuen Datensatz **ignorieren**

1 : neuen Datensatz **einmischen** (in die durch Option **-n** angegebene Datei).

Ein häufiger Fall ist **-fm41** : nur die in den neuen Fremddaten nicht belegten Kategorien bleiben in den alten Datensätzen erhalten (z.B. können dies Signaturen sein oder lokale Sacherschließungsdaten), die anderen werden alle ersetzt, und neue Aufnahmen werden in die Datenbank eingefügt. Man wird diesen Fall z.B. anwenden, wenn man per ISBN Fremddaten aus einer CD- oder online-Quelle abzieht und diese dann zwecks Qualitätsverbesserung in die eigene Datenbank einmischt. Die Fremddaten müssen in jedem Fall zunächst per Import konvertiert werden. Dabei kann man problematische Kategorien entweder ausfiltern, also nicht übernehmen, oder auf solche Nummern legen, die man in den eigenen Daten nicht besetzt hat. So vermeidet man ungewollte Kategorieersetzungen.
Anmerkung: Die Kombination **-fm20** wäre sinnlos, denn das Programm würde gar nichts tun.

Teilfelder aktualisieren: Wenn in der Update-Datei eine Kategorie steht mit ▼▼ am Anfang und dann einem oder mehreren Teilfeldern, werden die Teilfelder ausgetauscht. *Beispiel:* in der Update-Datei steht (hier MARC-Format, U.CFG)

```
#260   ▼▼bRowohlt▼c1980                  im Titelsatz (der aktualisiert werden soll) steht
#260   ▼aBerlin▼bFischer▼c1977.          Dann kommt heraus:
#260   ▼aBerlin▼bRowohlt▼c1980
```

Sonderfall: Steht in der Update-Datei `#260 ▼▼b` (ohne Text hinter ▼b), wird Teilfeld ▼b gelöscht.

Obacht: **Hierarchische Sätze** werden bei -fm3 bzw. -fm4 so gemischt, daß die alten Untersätze komplett erhalten bleiben bzw. komplett ersetzt werden. Das Einmischen funktioniert also nicht auf den unteren Ebenen. Hier wird man in Zukunft besser mit verknüpften statt hierarchischen Aufnahmen arbeiten (→ Kap.10.2.6.7). Das Konsolprogramm ***acon*** (→ 9.5) kann allerdings auch Untersätze von hierarchischen Sätzen gezielt einzeln austauschen.

9.2 LOG-Datei auswerten/umwandeln

***a99*: `h super`**

Gelegentlich entsteht der Wunsch, die .LOG-Datei einer Datenbank untersuchen zu können. Ohne andere Mittel kann man folgendes tun: die Datei von *dbn*.LOG auf *dbn*.ALD umbenennen, dann läßt sie sich mit dem Programm SRCH durchsuchen und exportieren. Ihre Struktur ist formal identisch mit der Struktur einer **Datenbank**datei, deshalb kann SRCH sie lesen. Dieser Trick war mit Version 13 aber unnötig geworden, denn es kam das Hilfsprogramm namens **LOG2ALG.EXE** (lies "LOG to ALG"). Es wandelt die Datei *dbn*.LOG um in eine **Grund**datei *dbn*.ALG. Dabei werden die internen Satznummern, die jeweils am Beginn eines Satzes stehen, weggelassen. Jeder (Haupt-)Datensatz beginnt mit dem Code 1, gelöschte Sätze beginnen mit der Sonderkategorie **`#u1 @@@@@`**. Neu erfaßte Sätze beginnen mit **`#u1 #####`***N*, wobei *N* die Nummer der Datei ist, in welcher der Satz gespeichert wurde. Die Datei *dbn*.ALG kann man ebenfalls mit UPD einmischen, dann aber mit Funktion `-fm11` statt `-fp`, und NUR dann, wenn es einen eindeutigen Primärschlüssel gibt.
Was hat man davon? Wenn man z.B. regelmäßig zwei getrennte, unabhängig voneinander betriebene Datenbanken aktualisieren will, kann man mit UPD und vorgeschaltetem LOG2ALG die .LOG-Dateien zwischen beiden Datenbanken austauschen, um die jeweils andere zu aktualisieren. Dieser Wunsch entsteht immer dann, wenn man auf zwei oder drei nicht vernetzten Einzelrechnern Katalogisierung betreibt und die Daten untereinander austauschen will, so daß jeder über denselben Datenbestand verfügt.
Bitte beachten: In der Konfiguration der Zieldatenbank sollte man die Befehle `cn` und `ce` (für Erfassungs- bzw. Änderungsdatum) deaktivieren, damit diese Angaben in den einzumischenden Daten erhalten bleiben.

Beispiel: die Datenbanken heißen beide CAT (mit A.CFG) und liegen jeweils auf C:\ALLEGRO\KATALOG.
Schritt 1: Mit **`log2alg cat`** erhält man die Datei CAT.ALG; man kopiert sie auf die Diskette im Laufwerk A: und bringt diese zu dem anderen Rechner.
Schritt 2: Man gibt auf dem anderen Rechner den Befehl
`upd -fm11 -u a:cat.alg -dkatalog -n1`
(die Nummer hinter **`-n`** ist beliebig zwischen 1 und 255).
Sätze mit gleichem Primärschlüssel werden dann ausgetauscht, gelöschte werden in der Zieldatenbank ebenfalls gelöscht, und neu erfaßte werden in dieselbe Datei geschrieben wie in der Quelldatenbank. Dieser Vorgang kann wöchentlich oder sogar täglich durchgeführt werden. Es ist zu raten, die .LOG-Datei dann jeweils zu löschen, damit nicht dieselben Daten immer wieder zurück übertragen werden. (Schlimm wäre das nicht, sie würden einfach nochmals ersetzt.)
Ausnahme: wenn man neue Daten mit INDEX einmischt, werden diese nicht in der .LOG aufgezeichnet! Man muß daher den INDEX-Vorgang an beiden Datenbanken durchführen, um die Datenbanken beide zu aktualisieren.

9.3 Sätze löschen mit UPD

Der Code 9 zum Löschen funktioniert auch dann, wenn die .ALG-Datei nicht aus einer .LOG-Datei hervorgegangen ist. Dadurch kann man das Programm UPD auch benutzen, um Löschungen automatisch durchzuführen. Die zu löschenden Sätze müssen nur mit einem ASCII-Code 9 statt 1 beginnen und brauchen nur diejenigen Kategorien zu enthalten, die zur Bildung des Primärschlüssels notwendig sind. *Alternative:* der zu löschende Satz beginnt mit `#u1 @@@@@`. Wenn UPD einen solchen Satz aus der .ALG-Datei einliest, bildet es den Primärschlüssel, sucht in der Datenbank den zugehörigen Satz und löscht diesen. Wird er nicht gefunden, passiert nichts.

Bemerkungen über Primärschlüssel
Der **Primärschlüssel** ist der erste Schlüssel, der bei der Abarbeitung der ak-Befehle (→ 10.2.1) in der Index-Parameterdatei dbn.API entsteht. Der Primärschlüssel muß, um für UPD tauglich zu sein,

- **eindeutig** sein: wenn mehrere Sätze mit demselben Primärschlüssel vorhanden wären, würde der mit der kleinsten Satznummer genommen, was zu ganz falschen Ersetzungen führen könnte!
- bei hierarchischen mehrbändigen Werken **aus der Hauptaufnahme** hervorgehen; es darf sich also nicht z.B. um eine Zugangsnummer handeln, die evtl. nur in einer Unteraufnahme vorkommt.

Jedoch hat der Primärschlüssel ansonsten keine Sonderstellung für das Datenbanksystem. Das bedeutet: man kann für einen speziellen Update-Vorgang ad hoc einen anderen Schlüssel zum Primärschlüssel deklarieren. Dazu braucht man nur die ak-Zeilen umzuordnen, also den bewußten Schlüssel an die erste Stelle zu rücken; keineswegs muß man neu indexieren! Dasselbe gilt für die Check-Funktion: hintereinander kann man mehrere Check-Durchläufe mit unterschiedlichen Primärschlüsseln durchführen.

Hinweise zur Programmierung des Primärschlüssels:
Nehmen wir als Beispiel an, wir hätten keine bestimmte Kategorie wie z.B. #00 mit eindeutiger Identnummer, sondern der Primärschlüssel soll aus Kategorie #89D (DB-Nummer) gebildet werden, und wenn diese nicht vorhanden ist, aus der immer vorhandenen Signatur #90. Die beste Lösung ist diese:

ak=zz+@ Dieser Sonderbefehl wird in jedem Fall ausgeführt, nur gibt es dann keine #u1
...
#-@ hier beginnt die Ausführung
#89D +# e";" p"|9DB" die DB-Nummer soll im Reg. 9 stehen mit Präfix DB
#90 p"|8" Wenn #89D nicht besetzt ist, kommt die Signatur dagegen im Reg. 8
#+#

Zwar ist es nicht zwingend, eine bestimmte Sprungmarke für den Primärschlüssel zu reservieren, es fördert jedoch die Übersichtlichkeit. Und: die Programme ***aLF*** und ***ORDER*** erwarten, anders als UPD, daß bei Sprungmarke **#-@** der Primärschlüssel gebildet wird, und daß dies ohne die Hilfskategorie **#u1** geschieht. Daher wird mit dem Befehl **ak=zz+@** gearbeitet (und nicht mit zwei Befehlen für #89D und #90).
Vorsicht: Im Abschnitt `#-@` keine Setzung von Anwendervariablen (#u-Variablen) vornehmen! Diese sind sonst beim zweiten Durchlauf nicht verfügbar, und man erhält fehlerhafte Register. *Besser:* Abschnitt `#-0` dafür benutzen.

9.4 Datenbank ganz neu aufbauen mit UPD [ab V33 obsolet]

Statt mit INDEX konnte man **bis V32** eine Datenbank auch mit UPD ganz neu aus einer Grunddatei aufbauen. Der Startbefehl sieht dann z.B. so aus, gegeben auf dem Programmverzeichnis C:\ALLEGRO:

upd -fm41 -kp -dpica -ipica -uc:\allegro\pica\pica.plg -n1 -m0 -S -L
Die Ausgangsdatei `pica.plg` auf dem Verzeichnis `c:\allegro\pica` wird dann mit den Indexparametern `pica.ppi` unter Konfiguration `p.cfg` zu einer neuen Datenbank aufgebaut. (Wenn jedoch auf `c:\allegro\pica` schon eine Datenbank namens `pica` existiert, werden die neuen Daten eingemischt.)
Allein schon wegen der viel geringeren Geschwindigkeit ist dies nicht mehr sinnvoll und **ab V33** geht es nicht mehr. Kapitel 7.1 beschreibt genau, wie es mit dem Programm INDEX gemacht wird.

9.5 Datenbanken aktualisieren mit *avanti*-Server oder *a99*

Ab V16 gibt es die Alternative, Datenbanken nicht mehr mit `upd -fm...` zu aktualisieren, sondern mit ***a99***. oder ***acon***

Der Job für `acon` muß diese zwei Zeilen enthalten, wenn eine Datei `f:\user\xy\neudat.alg` eingemischt werden soll:

```
set u41
upd f:\user\xy\newdata.alg
```

Die mit **set u41** angegebenen Ziffern entsprechen genau den in 9.1 angegebenen Werten für die Option **-fm**.

a99 kann nebenbei Funktionen eines Datenbankservers ausüben. Die Aufträge, die man ihm als solchem gibt, heißen FLEXe und sind zu schreiben in der FLEX-Sprache, die mit der ***avanti***-Sprache kompatibel ist, aber noch andere Elemente enthält. Daher kann man den gleichen Auftrag, der oben angegeben wurde, auch in einen FLEX für ***a99*** verpacken. Entweder startet man diesen interaktiv (manuelle Eingabe der Befehle), oder man übergibt ihn von außen als sog. ExFLEX. Dazu schreibt man die beiden Zeilen in eine Datei `misch.flx` und ruft dann das Hilfsprogramm FLEX.EXE am DOS-Prompt auf, indem man eingibt:

```
flex misch
```

Es gibt bei ***a99*** noch eine weitere Möglichkeit: Update zur Probe. Man setzt dann hinter den Befehl `set u..` noch die Ziffer 0, also z.B. **set u410.** Dieser Vorgang läuft dann sehr schnell ab, und im Offline-Speicher von a99 kann man sich die veränderten Sätze gleich danach ansehen (Alt+q) – in der Datenbank ist aber noch nichts passiert. Danach verläßt man a99 entweder ohne weitere Aktion, oder man veranlaßt das Programm über das Menü „Datei | alle bearb. Daten speichern“, die Daten dann echt einzumischen. Das dauert genauso lange wie bei UPD. Mehr dazu: im Schreibfeld **h xupdate** eingeben.

9.6 Besondere Möglichkeiten beim Updating

Zum Aktualisieren einer Datenbank kann man mehr tun als nur Sätze vollständig ersetzen. Dies macht man mit dem Modus `-fm11` oder `-fm10` (→ 9.1). Der Modus `-fm31` ermöglicht es, nur diejenigen Kategorien zu ergänzen, die im Datensatz noch nicht vorkommen, die vorhandenen aber unberührt zu lassen. Das ist ein eher seltener Fall.
Wenn man den Modus 41 anwendet, kann man sowohl mit dem Programm UPD wie mit dem ***avanti***-Server einige zusätzliche Möglichkeiten nutzen. Zu Teil wurden sie schon im Zusammenhang weiter oben erwähnt, aber wir stellen sie hier nochmals als Übersicht zusammen:
Für die Beispiele setzen wir voraus, daß der Primärschlüssel aus der Kategorie `#00` gebildet wird.

Datensatz löschen
Beispiel: Es soll der Satz mit `#00 1234567` gelöscht werden. Dann braucht in der Update-Datei nur zu stehen:
```
#u1 @@@@@
#00 1234567
```
Es muß nur das Feld da sein (hier #00), aus dem der Primärschlüssel gebildet wird. Wenn noch andere Kategorien angegeben sind, hat das keine Auswirkung. In der .LOG-Datei wird aber in jedem Fall der vollständige Satz als Kopie aus der Datenbank protokolliert.

Datensatz als neuen Satz einspeisen, ohne Rücksicht auf Primärschlüssel
Beispiel: Ein neuer Satz hat noch keine #00, er soll auf jeden Fall als neuer Satz eingemischt werden, und zwar in die Datei 72. Dann muß der Satz mit einer speziellen #u1 beginnen:
```
#u1 #####72
#...          [Datensatz]
```
Ab V29: Mit `#u1 ####?N` (nur 4mal #) erreicht man, daß der Primärschlüssel verglichen und nur dann der Satz als Neusatz in die Datei N kommt, wenn es noch keinen Satz gleichen Primärschlüssels gibt, ansonsten wird er normal eingemischt. #u1 verschwindet in jedem Fall.

Kategorie löschen
Beispiel: Aus dem Satz mit `#00 1234567` soll die Kategorie `#32` entfernt werden. Dann muß der Update-Satz folgendes enthalten:
```
#00 1234567
#32 _                    (Ein Unterstrich auf der ersten Textposition, sonst nichts)
#...
```
Es können noch beliebige andere Kategorien kommen, die ergänzt oder ausgetauscht werden sollen.

Teilfelder aktualisieren (ersetzen oder löschen) dies wurde schon am Ende von 9.1 behandelt, siehe oben.

Anhängen von Wiederholungskategorien
Wenn z.B. #31 die Schlagwortkategorie ist und #312 das zweite, #313 das dritte Schlagwort usw., kann das Problem entstehen: es sollen neue Schlagwörter hinzugefügt werden, vorhandene also bestehen bleiben. Dann muß man die neuen Schlagwörter so anliefern:
```
#31 ~Inhalt1...
#31 ~Inhalt2...
```
Die Tilde auf der ersten Textposition bewirkt, daß diese Inhalte auf die nächsten freien Plätze hinter die schon vorhandenen Kategorien kommen. (Bei FLEX-update kann es auch die Position direkt hinter `#31` sein.)

Parametrierte Vorbehandlung
Ein potentiell sehr mächtiges Konzept für den fortgeschrittenen *Allegrologen*. Man startet UPD mit der Option `-emanip/nul` und macht sich vorher eine Parameterdatei MANIP.APR, in der steht, was im Einzelnen passieren kann. Wie bei den Globalen Manipulationen (→ 1.5.4) muß es in MANIP.APR eine Sprungmarke `#-#` geben, unter der dann steht, was gemacht werden soll. Für jeden Satz wird dieser Abschnitt ausgeführt, und zwar unmittelbar vor dem Speichern (egal ob es ein neuer Satz ist oder ein ersetzter), d.h. nach der Nummernvergabe und Zeitstempelung.

Untersätze von hierarchischen Sätzen austauschen
Nur der avanti-Server kann die Untersätze gezielt einzeln ersetzen, das Programm UPD löscht einfach die alten Untersätze und hängt die neuen an. Voraussetzung jedoch ist, damit der Server das korrekt durchführt, daß die Bandnummern in den Kategorien `#01` übereinstimmen. Wenn im neuen Satz ein Untersatz mit `#01 47` vorkommt, wird ein Untersatz gesucht und ersetzt, der gleichfalls mit `#01 47` beginnt. Ist das nicht der Fall, wird der neue Untersatz hinten angehängt, nicht dazwischen eingeordnet.

10 Export-Parameter

***a99*: `h ac10`**

10.0 Zum Konzept (Einfache Einführung: http://www.allegro-c.de/expar.htm)

Mit **Export** wird im ***allegro***-System jeder Vorgang bezeichnet, bei dem aus den Anwenderdaten, also den Datenbank- oder Grunddateien (Typen `.ald` bzw. `.alg`) irgendeine Ausgabe produziert wird. Ganz gleich, ob diese Ausgabe auf den Bildschirm, den Drucker oder in eine Datei geht. Das Ergebnis eines Exportvorgangs wird "Exportprodukt" genannt.
Was bei einem konkreten Ausgabevorgang im einzelnen mit welchen Feldern geschehen soll, das entnehmen die Programme aus sog. **Export-Parameterdateien** (Typen `.cpr` und `.cpi`, → 0.3). Eine solche muß also existieren und geladen werden, bevor ein Programm eine Ausgabe produzieren kann.
Hinweis: Im Windows-System kann man sehr oft auf Exportparametrierung verzichten, weil die Makrosprache **FLEX** auch sehr gute Möglichkeiten bietet, Daten z.B. mit dem `write`-Befehl in Dateien auszugeben (→ Kap.2.3).

Exportprodukte und die dazu nötigen Parameterdateien lassen sich in folgende 11 Typen einteilen:

Typ	Produkt	Beispiele
DISPLAY	Bildschirmanzeigen, Katalogkarten. `d-1.apr` wird von PRESTO für die Anzeige geladen, `d-wrtf.apr` oder `d-krtf.apr` (mit `d-k.apt`) von ***a99*** für die Windows-Anzeige	`d-1.apr` `d-k.apt`
PRINT	Ausdrucke mit Seitenumbruch, Literaturlisten, gedruckte Kataloge	`p-*.apr`
INTERN	Selektierte Teilmengen (downloads) zum Aufbau neuer Datenbanken oder Einmischen in andere. Ausgabedatei vom Typ `.xlg` (Grunddatei). Das Ergebnis kann auch einem anderen Datenformat entsprechen, wie im Fall `i-neut.apr`: Wandlung von `a.cfg` nach `n.cfg`	`i-1.apr` `i-neut.apr`
FORM	Formularähnliche Ausdrucke, z.B. Bestellzettel	`f-0.apr`
SORT	Sortierfähig aufbereitete Dateien, die anschließend mit ASORT sortiert und dann mit SRCH nochmals z.B. als gedruckte Liste exportiert werden können. Ausgabe ebenfalls im allegro-Internformat (Typ `.xlg`), aber jeder Datensatz beginnt mit einem Sortierbegriff in Sonderkategorie #u1	`s-*.apr`
EXTERN	Dateien in anderen Formaten für Anwendungen außerhalb der allegro-Umgebung (z.B. Weiterbearbeitung mit Textsystemen oder sonstiger Software oder Übermittlung an andere Katalogsysteme). Wichtige Beispiele: `MARCTXT.APR` und `MARCXML.APR`	`e-1.apr`
RESULT	Rechenergebnisse, Auswertungen von Dateien oder Ergebnismengen	`r-0.apr`
ALPHA	Alphabetisches Register ausgewählter Kategorien, meist in Verbindung mit einer Listenproduktion (mit Typ PRINT)	`a-0.apr`
COUNT	Ermittlung der Häufigkeit von Kategorieinhalten, z.B. Liste der vergebenen Schlagwörter mit Anzahl der Datensätze	`c-0.apr`
INDX	Sonderfall : Registereinträge, die in den Index einer Datenbank einzuordnen sind. Auch dafür ist die Exportsprache zuständig!	`cat.api` `indx.api`
TABLE	Wird von a99 automatisch erzeugt für die Funktion **Tabellen erstellen**, Dateityp `t-*.apr`	`---`

Empfehlung zur Verbesserung der Übersicht: geben Sie Ihren Parameterdateien Namen, die mit dem Typbuchstaben und Bindestrich anfangen: eine Datei vom Typ EXTERN sollte z.B. einen Namen erhalten, der mit **`e-`** beginnt. **`e-0.apr`** ist der **Prototyp** dafür.
CockPit unterstützt die Typen SORT, PRINT und COUNT. (Diese spielen auch eine Rolle in den Paketen QUEX und QUANT, → 6.3/4.) Im Menü "Optionen" kann man je eine der vorhandenen Dateien **`s-*.apr`** und **`p-*.apr`** auswählen. Diese wird dann kopiert auf den Namen **`sort.apr`** bzw. **`print.apr`** und bei einer nachfolgenden Listenproduktion über Menü "Routinen/exportieren" benutzt. Die Windows-Programme können dieselben Parameterdateien verarbeiten, d.h. man muß keine neuen entwickeln, mit Ausnahme der Parameter vom Typ DISPLAY.
Wenn Sie darangehen, eine neue Export-Parameterdatei vom Typ **`x`** zu erstellen, können Sie sich an den Prototypen **`x-0.apr`** anlehnen. Stellen Sie also zunächst fest, zu welchem Typ Ihr Vorhaben gehört, nehmen Sie dann die zugehörige Prototyp-Parameterdatei als Muster (→ 10.1, 10.4) und wählen einen geeigneten Namen.

10.0.1 Teilbereiche des Exportverfahrens

Mit einem Export können zahlreiche Einzelaufgaben verbunden sein. Man gewinnt leichter den Überblick, wenn man das Verfahren zunächst in vier oder fünf Teilbereiche oder **Ebenen** aufgliedert.
Die Datenfelder (Kategorien) der ***allegro***-Daten können in beliebiger Auswahl und Reihenfolge, mit individueller Vorbearbeitung und bei Bedarf auch mit variablem Zeilen- und Seitenumbruch exportiert werden.
Hinweis: Die FLEX-Makrosprache der Windows-Programme ermöglicht in vielen Fällen die Programmierung eines Exports ohne Nutzung der Exportsprache. Dazu dient vor allem der FLEX-Befehl **`write`**, mit dem man Datenfelder und andere Elemente in beliebiger Folge in eine Datei schreiben kann.

Das Programm muß wissen und berücksichtigen: (vgl. das Ebenen-Modell in Kap.0.3)

auf der **Satzebene** (auch "Aufnahme-Ebene" genannt)

- ob **aus einem Datensatz mehrere Ausgabesätze** entstehen sollen oder ob in bestimmten Fällen keine Ausgabe gewünscht ist, z.B. Karten mit verschiedenen Köpfen, mehrere unterschiedliche Zugriffsschlüssel je nach Vorhandensein von Datenfeldern (jeder Schlüssel gilt als "Ausgabesatz"), etc.
- wie die **Trennung** zwischen zwei Ausgabesätzen auszusehen hat (was für Steuer- oder Textzeichen),
- wie **hierarchische Sätze** ("Stufenaufnahmen") zu behandeln sind
- in welchen Fällen und an welchen Stellen andere Sätze (z.B. **Stammsätze** oder hierarchisch über- oder untergeordnete Datensätze) mit heranzuziehen ("nachzuladen") sind

auf der **Feldebene** (auch: Kategorie-Ebene)

- **welche Felder** oder Bestandteile (**Teilfelder** etc.) überhaupt auszugeben sind, unter **welchen Bedingungen** und in welcher **Reihenfolge**,
- daß viele Felder **nicht in jedem Datensatz vorkommen**,
- daß manche **mehrfach vorkommen** können,
- ob ein Feld vor seiner Ausgabe irgendwie verändert werden muß (**Manipulationen** vorzunehmen sind),
- was für **Zeichen zwischen den Feldern** stehen sollen (Interpunktion, Steuerzeichen, sog. "Zwischenteile"),
- und daß die **Länge und interne Struktur** der einzelnen Datenfelder völlig variabel ist. Dies erfordert evtl. Bedingungsprüfungen, Zerlegungen und flexible Umbruchfähigkeiten.
- ob es Identnummern in einzelnen Feldern gibt, die vor der Ausgabe durch Klartexte ersetzt werden müssen (**Normdaten**-Verknüpfung). Das ist erst ab Version 14 möglich (→ 10.2.6.8)

auf der **Formularebene** (unterteilbar in **Seiten-** und **Zeilenebene)**

- ob **Seitenumbruch** oder ein **formular**ähnlicher Aufbau stattfinden soll
- wie bei Listen und Karten **Kopf** und **Fuß** der Seite bzw. Karte aussehen sollen.
- wie der **Zeilenumbruch** durchzuführen ist (Breite, Einrückung, Zeilenende-Steuerzeichen etc.)

auf der **Zeichenebene**

- ob **Zeichencodes** zu ändern sind, d.h. ob einzelne Zeichen umcodiert, weggelassen oder durch mehrere Zeichen ersetzt werden sollen. Diese Dinge können auch feldabhängig sein, d.h. nur bei bestimmten Kategorien nötig. Bei Druckausgaben oder bei der Produktion von RTF-Dateien für WinWord sind oft einzelne Zeichen durch sog. "Escape-Sequenzen" o.a. zu ersetzen. (Früher war so etwas z.T. sogar abhängig vom benutzten Drucker.)
- ob bestimmte Textersetzungen im gesamten Datensatz (global) oder in einzelnen Feldern (lokal) vorzunehmen, also **Zeichenfolgen** durch andere zu ersetzen sind.

Zur Veranschaulichung soll hier ein Beispiel dienen, das Sie selbst am Bildschirm nachvollziehen können (und sollten: geben Sie diesen Satz in die Demodatenbank ein):

Interne Gestalt des Datensatzes (wie man die Daten bei der Eingabe und Bearbeitung sieht):

```
#20 Einsicht ins Ich : Fantasien und Reflexionen über Selbst und Seele
#22 ¬The¬ mind's I <dt.>
#30aCS PH
#31 Künstliche Intelligenz; Selbstreferenz; Individualität
#39 Ausgewählt und in Szene gesetzt von Douglas R. Hofstadter und Daniel C. De
nnett. [Aus d. Amerik. übers. von Ulrich Enderwitz]
#40 Hofstadter, Douglas R.
#402Dennett, Daniel C.
#74 Stuttgart
#75 Klett-Cotta
#76 1986
#77 485 S.
#81 Orig. Ausg. bei Basic Books, 1982
#82 2.Verf.; EST
#87 3-608-93038-8
```

Mit den standardmäßig für die Bildschirmanzeige verwendeten Parametern D-1.APR (Windows: D-WRTF.APR) entsteht das folgende Bild:

Hofstadter, Douglas R.:
Einsicht ins Ich : Fantasien und Reflexionen über Selbst und Seele /
Ausgewählt und in Szene gesetzt von Douglas R. Hofstadter und Daniel C.
Dennett. [Aus d. Amerik. übers. von Ulrich Enderwitz]. -
   Stuttgart : Klett-Cotta, 1986. - 485 S.
ISBN 3-608-93038-8
Orig. Ausg. bei Basic Books, 1982
Originaltitel: The mind's I <dt.>
Themen:
Künstliche Intelligenz; Selbstreferenz; Individualität

Ist man damit nicht zufrieden: mit Eingriffen in die genannten Parameter kann man diese Anzeige beliebig ändern.
In PRESTO : Will man es zu Papier bringen, übergibt man es mit F2 'a' an den Drucker oder schaut sich zuerst mit F3 das Ergebnis an. Dabei entstehen zusätzlich noch Karten mit anderen Köpfen (Nebeneintragungen), z.B für den Einheitssachtitel, weil in der P-KARTE.APR mit sog. "Kopfbefehlen" (→ 10.2.1.2) dafür Vorsorge getroffen ist.

Versuchen Sie, einmal bis in die Einzelheiten gedanklich nachzuvollziehen, was alles zu tun ist, um aus der kategorisierten Form, die Sie oben sehen, die Anzeige- oder Druckform zu machen. Alles das macht das Programm jedesmal, wenn es einen Datensatz in dieser Gestalt auf den Bildschirm bringt oder ausdruckt.

Anmerkung: Ob dieses oder andere Beispiele in diesem Handbuch oder in der Demodatenbank in allen Details RAK- oder RDA-gerecht sind, darum geht es nicht. Was gezeigt werden soll, sind die Möglichkeiten des Umgangs mit Daten. Dies Buch ist nicht für KatalogisiererInnen geschrieben, sondern es soll denen eine Hilfe sein, die kreativ mit dem Datenbanksystem umgehen wollen. Wer dies vorhat, muß in der Lage sein, über den Zaun einer Norm hinauszublicken. Und wer dies gelernt hat, wird die Norm dann mit den Mitteln des Systems schon realisieren können. Und wird erkennen, daß man **auch**, aber nicht **nur**, RAK-Produkte damit herstellen kann. RDA macht keine Vorschriften über die formale Gestalt der Datenanzeige. Dazu kann die ISBD genutzt werden, wie in RAK, oder auch eine Art tabellarische Darstellung mit Textlabels. Beispiele dafür sind `d-krtf.apr` für a99, `d-kdos.apr` für PRESTO und `d-khtm.apr` für Web-Kataloge.

Hatte man in PRESTO die Parameterdatei P-NORMAL.APR geladen (mit Option **`-p p-normal`** schon beim Start oder mit **F2 F2** + Auswahl), so entsteht dagegen dieses Anzeigebild:

Hofstadter, Douglas R.:
Einsicht ins Ich : Fantasien und Reflexionen über Selbst und Seele / Ausgewählt und in Szene gesetzt von Douglas R. Hofstadter und Daniel C. Dennett. [Aus d. Amerik. übers. von Ulrich Enderwitz]. - Stuttgart: Klett-Cotta, 1986. - 485 S. Orig. Ausg. bei Basic Books, 1982. -
ISBN 3-608-93038-8

Versuchen Sie, alle Unterschiede zwischen dieser und der vorher gezeigten Anzeigeform festzustellen. Die interne, kategorisierte Form ist in beiden Fällen exakt dieselbe - die Unterschiede werden allein durch die Exportparameter bewirkt.

Zu den mitgelieferten Parameterdateien gehören die folgenden:

`d-1.apr`	Damit wird in PRESTO eine Ausgabe in Katalogkartenform nach internationalen bibliothekarischen Gepflogenheiten erstellt (sog. ISBD = International Standard Bibliographic Description). Die Köpfe für die Nebeneintragungen werden (weitgehend) nach den Vorschriften der RAK angelegt.
`d-wrtf.apr`	ist die entsprechende Datei für die Windows-Programme a99 + alcarta.
`d-*.apr`	weitere Parameter für unterschiedliche Formen der Anzeige..
`p-w.apr`	liefert einen Output in Listenform mit Seitenumbruch in Anlehnung an DIN 1505 "Titelangaben von Schrifttum" welches die deutsche Norm für die Gestaltung von wissenschaftlichen Literaturangaben ist.

Diese Beispiele machen zwar noch nicht das gesamte Potential des Exportkonzepts deutlich, vermitteln aber eine Vorstellung einiger Möglichkeiten.

10.0.2 Inhalt einer Parameterdatei

Die Anforderungen sind nun formuliert, die Problembereiche in eine Schichtenstruktur logisch gegliedert und durch Beispiele veranschaulicht. Jetzt läßt sich präzisieren, was eine ***allegro***-Parameterdatei alles enthalten muß:

a) eine Anzahl **Konstanten** für die verschiedenen Ebenen: Zeilenlänge, Einrückung am Zeilenanfang, Zeilenzahl je Karte, Anzahl Kopf- bzw. Fußzeilen je Karte, Befehl für die Druckerinitialisierung etc., und für Indexparameter kommen noch andere Werte hinzu.
Außerdem:
Steuerzeichen, die zwischen den Zeilen, Datensätzen und Seiten oder Karten anzuwenden sind.

b) **"Kopfbefehle"** legen fest, ob und unter welchen Bedingungen aus einem Datensatz mehrere Ausgabesätze zu machen sind. So z.B. mehrere Katalogkarten mit unterschiedlichen "Köpfen" (daher der Name) für alle vorhandenen Personen, Körperschaften, Schlagwörter etc.
Allgemeiner lautet das Problem: aus einem Datensatz sollen mehrere unterschiedlich gestaltete Ausgabesätze erzeugt werden; nicht nur bei Katalogkarten tritt diese Forderung auf, auch z.B. bei Eintragungen für gedruckte Register oder Zugriffsschlüsseln für einen Index. Die "Köpfe" sind dann die dafür heranzuziehenden Kategorien, mit denen jeweils etwas Spezifisches zu geschehen hat, wenn sie existieren. (→ 10.2.1.2)

c) Eine **Liste der Datenfelder** (→ 10.2.6 "Kategorieliste"), die überhaupt ausgegeben werden sollen, und in welcher Reihenfolge sie erscheinen sollen. Z.B. erscheint in obigen Beispielen nicht die #31. Sie könnte auf Wunsch an beliebiger Stelle eingefügt werden oder z.B. auch als Kopfzeile verwendet werden (→ 10.2.1.2).

Innerhalb der Liste können Sprungbefehle und Unterprogramme vorkommen, die den Ablauf situationsabhängig steuern. Die Kopfbefehle können auf Abschnitte dieser Liste verweisen.

d) Zu jedem einzelnen Datenfeld in der Liste können etliche Angaben notwendig sein:

d1. ob das Feld durch seine **Anwesenheit**, sein **Fehlen** oder sein **mehrfaches Auftreten** innerhalb eines Satzes den Ablauf der Ausgabe beeinflussen soll. In der Parameterdatei kann bei einer Kategorie #xx ein **bedingter Sprungbefehl** zu einer Sprungmarke oder einer Kategorie #yy angegeben sein. Der Befehl sagt dann dem Programm: wenn das Feld #xx vorkommt, dann gib diese aus und mache weiter bei der Sprungmarke bzw. der nächsten Zeile mit der Kategorie #yy.

d.2. Angaben über **Zeichen, die vor dem Kategorietext erscheinen sollen**: z.B. ist in D-1.APR festgelegt, daß bei #87 zuerst ein Zeilenvorschub ausgeführt und dann "ISBN " gedruckt wird, bevor der Inhalt von #87 ausgegeben wird. Beispiel: Die Zeichenfolge "<Zeilenvorschub> ISBN " gehört zu den sog. **Zwischenteilen** oder **Textkonstanten** (→ 10.2.0) und wird hier als **Präfix** (Vorspann) für #87 benutzt.

d.3. sogenannte **Manipulationsbefehle** zur Vorbearbeitung des Kategorieinhaltes vor der Ausgabe: Herausnahme von Bestandteilen, Hinzufügen anderer Teile vorne oder hinten, Ersetzungen, Längenveränderungen, rechtsbündige Ausgabe etc. (→ 10.2.6.3)
Die Möglichkeiten sind sehr umfangreich; es gibt mehr als 30 verschiedene Befehle, darunter auch Rechenoperationen sowie Funktionen zur Verknüpfung mit anderen Datensätzen (Mehrdateistruktur). Das bedeutet: andere Datensätze können während der Bearbeitung ad hoc herangezogen und Teile davon in die Ausgabe eingebaut werden.

d.4. Angaben über **Zeichen, die hinter dem Datenfeld erscheinen sollen**. Hier kann man, im Gegensatz zu d.2., dem Programm sogar mehrere Zwischenteile zur Auswahl vorgeben. Z.B. kann hinter #20 (Sachtitel) eine #40 (Verfasser) folgen, jedoch kann es auch sein, daß der Datensatz keinerlei Personendaten enthält (anonymes Werk) und sofort #74 (Ort) folgt, oder es gibt die #40, aber zusätzlich #42 (Mitarbeiter). Die Regeln verlangen im ersten Fall ein " / " als Trennung, im zweiten jedoch ". - ", im dritten muß "; " zwischen #40 und #42 und ". - " hinter die #42. Man mag das für Spitzfindigkeiten halten (was Katalogisierer selten tun), aber ***allegro*** ist fähig, auch solches zu leisten. Hinter eine Kategorie können also als **Postfix** ("Nachspann") unterschiedliche Zeichenkombinationen eingesetzt werden, abhängig davon, welche der anschließend auszugebenden Kategorien wirklich vorhanden sind.

e) Außerdem gehört zu den Druckparametern noch die Liste der in d.2. und d.3. erwähnten Zwischenteile. Dies sind maximal 100 Zeichenfolgen, die, wie oben erläutert, vor und hinter jede Kategorie eingesetzt werden können (dann als **Präfix** bzw. **Postfix** bezeichnet). Auch diese Textelemente können vom Anwender frei definiert werden (→ 10.2.0). In einem Zwischenteil können auch **Druckersteuerzeichen** zur Änderung der Schriftart enthalten sein sowie **Strukturbefehle** zur Veränderung von Zeilenlänge, Einrückung etc.

f) Zuletzt sind die **Codiertabellen** zu nennen (→ 10.2.4). Man kann sich zu jeder Parameterdatei zwei alternativ zu benutzende Tabellen anlegen, mit denen das Umcodieren der Zeichen für den Druck besorgt wird, aber auch z.B. für die Anzeige von PC-ASCII-Daten unter Windows oder UNIX. Diese Tabellen steuern die Ersetzung bestimmter Zeichen durch andere Zeichen oder durch ganze Zeichenfolgen, oder auch die Unterdrückung von Zeichen, die nicht in der Ausgabe erscheinen sollen. Die Alternativtabelle kann dazu dienen, in bestimmten Kategorien z.B. die Umlaute aufzulösen und Klein- in Großbuchstaben zu wandeln.

Bei den unter d.4, e) und f) genannten Elementen kann es sich auch um HTML-, XML- oder RTF-Tags handeln. Das hat dazu geführt, daß vorhandene Parameter schnell für WWW- oder Windows-Formatierungen genutzt werden können.

Sprungbefehle und **Unterprogramme** (→ 10.2.6.5), die den Ablauf des Geschehens strukturieren können, machen das Parametersystem von ***allegro*** zu einer Art höheren Programmiersprache, eben einer "Datenmanipulationssprache" speziell für Daten mit der Variationsbreite, die für bibliographische Daten charakteristisch ist.

Befehle zum **Nachladen** anderer Sätze (→ 10.2.6.7/8) ermöglichen viele Funktionen, bei denen der Satz allein nicht reicht, um die gewünschte Ausgabe zu erzielen, sondern dazu noch ein oder mehrere andere Sätze gebraucht werden.

10.1 Wie macht man eine Parameterdatei?

Eine Parameterdatei ist eine Textdatei (auch "ASCII-Datei" genannt). Man erstellt und bearbeitet solche Dateien mit einem beliebigen **Texteditor**. Ab DOS Version 5.0 gab es den halbwegs komfortablen EDIT. Ansonsten war der mitgelieferte Editor X sehr praktisch (Beschreibung → Anh.D, das Wesentliche lernt man in zwei Minuten). ***CockPit*** ruft standardmäßig diesen Editor auf, wenn eine Parameterdatei zu bearbeiten ist. Sie können ***CockPit*** aber anweisen, einen anderen Editor zu benutzen (→ 0.11.6). Und unter Windows? Vorsicht, Notpad u.a. arbeiten mit ANSI-Codes, es wird aber ASCII gebraucht. In einem DOS-Fenster geht immer noch der X-Editor. Ansonsten wird das freie Programm `WinVi` empfohlen, denn man kann es auf DOS-ASCII-Code umschalten.

Hinweis: Im Windows-System kann man bequem neue Textdateien im Anzeigefeld des Programms a99 schreiben und bearbeiten: Klicken Sie auf die Funktion "Neue Textdatei schreiben" im Hilfemenü "?". Dann eine erste Version im Anzeigefeld schreiben und mit dem Button [Speichern] unter einem geeigneten Namen mit Typ *.d*PR speichern.

Der Name einer Export-Parameterdatei muß vom Typ .cPR sein, wobei das '*c*' der Kennbuchstabe Ihrer Konfigurationsdatei ist (z.B. 'A', wenn Sie A.CFG verwenden, → 0.3). An dem Dateityp ".*d*PR" erkennen alle ***allegro***-Programme, um was es sich handelt.

Zur Einarbeitung nimmt man sich am besten keine zu komplizierten Aufgaben vor. Der Abschnitt 10.4 enthält einfache Beispiele ("Prototypen") zu einigen der in 10.0 vorgestellten Typen, die Sie ausprobieren und beliebig ausbauen können, denn die Beispiel-Parameterdateien werden auch alle mitgeliefert. Die umfangreichsten und kompliziertesten Dateien sind D-1.APR und CAT.API zur Erzeugung der Standard-Bildschirmanzeige bzw. des Index der DEMO-Datenbank.

Gehen Sie so vor, wenn Sie einen neuen Export oder eine neue Indexierung programmieren wollen:
(Vorausgesetzt sei, daß mit A.CFG gearbeitet wird)

1. Stellen Sie fest, zu welchem **Typ `t`** Ihr Vorhaben gehört (→ 10.0)
2. Geben Sie Ihrem Vorhaben einen bis 8 Zeichen langen **Namen** `t-`*xyz*, d.h. der Name sollte mit dem Typbuchstaben und Bindestrich beginnen, für *xyz* bleiben dann 6 Zeichen. *Sonderfall:* Eine Indexparameterdatei hat den Dateityp .API statt .APR, der Name ist beliebig, aber nur bis zu 4 Buchstaben lang.
3. **Kopieren** Sie die zugehörige **Prototyp-Parameterdatei** auf den gewählten Namen, z.B.

   ```
   copy p-0.apr p-xyz.apr
   ```

   oder
   ```
   copy indx.api xyz.api
   ```

   oder **besser**: Sie wählen im ***CockPit*** über das Menü "Dateien" die Prototyp-Datei an und aktivieren die Unterfunktion "Neue Datei vom selben Typ erstellen". Hier müssen Sie dann einen neuen Namen eingeben, dann bekommen Sie eine Kopie des Prototyps unter dem Namen *xyz*.APR zum Bearbeiten bereitgestellt. Die wesentlichen Dinge sind dann bereits vorhanden.
4. **Bearbeiten** Sie diese neue Parameterdatei mit dem Texteditor:
   - Entfernen Sie alles, was für Ihre Zwecke unnötig ist
   - Ändern Sie die Basisparameter (z.B. Zeilenlänge), soweit nötig
   - Bearbeiten oder erstellen Sie die Kategorieliste in Verbindung mit den Zwischenteilen.
     Falls Sie eine ganz neue Parameterdatei erstellen wollen:
     - schreiben Sie zunächst alle auszugebenden Kategorien in der gewünschten Reihenfolge als Liste untereinander (siehe Beispiel 1 → 10.4)
     - ergänzen Sie zuerst Präfixe, wo nötig auch Postfixe, und schließlich zur Verfeinerung und Differenzierung Manipulations- und Sprungbefehle (→ 10.2.6.1).
   - **Kommentieren** Sie alles ausführlich (Zeilen mit Leerzeichen am Anfang!)
   - **Testen** Sie Ihre Parameterdatei direkt vom Anzeigemenü aus (→ 10.3).

Nummer 4 ist der umfangreichste Teil der Aufgabe. Der Rest des Kapitels beschreibt die Ausdrucksmittel, die Sie dafür nutzen können.

Wenn Sie mit ***allegro*** nichtbibliographische Daten bearbeiten oder für bestimmte Zwecke ganz neue Dateien mit eigener Struktur erstellen wollen, dann werden Sie etwas mehr zu tun haben, bis hin zu einem kompletten Neuentwurf, der allerdings ein gründliches Studium dieses Kapitels erfordert.

10.2 Die Exportsprache

Einfache Einführung:
http://www.allegro-c.de/expar.htm

Vorbemerkungen zu diesem Abschnitt

Es folgt jetzt, nach der Einführung in das Konzept, die systematische Erläuterung aller Elemente der ***allegro***-Exportsprache. Die Reihenfolge dieser Elemente in einer Parameterdatei ist nicht vorgeschrieben (bis auf die Liste der Kategorien), aber es ist wegen der Übersichtlichkeit günstig, sich an die hier gezeigte Einteilung zu halten bzw. die der Prototypen oder anderer Beispieldateien zu übernehmen.

Es gibt in einer Parameterdatei

- 5 Gruppen von **Basis-Parametern** (entsprechend den in 10.0.1 beschriebenen Ebenen), die man auch als Rand- oder Nebenbedingungen bezeichnen könnte (→ 10.2.1 - 10.2.5)

- Eine Liste von Konstanten, die sog. **Zwischenteile** (→10.2.0). Das sind Zeichenfolgen, die aus Text-, Interpunktions- und Steuerzeichen bestehen können. Jedes hat eine Nummer, und diese Nummern benutzt man in der Kategorieliste, um die zugehörigen Zwischenteile als "Versatzstücke" (oder "Vor- und Nachspann", meistens Interpunktionszeichen) zwischen die Kategorietexte zu plazieren

- als Kernstück die **Kategorieliste**, in der die Ausgabe jedes einzelnen Datenfeldes mit speziellen Befehlen, den **Exportbefehlen**, beschrieben wird (→ 10.2.6). Jede solche Befehlszeile kann eine Anzahl von sog. **Manipulationsbefehlen** enthalten, die den Kategorietext vorbearbeiten. In der Kategorieliste können außerdem Bedingungsprüfungen, Wertzuweisungen, Berechnungen, Sprünge und Unterprogramme vorkommen.

Konventionen für Bezeichnungen

In der nun folgenden Beschreibung der Exportsprache gelten diese Bezeichnungen:

n	eine ganze Zahl >= 0
k	eine ganze Zahl zwischen 0 und 255 (ein "Byte-Wert")
z	eine ganze Zahl zwischen 0 und 127 (Nummer eines Zwischenteils, →10.2.0))
x , y	einzelne druckbare Zeichen (Ziffern, Buchstaben, Sonderzeichen)
CS	eine Steuerzeichenfolge ("character string") (Definition in 10.2.0)
#xxf	und **#yyf** sind Kategorienummern; je nach Konfiguration hat sie 2, 3 oder 4 Ziffern, ein Folgezeichen **f** tritt nur bei mehrfach besetzten Kategorien auf, z.B. `#31s`
X	eine Zeichenfolge "abc" in Anführungszeichen oder Apostrophen. Die Schreibweisen "abc" und 'abc' sind völlig gleichwertig, man kann es abwechselnd so oder so machen. Wenn allerdings darinnen ein Anführungszeichen vorkommt, muß man Apostrophe nehmen und umgekehrt. Ausgegeben wird nur, was zwischen den Anführungen steht. Nicht eingebbare Zeichen kann man mit `\nnn` verschlüsseln,. z.B. `\31` für das Zeichen ▼. und `\92` für den \.

Für jeden Basis-Parameter ist ein <Normalwert> (engl. "default") angegeben. Dieser wird vom Programm automatisch angenommen, wenn der betreffende Parameter in der Datei nicht vorkommt. Das Zeichen '*' bei einem Normalwert bedeutet, daß dieser Parameter keine Wirkung auf den Zeilenumbruch, nur auf die Druckausgabe hat.

Achten Sie auf **Groß- und Kleinschreibung**: für alle Befehle ist die im folgenden gezeigte Schreibung verbindlich.

Basis-Parameter (= Grundeinstellungen)

Alle Basis-Parameter haben zweibuchstabige Namen. Der **erste Buchstabe** des Namens bezieht sich auf die Ebene, zu der der Parameter gehört (→ 10.0.1). Es gibt dazu jeweils eine **englische Entsprechung**, die man gleichwertig benutzen kann (d.h. statt z.B. **ae** kann man auch **re** schreiben):

a	**A**ufnahme-Parameter	(**r**ecord-related parameters)	(→ 10.2.1)
s	**S**eiten- oder **F**ormularparameter	(**f**orm-related parameters)	(→ 10.2.2)
z	**Z**eilen-Parameter	(**l**ine-related parameters)	(→ 10.2.3)
d	**D**rucker- u. Codierungsparameter	(**p**rinter-related parameters)	(→ 10.2.4)
k	allgemeine **K**ategorie-Parameter		(→ 10.2.5)

Die Basisparameter sind **Einstellungen**, die während der gesamten Abarbeitung der Parameterdatei gelten. Allerdings gibt es eine Möglichkeit, einzelne Basisparameter während der Abarbeitung zu ändern (→ 10.2.6.4 Befehl `#b...`)

Exportbefehle

**Exportbefehle** stehen in Zeilen, die mit '#' oder '!' beginnen; (→ 10.2.6)

! diese Befehle bilden die **Kategorieliste** = Liste der auszugebenden Kategorien mit Angaben für die Behandlung jeder einzelnen Kategorie

Die Kategorieliste ist sozusagen das eigentliche Programm; nur diese Zeilen beschreiben wirklich Ausgabevorgänge oder stellen Anweisungen dar, die auf Daten angewendet werden.

Kommentare *äußerst wichtig! Aufpassen, weil ungewöhnlich.*

- Jede Zeile, die mit einem **Leerzeichen** oder Tabulatorzeichen **beginnt**, wird vom Programm als Kommentar angesehen und übergangen. In die erste Zeile sollte man eine Kurzcharakteristik der Datei schreiben, weil diese Zeile in der Auswahlliste (→ 0.5) als Hilfe abrufbar ist.
- *Besonders eigenartig und fehlerträchtig:*
  In jeder Parameterzeile können nach den eigentlichen Parametern auch noch Kommentare stehen, abgetrennt durch mindestens **zwei Leerzeichen** oder Tabulatorzeichen (letzteres wird nicht empfohlen). Leerzeichen innerhalb von Anführungszeichen "…" und '…' haben nicht diese Wirkung, d.h. dort sind mehrere erlaubt.

Parameter hinzuladen

Wenn gewisse Parameter in mehreren Dateien immer gleich sind (vor allem die Codiertabellen, Zwischenteile oder Stopwörter), kann man diese zu einer eigenen Datei zusammenfassen, deren Namenstyp .cPT (standardmäßig also .APT) sein muß.
Ist z.B. `printer.apt` eine solche Datei, dann wird diese mit dem Befehl

`tprinter`

den man in eine **`.apr`**-Datei schreibt, nachgeladen, d.h. in die betreffende Datei hineingenommen.
Genauer: Es können mehrere **`t`**-Befehle an beliebigen Stellen innerhalb einer Parameterdatei vorkommen, um mehrere Standardteile in eine Datei einzubeziehen, so als stünde deren Text an derselben Stelle.

Die Datei PRINTER.cPT hat eine Sonderbedeutung: man kann sie als **Druckertreiber** bezeichnen. Mitgeliefert werden die Tabellen P-EPSON.APT, P-NECP2.APT, P-DESKJET.APT und P-LEER.APT. Wenn man im ***CockPit*** über den Menüpunkt "Optionen / Drucker" eine dieser Dateien auswählt, wird sie auf den Namen PRINTER.APT umkopiert. Sie enthalten Zwischenteile mit den Nummern 79-99, die zur Druckersteuerung gedacht sind. Wenn man diese Zwischenteile konsequent verwendet, kann man Parameterdateien schreiben, die weitgehend druckerunabhängig sind. Anpassungen an andere Drucker sind nicht schwierig, denn die genannten Tabellen sind kommentiert.

Hinweis: Prototyp `p-0.apt` enthält eine Tabelle aller Zeichen des OSTWEST-Zeichensatzes. Diese Datei kann man als Vorlage für einen neuen Druckertreiber nehmen.

Wie wird eine Parameterdatei abgearbeitet?

Eine Parameterdatei wird immer als Ganzes eingelesen und bleibt dann geladen - bis zum Ende des Programmlaufs oder bis man eine andere anfordert (z.B. in PRESTO mit **F2** oder Sh+**F4** (→ 1.5), in FLEX mit `display p ...`. Dabei werden die Basisparameter und Zwischenteile registriert. Für jeden konkreten Exportvorgang wird dann die Kategorieliste durchgearbeitet (→ 10.2.6, dort wird in allen Details erläutert, wie das abläuft). Mittels des Kopfbefehls **`ak`** (→ 10.2.1) können je Datensatz mehrere Durchläufe mit jeweils unterschiedlichen Startpunkten (= Sprungmarken) in der Kategorieliste veranlaßt werden.
Die Abarbeitung konkreter Parameterdateien oder einzelner Teile davon wird an vielen Stellen auf unterschiedliche Art ausgelöst. Automatisch passiert es, wenn über den Index ein Datensatz angefordert wurde: um sie anzuzeigen, wird die (vorher geladene) Anzeigeparameterdatei durchgearbeitet. Drückt man **F7**, wird die (ebenfalls geladene) Indexparameterdatei abgearbeitet und die zum Satz gehörigen Registereinträge dann angezeigt. Drückt man **F4**, wird die (ebenfalls geladene) Exportparameterdatei abgearbeitet und der Datensatz auf diese Weise ausgegeben. Die Programme SRCH, IMPORT und INDEX benutzen die jeweils für ihre Aufgaben angeforderten Parameterdateien vollautomatisch, um ihre Produktion durchzuführen. Im Detail jedoch spielen sich immer die gleichen Dinge ab, nur mit unterschiedlichen Randbedingungen.

10.2.0 Zwischenteile

Ein **Zwischenteil** ist eine **Zeichenfolge**, die bei der Druckaufbereitung zwischen einzelne Kategorien eingefügt werden soll, z.B. die Zeichenfolge **`". - "`** als Interpunktion zwischen bestimmten Teilen der bibligraphischen Beschreibung. Es können 128 solche Zwischenteile (Nummern 0 bis 127) vom Programmbenutzer definiert werden, im Windows-Programm sogar 400, und jedes davon kann als **Präfix** vor bzw. als **Postfix** hinter jeder auszugebenden Kategorie eingesetzt werden (→ 10.2.6)
Man könnte auch von Text**bausteinen** oder Text**konstanten** sprechen, da es sich um beliebige Zeichenfolgen handelt, die an jeder Stelle in einen Ausgabetext eingefügt werden können und mehrfach verwendbar sind.

So definiert man ein Zwischenteil:

`z=CS`

`z` ist eine Zahl zwischen 1 und 127 (0 ist reserviert für die leere Zeichenfolge und darf nicht umdefiniert werden).

`CS` ist eine Steuerzeichenfolge ("**c**haracter **s**tring", siehe unten)

Beispiel:

`19=C "ISBN: "`

C bewirkt einen Zeilenvorschub, dann wird "ISBN: " hingeschrieben.

Die Zwischenteil-Definitionen können in der Parameterdatei in beliebiger Reihenfolge und an beliebiger Stelle stehen, auch am Ende: jede Parameterzeile, die mit einer Zahl beginnt, wird als Definition eines Zwischenteils angesehen. *Empfohlen wird*, alle Zwischenteile zusammenhängend und in aufsteigender Folge zu schreiben, dann bleibt's übersichtlich.
Wenn eine Parameterdatei eingelesen wird, so liest das Programm sie zunächst bis zum Ende durch und registriert alle Zwischenteile. Daraus folgt: Wenn man eine Nummer in der Parameterdatei *mehrfach* definiert, gilt nur die *letzte* Angabe.
Achtung: Die Zwischenteile sind **Konstanten**, es ist nicht möglich, Zwischenteile während des Programmlaufs zu ändern, indem man etwa Definitionszeilen zwischen die Exportbefehle schreibt. Jedoch kann man in PRESTO vom Editor aus die Werte der Zwischenteile anzeigen lassen und auch zum Testen interaktiv ändern (Befehl #p, → 10.3.1). (In *a99* geht das nicht.)

Sie finden im folgenden einige Beispiele, weitere natürlich in den mitgelieferten Parameterdateien.

Steuerzeichenfolgen (Character Strings)

Dies ist ein Konzept zur Darstellung von **gemischten Zeichenfolgen**, das für Zwischenteile und Basis-Parameter (→ 10.2.1-5) gleichermaßen anwendbar ist.
Für das Symbol CS kann in den oben und nachfolgend beschriebenen Parametern jeweils eine Zeichenfolge eingesetzt werden, die formal so aussieht:

`z1 z2 z3 ... zn`

Für jedes dieser beliebig vielen **`zi`**, getrennt durch jeweils genau ein Leerzeichen, kann eines der drei folgenden Elemente stehen:

a) eine **Zahl zwischen 0 und 255** (also ein Byte-Wert, dezimal angegeben)

b) eine **Folge druckbarer Zeichen**, zwischen Anführungsstriche bzw. Apostrophe eingeschlossen. Darin kann ein Apostroph bzw. Anführungszeichen vorkommen, also z.B. `"abc'def'ghi"`, oder auch umgekehrt: `'abc"def"ghi'`, aber nicht `"abc'def"ghi'`. Das gilt überall da, wo druckbare Zeichenfolgen anzugeben sind, also auch bei den Manipulationsbefehlen (→ 10.2.6.3).
Nicht eingebbare Zeichen kann man codiert angeben: z.B. `\31` für den ASCII-Code 31 (das Unterfeld-Dreieck) oder `\20` für das Absatz-Endezeichen ¶ .

Direkt hinter a) bzw. b) kann ein **Wiederholungsfaktor** mit '*' angeschlossen werden, z.B. ist `" --"*20` dasselbe wie 40 Bindestriche. Das erste Zeichen hinter " wird nicht vervielfältigt.

c) ein **Strukturbefehl**; dazu dienen die Steuerzeichen **`& % # $ h s t C N K`** (siehe nächste Seite) außerhalb von Anführungszeichen.

Eine Steuerzeichenfolge wird wie folgt abgearbeitet:

- eine **Zahl** wird als ein Byte an die Ausgabe übergeben
  (mehrfach, wenn ein Wiederholungsfaktor angeschlossen ist)
  Wichtig: bei der Zeilenlängenberechnung für den Umbruch zählen diese Bytes nicht mit.
  Die Zahlencodes werden normalerweisen nicht auf den Bildschirm ausgegeben, sondern nur auf Datei oder Drucker. Mit dx=1 erreicht man, daß sie auch für die Bildschirmausgabe benutzt werden. Das ist notwendig, wenn man ANSI-Steuersequenzen für Bildschirmattribute einbauen will (Beispiele in COLOR.APR, → 10.2.4).

  Sonderfälle: Die Zahl 0 wird auch als Code 0 ausgegeben (d.h. sie ist kein Sonderfall)
  Die Zahl 13 in einem Zwischenteil bewirkt einen Zeilenumbruch (wie 'C'). Unter DOS wird die Kombination 13 10 ausgegeben, unter UNIX nur 10.

- die **Zeichen zwischen Anführungszeichen** werden in den laufenden Text eingebaut (mit Umbruchbehandlung, wenn zl>0), ggfls. mehrfach hintereinander, falls ein Wiederholungsfaktor folgt.
  Sonderfall: (C-Programmierer, aufgepaßt!) die **leere** Zeichenfolge `""` bedeutet: Ausgabe entfällt, d.h. es wird **kein einziges** Zeichen ausgegeben, auch kein Code 0. Der Code 0, siehe oben, ist ein Zeichen wie andere auch (ein Byte mit dem Wert 0). Sinnvoll ist die leere Zeichenfolge bei bestimmten Basisparametern wie z.B. **`as`** oder **`ke`** (→ 10.2.1...). Zwischenteil 0 hat automatisch diesen Wert, d.h. man braucht nicht **`0=""`** zu schreiben.

- **Strukturbefehle** werden ihrer Bedeutung entsprechend ausgeführt (siehe unten).
  Wenn z.B. der Befehl %86 kommt, wird für den nachfolgenden Text bis zum Ende des Datensatzes oder einem weiteren %-Befehl mit einer Zeilenlänge von 86 Zeichen gearbeitet.

Der ASCII-Code 0 hat in ***allegro***-Grunddateien die Funktion "Feldende" (→ 0.2.2). Wenn man also Grunddateien erzeugt, wird dieser Code gebraucht. Das kommt in den Dateien des Typs **I** vor, z.B. **`I-1.APR`**.

Das Wichtige am Konzept der Steuerzeichenfolge ist, daß sie Steuerzeichen und druckbare Zeichen in beliebiger Mischung enthalten kann. Das ermöglicht eine flexible **Druckersteuerung, Farbumschaltungen** in der Bildschirmanzeige (Beispiele siehe unten), eine freie **Positionierung** im Druckbild, Änderung von Umbruchparametern mitten im Text und zudem die Einschaltung jeder gewünschten **Interpunktion** und anderer Elemente in den Text. Nur die druckbaren Zeichen innerhalb "..." werden bei der Längenberechnung für den Zeilenumbruch berücksichtigt.

Strukturbefehle : `& % # $ h s t C N K`

Diese dienen u.a. zur Änderung von Grundparametern während der Ausgabe. Jeder Bestandteil zi einer Steuerzeichenfolge kann ein solcher Strukturbefehl sein.
Einschränkung: Die Befehle `& % # $ s` können nur in Zwischenteilen, nicht in Basisparametern vorkommen. Oder anders ausgedrückt: in Basisparametern dürfen nur die Buchstabenbefehle vorkommen.
Beispiele für die einzelnen Befehle: (beachten Sie: keine Leerzeichen zwischen Befehlszeichen und den zugehörigen Zahlenwerten)
Achtung: Die ersten drei (% # $) funktionieren nicht, wenn `zl=0` gesetzt ist.

`%100` Umstellung der **Zeilenlänge** (Parameter zl) auf 100 Zeichen (max.120). Dabei zählt die Einrückung mit:

`#10` Änderung der **Einrückung** (zi) auf 10 Zeichen ab der nächsten Zeile; wirksam nach dem nächsten Zeilenvorschub. Wenn die Einrückung größer ist als die aktuelle Zeilenlänge (z.B. #250), wird sie auf die aktuelle Cursorposition eingestellt. Dadurch kann die Einrückung variabel gestaltet werden.
Einrückung abschalten: `#0`.

`$4,30` **Positionierung** auf Zeile 4, Zeichen 30 ($ allein : HOME-Position = $0,0) d.h. es wird anschließend auf dieser Stelle weitergedruckt, wobei die aktuelle Einrückung berücksichtigt, d.h. zu 30 addiert wird. Damit hat man Möglichkeiten zur Gestaltung von Formularen. Die zweite Zahl kann fehlen, wenn die Anfangsposition der Zeile anzusteuern ist, also z.B. $4 statt $4,0 .
Der Strukturbefehl $ ist auch für folgenden Fall geeignet: man will mehrere kurze Angaben (Signatur, Notationen etc.) am linken Rand der Karte unterbringen und den eigentlichen Text mit einer gewissen Einrückung daneben. Dann definiert man z.B. ein Zwischenteil 35=%10 #0 und setzt dieses als Präfix vor die betr. Kategorie (z.B. #90). Vor die nächste Kategorie, die dann wieder auf der oberen Zeile, aber eingerückt erscheinen soll, setzt man z.B. das Präfix 36=%56 #15 $. Dieses bewirkt, daß die Zeilenlänge auf 56 eingestellt wird und die Einrückung auf 15, und der nachfolgende Text beginnt dann in der ersten Zeile und läßt die links schon stehenden Angaben unberührt, da die Einrückung ein Überschreiben verhindert. *Nebenwirkung:* Wenn man $ benutzt, wird die Einrückung auf 1 gesetzt, falls sie auf 0 steht.

`&18` **Tabulatorsprung** auf Position 18 der laufenden Zeile; Achtung: wenn die aktuelle Zeile zum Zeitpunkt dieses Befehls schon länger ist, wird der Teil ab Position 18 überschrieben. Zeilenanfang: `&1` .
Falls `zl=0` ist, hat dieser Befehl nur Sinn bei den Indexparametern, denn ein Registereintrag gilt dann als Zeile sowohl als auch Ausgabesatz und kann nur bis zu 240 Zeichen lang sein. (Einstellung der max. Länge mit `il=`...)

`h0` **Hierarchie-Code**
Eine Hauptaufnahme (Stufe #00) hat den Hierarchie-Code 1, auf Stufe #01 ist es der Wert 2, usw. Dieser Code wird an der aktuellen Schreibposition ausgegeben, sobald der Strukturbefehl h kommt. Folgt dem h eine Zahl ungleich 0, so wird der Hierarchiecode auf diese Zahl aufaddiert. Z.B. würde h47 bewirken, daß statt des Byte-Wertes 1 der Wert 48 ausgegeben wird, und das ist der ASCII-Wert der Ziffer '0'.
(Benötigt wird dieser Strukturbefehl hauptsächlich für Exporte in andere Formate, aber auch für das Grundformat: siehe dazu I-1.APR und S-*.APR)

`s4` Basisparameter `ks` (→ 10.5.2) ändern. Von diesem Punkt ab werden die Kategorietexte ab dem 4-ten Zeichen ausgegeben, bis ein neuer **`s`**-Befehl kommt oder der nächste Satz bearbeitet wird (dann wird der Parameter `ks` auf den Ausgangswert zurückgesetzt).

`t29` **Verschachtelung** : Zwischenteil Nummer 29 wird in das aktuelle Zwischenteil eingebaut. Das ist z.B. nützlich, wenn man z.B. Steuersequenzen für Drucker oder Bildschirm (Schriftart- bzw. Farbumstellung) in mehreren Zwischenteilen benötigt. Man legt diese Sequenzen dann auf bestimmte Nummern und verwendet sie an beliebiger Stelle mit Hilfe dieses Strukturbefehls.

Sondercodes:

C **Zeilenvorschub** ("Carriage Return", in Zwischenteilen gleichwertig mit dem Code 13).
Sobald dieser Code auftritt, wird eine neue Zeile begonnen. Anstelle des 'C' wird der Wert des Parameters **ze** ausgegeben (Zeilenende-Steuerzeichen → 10.2.3).

N **Neue Seite** : Sobald dieser Code auftritt, wird eine neue Seite begonnen, was eine komplizierte Aktion ist: es wird zuerst der Fußabschnitt ausgeführt, dann der Parameter **sv** ausgegeben, dann der Kopfabschnitt ausgeführt, und dann der Export fortgesetzt (→ 10.2.6.5.3). Ab Version 12.2 wirkt dieser Befehl bei Kartendruck als Startbefehl für eine **Fortsetzungskarte**.
Ansonsten ist der Seitenumbruch ein völlig automatisch ablaufender Vorgang, den man durch die Parameter `z1, z2, zm, sz` und den selbstdefinierten Fußabschnitt beeinflussen kann. Der Code N braucht nur dann benutzt zu werden, wenn an einer bestimmten Stelle ein Seitenumbruch erzwungen werden soll.

K **Nummer der Fortsetzungskarte**. Sinnvoll ist dieser Befehl praktisch nur in dem Grundparameter **fa** (→ 10.2.2, Beispiel siehe dort), um an bestimmter Stelle eine Kartennummer anzubringen.

Anmerkungen

- Zeilenlänge, Einrückung und Textbeginn werden automatisch auf die Werte `zl, zi` bzw. `ks` (→ 10.2.3/5) zurückgesetzt, jedesmal nachdem ein Satz fertig ausgegeben ist. Man muß also nicht mit einem eigenen Befehl in einem abschließenden Zwischenteil dafür sorgen, daß dies geschieht.

- Sollte der Drucker durch eins der Zwischenteile in seiner Einstellung beeinflußt worden sein, so wird er nicht automatisch in den Normalmodus zurückgesetzt! Wenn man z.B. als letztes Datenelement ein Abstract (#98 oder #99) drucken und dafür den Drucker auf Schmalschrift umstellen läßt, muß diese Einstellung (am besten durch das auf #98 folgende Postfix) wieder rückgängig gemacht werden. Man kann auch in dem Parameter `as`, der vor der eigentlichen Druckausgabe dem Drucker übermittelt wird, die Normaleinstellung vornehmen, dann kann nichts schiefgehen.

- Die Reihenfolge der Strukturbefehle kann wichtig sein: z.B. könnte `#50 %76` nicht ausgeführt werden, wenn vorher die Zeilenlänge auf 45 steht; es müßte umgekehrt `%76 #50` heißen, sonst wirkt #50 nicht.

***Beispiele* für Zwischenteil-Definitionen und Steuerzeichenfolgen:**

2=" / "	Leerzeichen, Schrägstrich, Leerzeichen
13=C " "	Zeilenvorschub, dann drei Leerzeichen (d.h. z1=C, z2=" ")
67=C "ISBN "	Zeilenvorschub, dann "ISBN "
18=0	Der ASCII-Code 00 (ein einzelnes Byte mit dem Wert 0) wird ausgegeben. Das ist zu unterscheiden von der "leeren Zeichenfolge", die aus überhaupt keinem Zeichen besteht:
17=s0	Wenn dieses Zwischenteil benutzt wird: ab der nächsten Kategorie wird der gesamte Text, einschließlich '#' und Kategorienummer, ausgegeben. Manchmal wird für gewisse Kategorien auch deren Nummer gebraucht, dann ist das sehr nützlich.
88=27 52	<Escape> '4' (bewirkt Umschaltung auf Kursiv bei Epson-kompatiblen Druckern) Vorsicht: **27 "4"** würde auch gehen, aber die '4' würde in die Zeilenlängen-Berechnung einbezogen!

9=27 52 '"Anmerkung:" ' 27 53 27 103
10=27 77

Damit würde die Befehlszeile `#81 p9 P10` auf einem EPSON-Drucker dieses produzieren:
"Anmerkung:" Text der Kategorie #81 in Schmalschrift (wegen 27 103), die nachfolgende Kategorie würde wieder in Normalschrift erscheinen wegen 10=27 77 = Elite-Schrift.
Weil man eine Menge ändern müßte, wenn man solche Dinge auf andere Drucker übertragen will, gibt es standardisierte Zwischenteile in den Dateien P-*.APT, den sog. Druckertreibern. Wenn man einen davon auf PRINTER.APT umkopiert und ihn dann mit dem Befehl `tPRINTER` in die Parameterdatei einbindet, kann man schreiben

9=t88 '"Anmerkung:" ' t87 t81

und erreicht mit `#81 p9 p82` dasselbe, aber druckerunabhängig. Benutzt sind hierbei die Zwischenteile 88 (Kursiv ein), 87 (Kursiv aus), 83 (Schmalschrift) und 82 (Normalschrift) definiert mit den Codes für den jeweiligen Drucker. Siehe auch Anmerkung unten.

43="-.-"*5 erzeugt die Folge **"-.-.-.-.-.-"**

31=&10 "(" Sprung auf Position 10 und dort Ausgabe des Zeichens '('

71=27 91 49 59 51 49 59 52 55 109

"ANSI"-Befehl für Farbumschaltung: rot (1) auf weiß (7) (= `Esc[1;31;47m`)

77=27 91 49 59 51 55 59 52 52 109

... weiß(7) auf blau(4) (`Esc [1;37;44m`, siehe D-1.APR für die anderen Farbcodes)
Achtung: wenn man `77=27 "[1;37;44m"` schreibt, bewirkt das dieselbe Farbeinstellung, aber die Zeichen in **"."** werden beim Umbruch mitgerechnet, es passen weniger Zeichen in die Zeile!
Zu den Farbcodes Siehe auch → Anh. A.3 (Grundeinstellung der Farben in der CFG)

26=$1,60 t71 " <Signatur:> " t77

Positioniere auf Zeile 1, Zeichen 60, gib in rot auf weiß den Text " <Signatur:> " aus und schalte auf weiß auf blau zurück.
(Man sieht hier, daß gerade die Farbumschaltungen ein gutes Beispiel für solche Zwischenteile sind, die man mit dem t-Befehl in andere Zwischenteile einbauen kann.)

ae="99 ENDE" 0 13 10

schreibt ans Satzende beim Export die Kategorie "99 ENDE" und setzt korrekt den Code 0 sowie 13 und 10 dahinter, wie es bei einer Ausgabe im .ALG-Format sein muß.

22=". " #250 C

Es wird ein Punkt mit Leerzeichen dahinter gemacht. Dann wird die Einrückung auf die Position auf die aktuelle Position, also das Zeichen hinter dem Leerzeichen gestellt und in die nächste Zeile gegangen. Dort beginnt der nachfolgende Text auf der gerade vorher eingestellten Position! Der Rest des Satzes wird mit dieser Einrückung ausgegeben, wenn nicht ein weiterer #-Befehl folgt. Vor Beginn des nächsten Satzes wird der Wert wieder auf zi gestellt.

Druckertreiber: Die Nummern 79 - 99 sind in den mitgelieferten Parameterdateien für die Einstellung von Druckattributen vorgesehen, z.B. 99 = Fettschrift ein, 98 = Fettschrift aus usw. Es handelt sich um die Dateien **P-EPSON.APT**, **P-NECP2.APT** usw., die sog. "Druckertreiber". Über ***CockPit*** **µo d** wählt man davon einen aus, der dann auf **PRINTER.APT** umkopiert wird. Mit dem Befehl `tPRINTER` bindet man dies ein in die .APR-Datei, wo die Nummern benutzt werden sollen. Es genügt dann, in der Kategorieliste z.B. etwas wie `#xx p99 P98` zu schreiben, damit `#xx` fett gedruckt wird. So gestaltet man Parameterdateien druckerunabhängig.
Die Dateien P-*.APT sind unabhängig von der Konfiguration, d.h. man kann sie ohne weiteres z.B. auf **.MPT** umkopieren und hat damit die entsprechenden Druckertreiber für M.CFG.
Die Nummern 71 - 78 werden meistens für Bildschirmfarben benutzt (siehe D-1.APR). In Druckparametern würden diese stören, auch bei Ausgabedaten für Windows oder UNIX.
Dieses Konzept der Codierung von Schriftattributen erstreckt sich auch auf die Ausgabe von Daten für andere Oberflächen. Es gibt die Pseudo-Druckertreiber P-HTM.APT für HTML-Codierung (Einsatz an **WWW**-Schnittstellen) und P-RTF.APT für **Windows**-RTF-Ausgabe (Einsatz in ***a99/alcarta***). Dabei gelten dieselben Nummern wie für die Druckausgabe, z.B. steht dann in P-HTM.APT für das Ein- und Ausschalten der Fettschrift: `99="<b>"` und `98="</b>"`, in P-RTF.APT steht `99="{\b "` und `98="}"`. Der Vorteil: Hat man Parameter mit diesen einfachen Konventionen entwickelt, kann man sie leicht für andere Umgebungen umschreiben – es muß kaum mehr getan werden als eine andere .APT einzusetzen.

10.2.1-5 Basis-Parameter

10.2.1 Aufnahme-Parameter [beziehen sich auf den Datensatz (= Aufnahme) als Ganzes]

10.2.1.1 Konstanten

ab=CS **Aufnahme-Beginn** : Zeichen, die **vor dem Beginn** jeder Hauptaufnahme bzw. <>*

as=CS **Aufnahme-Start** : vor jeder Unteraufnahme (**subrecord**) geschrieben <>*
werden sollen, also **zwischen alle Teilsätze** einer Gruppe. Wenn **ab** nicht definiert ist, wird **as** auch vor der Hauptaufnahme ausgegeben. Wenn man das verhindern will, wenn also vor der Hauptaufnahme nichts stehen soll, muß man **ab=""** setzen.

ae=CS **Aufnahme-Ende** : Zeichen, die am **Ende** jeder Aufnahmegruppe <>*
(hinter Hauptsatz plus Untersätze) geschrieben werden sollen

ad=1 **Aufnahme exportieren auch ohne Kopf?** (**A**usgabe-**D**efault)

Wenn keine der in **ak** (→ 10.2.1.2) angegebenen Kopfkategorien vorkommt, soll dennoch ein

Ausgabesatz produziert werden, bzw. <1>

=0 **keine Ausgabe, wenn kein Kopf** : es soll in dem Falle keine Ausgabe erfolgen. Z.B. wird man in einem Namensregister keine Eintragung für Aufnahmen haben wollen, in denen gar kein Name vorkommt. Dann ist **ad=0** sinnvoll.

Wenn es **keine ak-Zeilen** gibt, wird für jeden Satz genau ein Ausgabesatz produziert.

ag **Aufnahme-Gliederungsmodus:** <1>

=0 Die untergeordneten Sätze einer hierarchischen Aufnahme (Bände von mehrbändigen Werken) sollen als selbständige Sätze (unabhängig vom Hauptsatz) behandelt werden (z.B. bei der Produktion von Registern oder Stücktitelkarten). D.h. es wird so getan, als ob die untergeordneten Sätze Hauptsätze wären: für jeden werden die vorhandenen Köpfe ermittelt (s.u., Parameter ak) und die entsprechenden Exportsätze produziert. Der Unterschied: nicht vorhandene Kategorien werden auf den nächsthöheren Stufen gesucht, bis hin zur Stufe #00, und von dort entnommen. So werden Stücktitelaufnahmen automatisch "zusammengebastelt". Siehe dazu auch die Sonderkategorien **#gt** und **#hi** (→ 10.2.6.2).
Hinweis: Wenn man in der Kategorieliste das Zeichen **'^'** vor eine Kategorienummer setzt, also z.B. **#^20** statt **#20**, wird die betreffende Kategorie in jedem Fall aus der Hauptaufnahme entnommen, ob sie nun auf der gerade bearbeiteten Stufe vorkommt oder nicht (→10.2.6.1).

=1 Untergeordnete Sätze werden nicht als eigene Datensätze abgearbeitet, d.h. sie erzeugen keine eigenen Exportsätze, sondern werden nur als "Bandaufführung" hinter den Hauptsatz gehängt. Anzuwenden ist dies z.B. bei der Produktion von Listenkatalogen. Die Bearbeitung geht so vor sich, daß für jeden Kopf, der sich aus der Hauptaufnahme ergibt, ein Exportsatz erstellt wird, an den sich für jeden Band eine Bandaufführung anschließt. Für jeden Band wird dazu die Kategorieliste vom Anfang an durchgearbeitet. Wenn man dort die Kategorie **#hi** verwendet, kann man mit einem bedingten Sprung eine Sonderbehandlung der Bandaufführungen programmieren. (→ 10.2.6; vgl. dazu D-1.APR und P-NORMAL.APR).

=2 Es wird beides durchgeführt: zuerst die Hauptaufnahme mit Untersätzen ausgegeben, dann die Untersätze nochmals als selbständige Sätze abgearbeitet. Beispiel: bei Kartendruck können sowohl Bandaufführungen als auch automatisch zusammengesetzte Stücktitelaufnahmen (= eigene Einheitsaufnahmen) entstehen.

am=n **Maximale Anzahl Exportsätze** je Aufnahme <0>
Begrenzung der Anzahl auszugebender Sätze, die sich aufgrund der **ak**-Befehle ergeben würden.

=0 **ohne Begrenzung**
Normalerweise werden alle **ak**-Definitionen abgearbeitet, d.h. jeder produzierbare Ausgabesatz wird auch ausgegeben. Wenn man dies einschränken will, muß man **am** auf den gewünschten Maximalwert setzen. Insbesondere bei der Herstellung einer Sortierdatei (→ Kap.6), die nur Haupteintragungen enthalten soll, ist es notwendig, **am=1** zu setzen. Dann müssen in **ak** die Definitionen in der für Haupteintragungen richtigen logischen Reihenfolge stehen.

an=n **Aufnahme-Nummer** der ersten Aufnahme (siehe 10.2.6.2: Sonderkat.), wenn <1>
Sonderkategorie #nr benutzt wird. (Automatische fortlaufende Zählung der Exportsätze.)

10.2.1.2 Kopfbefehle

ak **Kopfbefehl**. Dies ist der **wichtigste Parameter** überhaupt - jedenfalls in den Fällen, wo aus einem Datensatz evtl. mehrere Ausgabesätze entstehen sollen.
Seine Aufgabe: Ein Kopfbefehl veranlaßt die Produktion eines oder mehrerer Exportsätze, abhängig von der Existenz bestimmter Kategorien oder Bestandteile davon. Es können ca. 100 Kopfbefehle in einer Parameterdatei auftreten. Man kann sie beliebig zu mehreren auf einer Zeile zusammenfassen (siehe unten).
Z.B.
- Herstellung einer Katalogkarte für jede Kategorie, die für Eintragungen im Zettelkatalog heranzuziehen ist;
- Produktion je eines Sortiersatzes für jede von den Regeln geforderte Eintragung;
- Erzeugung einer Indexeintragung für jede relevante Kategorie.

Wenn ein Satz keine der per **ak** definierten Felder enthält, entsteht genau ein Ausgabesatz. Wenn aber **ad=0** ist, dann keiner (→ 10.2.1.1).
Wenn **keine ak-Befehle** vorhanden sind, wird für jeden Datensatz die Kategorieliste genau einmal abgearbeitet, und zwar natürlich am Anfang beginnend.

Ein vollständiger Kopfbefehl sieht schematisch so aus: (zur Abarbeitung → 10.2.6)

ak=xxf"xyz"/mmf+M

Hinter `ak=` können mehrere gleichartige Angaben stehen, getrennt durch je ein Leerzeichen.
Alle Elemente außer der Kopfkategorienummer **xx** können entfallen.
Die einzelnen Angaben bedeuten folgendes:

xxf Hauptkopf (Kategorienummer); Folgezeichen f kann entfallen.
Das zweite 'x' und das 'f' können durch '.' ersetzt werden. Z.B. bedeutet
`ak=4.` : jede Kategorie der Gruppe #4 soll eine Eintragung erzeugen
`ak=40.` : #40 und alle Mehrfachbesetzungen von #40 sollen je eine Eintragung erzeugen.

/mmf Zusatzkopf, `/mmf` kann entfallen; hier kann kein '.' als Ersatzzeichen benutzt werden (f kann auch hier wegfallen). Wenn **mmf** angegeben ist, wird der ak-Befehl nur ausgeführt, falls **#xxf und mmf** beide in dem aktuellen Datensatz vorkommen.

Für den Hauptkopf können in Anführungszeichen **Trennbefehle** gegeben werden, wenn eine Kategorie mehrere gleichartige Inhalte enthalten kann, die alle separat verarbeitet werden sollen:

"xyz" Zeichenkombination zur Zerlegung von #xxf: die Kategorie #xxf wird jeweils da auseinandergenommen, wo "xyz" steht, und jeder solche Bestandteil erzeugt dann einen eigenen Ausgabesatz. Wenn z.B. mehrere Schlagwörter in #31 enthalten sind und " ; " als Trennung verwendet wurde, würde durch ak=31" ; " eine separate Verarbeitung der einzelnen Schlagwörter erreicht.
Innerhalb **xyz** kann '?' als Jokerzeichen auftreten. **" ? "** würde bewirken, daß jedes einzeln stehende, von Leerzeichen eingeschlossene Zeichen als Trennung gilt.
Codierung nicht-eingebbarer Zeichen: mit `\nnn`., z.B. `\31` für das Unterfeld-Dreieck.

Es gibt hier noch zwei ähnliche Möglichkeiten:

"\xyz" dasselbe, aber der erste Bestandteil der Kategorie wird nicht berücksichtigt. Sinnvoll ist dies, wenn man z.B. Titelstichwörter und zusätzlich den Titelanfang in den Index einbringt.

"[abc]" anders als bei "xyz" wird durch diesen Befehl die Kategorie bei jedem der einzelnen Zeichen a b und c auseinandergenommen. Dies ist bei der Indexgenerierung wichtig, wenn man Titel in Stichwörter zerlegen will. Die Angabe `ak=20"[ ,.'-¶]"` würde bewirken, daß der Titel in Einzelwörter auseinandergenommen wird, wobei die in den eckigen Klammern stehenden Zeichen als Trennung gelten. Und `ak=20"\[ ,.'-¶]"` würde dasselbe machen, nur daß das erste Wort unberücksichtigt bliebe (siehe oben). Codierung nicht-eingebbarer Zeichen auch hier mit `\nnn`, z.B. `\31` für das Unterfeld-Dreieck.

+M **Sprungmarke**: die Verarbeitung beginnt für diesen Kopf bei der Sprungmarke **#-M** innerhalb der Kategorieliste (→ 10.2.6, 10.2.6.5.1).
Wenn **+M** fehlt, wird die Verarbeitung mit der ersten Zeile der Kategorieliste begonnen.
M muß ein **einzelnes** alphanumerisches Zeichen oder Sonderzeichen sein, kein Wort.
Empfehlung: verwenden Sie hier vorzugsweise Großbuchstaben und Ziffern, für Sprünge innerhalb der Kategorieliste dagegen Kleinbuchstaben (→ 10.2.6.1, 10.2.6.5.1)

ak=zz+M Dieser Sonderbefehl wird **in jedem Fall** ausgeführt, liefert aber keine Sonderkategorie `#u1`. Sinnvoll ist er, wenn Abschnitt `#-M` unter allen Umständen durchgeführt werden soll.

ak=uxy+M mit Anwendervariablen `#uxy` ist möglich (→ 10.2.6.3). Natürlich muß man vorher, in einem anderen Abschnitt, die Variable `#uxy` produziert haben, sonst kommt nichts Sinnvolles heraus. Anwendbar ist dies z.B., wenn beim Indexieren nur der Hauptsachtitel, nicht aber der Zusatz ins Stichwortregister soll: man kopiert in einem Abschnitt, der vorher drankommt, den HST in #uhs (mit `#20 dhs e" : " ahs`); der Kopfbefehl lautet dann `ak=uhs" "+C`.

Die Bezeichnungen "Hauptkopf" und "Zusatzkopf" dürfen nicht wörtlich so verstanden werden, daß sie auch zwangsläufig als Überschrift oder an erster Stelle des Exportsatzes erscheinen. Die Namen stammen aus den Anfängen der Entwicklung, als der Kartendruck im Vordergrund stand.
Die Kategorie #xxf bzw. ihr durch die Zerlegung erzeugter Bestandteil wird vor dem Ansteuern der Sprungmarke einer **Anwendervariablen** mit Namen **#u1** zugewiesen. Wenn ein Zusatzkopf #mmf vorkommt, wird daraus die Anwendervariable **#u2.** Diese können dann (aber müssen nicht!) als Sonderkategorien in der Kategorieliste an beliebiger Stelle verwendet werden, nicht nur als Kartenkopf (→ 10.2.6.2).
Im Editor werden die Kombinationen der ak-Liste von D-1.APR als Standard-Köpfe für den Kartendruck benutzt, wenn man F2 'a' oder F3 'a' gibt.
Der Parameter **ak** ist beliebig wiederholbar, jeweils auf neuer Zeile. (Die Zeilen können an beliebigen Stellen stehen und durch andere Teile getrennt sein.) Es können aber auch mehrere Kopfbefehle auf einer Zeile zusammengefaßt werden. D.h. hinter **ak=** darf eine Kette von Angaben stehen, und die einzelnen Angaben müssen durch **genau ein Leerzeichen** getrennt sein, z.B.

```
ak=40+a 20+b 87+i 31" ; "+W 22/40+e 21/40+S
```

(Zwei oder mehr Leerzeichen hätten zur Folge, daß der Rest wegfällt!)

Zur Arbeitsweise des Programms

Es werden normalerweise alle ak-Definitionen in der vorliegenden Reihenfolge abgearbeitet, und jedes in dem Datensatz vorkommende, zum ak-Befehl passende Feld führt zur Produktion eines Ausgabesatzes. Wenn man die Anzahl der Ausgabesätze beschränken will, geht dies mit dem Parameter **am** (siehe oben). Man wird insbesondere **am=1** setzen, wenn man eine Liste herstellen will, die nur *eine* Eintragung je Satz enthält.
Bei Indexparametern hat der erste **ak**-Befehl eine herausgehobene Bedeutung: er definiert den **Primärschlüssel**, den das Programm UPD für die Identifizierung eines zu ersetzenden Satzes braucht (→ Kap.9).

Zur Mehrfachbesetzung von Kategorien

Wenn eine mehrfach besetzte Kategorie z.B. als Kartenkopf heranzuziehen ist, dann sollen im allgemeinen mehrere Karten entstehen, je eine für jede Besetzung. Es gibt zwei Fälle, hier am Beispiel #40 erläutert:

- man erfaßt die verschiedenen Namen alle in #40, trennt sie aber z.B. immer durch "; " (Semikolon Leerzeichen).
  Statt z.B. `ak=40+V` schreibt man dann `ak=40"; "+V` . Für jeden der in #40 angegebenen Namen wird dann eine eigene Karte gemacht.
- man arbeitet mit Folgezeichen: z.B. `#40` `#402` `#403` usw. für Verfasser.
  Dann gibt man nicht `ak=40+V`, sondern `ak=40.+V` an, und das Programm wird alle `#40`er-Kategorien hintereinander bearbeiten.
- Gibt man, noch radikaler, `ak=4..+V` , so werden auch `#41, #42` usw. genommen, jeweils mit allen Mehrfachbesetzungen, und bei `ak=4.."; "+V` werden alle betroffenen Kategorien auch noch zerlegt. Das ist die knappste Formulierung, um die gleichartige Verarbeitung aller Namenskategorien zu erwirken. Solche Zusammenfassungen sind anzustreben, denn sie arbeiten schneller als eine Vielzahl von Einzelbefehlen.

Diese Verfahren sind anzuwenden bei der Herstellung von Exportdateien und Indexdateien (→ Kap.4 und 7), wenn ein Datensatz z.B. mehrere Eintragungen in einer Liste bzw. in einem Index erhalten soll und die betreffenden Elemente sich in Mehrfachkategorien befinden oder innerhalb einer Kategorie durch Steuerzeichen getrennt sind.

10.2.1.3 Spezialparameter für die Indexproduktion

Eine Indexparameterdatei hat immer den Typ .API statt .APR, die Sprache ist jedoch dieselbe. Mitgelieferte Beispiele sind INDX.API und CAT.API. Darin sehen Sie die Anwendung der Parameter.
Als "Ausgabesatz" muß bei der Schlüsselgenerierung jeder einzelne Zugriffsschlüssel angesehen werden, da diese völlig unabhängig voneinander in den Index eingeordnet werden.
Das gesamte Parameter-Instrumentarium und insbesondere die Befehle zur Umcodierung und Vorbearbeitung sind auch für die Produktion von Zugriffsschlüsseln einsetzbar. Indexeintragungen können daher bei ***allegro*** aus beliebigen Teilen eines Datensatzes bestehen und auch aus mehreren Elementen zusammengesetzt werden. Und: man braucht für die Datenbankdefinition keine eigene Sprache oder Prozedur zu erlernen, sondern die Exportsprache erledigt auch diese Aufgabe!
Der Vorgang der Indexerstellung wird in Kap. 7 genau beschrieben; hier werden nur die dafür nötigen Spezialparameter näher erläutert. Beispiele für alle findet man in CAT.API.
Im Prinzip muß man die folgenden Werte, die nur bei der Indexproduktion relevant sind, zu den Aufnahmeparametern rechnen. Sie haben aber zur Unterscheidung den Anfangsbuchstaben 'i':

i0=k **Länge der Kurzanzeige** <0>

Der Wert k kann ein anderer sein als beim Befehl **il** (siehe unten), aber ein größerer Wert als 72 ist nur für ***avanti*** und ***a99/alcarta*** sinnvoll. Die Kurzanzeige wird dann auf diese Länge festgesetzt.
Die Aufbereitung der Kurzanzeige muß bei der dafür reservierten Sprungmarke **#-0** in der Kategorieliste definiert sein, und ein Befehl **ak=zz+0** muß dafür existieren. Dort muß ohne #u1 gearbeitet werden (Beispiel → CAT.API). Nachdem man in einer älteren Parameterdatei solche Vorkehrungen getroffen hat, muß man per ***CockPit*** **µr o k** das Kurztitelregister anlegen lassen. Erst dann funktionieren **Shift+F8** und **Shift+F9**.

i1=CS **Trennzeichen** für zweiteilige Schlüssel <"=:>➢➢">
die in CS angegebenen Zeichen sollen als Trennzeichen gelten. Das bewirkt insbesondere: der erste Teil wird untersucht, ob es sich um ein Stopwort handelt. In diesem Fall wird der gesamte Schlüssel nicht einsortiert. Maximum für die Länge von CS: 10 Zeichen

i2=CS **Verbotszeichen** <58 61 62 32>
Wenn das erste Zeichen eines Schlüssels in CS vorkommt, wird dieser Schlüssel nicht einsortiert. Wichtig bei kombinierten Schlüsseln, wenn es sein kann, daß der erste Teil entfällt. Maximale Länge von CS: 10 Zeichen. *Empfehlung:* für i2 ASCII-Codes angeben, nicht Zeichen in "..."

i3=CS **ungültige Endzeichen** <"., ;:)=">
CS enthält Zeichen, die am Ende von Schlüsseln automatisch beseitigt werden sollen. Die meisten Sonderzeichen wird man zwar über p-Befehle ausblenden (→ 10.2.4), aber z.B. das Komma soll in der Regel, weil es in Namen vorkommt, auch im Index erhalten bleiben, nicht aber am Ende eines Schlüssels. Dasselbe kann für Klammern gelten.
Im übrigen kann man mit dem Manipulationsbefehl F gezielt auch einzelne andere Zeichen bei bestimmten Schlüsseln am Ende wegnehmen (→ 10.2.6.3).

i4, i5, i6 : Diese werden in 10.2.6.8 behandelt (Stammsatz-Ersetzungen)

i7=ki **Nummer des SR-Hilfsregisters** (für satzübergreifennde Suche; → 10.2.6.9)

i8=CS **Verweisungszeichen** <">➢">

i9=CS Wenn eines der Zeichen von i8 in einer Registerzeile vorkommt, löst die Betätigung der <Enter>-Taste auf dieser Zeile ein Umblättern auf den Registereintrag aus, der hinter dem Zeichen steht. Für `i9` gilt dasselbe, jedoch mit der <Cursor rechts>-Taste statt <Enter>. ***a99/alcarta***: Nur "`-->`" löst Umblättern aus.

I XXX k "Registername" Für ***a99/alcarta*** und den "avanti"-Server braucht jedes Register eine solche Zeile. XXX ist der symbolische Name, k die Nummer des Registers. Beispiele in `cat.api`.
Bis zu 50 solche symbolische Registernamen kann es geben, z.B.
`I PER 1 "Personen"` **MultiX** (ab V27) z.B.: **`I ALL e1 "Alle Wörter"`**
`I TIT 3 "Wörter"` d.h. Indexdatei `cat.aex` statt `cat.adx`
`I DIS 1D. "Dissertationen"` (`1D.` bedeutet: Präfix D plus Spatium im Reg. 1)

R XXX mp "Restriktion" : Zur Einschränkung einer Suche kann man evtl. Kriterien wie das Erscheinungsjahr verwenden (ab V15). **m**=r,R,s,S. Dazu muß eine Restriktionsdatei angelegt werden. Siehe → 10.2.9

ic=1 **Umcodierung der Benutzeranfrage gewünscht** <0>
Was der Benutzer eintippt, kann intern umcodiert werden, bevor die Suche ausgeführt wird. Dazu muß es aber Sprungmarken **#-1** ... **#-9** für die einzelnen Register geben, wo die Umcodierung definiert wird. So ist erreichbar, daß man im Register 1 "mueller" findet, wenn man "Müller" eingetippt hat, im Register 8 aber "ABC", obwohl man "abc" eingegeben hat.
Unter jeder dieser Sprungmarken stellt die Sonderkategorie `#u1` die Benutzervariable dar und kann mit Hilfe der Exportsprache zu einem beliebigen Arbeitstext umgewandelt werden.
Tip: Soll diese Umcodierung in bestimmten Fällen nicht wirken: einen '_' (Unterstrich) vor das Suchwort setzen.

il=k **Maximale Länge der Registereinträge** <60>
Die maximale Schlüssellänge muß bei der Indexgenerierung festgelegt werden. Aus historischen Gründen, damit "alte" Datenbanken weiter funktionieren, wird 60 als Standardwert genommen. Empfohlen wird jedoch `il=72`, weil damit die Bildschirmbreite voll ausgenutzt wird. Das mögliche Maximum ist `il=246`. Einen überlangen Schlüssel kann man sich vollständig zeigen lassen, indem man '#' drückt. Mit der Tabtaste kann man die Anzeige der langen Schlüssel nach rechts und links verschieben (→ Kap.1.4).
Empfehlung: Wenn man V14-Ersetzungen anwendet, und wenn dabei sehr lange EInträge auftregen können (z.B. Körperschaftsnamen) setze man `il` auf einen genügend großen Wert.
Nachträgliche Änderung: wenn man **`il`** ändert, muß man anschließend den Index erneuern lassen (***CockPit*** **`µr o i`**), sonst funktioniert er nicht mehr.

ii=k **Multiplikator für maximale Dateigröße** <1>
Eine Datenbankdatei kann eine maximale Größe von 16 MB erreichen. 255 Solche Dateien können zusammen eine Datenbank bilden. Wenn das noch nicht reicht, kann man mit **`ii=2`** die Maximalgröße verdoppeln. Die Zahl k darf bis zu 125 groß sein! Diese Zahl wird auch "Aufbohrfaktor" genannt.
Vorsicht bei Veränderung dieses Wertes. Man muß dann die **Datenbank neu aufbauen** lassen (dabei wird **`INDEX -f70 ...`** gestartet), bevor man weiterarbeiten kann.

ix=ddeddddddd : Register 3 liegt in der Indexdatei *.aex, die anderen in *.adx (s. Hilfetext zu `MultiX`)

ia, ib **Steuerzeichen** für die Verwendung von V-Sequenzen. (Siehe dazu die "Verlautbarung" 164: `h vb164` eingeben in a99). Damit können z.B. Unicode-Sequenzen der Form `&#nnn;` für Sonderzeichen verwendet werden.

Empfehlung: Verwenden Sie in Index-Parametern immer diese zwei Zeilen als erste ak-Befehle:
ak=zz+@
ak=zz+0
und machen Sie im Abschnitt `#-@` die Aufbereitung des Primärschlüssels, im Abschnitt `#-0` die Erstellung der Kurzzeile. In beiden Abschnitten muß man ohne `#u1` arbeiten, denn `ak=zz` erzeugt keine `#u1`. Dafür werden beide Befehle in jedem Fall ausgeführt, d.h. jeder Satz bekommt einen Primärschlüssel und eine Kurzzeile.
Die Sprungmarken `#-@` und `#-0` sind auf diese Aufgaben festgelegt: die Programme müssen manchmal den Primärschlüssel eines Satzes oder dessen Kurzzeile errechnen, dann verwenden sie diese Sprungmarken.

Anmerkung: Ansonsten gibt es für die Schlüsselgenerierung noch einen speziellen Vorbearbeitungsbefehl, 'K', zur Erzeugung sogenannter Matchcodes oder Akronyme.
Ferner eine besondere Art von Präfix (z.B. **`"|5"`**), der die Zuordnung des Schlüssels zu einem bestimmten Index bewirkt. (Für beides → 10.2.6.3)

`|i=`*Überschrift für Register i* (i=1,...,9, : , ; , <)
Register-Überschriften
Für die Register 1 bis 9 kann man Überschriften vorgeben, die dann bei der Index-Anzeige in der obersten Zeile erscheinen (→ 1.4). Man setzt für **`i`** die Registerziffer ein, für *Überschrift* den Text, der gewünscht wird. In CAT.API sind Beispiele zu sehen. Für Register 10 ist **`:`** (Doppelpunkt), für Register 11 **`;`** (Semikolon) anstatt einer Ziffer anzugeben,

`|<=`*abc* **Überschrift für die Kurzanzeige** der Ergebnismengen.

`|a="`*Überschrift für den Anzeige-Bildschirm***"** (a99: Titel der Datenbank; überschreibt die Zeile 186 in UIF1)

Tip: Die Überschriften verlagern in eine eigene Datei namens *dbn.d*PT (*dbn*=Datenbankname). Bei Umschaltung auf eine andere Datenbank mit Alt+a (in PRESTO) werden solche Dateien automatisch geladen, und dadurch hat man beim Erscheinen der anderen Datenbank immer die richtigen Überschriften.

`N` **Stopwort-Tabelle**

`>` **Zweite Tabelle**

Wörter und Zeichenfolgen, die bei der Indexgenerierung entstehen können, aber nicht in die Indexdatei einsortiert werden sollen, nennt man "Stopwörter". Für jede Datenbank kann die Liste der Stopwörter individuell zusammengestellt werden. Entweder wird sie mit in die .API-Datei aufgenommen, oder man legt sie in eine eigene Datei des Typs .APT, die dann für mehrere .API-Dateien nutzbar ist, indem sie mit dem t-Befehl eingebunden wird. (Vgl. CAT.API und SWL1.APT)
In jedem Fall sieht die Stopworttabelle formal so aus:

N Anfang der Liste
0 alle einstelligen Zahlen sollen Stopwörter sein
1 möglich wäre hier: **`19??`** um alle Jahreszahlen auszublenden
9
a
ab nun kommen Artikel, Präpositionen etc.,
... in alphabetischer Reihenfolge
zur letztes Wort
N Ende der Liste

Kommentare können, wie bei den anderen Parametern, auch hier frei eingestreut werden. (Mindestens zwei Leerzeichen vorweg. Kein Leerzeichen innerhalb eines Stopworts!)
Beim Einfügen neuer Wörter achte man streng auf korrekte alphabetische Einordnung (d.h. exakte ASCII-Reihenfolge), sonst wundert man sich hinterher, warum bestimmte Wörter trotzdem im Index erscheinen.
Es kann in einzelnen Sätzen Fälle geben, wo bestimmte Wörter, die mit einem Stopwort buchstabenweise identisch sind, trotzdem in den Index sollen. Beispiele: UNO und MIT. Vor diese Wörter setzt man bei der Erfassung das "Entstoppungszeichen" '@' (→ Anh.A.1). Wenn man zusammengesetzte Schlüssel macht, z.B. `ort:jahr` , muß das Trennzeichen in **`i1`** deklariert sein, falls der Schlüssel eliminiert werden soll, wenn `ort` ein Stopwort ist.
Nichts zu tun hat die Stopworttabelle mit der Artikelliste, die in der .CFG-Datei anzulegen ist. Letztere dient **nur** dazu, bei der Eingabe die Artikel am Titel**anfang** oder Namens**anfang** von Körperschaften zu erkennen und mit Nichtsortierzeichen zu markieren, damit sie beim Sortieren übergangen werden können (→ Anh.A.1).
Zwischen den beiden N-Zeilen kann eine Zeile stehen, die nur das Zeichen > enthält. Damit leitet man eine zweite, wiederum alphabetisch sortierte Liste ein. Diese wird anders benutzt: Man schreibt in einer Parameterzeile den Manipulationsbefehl `u>>` , womit dann die Zeile ergebnislos bleibt, wenn der Arbeitstext in dem Moment einem Wort der zweiten Liste entspricht. In der zweiten Liste kann `?` als "Joker" auftreten: untersuchung?? würde erreichen, daß auch das Wort "untersuchungen" ausgeblendet wird.

10.2.2 Formular- und Seiten-Parameter

fl=n **Formularlänge** : Anzahl Zeilen je Datensatz <14>
wenn mehr als n Zeilen gebraucht werden, wird eine Fortsetzungskarte gemacht.

=0 Listendruck: keine Formularaufteilung (dann sind fb, ff und fn wirkungslos!)

fm=0 **Formular-Modus** : Listendruck (automatisch bei fl=0)

=1 Kartendruck, Unteraufnahmen nicht auf getrennte Karten <1>

=2 Kartendruck, für jede Unteraufnahme (Bandaufführung) eine eigene Karte.

fa=CS **Fortsetzungskarten-Anfang** <>
Wenn eine Fortsetzungskarte nötig ist, wird diese Zeichenfolge an die Position geschrieben, die durch den Befehl **#wh** markiert wurde (→ 10.2.6.4). Beispiel:
fa="..." 0 $ &60 K ")"
Dadurch erscheinen dann an der #wh-Position zuerst "...", dann geht es in die erste Zeile auf Position 60, dort wird die Nummer plaziert und eine Klammer dahinter geschrieben. Automatisch geht anschließend das Programm zur #wh-Position zurück, macht zwei Zeilenvorschübe und fährt mit der Ausgabe fort.

fb=CS Zeichenfolge, die zwischen zwei Formularen (Karten) <>*
geschrieben werden soll (**Formularbruch**)

ff=i i=0,1,2: **Anzahl der Fußzeilen** je Formular (Karte) <0>
siehe Befehl F für Fußabschnitt in 10.2.6.5.3.
fl+ff ist die Gesamtzahl der druckbaren Zeilen auf einer Karte.

fn=n **Formular-Nutzen** je Seite <0>
Formulare in Endlosform können so gestaltet sein, daß mehrere, etwa 4, eine Seite bilden, und zwischen den Seiten ein Streifen ist, der beim Drucken übersprungen werden muß. fn gibt dann die Anzahl der Formulare je Seite an. Zusätzlich benötigt man dann:

sv=CS **Seiten-Vorschub**: <12>*

bei **Listendruck** (fl=0): Druckersteuerung für den Seitenvorschub
bei **Kartendruck** (fl>0): nach einer Anzahl von fn Formularen wird der Drucker mit sv gesteuert.
Wenn fn=0 ist, wird sv nicht benötigt, sondern nur fb benutzt.

sz=n **Seitenzahl** der ersten Seite (bei Listendruck) <1>

=0 kein Seitenumbruch (**zm** dann wirkungslos, → 10.2.3)

10.2.3 Zeilen-Parameter

zb=CS **Zeilenbruch**-Zeichen <" -/">
Wenn Zeilenumbruch durchzuführen ist, werden diese Zeichen benutzt, um Unterbrechungsstellen zu finden.

zi=z **Zeileneinrückung** (indentation). Ab der 2. Zeile des Datensatzes wird der Text **z** Zeichen eingerückt. <0>
Wird keine Einrückung gewünscht: Angabe weglassen, denn **zi=0** ist Standard.
Sonderfall: Bei einem Pauschal-Export (mit Befehl **##**) beginnt jede Kategorie wieder am Zeilenanfang, aber bei langen Kategorien erfolgt Einrückung der umgebrochenen Zeilen.

zl=k **Zeilenlänge** in Zeichen (k <= 132) <56>
Nach maximal k Zeichen wird ein Zeilenumbruch an den im Parameter **zb** aufgeführten Zeichen durchgeführt.
Sonderfall: Fortlaufende Ausgabe ohne Zeilenumbruch (z.B. zur Übergabe an Textprogramme wie WORD oder WordPerfect): erreichbar ist das durch die schlichte Zeile

zl=0

Wenn der Befehl C (entweder in einem Zwischenteil oder als Manipulationsbefehl) verwendet wird, gibt das Programm trotzdem den Inhalt von **ze** aus (s.u.)
Achtung: Bei `zl=0` sind `zi` und `zm` nicht nutzbar, auch nicht die Strukturbefehle % , $ und # (→ 10.2.0)

zm=n **Zeilenmaximum** (max. Anzahl Zeilen) je Seite <65>
Wieviele Zeilen sollen tatsächlich bedruckt werden? Nach n ausgegebenen Zeilen wird ein Seitenvorschub ausgeführt. Genauer: es wird zuerst der Seitenfuß, dann der nächste Seitenkopf aufbereitet.

=0 fortlaufende Ausgabe **ohne Seitenumbruch** (dann auch sz=0). Diesen Wert setzt man insbesondere dann, wenn die Ergebnisdatei mit einem Textsystem weiterverarbeitet werden soll.

z1=k **minimale Zeilenzahl vor Umbruch** : mindestens k Zeilen müssen noch auf die Seite passen, sonst wird zuerst ein Seitenumbruch gemacht, bevor das Programm mit dem aktuellen Satz beginnt. Das verhindert die Entstehung sogenannter "Schusterjungen", d.h. alleinstehender Kopfzeilen am Seitenende. <3>
Sonderfall: **z1=0** : wenn der Datensatz nicht mehr komplett auf die Seite paßt, wird vorher ein Seitenvorschub gemacht.

z2=k **maximaler Überlauf über zm** : eine Toleranz von k Zeilen wird eingeräumt, d.h. das wahre Maximum je Seite ist zm+z2. Ausgenutzt wird das nur, wenn der Datensatz dann gerade noch als Ganzes auf die Seite geht. Wenn nicht, wird schon nach der Zeilenzahl `zm` unterbrochen. Dies ist das Mittel gegen "Hurenkinder", vereinzelte Schlußzeilen am Seitenkopf. <2>

Je höher man `z1` und `z2` setzt, um so seltener wird ein Datensatz am Seitenende getrennt. Jedoch werden die Seiten ungleichmäßiger gefüllt. Es ist klar, daß **zm + z2 + ff** nicht größer als die tatsächliche Seitenlänge des Papiers sein darf. Der Wert `ff` entfällt, wenn kein Fußabschnitt definiert ist.

ze=CS **Zeilenende**-Steuerzeichen <13 10>*

Nur anzugeben, wenn nicht das normale Zeilenende "Wagenrücklauf / Zeilenvorschub" anzuwenden ist. Auf die Bildschirmanzeige wirkt sich **ze** nicht aus. Nur bei der Ausgabe auf Drucker oder Datei wird **ze** hinter jede fertig aufbereitete Zeile gesetzt, auch hinter die letzte! Für UNIX ist der Standardwert das Zeichen "newline", ASCII-Code 10.
Zur Verdeutlichung: Ein Ausgabesatz wird zuerst komplett im Arbeitsspeicher aufgebaut, wobei das Programm noch keine Trennzeichen setzt, sondern sich nur die Zeilenenden "merkt". Danach wird zeilenweise auf den Drucker bzw. in die Exportdatei geschrieben. Erst zu diesem Zeitpunkt wird dann der Parameter **ze** verwendet, d.h. am Ende jeder Zeile ausgegeben. Für den Umbruch innerhalb des Arbeitsspeichers und die Anzeige auf dem Bildschirm hat **ze** daher keine Bedeutung.

10.2.4 Drucker- und Codierungs-Parameter

di=CS **Initialisierung** des Druckers. Wird vor dem ersten Exportsatz <"">*
automatisch ausgegeben (vgl. Kap.3.5.3, Befehl #p) und wenn #pi im Eingabemodus als Befehl gegeben wird. (siehe auch → 10.2.6.5 "Kopfabschnitt")
Keine Wirkung auf Umbruch und Bildschirm.
Auch verwendbar, um einen **Dateikopf** zu erzeugen.

dx=1 **Zeichenumwandlung** über Tabelle <0>
=0 1: auch für die Bildschirmanzeige,
0: nur bei Druck- bzw. Dateiausgabe wirksam
Wenn die Parameterdatei für Bildschirmausgabe gedacht ist und spezielle Bildschirmsteuerungen (sog. ANSI-Escape-Sequenzen) vorhanden sind: dx=1.

dn=X **Name der Parameterdatei**: ohne Bedeutung für das Programm, <>
nur als Merkhilfe für den Benutzer gedacht; Anzeige bei F2

10.2.4.1 Codiertabellen und Ersetzungsbefehle

Auf der Ebene des einzelnen Zeichens gibt es eine Reihe von Möglichkeiten, Zeichen durch andere Zeichen oder durch Zeichenfolgen zu ersetzen.
Es existieren während des Programmlaufs zwei Codiertabellen, die **Normaltabelle** und die **Alternativtabelle**, die für jedes Datenelement wahlweise benutzt werden können. Das erste Zeichen des Exportbefehls, '#' oder '!', entscheidet darüber, welche von beiden jeweils benutzt wird (→10.2.6.1).
Warum zwei Tabellen? Es kann z.B. vorkommen, daß bestimmte Datenfelder in Großbuchstaben und/oder mit aufgelösten Umlauten gedruckt werden sollen, obwohl sie mit normaler Schreibweise erfaßt sind. Das kann etwa bei den Köpfen von Katalogkarten der Fall sein oder bei den Ordnungskriterien in sortierten Listen. Dazu ist dann die Alternativtabelle nützlich.
Wahlweise kann man auch befehlen, eine bestimmte Kategorie überhaupt nicht umzucodieren: dazu gibt man den Manipulationsbefehl `y0` .
Für die Bildschirmanzeige ist die Umcodierung normalerweise abgeschaltet. Mit dem Parameter `dx=1` wird sie auch für den Bildschirm eingeschaltet. Diese Möglichkeit ist wichtig, wenn z.B. unterschiedliche Attribute, auch Farben, auf den Bildschirm zu bringen sind und dazu Steuerzeichenfolgen (sog. ANSI-Escape-Sequenzen) verwendet werden.

Die Codiertabellen enthalten für jedes Zeichen (256 ASCII-Codes) eine Anweisung, was statt dieses Zeichens wirklich auszugeben ist. Vordefiniert ist, daß jedes Zeichen unverändert ausgegeben, also nicht umcodiert wird. Nur für abweichende Fälle muß man Angaben für die Codiertabellen machen, und zwar in folgender Form: (statt 'p' kann immer auch 'd' benutzt werden!)

p Codierbefehl für die Normaltabelle,
wirkt in den mit '#' beginnenden Exportbefehlen (→ 10.2.6.1);

q Codierbefehl für die Alternativtabelle,
wirkt in den mit '!' beginnenden Exportbefehlen;
In den folgenden Beispielen wird nur 'p' benutzt. Die Form der Befehle ist für 'q' dieselbe.

Allgemeine Form:

p *zeichen ersatzcode*

Beispiele:

p ä 123 ä wird in den Code 123 übersetzt

p § 192 § erhält den Code 192

p a/z 97 a bis z werden mit 97,98,99 ... 122 codiert

(das entspricht den normalen Rechnercodes und kann deshalb entfallen; dagegen:)

p a/z A oder

p a/z 65 diese Befehle würden Kleinbuchstaben a...z in große (Codes 65...90 = A...Z) umsetzen

p ä Ä Für die Umlaute muß man eigene Befehle einsetzen (sie liegen nicht zwischen a und z)

p .15 95 Ersetzt den (nicht druckbaren) Code 15 durch 95 (= '_')

p A/Z 97 macht das Umgekehrte: A -> a, B -> b, ...

p x/y =z Die Codes zwischen x und y (einschließlich) werden alle durch den Wert z ersetzt. Für z=1 heißt das: die Codes x bis y werden alle ausgeblendet. Bei Sortier- und Indexparametern kann man so sehr leicht ganze Bereiche von Sonderzeichen eliminieren.

p x 0 Zeichen x wirkt als **globales Endezeichen**. Wenn es in einer Kategorie vorkommt, wird ihre Ausgabe an dieser Stelle beendet. Ausnahme: wenn das Zeichen gleich am Anfang der Kategorie steht, wird der Text ausgegeben, weil nichts anderes in der Kategorie vorhanden ist.
Empfehlung: **p ⇳ 0** (Code 18)

p x 1 Zeichen x soll gar nicht gedruckt werden (ersatzlose Übergehung)

Die folgenden 2 Codes ermöglichen es, Codierungen von Teilfeldern, wie sie bei manchen Kategoriensystemen (z.B. MARC) vorkommen, auf unterschiedliche Art zu behandeln:

p x 2 Wenn x auftritt, soll es mitsamt dem nachfolgenden Zeichen ersatzlos beseitigt werden

p x 3 Wenn x auftritt, soll es durch Leerzeichen ersetzt und das nachfolgende Zeichen ignoriert werden. D.h. x wird zusammen mit dem nachfolgenden Zeichen durch nur ein Leerzeichen ersetzt.

p x 5 Zeichen x zusammen mit dem nachfolgenden Zeichen soll unverändert stehen bleiben. Wichtig ist dies, wenn es 2-Zeichen-Kombinationen gibt, wo bestimmte Zeichen zusammen mit ihrem Nachfolger einen Code für ein anderes Zeichen bilden. Der Befehl
p .161/.255 =5 würde bewirken, daß alle Zeichen ab Code 161 so behandelt werden.

p x 8 **Texel-Ersetzung.** Wenn x auftritt, soll es entfallen und das nachfolgende Zeichen soll durch das Zwischenteil ersetzt werden, das seiner ASCII-Nummer entspricht. Zum Beispiel: p _ 8, dann wird aus _K das Zw.T. 75 (= Code von K) Vgl. Anwendung in der N-Parametrierung, Kommentar in N.CFG.

p x 9 **Sequenzen-Ersetzungstechnik.** Wenn x auftritt, soll die nachfolgende Zeichensequenz bis zum Zeichen `ib` (Default: ;) im Ersetzungsregister gesucht werden (→ **10.2.1.3)**. Was im Register hinter dieser Sequenz steht, soll statt der Sequenz eingesetzt werden. Anw. für Unicode z.B. in der N-Parametrierung.
(Siehe zu den zwei letzten Befehlen die "Verlautbarung" 164: h vb164 eingeben in a99).

Wenn man **Druckertreiber** als Nachladetabellen organisiert (→ 10.2), kann man darin auch alle Codierungsbefehle unterbringen, die druckerspezifisch sind.
In der Tabelle **S.APT** sind die Codewandlungen untergebracht, die man für eine Sortieraufbereitung braucht; zur Anwendung siehe Kap.8.
In den Indexparameterdateien, z.B. CAT.API, findet man lange Listen von Codierbefehlen, die man jeweils in neue Parametersätze übernehmen kann, denn für den Index ist die Vereinheitlichung der Schreibweise, die Umlautauflösung (siehe nächster Abschnitt) und die Eliminierung von Akzenten und Sonderzeichen sehr wichtig. Die Datei **I.APT** enthält diese Codierungen; der Zeichensatz OSTWEST.FON ist berücksichtigt..
Alle diese Tabellen sind unabhängig vom Kategorienschema, also nach Umbenennung auch für andere nutzbar.

10.2.4.2 Ersatzdarstellungen für Einzelzeichen

Für jedes Zeichen kann man über die Codiertabelle ferner eine **Ersatzdarstellung** definieren, wobei es zwei Möglichkeiten gibt:

Text-Ersatzdarstellung

p *zeichen "ersatztext"*

Beispiel: **`p ä "ae"`**
z.B. zur Auflösung von Umlauten. Solche Ersatzdarstellungen werden, wie man erwarten wird, beim Zeilenumbruch berücksichtigt. Solche Befehle wirken nicht auf Zwischenteile!
Ausnahme: das Nichtsortierzeichen kann so nicht umcodiert werden. Bei diesem Zeichen funktioniert nur der Befehlstyp **`p x k`**.

Drucker-Ersatzdarstellung : statt *"ersatztext"* eine Codesequenz mit führendem Punkt:

`p [ .27 82 0 91 27 82 2` bzw.

`p .91 .27 82 0 91 27 82 2`
wenn z.B. der Drucker zwar die eckige Klammer hat, aber in einem anderen nationalen Zeichensatz. Die hier gezeigten Zahlen sind die Ersatzdarstellung von [für EPSON-Drucker:
Esc R 0 91 Esc R 2; dabei schaltet Esc R 0 auf den US-ASCII-Satz um, 91 ist der Code für [, und Esc R 2 schaltet wieder auf den deutschen Zeichensatz zurück. Alle nationalen Sonderzeichen kann man auf diese Weise codieren.
Drucker-Ersatzdarstellungen haben sinnvollerweise keinen Einfluß auf den Zeilenumbruch: die gesamte Sequenz wird als **ein** Zeichen gerechnet. Ausgeführt werden diese Befehle erst am Schluß, unmittelbar vor der physischen Ausgabe des Zeichens, die anderen Befehle dagegen schon vor der Umbruchbearbeitung.
Trotz des Namens können Drucker-Ersatzdarstellungen natürlich auch zum Datei-Export verwendet werden.
Drucker-Ersatzdarstellungen können aus mehreren Zeichen bestehen. Hinter dem Punkt sind die Druckercodes einzugeben, durch die das Zeichen zu ersetzen ist. Die Umwandlung erfolgt in zwei Schritten: zuerst werden Zeichen, für die eine Text-Ersatzdarstellung definiert ist, durch diese ersetzt, dann der Zeilenumbruch durchgeführt, und dann werden die Rechnercodes vor dem Drucken durch die Druckercodes ersetzt, wenn für ein Zeichen eine Drucker-Ersatzdarstellung definiert ist. Für die Ausgabe auf dem Bildschirm (Befehl #D) hat eine Drucker-Ersatzdarstellung keine Wirkung.
Text-Ersatzdarstellungen sind dagegen auch wirksam, wenn `dx=0` gesetzt wurde, wenn also keine Codeumwandlung für den Bildschirm gewünscht wird.

Eine extreme Anwendung von Ersatzdarstellungen wäre diese: wenn bestimmte Wörter oder Wendungen in einem Datenmaterial häufig vorkommen, wähle man ein ansonsten nicht benutztes Zeichen (z.B. Graphikzeichen) und teile diesem die betreffende Zeichenfolge als Text-Ersatzdarstellung zu. Würde man etwa eine große Sammlung von Dissertationen einiger Technischer Universitäten erfassen und später drucken wollen, könnte man ein Zeichen x auswählen und schreiben

`p x "Technische Universität"` .
Beim Drucken wird dann der Umbruch so durchgeführt, als ob tatsächlich dieses anstelle von x stünde. Bei der Volltextsuche allerdings könnte man diese Sätze nicht finden, wenn man z.B. `"universitaet"` als Suchwort eingäbe!

[10.2.4.3] Backspace-Liste

Anmerkung: Von der Verwendung dieses Befehls wird abgeraten, da nicht mehr zeitgemäß.

Für Druckausgaben auf älteren Druckern gibt es noch eine kleine Sondertabelle, die **Backspace-Liste**, die mit dem Befehl pb definiert wird:

pb 94 95 96 126 die Zeichen ^ _ ` ~ sollen einen Rückschritt auslösen

Die Codes 94,95,96 und 126 sollen für den Druck als "backspace"-Zeichen gelten, d.h. hinter jedem dieser Codes muß ein backspace (= Code 8) gedruckt werden (Akzente bzw. Unterstreichungsstrich!)

In die "Backspace-Liste" kann man beliebig viele Codes aufnehmen. Bei der Ausgabe wird dann geprüft, ob das zu druckende Zeichen in dieser Liste vorkommt. Wenn ja, wird es gedruckt und ein Code 8 hinterhergeschickt, so daß das nächste Zeichen auf dieselbe Position gedruckt wird. Dies funktioniert z.B. auf EPSON-Druckern; bei anderen nicht unbedingt, aber meistens.

Wenn man den Zeichensatz DIN 31628 einsetzt, gibt es noch eine Anzahl andere Akzente, die auch berücksichtigt werden müssen.

10.2.4.4 Absatz-Endezeichen (Zeilentrennzeichen)

Wenn man in gewissen Kategorien längere Texte hat, kann es sein, daß man beim Umbruch gern an bestimmten Stellen einen Absatz (Zeilenwechsel) erzwingen möchte, d.h. einen Zeilenvorschub, obwohl noch Platz in der Zeile wäre. Dazu kann man ein Zeilentrennzeichen definieren, indem man den Codierbefehl

p x 255

gibt, wobei x ein sonst nicht benutztes Zeichen ist.

Empfehlung: (verwendet in den mitgelieferten Parameterdateien) ist der Code 20 = '¶' , auf MS-DOS-Rechnern mit CTRL+t einzugeben. Die Auswirkung: wo immer im Text dieses Zeichen steht, wird beim Umbruch eine neue Zeile begonnen.

Nützlich ist dieser Code auch als Trennung zwischen mehreren gleichwertigen Eintragungen innerhalb einer Kategorie (→ 10.2.1).

Empfehlung: wenn Sie beide Mechanismen zugleich verwenden wollen, definieren Sie ein anderes Zeichen als Zeilentrennung.

10.2.4.5 Indexcodes

Diese sehen aus wie p- und q-Befehle, machen aber keine Umcodierung. Sie definieren die Sortierwerte einzelner Zeichen, wenn diese vom ASCII-Wert abweichen sollen. Die Zeichen selbst bleiben dabei unverändert. Wenn man etwa das türkische "dumpfe i" (ohne Punkt), zwischen i und j ordnen wollte, wie es die Türken tun, müßte man schreiben:

i ı 106 Zeichen **ı** erhält den Sortierwert 106

i j/z 107 j erhält 107, k 108 usw. bis z (normalerweise wäre j=106 usw.)

Das heißt: **ı** bekommt den Sortierwert 106, das ist sonst der Wert von j. Das j und alle nachfolgenden Buchstaben bis z rücken eine Position weiter.

In deutsche OPACs wird man genau dieses nicht einbauen, sondern vielmehr schreiben:

```
p ı I
q ı i
```

Dann wird in den Registern das türkische durch ein "normales" i ersetzt und folglich auch so geordnet. Die i-Befehle wurden übrigens ursprünglich entwickelt, um kyrillische Zeichen mit ukrainischen Sonderzeichen benutzen und diese korrekt ordnen zu können.

o x y ASCII ←→ ANSI Zuordnung

Für die Windows- und UNIX-Programme braucht man eine Liste solcher Zuordnungen, die in beiden Richtungen eindeutig sein müssen. Dadurch wird es möglich, daß die Datenbank selbst ASCII-Daten enthält (und somit von den DOS-Programmen benutzt werden kann), daß sie aber unter Windows und UNIX korrekt aussieht, denn an allen relevanten Stellen werden dann die Zeichen mit Hilfe dieser Tabelle hin und zurück konvertiert. Eine komplette Tabelle dieser Zuordnungen für den OSTWEST-Zeichenvorrat ist die Datei `o.apt`, die man in die Indexparameter einfach mit dem Befehl

to

einbindet.

Man muß eine andere solche Tabelle aufstellen, wenn man nicht den Ostwest-Zeichenvorrat benutzt, sondern z.B. den Pica-Code. Ansonsten kann man die O.APT unbesehen auch für andere Konfigurationen übernehmen.

Hinweis: Die DOS-Programme ignorieren die o-Befehle – sie arbeiten ja nur mit ASCII.

10.2.4.6 Globale Zeichenfolgen-Ersetzungsbefehle

("Lokale Ersetzung" → 10.2.6.3)

Neben den verschiedenen Möglichkeiten, einzelne Zeichen durch ein oder mehrere andere Zeichen zu ersetzen, kann man jedoch auch ganze Wörter, oder allgemeiner: Zeichenfolgen, durch andere Zeichenfolgen ersetzen lassen. (Aber: dies kostet viel mehr Zeit als Einzelzeichen-Ersetzungen.)
Für die Erfassung haben Sie den Befehl **Suchen und Ersetzen** (Befehlszeichen '**_**' oder '**,**') kennengelernt (→ Kap.3.5.3). Dieselben Befehle, die im Editor anwendbar sind, kann man als Befehle in eine Export-Parameterdatei einbauen:
Um eine automatische Ersetzung in jedem zu bearbeitenden Satz ("globale Ersetzung") zu erreichen, braucht man nur einen oder mehrere solcher Befehle als eigene Zeilen in die Parameterdatei einzusetzen. Die Wirkung erstreckt sich auch auf nachgeladene Sätze.
Es spielt keine Rolle, an welcher Stelle in der Datei diese Befehle stehen, d.h. sie wirken immer global auf den ganzen aktuellen Datensatz, der gerade exportiert wird.
Wenn Sie z.B.

`_PC_Personal Computer_`
in die Parameterdatei schreiben, ganz gleich wo, wird jedes Vorkommnis von "PC" durch "Personal Computer" ersetzt. Allerdings auch da, wo es vielleicht etwas anderes bedeutet! Anders als bei der Erfassung fragt das Programm bei der Ausgabeproduktion nämlich nicht an jeder Stelle, ob hier die Ersetzung gewünscht ist, sondern führt sie kommentarlos aus.
Zusätzlich gibt es aber die "Kategoriespezifische Ersetzung". Wenn Sie z.B.

`*#31_PC_Personal Computer_`
schreiben, wird nur in der Kategorie #31 (Schlagwort) diese Ersetzung vorgenommen (auch in den evtl. vorkommenden Mehrfachkategorien #31a, #31b, etc.), aber nicht in den anderen Kategorien. Will man dagegen die Ersetzung in allen Titelkategorien (#20, #22, ...) vornehmen, so würde man dieses schreiben:

`*#2_PC_Personal Computer_`
Das dritte '_' am Ende der Ersetzungskette darf man auf keinen Fall vergessen!
Innerhalb der beiden Zeichenfolgen können Codes der Gestalt **`\n`** mit 1<n<255 (Dezimalzahl, nicht Hexadezimal) verwendet werden für Zeichen, die man mit seinem Texteditor nicht eingeben kann.
Vorsicht: Das Dollarzeichen kann nicht ersetzt werden, d.h. es kann nicht in der ersten Zeichenfolge vorkommen, denn es hat eine Sonderwirkung: es steht stellvertretend für den ganzen Rest der Kategorie, siehe Beschreibung in Kap.3.5.3.
Beispiel:

`*#31p_Shakespeare$_Shakespeare, William <1564-1616>_`
Dadurch wird in Kategorie #31p "Shakespeare" in allen Schreibweisen durch die angegebene Ansetzungsform ersetzt.

Sonderformen: (siehe auch Kap. 3.5.3.)

`_#980_#270_` wandelt #980 in #270 um (sorgt auch für die Umordnung innerhalb des Satzes!)

`_#33a__` löscht die Kategorie #33a (2 Unterstriche hinter #33a)

`__#33aText_` fügt in jeden bearbeiteten Satz die Kategorie `#33aText` ein (2 Unterstr. vor #33a)

Anmerkung: Bei den Import-Parametern hat der Ersetzungsbefehl eine etwas andere Form (→ Kap.11.2.2.2), weil dort auch häufig nicht-druckbare Zeichen (Steuerzeichen) ersetzt werden müssen.

Empfehlung: Für Ersetzungen von **Einzelzeichen** verwendet man besser nicht diese Befehle, sondern die p- und q-Befehle, denn die sind erheblich schneller. Statt **`_X_U_`** schreibt man also besser **`p X U`**. Das sind exakt gleich viele Tastenanschläge, aber eine mindestens 10fach schnellere Abarbeitung (→ Kap.10.2.4.2).

10.2.4.7 Unicode-Ersetzungsbefehle für UTF-8-Codes

Ab Version 23.2 kann man in einer Datenbank statt der OstWest-Codierung oder einer anderen ASCII-Codierung die Unicode-Methode **UTF-8** verwenden. Dafür gibt es neue Befehle, mit P und Q eingeleitet, mit denen man Kombinationen von 2 bzw. 3 Zeichen durch andere Zeichen oder Kombinationen ersetzen kann. Sinnvoll ist das vo allem dann, wenn man eine Datenbank im Internet benutzen und auch bearbeiten will. Dazu gibt es die Web-Anbindungen PHPAC und a35. Alles Notwendige dazu ist über die Homepage zu finden: **http://www.allegro-c.de/unicode/**

10.2.5 Kategorie-Parameter

kb=1 **Folgezeichen** (Wiederholungszeichen) der Kategorienummer durch Leerzeichen ersetzen <0>

=0 Folgezeichen unverändert lassen

relevant ist dieser Parameter **nur**, wenn **ks** (s.u.) gesetzt ist: das Wiederholungszeichen der Kategorienummer wird bei `kb=1` durch ein Leerzeichen ersetzt. Man braucht diese Funktion evtl. dann, wenn die Daten als ASCII-Text an andere Software übergeben werden soll, die wiederholbare Kategorien ohne unterscheidendes Folgezeichen verlangt.

=2 Folgezeichen ersatzlos beseitigen

Das Zeichen an der Wiederholungsposition fällt dann ganz weg.

ke=CS **Kategorie-Ende-Codes** (= Standard-Postfix) <"">

ke wird als Standardwert hinter einer Kategorie ausgegeben, wenn in der Kategorieliste (→ 10.2.6) für diese Kategorie kein Postfix angegeben ist.
Notwendig ist **ke** bei Pauschalexport (Befehl **##**, → 10.2.6.1), es wird dann hinter jeder einzelnen Kategorie, also als Feldtrennung, ausgegeben.
Im Gegensatz zu **ze** wird **ke** in den Zeilenumbruch einbezogen, wie jedes andere Postfix auch.
Aufpassen: Wenn **ke** gesetzt wurde, aber bei einer Kategorie dann ausnahmsweise **kein** Postfix gewünscht wird, muß man **#zz 0** an den Kategoriebefehl anhängen
Weil man das leicht vergißt, wird empfohlen: ke nur verwenden, wenn mit Pauschalexport gearbeitet, also Befehl **##** verwendet wird.
Hinweis: Ohne Wirkung in dem Abschnitt für Globale Manipulation, der durch `#-#` eingeleitet wird.

kh=x **Hierarchie-Separator** <,>

Mehrteilige Bandbezeichnungen werden auf den Hierarchiestufen #02... durch Aneinanderhängen der zugehörigen übergeordneten Stufen gebildet (siehe Sonderkategorie #hi, → 10.2.6.5). Zur Trennung der einzelnen Zählungselemente wird dabei ein Komma verwendet. Wenn man z.B.
kh=/
gibt, wird statt dessen ein Schrägstrich benutzt.

ks=n **Startposition** des Ausgabetextes innerhalb der Kategorie (**wichtig**) <4/5>

Als Standard wird der Wert genommen, der in der .CFG-Datei mit dem Befehl **k** gesetzt wird. Bei A.CFG ist dies 4, bei U.CFG 7, bei D.CFG 6.
1 würde bewirken, daß die Kategorienummer mit ausgegeben wird
0 würde außerdem noch das Zeichen '#' mit ausgeben.
Dieser Wert gilt global für alle Kategorien. Jede einzelne kann mit dem Befehl **bn** individuell zugeschnitten werden (→ 10.2.6.3).
Der Wert `ks` kann mit Strukturbefehl `si` temporär verändert werden (→10.2.0).

kt **Kategorietextlänge begrenzen**

=0 gesamten Kategorietext ausgeben, keine Längenbegrenzung <0>

=n alle Kategorien pauschal beim n. Zeichen abschneiden (z.B. für Tabellen)

Auch dieser Wert gilt global. Zum Beschneiden einzelner Kategorien verwendet man die Manipulationsbefehle **e** und **b** (→ 10.2.6.3) (Es wurde noch kein Fall bekannt, wo man **kt** brauchte. Seine Begründung ist in Vergessenheit geraten, Abschaffung wird erwogen.)

10.2.6 Die Kategorieliste - das Kernprogramm beim Export

Das notwendige Beiwerk ist beschrieben, jetzt kommt der Hauptteil.
Alle Zeilen, die mit '**#**' oder '**!**' beginnen, sind **Exportbefehle** und bilden zusammen die **Kategorieliste**.
Es gibt zwei Sorten solcher Befehle:

- **Anweisungen** (mit Kategorienummer hinter dem Zeichen, z.B. `#20 ...`) und
- **Steuerbefehle** (Sprungbefehle und andere Ablaufsteuerungen, –➢10.2.6.4, z.B. `#t{ ... }, #+X, #-Y`).

Zwischen den Zeilen können, wie überall, Kommentare (d.h. mit Leerzeichen beginnende Zeilen) oder andere Parameterangaben stehen, ohne den Ablauf zu stören.
Die Zeilen der Kategorieliste werden von oben nach unten abgearbeitet, wenn nicht Sprungbefehle oder Unterprogrammaufrufe Verzweigungen verursachen.
Nur in diesem Teil der Parameterdatei **kommt es auf die Reihenfolge an**, denn daraus ergibt sich unmittelbar die Reihenfolge der Ausgabedaten. Alle anderen Teile der Parameterdatei können an beliebigen Stellen stehen. Kommt ein Befehl mehrfach vor, z.B. **`ks`**, gilt nur der letzte! Übersichtlichkeit läßt sich folglich nur erzielen, wenn man selber darauf achtet.

So wird ein Datensatz exportiert:

1. Wenn es *keinen* **`ak`**-Befehl gibt, geht es gleich bei Schritt 3**.** los.
Das Programm nimmt sich den ersten **`ak`**-Befehl vor (→10.2.1) und bestimmt das erste sich daraus ergebende Datenelement - eine Kategorie oder ein Teilstück einer solchen. Dieses merkt es sich als Sonderkategorie **`#u1`** (und evtl. **`#u2`**, wenn der Kopfbefehl noch eine Zusatzkategorie angibt). Und weiter mit 3.
Wenn der erste **`ak`**-Befehl nichts ergibt, wird der nächste genommen.

2. Wenn zu dem **`ak`**-Element ein Sprungbefehl **`+M`** gehört, wird in der Kategorieliste nach der Sprungmarke **`#-M`** gesucht (→10.2.6.5).

3. Wenn bei dem **`ak`**-Element **kein Sprungbefehl** steht oder wenn gar **keine** **`ak`**-Zeile vorhanden ist, begibt sich das Programm zur ersten Zeile der Kategorieliste, d.h. zur ersten mit '#' oder '!' beginnenden Zeile. (→10.2.6.1).

4. Die Export-Bearbeitung beginnt mit der so ermittelten Zeile der Kategorieliste und arbeitet die Zeilen der Reihe nach ab. Wenn dabei die Sonderkategorie **`#u1`** vorkommt, hat sie den vorher ermittelten Inhalt, **`#ch`** enthält außerdem den gesamten Kategorietext, aus dem **`#u1`** entnommen ist.
Die Abarbeitung einer einzelnen Anweisungszeile kann sehr aufwendig sein, wenn diese mehrere Manipulationsbefehle enthält. Im nächsten Abschnitt wird der Aufbau einer solchen Zeile gezeigt und ihre Abarbeitung dann erklärt.
Die Anweisungen beziehen sich immer auf eine zu bearbeitende Kategorie oder Variable. Dazwischen können **Steuerbefehle** stehen für Bedingungsprüfungen, Sprünge, Unterprogrammaufrufe (→ 10.2.6.4/5).

5. Die Bearbeitung endet, wenn die letzte Zeile erledigt ist, **oder** wenn der Endebefehl **`#+#`** oder der Abbruchbefehl **`#+-`** auftritt, **oder** wenn ein Sprung auf eine nicht existierende Marke verlangt wird.
Die Ausgabe erfolgt nicht unmittelbar: zuerst wird der gesamte Ausgabetext intern aufbereitet, erst am Ende werden die entstandenen Zeilen hintereinander, durch **`ze`** getrennt, ausgegeben. Kompliziert, aber sonst könnten z.B. einige der Strukturbefehle nicht funktionieren (→ 10.2.0).

6. Wenn es hierarchische **Untersätze** gibt, werden diese im Anschluß an den Hauptsatz einzeln durchgearbeitet, und zwar beginnt das Programm jeweils mit der ersten Zeile der Kategorieliste, also nicht wieder bei der Sprungmarke! Ist für die Unteraufnahmen eine Sonderbehandlung nötig, sollte man an den Anfang der Kategorieliste den Befehl `#hi +U e0` schreiben, wobei dann der Abschnitt für die Sonderbehandlung mit Sprungmarke `#-U` beginnt. (Vgl. dazu auch Param. **`ag`**, → 10.2.1).

7. Das nächste Element, das sich aus den **`ak`**-Zeilen ergibt, wird bestimmt und unter **`#u1`** zwischengespeichert, und der nächste Exportsatz wird entsprechend erzeugt, d.h. es geht wieder bei Nr. 2. weiter, nun mit einem neuen Inhalt in **`#u1`**.

8. Wenn es keine **`ak`**-Befehle gibt oder wenn keines der in den **`ak`**-Zeilen definierten Elemente im zu exportierenden Datensatz vorhanden ist, beginnt das Programm mit der ersten Zeile der Kategorieliste und produziert genau einen Exportsatz, es sei denn, man hat `ad=0` gesetzt – dann kommt für den betreffenden Satz nichts heraus.

10.2.6.1 Anweisungen (= Zeilen, die mit # oder ! anfangen)

Eine Anweisung bezieht sich auf eine bestimmte Kategorie oder Variable (→ 10.2.6.2) und beschreibt , **wie** diese auszugeben ist, was **vorher** und was **danach** passieren soll. Zwischen den Anweisungen können Zeilen mit Steuerbefehlen stehen, die Verzweigungen und andere Aktionen auslösen (→ 10.2.6.4/5).
Jede Anweisung beginnt auf Position 1 einer eigenen Zeile und sieht schematisch so aus:

`Kxx[f] [BS] [MB] [#kt1 z1 #kt2 z2 #kt3 z3 ... #ktn zn]`

Kategorie Bedingter Spr . Manipulation Bedingte Postfixe

Sehr wichtig: Zwischen den einzelnen Elementen muß jeweils **genau ein Leerzeichen** stehen.
Es darf nicht `#kt1` direkt auf `Kxx` folgen, dann muß zumindest `0` dazwischen.
Die eckig geklammerten Teile können entfallen. Im einfachsten Fall besteht eine Anweisung also nur aus der Angabe **`#xx`** für eine auszugebende Kategorie, d.h. die einfachste Parameterdatei ist nichts als eine Liste der auszugebenden Kategorien. Die einzelnen Teile bedeuten:

`K` = **`#`** bzw. **`!`** Kategorie #xxf nehmen, umwandeln per Normal- bzw. Alternativtabelle (→ 10.2.4)
oder **`#^`** bzw. **`!^`** wenn zusätzlich der '**^**' steht, wird die Kategorie #xxf aus der Hauptaufnahme entnommen, falls gerade ein hierarchischer Untersatz in Arbeit ist.
oder **`#_`** bzw. **`!_`** wenn zusätzlich der '_' steht, wird die Kategorie **nur** aus der Unteraufnahme genommen, wenn gerade eine solche in Arbeit ist
oder **`#↓`** bzw. **`#↓`** mit dem Zusatzzeichen ↓ erreicht man, daß die Kategorie in den evtl. vorhandenen Unteraufnahmen gesucht wird, wenn sie in der Hauptaufnahme fehlt.

`xxf` ist eine gültige Kategorie-Nummer (das Folgezeichen 'f' kann fehlen), oder auch eine Sonderkategorie (→ 10.2.6.2). Wenn für **`f`** ein **`.`** steht, wird die erste vorhandene Kategorie #xxf genommen. (Nützlich, wenn man nicht weiß, welche besetzt sein wird.)

`BS` der **bedingte Sprung** wird benötigt, wenn es **nach** der Verarbeitung von #xxf nicht mit der nächsten Zeile weitergehen soll. (Wenn **`#xxf`** unbesetzt ist, passiert nichts.) Für **`BS`** kann stehen:

a) ein **Sprungbefehl** der Form **`+M`** dann geht's anschließend hinter der Sprungmarke **`#-M`** weiter. Sie kann auch oberhalb der Anweisung stehen.

b) der **Wiederholungsbefehl** **`++`** um Folgekategorien von #xxf gleichfalls abzuarbeiten

c) der **Endebefehl** **`+#`** der laufende Exportsatz wird abgeschlossen.

d) der **Abbruchbefehl** **`+-`** der laufende Exportsatz wird nicht ausgegeben. Im Unterprogramm bewirkt das nur Abbruch des UP.

e) ein **Sprungbefehl** der Form **`+#yyf`**
d.h. statt der Sprungmarke **`M`** steht hier eine Kategorienummer **`#yyf`**, die in der Kategorieliste irgendwo weiter unten vorkommt. Es muß eine zulässige Nummer sein, die Kategorie braucht aber nicht besetzt zu sein. (Diese Form spart Zeit und eine Sprungmarke!) Bis zu dieser Stelle wird gesprungen, wenn #xxf im aktuellen Satz besetzt ist und MB ausführbar ist.

`MB` ist ein sogenannter **Manipulationsbefehl** (auch **Vorbearbeitungsbefehl**), oder eine Kette derartiger Befehle. (Diese Befehle sind die mächtigsten Werkzeuge der Exportsprache.)
Mit solchen Befehlen kann man den Kategorietext in vielfacher Weise "vorbehandeln" lassen, bevor er ausgegeben wird. Diese Befehle werden in einem eigenen Abschnitt beschrieben (→ 10.2.6.3). Unter anderem kann hier ein **Präfix** gesetzt werden: eine Zeichenfolge, die vor den eigentlichen Kategorietext zu setzen ist.

`#kt1 z1 ... #ktn zn` [Diese "Bedingten Postfixe" sind nur sehr selten notwendig!]
Die Kategorienummern **`#kti`** müssen in aufsteigender Reihenfolge (laut CFG) angeordnet sein.
`zi` sind Nummern (0..127) von **Zwischenteilen**, die als Postfixe benutzt werden sollen
oder **Unterprogramm-Aufrufe**, z.B. **`>U`** (→10.2.6.5)
aber nicht Textangaben der Form `P"..."` wie bei direkten Postfixen! (→ 10.2.6.3)
Bedingung: Wenn #kti vorkommt, darf auch zi nicht fehlen.
Hinweis:
Wenn die #kti-Angaben fehlen, wird hinter die Kategorie die Zeichenfolge `ke`, das Standard-Postfix, gesetzt (→ 10.2.5). Um das zu verhindern, muß man die Zeile mit **`#zz 0`** abschließen.
Wie die bedingten Postfixe abgearbeitet werden, wird auf der nächsten Seite unter 7. und 8. beschrieben.

Eine Anweisungszeile wird in folgenden Schritten abgearbeitet:

1. Stelle fest, ob Kategorie **#xxf** im aktuellen Satz vorhanden ist, sonst ignoriere diese Zeile völlig (auch den bedingten Sprung) und bearbeite die nächste. (Wenn hinter '#' eines der drei Zeichen **^**, **_** oder **↓** steht, wird die Kategorie evtl. in angrenzenden Datensatzteilen gesucht (s.o.)).
Mache eine **Kopie** des Inhalts von Feld `#xxf` ab Zeichen `ks`. Diese Kopie gilt für alles weitere als **Arbeitstext** (d.h. zunächst wird noch nichts ausgegeben - der Arbeitstext muß evtl. noch vorbearbeitet werden:)

2. wende Befehlskette **MB** auf den Arbeitstext an (→ 10.2.6.3); ist ein Befehl dieser Kette nicht ausführbar, so wird die Befehlszeile abgebrochen, ohne daß etwas passiert, und zur nächsten übergegangen, und auch der bedingte Sprung wird ignoriert. Durch jeden einzelnen Manipulationsbefehl wird aus dem Arbeitstext ein neuer, veränderter Arbeitstext. Wenn einer der Befehle `bn, Bn, en, En, s` oder `r` in **MB** vorkommt, wird unmittelbar vor seiner Ausführung die Zeichenumcodierung erledigt, denn dabei könnte sich die Länge des Textes ändern. Sonst kommt die Umcodierung erst nach der MB-Abarbeitung:

3. Führe die Zeichenumcodierung aus, und zwar über die Normaltabelle (wenn **K** = '#') bzw. über die Alternativtabelle (wenn **K** = **'!'**) (→ 10.2.4),

4. Erst jetzt wird wirklich etwas ausgegeben, und zwar hintereinander in dieser Reihenfolge:
1. das bedingte Postfix, das sich aus der *voran*gegangenen Anweisung ergibt (siehe unten),
2. die indirekten Präfixe der aktuellen Kategorie
3. das Ergebnis der aktuellen Zeile (die letzte Version des Arbeitstextes: darin sind direkte Prä- und Postfixe enthalten, also Befehle **p"abc"** und **P"xyz"**.
4. das indirekte Postfix (aus dem evtl. vorhandenen Manip.Befehl **P{ ... }**)

Wenn jedoch infolge der Manipulationen sich ein Abbruch ergibt (weil z.B. eine Manipulation nicht ausführbar ist), gehe zur nächsten Kategoriezeile (ignoriere also auch den bedingten Sprung) und beginne damit wieder bei Punkt 1., andernfalls:

5. wenn ein bedingter Sprung **BS** angegeben ist, führe diesen aus. Dabei gibt es 3 Möglichkeiten:
a) hat **BS** die Form **+#yyf**, suche in der Kategorieliste **weiter unten** die Zeile, die mit der Kategorienummer **#yyf** anfängt, und mache dort weiter,
b) wenn **BS** die Form **+M** hat, fahre fort hinter der **Sprungmarke** **#-M**; diese **kann auch oberhalb** der aktuellen Zeile liegen (Achtung: Schleifengefahr!)
In beiden Fällen gilt: wird das Sprungziel nicht gefunden, endet die Verarbeitung, d.h. der Ausgabesatz gilt als fertig. Dann wird noch Parameter **ae** ausgegeben (→ 10.2.1).
Sonderformen: **+#** wirkt als **bedingter Endebefehl**, **+-** als **Abbruchbefehl**.
c) wenn **BS** = ++ ist, **wiederhole** die Anweisung, wenn die Kategorie #xx **mehrfach** vorkommt; aus den Postfix-Angaben ergibt sich, welche Zeichen zwischen die einzelnen Kategorien zu setzen sind. Ein Präfix (evtl. in **MB** enthalten) kommt nur vor die erste, vor die weiteren kommt das, was mit Manipulationsbefehl **m** dafür definiert ist.

6. wenn **kein** **BS** angegeben ist, fahre mit der auf #xxf folgenden Exportzeile fort.
Das hinter der Kategorie auszugebende **bedingte Postfix** steht jetzt noch nicht fest, sondern zuerst ist festzustellen, welche Kategorie als nächste drankommt:

7. Arbeite (nach Ausführung von **BS**, wenn ein solcher vorhanden ist) die nachfolgenden Zeilen der Kategorieliste durch und finde die nächste ausführbare Zeile, d.h. diejenige, deren **#xxf** im aktuellen Satz vorkommt, und deren **MB** ein Ergebnis liefert. Bevor dann deren Text ausgegeben wird, kommt das bedingte Postfix, das von der vorigen Anweisung noch ansteht:
Ist die aktuelle Kategorie **<= #kt1**, dann ist Zwischenteil **z1** (bzw. das durch z1 bezeichnete Unterprogramm) das bedingte Postfix, ist sie **> #kt1, aber <= #kt2**, dann **z2** usw.
(Deshalb sind die #kti in aufsteigender Reihenfolge, der .CFG entsprechend, anzuordnen.)
Wenn ein Befehl **#t...** kommt, wirkt er wie eine "unendlich kleine" Kategorie, d.h. **z1** wird wirksam.
Sonderfall: **#zz** anstelle #ktn bedeutet: unbedingte Ausgabe von zn, wenn die nachfolgende Kategorie oberhalb aller #kti liegt.
Wenn die nächste gefundene Kategorie eine **#u**-Variable ist ODER wenn statt einer Kategorie ein **#t**-Befehl (Direkt-Ausgabebefehl) kommt, dann wird in jedem Fall Zwischenteil **z1** genommen.
Mit **#zz 0** erreicht man, daß **kein bedingtes Postfix** ausgegeben wird, auch nicht **ke**. Folglich **muß** man **#zz 0** angeben, wenn **ke** gesetzt ist, aber im konkreten Fall entfallen soll.

8. Fehlt `#kt1 ...`, d.h. sind die bed. Postfixe weggelassen, so wird statt dessen der Parameter **ke** benutzt (→ 10.2.5). Dies geschieht auch, wenn nach `#xxf` als nächste eine Kategorie bearbeitet wird, die größer als `#ktn` ist. Und nun geht das ganze mit der neuen Zeile wieder bei 1. los.

In bestimmten Fällen ist es sinnvoll, alle Kategorien einfach hintereinander in der internen Reihenfolge, mit immer gleicher Interpunktion auszugeben. Ein Beispiel wäre der Abdruck der erfaßten Daten mit Kategorienummern, um eine Liste zum Korrekturlesen zu haben. Dafür gibt es einen sehr einfachen Befehl:

**Pauschalexport** des gesamten Datensatzes, evtl. mit Ausnahme bestimmter Kategorien (siehe unten)
Wenn diese Zeile innerhalb einer Kategorieliste auftritt, werden an dieser Stelle alle Kategorien hintereinander ausgegeben, getrennt jeweils durch den Parameter **ke**.
(Normalerweise setzt man **ke=C** (= Zeilenvorschub).)
Es kann natürlich sein, daß bestimmte Kategorien weggelassen werden sollen. Diese gibt man auf jeweils eigenen Zeilen mit dem Befehlszeichen '/' an:, z.B.

/39 bedeutet: #39 weglassen
/27. alle #27er Kategorien (beliebige Folgezeichen) weglassen
/852 #85 mit Folgezeichen 2 weglassen
/4.2 alle #4er Kategorien mit Folgeziffer 2 weglassen
/9 oder **/9.** alle #9er Kategorien weglassen

In solchen Befehlszeilen dürfen **keine Kommentare** stehen! Gesamtlänge dieser Zeilen 255 Zeichen.
Die Parameterdatei E-0.APR zeigt, wie man mit diesen Mitteln eine Ausgabe aller Kategorien in interner Reihenfolge mit Nummern erreicht. Das ergibt eine mit anderen Programmen leicht einlesbare Textdatei. (In anderen Zusammenhängen wird so etwas oft "ASCII file" genannt). Auch in den Dateien S-LANG.APT und I-0.APR (früher PA.APR) wird vom Pauschal-Exportbefehl Gebrauch gemacht.

#L **Pauschalexport** aller Kategorien, aber mit Feldbezeichnungen, und zwar denen, die in der CFG definiert sind. Das funktioniert nur, wenn eine CFG der neueren Art (ab V13) vorhanden ist (→ Anh. A.1).

10.2.6.2 Sonderkategorien

An der Stelle von `#xxf` in der Anweisungszeile können verschiedene Elemente vorkommen, die keine echten Kategorien sind, die aber genauso behandelt werden können.

#ax **Aktuelle Indexdatei und Register**, z.B. d4 oder e1, : für 10 und ; für 11
Diese Sonderkategorie gibt es erst ab V27.2 wegen der MultiX-Funktion.

#cc **aktuelle Kategorie** ("current category") Zu Empfehlen nur in Unterprogrammen, sonst unsicher.
für **#cc** wird der Wert der zuletzt aufgerufenen Kategorie angenommen und zwar der noch unveränderte Text. Insbesondere ist dies in Unterprogrammen (→10.2.6.5) nützlich: in **#cc** steht dann der Wert der zuletzt vor dem UP-Aufruf benutzten Kategorie. Daher lassen sich UPs auch kategorieunabhängig formulieren. Innerhalb eines UP bleibt der Wert von `#cc` erhalten, auch wenn dort zunächst andere Kategorien benutzt oder noch weitere UPs aufgerufen werden. Interessant ist dies im Zusammenhang mit dem Steuerbefehl **Indikatoraktion** (→ 10.2.6.4).

#cca **aktuelle Kategorie** incl. # und Kategorienummer Zu empfehlen nur in Unterprogrammen, sonst unsicher. Ansonsten dasselbe wie #cc. Bequem, wenn man im UP mal die Nummer braucht. **ks** hat keine Wirkung.

#ch **aktuelle Kopfkategorie** ("current heading")
Fast dasselbe wie **#u1** , aber: unter **#ch** hat man jederzeit den gesamten Text der mit dem aktuellen ak-Befehl ausgewählten Kategorie, während **#u1** nur den separierten Bestandteil enthält!
Von einem Kopfbefehl **ak=4." = "+G** ausgehend, hat man unter Sprungmarke **#-G** mit **#u1** nur den jeweiligen Ausschnitt und weiß nicht, woher er stammt, unter **#ch** steht jedoch die gesamte Kategorie. Das ist z.B. wichtig für Indikatorprüfungen (→ 10.2.6.3).
Bei einfachen Kopfbefehlen (ohne Trennvorschrift) sind **#u1** und **#ch** gleich.

#dt **Datum** in deutscher Form, z.B. `Mo., 24. Sep 1990` . Mit dem Manipulationsbefehl `b5` oder mit `b", "` kann man z.B. erreichen, daß der Wochentag wegfällt.

#dts **Datum/Uhrzeit in Sortierform** : für Sortieraufgaben wichtig. Dasselbe Datum würde mit Uhrzeit in der Form **YYYYMMDD/hh:mm:ss** erscheinen, also z.B. `19920819/13:32:55`. Mit Hilfe der Manipulationsbefehle hat man nun alle Möglichkeiten, Datum und Uhrzeit in jeder Weise umzugestalten: z.B. würde `#dts e4 p"(" P")"` das Jahr nehmen und einklammern.

#fna **Dateiname** (File Name) der aktuellen .ALD- bzw. .ALG-Datei

#mv **Multi-Volume Indicator** : diese Kategorie existiert, wenn es sich um ein mehrbändiges Werk handelt. Sie enthält dann die Bandnummer des ersten Bandes, genauer: den Inhalt der ersten vorkommenden Kategorie #01. Sinnvoll ist #mv, weil man damit innerhalb der Kategorieliste feststellen kann, **ob** es sich um ein mehrbändiges Werk handelt. Der Befehl

#mv +K e0

würde einen Sprung zur Sprungmarke `#-K` auslösen, wenn es sich bei dem gerade bearbeiteten Satz um einen hierarchischen handelt. In KAT.API wird davon Gebrauch gemacht: wenn auf Stufe #00 die #75 besetzt ist, aber nicht die #76 (Verlag, aber kein Jahr), dann soll keine Verlagseintragung für Index 6 gemacht werden, denn die Erscheinungsjahre befinden sich dann bei den Bänden und erzeugen von dort aus die richtigen Einträge des Typs "Verlag:Jahr".

#nr **Satznummer** : wenn eine .ALD-Datei exportiert wird (mit PRESTO oder SRCH), ist #nr die interne Satznummer der Datenbank, bei .ALG-Dateien dagegen wird eine fortlaufende Nummer, beginnend mit **an** am Dateianfang (→ 10.2.1), automatisch gebildet.

#nra **Satznummer des nachgeladenen Satzes** : wenn in dem Moment der aktuelle Satz einer ist, den man mit einem Nachladebefehl geladen hat (→ 10.2.6.7). Sonst ist `#nra` unbesetzt.

#op **Bearbeiter** ("Operator"). Wenn beim Programmstart die Option `-O Meyer` gegeben wird, hat die Sonderkategorie den Inhalt `Meyer` .

#p0 Seitenzahl der aktuellen Seite, aber nur wenn sie gerade ist, sonst leer

#p1 Seitenzahl, wenn sie ungerade ist, sonst leer. Zu verwenden für Seitenkopf oder -fuß, wenn man die geraden Nummern rechts und die ungeraden links plazieren will. (Beispiel in P-0.APR)

#p2 Seitenzahl (ohne Berücksichtigung von gerade und ungerade)

#pa aktuelle Seitenzahl von Parameter-Set 1. Bekanntlich kann man im Exportprogramm (SRCH) mehrere Exporte parallel fahren. Wenn nun der erste Export (Parameter-Set 1) eine gedruckte Liste mit Seitenzahlen erstellt, kann dazu als zweiter Export (Set 2) eine sortierfähige Namensdatei entstehen, die die Seitenzahlen der Hauptliste mit enthält. Daran kann sich das Sortieren der Namensliste und schließlich eine Druckausgabe eines Namensregisters mit Angabe der Seitenzahlen des Hauptteils anschließen. Beispiel in A-0.APR .

#pz0 Nummer der aktuellen Zeile, auf der gerade gedruckt wird (beginnend bei 1)

#pz1 Anzahl insgesamt produzierter Zeilen für den aktellen Datensatz

#pz2 Anzahl bereits erzeugter Zeilen auf der aktuellen Seite (wenn `zm >0` ist).

#pzp Position in der aktuellen Zeile, wo gerade geschrieben wird

#u1 Sonderkategorien z.B. für Kartenköpfe

#u2 Die Kategorien #u1 und #u2 spielen eine Sonderrolle. Wenn man sie von Hand eingibt, stehen sie vor der Kategorie #00. (Das kann man nutzen, um von vornherein sortierfähige Sätze zu haben.) Wenn in der Kategorieliste #u1 und #u2 vorkommen, werden sie jedoch temporär mit den Inhalten gefüllt, die sich aus der aktuellen ak-Zeile ergeben (→ 10.2.1) bzw. die im Befehl `#dxxf/yyf` angegeben wurden (→ 3.3, Befehl #d).
Sonderfall: Wenn keine **ak**-Befehle vorhanden sind, #u1 und/oder #u2 jedoch wie normale Kategorien im Datensatz belegt sind, dann sind sie auch wie normale Kategorien exportierbar.
Der Sinn ist dieser: wenn man eine Sortierdatei hat, dann sind in der Regel #u1 und #u2 mit Sortierbegriffen gefüllt. Diese können dann gleich für den Druck benutzt werden. Z.B. in den Dateien P-*.APR gibt es deswegen keine **ak**-Befehle.
#u1 und #u2 sind spezielle **Anwendervariablen**:

#u*xy* **Anwendervariable**: Diese werden erzeugt durch die Manipulationsbefehle **a***xy* und **A***xy* sowie **=***xy* (→ 10.2.6.3, Typ 2), oder durch Vorbesetzung mit der Startoption **-U***xyTEXT*: dadurch gelangt *TEXT* beim Start in **#u***xy* . In a99 werden sie ständig in FLEX-Befehlen verwendet. Diese Variablen lassen sich wie normale Kategorien exportieren. Besonders wichtig sind sie in Fällen, wo man Inhalte eines Satzes aufbewahren will, um sie bei der nächsten wieder zu verwenden. Solche Anforderungen treten auf, wenn man z.B.

- Listen druckt, in denen ein sich wiederholender Kopf nicht jedesmal ausgedruckt werden soll, sondern ab dem zweiten Mal ersetzt durch etwas wie "***" (Beispiel: P-NORMAL.APR),
- Rechnungen durchführt, wobei Zwischenergebnisse aufbewahrt und von einem Satz zum nächsten jedesmal fortgeschrieben werden müssen (→ 10.2.6.3, Befehle **=, a, A, x**). (Beispiel: R-0.APR)

a99/alcarta: Variablen mit *x*=X,Y,Z haben Sonderfunktionen ("Flip-Variable"), solche mit *x*=v werden oft in System-FLEXen benutzt. Die aktuelle Belegung in einer Sitzung kann man jederzeit sehen, wenn man mit Alt+r auf den Reservespeicher umschaltet, dort stehen diese Variablen. Sie gelten für alle gelandenen Parameter gemeinsam UND für FLEX.

#u0*k* (*k* = 0..3) #u00 enthält die **Anzahl der Felder** des aktuellen Satzes, #u01 das **erste** Feld, #u02 das jeweils **nächste** (zum Programmieren von Schleifen!), und #u03 das **letzte**. Sobald man (in einer Schleife) das letzte Feld überschritten hat, ist #u02 nicht mehr vorhanden.

#ui0 Letzte Nutzereingabe im Register; verwendbar für Flips

#uxa Diese Sonderkategorie enthält die **Indexzeile**, von der aus zuletzt zugegriffen wurde.

#uxb Enthält die Nummer des aktuellen Registers, also 1, 2, ... 9 oder : oder ;

#uxc Enthält die Kurzzeile, über die zugegriffen wurde – falls der Zugriff so stattfand

#uxd Die Nummer (1,2,3) der aktuellen Datenbank, wenn mehr als eine beim Start angegeben wurden (nur DOS)

#uxq Hier steht die **Antwort** auf den letzten **#q**-Befehl (→ 10.2.6.4).

#ux*i* (*i* = 1..9 : ;) Diese Sonderkategorie enthält die **Indexzeile**, die mit dem letzten Nachladebefehl im Index *i* gefunden wurde (→ 10.2.6.7). Besonders wichtig bei Nachlademodus 8 und 9.
Tip: Die #u-Variablen kann man in ***a99*** mit Alt+r, im PRESTO-Editor mit Befehl #a betrachten (→ Kap.3.5.3).

Für ***a99/alcarta*** gibt es noch weitere: Übersicht anfordern mit **h ac10-5=Sonderkategorien**

#uzo Hier steht 000 drin bzw. 001, falls der aktuelle Satz ein online- bzw. offline-Satz ist.

Wenn man in einem FLEX den Befehl **var zc** gegeben hat, steht anschließend in der "internen Variablen" die aktuelle Anzeigezeile (wo sich die Schreibmarke befindet) , und

#uzC dieselbe Zeile, aber __ (2x Unterstrich) an der Stelle, wo die Schreibmarke sich befindet

#uzD Fliptext, falls Schreibmarke innerhalb eines solchen

#uzF Flipbefehl (zugehörige Kategorie #uZi, einschl. der Nummer, also #uzF#uZi...)

Sonderkategorien für mehrbändige Werke und Stücktitel

Die folgenden zwei Sonderkategorien sind nur in hierarchischen Untersätzen belegt:

#gt Gesamttitel: bei Stücktitel-Aufnahmen von Bänden (auf beliebigen Hierarchiestufen!) wird für #gt der Sachtitel #20 der Hauptaufnahme (Stufe #00) mit der Verfasserangabe #40 und der Hierarchiegliederung #01/#02... dieses Bandes eingesetzt (funktioniert nur bei A.CFG!).

#hi Hierarchie-Gliederung in Sortierform bzw. in Druckform

#hi1 Hat man z.B. folgende Gliederung einer hierarchischen Aufnahme (nur angedeutet):

```
#00
#20 übergeordneter Gesamttitel
...
#01 1 = Bd. 1          (Sortierform = Druckform)
#20 Sachtitel von Band 1
...
#01 5 = Bd. 5
...
#02 1 = Teil 1
...
#02 3 = Teil 3
...
```

dann gilt, während der letzte Teil abgearbeitet wird, #hi 5,3 und #hi1 Bd.5,Teil 3 . Sollen die einzelnen Zählungselemente durch '/' statt Komma getrennt werden, müßte man den Hierarchie-Separator mit dem Befehl kh=/ ändern (→ 10.2.5).
Ein Tip: wenn man das Postfix **P{ ". " #250 }** oder ein entsprechendes Zwischenteil hinter #hi verwendet, werden Folgezeilen sinnvoll eingerückt.
#hi enthält nur dann etwas, während ein Untersatz verarbeitet wird. Folglich kann man bedingte Sprünge konstruieren: #hi +V e0 bewirkt: wenn der aktuelle Satz ein Untersatz ist, erfolgt Sprung nach #-V ohne Ausgabe von #hi (e0 setzt die Länge auf 0). #hi existiert nicht, wenn ag=0 gesetzt wurde, die Untersätze also selbständig abgearbeitet werden, als wären sie Hauptsätze.
Gibt man im Editor den Druckbefehl #d, wenn #02 3 gerade der aktuelle Untersatz ist, so wird eine "Stücktitelkarte" gemacht, d.h. die fehlenden Kategorien werden aus den hierarchisch übergeordneten Einheiten zusammengesucht. #gt wird dabei benutzt, um den Titel #20 des übergeordneten Gesamtwerks auf der Stücktitelkarte mit auszugeben. Er erscheint dann bei diesem Beispiel in der Form

```
Übergeordneter Gesamttitel / Verfassername ; 5,3.
```

Einschränkung: #gt funktioniert in dieser Weise nur, wenn das Standard-Kategorienschema A.CFG benutzt wird! Bei anderen Konfigurationen oder anderen Wünschen muß man den Effekt mittels Anwendervariablen nachbilden (→ 10.2.6.3) und/oder mit #^xx arbeiten, um Elemente aus der Hauptaufnahme "herunterzuholen". (→ 10.2.6.1). Studieren Sie die Parameterdatei **D-1.APR**, um die Zusammenhänge genau zu verstehen, und probieren Sie alles damit aus.

10.2.6.3 Manipulation : Vorbehandlung von Feldinhalten

Nach der bisherigen Beschreibung sieht es so aus, als könnten Kategorien nur als Ganzes oder in globaler Verkürzung (s. Parameter **ks** und **kt**) ausgegeben werden. Es gibt aber sehr viele Möglichkeiten, einen Kategorietext als **Arbeitstext** vorzubehandeln: z.B. kann man vorn und hinten etwas abschneiden, Teile ausschneiden und/oder Textelemente davor und dahinter setzen, den Inhalt prüfen, zwischenspeichern oder mit Zahlenwerten verrechnen. Das alles macht man mit den sog. **Manipulationsbefehlen**, die in den Anweisungszeilen (→ 10.2.6.1) zwischen dem bedingten Sprung **BS** und den bedingten Postfixen **#kt1 z1 ... #ktn zn** stehen können.

Fast alle Manipulationsbefehle können in einer Befehlsfolge sogar mehrfach auftreten und beliebig miteinander kombiniert werden. Man schreibt einfach mehrere Befehle hintereinander, immer genau ein Leerzeichen dazwischen. (Zwei Leerzeichen bewirken auch hier, daß der Rest der Zeile als Kommentar behandelt, also ignoriert wird. Aufpassen!)

Für das folgende sind weiterhin die Bezeichnungsregeln (**n k X x CS ...**) von 10.2 gültig.

5 Typen von Befehlen kann man unterscheiden:

Typ 1 : Befehle für sofortige Veränderungen am Arbeitstext (AT),
Typ 2 : arbeitet mit Anwendervariablen (AV),
Typ 3 : veranlaßt Zeichenausgaben vor oder nach dem Arbeitstext,
Typ 4 : besteht aus Bedingungsprüfungen, die dann Reaktionen auslösen können, und
Typ 5 : Sonderaktionen, z.B. Vorschübe und Nachladungen.

*** heißt: Befehl nicht mehrfach in einer Zeile anwendbar**

Befehl	Typ	Wirkung	S.
!	5	nur zum Testen in PRESTO: Anzeige des aktuellen Arbeitstextes u.a. am Bildschirm	202
=*xy*	2	Vergleiche AT mit Inhalt von **#uxy** und speichere AT in **#uxy** gleich: nächster Befehl, ungleich: Ausgabe des AT und bed.Sprung	199
|im *	5	AT als Schlüssel nehmen, im Index **i** mit Modus **m** suchen, Satz laden. m=8 oder 9: gefundenen Schlüssel in #uxi laden	212
$x oder **▼x**	1	Unterfeld **x** aus AT herausholen und als neuen AT verwenden	198
~x	1	Unterfeld **x** aus AT entfernen (wenn es mehrfach vorkommt, nur das erste) (ab V15)	198
,"_X_Y_"	1	Lokale Ersetzung: ersetze im Arbeitstext X durch Y	197
a*xy*	2	AT + **#uxy** in **#uxy** speichern (AT nicht ausgeben)	199
A*xy*	2	**#uxy** + AT in **#uxy** speichern (AT nicht ausgeben)	199
bk	1	Beginne AT nach Position k. Wenn AT kürzer ist: Abbruch der Anweisung, keine Ausgabe, kein bedingter Sprung	196
bX	1	Beginne AT hinter Zeichenfolge **X**. Wenn **X** nicht vorhanden: Abbruch	196
Bk	1	Beginne AT nach dem k-ten Zeichen. Wenn AT kürzer, bleibt er unverändert.	196
BX	1	Beginne AT hinter Zeichenfolge **X**; wenn **X** nicht vorhanden, AT unverändert	197
C	5	Zeilenvorschub vor Ausgabe des AT (Param. **ze** ausgeben)	201
cX	4	Prüfe, ob AT die Zeichenfolge **X** enthält. Wenn ja: Anweisung fortsetzen (AT bleibt unverändert); wenn nein: Abbruch. Viele Sonderfälle!	201
dxy	2	Anwendervariable **#uxy** löschen	200
ek	1	Beende AT hinter dem **k**ten Zeichen (*Sonderfall* **k=0**, siehe **Z**)	197
Ek	1	wie **ek** , aber letztes Wort weglassen, wenn es sonst zerschnitten würde	197
eX	1	Beende AT vor Zeichenfolge **X**; wenn **X** nicht vorh., bleibt AT unverändert	197
EX	1	Beende AT hinter der Zeichenfolge **X**. (**X** nicht vorhanden? Dann AT unverändert)	197
fk / Fk	1	Entferne ASCII-Code **k** am Anfang/Ende des AT (evtl. mehrere)	197
fx / Fx	1	Entferne Zeichen **x** am Anfang / Ende des AT (evtl. mehrere)	197
fX / FX	1	Entferne alle in **X** enthaltenen Zeichen am Anfang/Ende des AT	197
ik,x	4	Anweisung nur fortsetzen, wenn auf Position **k** das Zeichen **x** steht (Position 1 = erste Kategorieziffer)	201

Befehl	Typ	Wirkung	S.
Ik,x	4	Anweisung nur fortsetzen, wenn auf Position **k** **nicht** **x** steht	201
Kz kz	1	"Matchcode" bilden, z.B. Titelschlüssel; für Index nutzbar. *Beispiel:* #20 K4 K2 K2 K1 : 4221-Schlüssel aus Titel bilden	203
l*xy*	5	Länge des aktuellen Arbeitstextes in #u*xy* schreiben	202
L*xy*	5	Label des im AT befindlichen Feldes aus CFG in die #uxy kopieren	202
l<k l>k	4	Ist AT kürzer/länger als k Zeichen? Dann weiter	201
mX m{CS}*	3	Direktes und Indirektes Mehrfachpräfix (Zeichenfolge/Steuerkette)	196
m>K *	3	Unterprogramm **#(K ... #)K** vor Mehrfachkategorie ausführen	201
mz *	3	Indirektes Mehrfachpräfix (Zwischenteil)	201
M *	5	AT als neues Datenfeld in den aktuellen Datensatz einordnen. Notwendig für "Glob. Manipulation" (Kap. 1.5) (AT muß mit gült. Kategorienummer beginnen)	202
N	5	Seitenumbruch vor Ausgabe des AT	202
p{CS} *	3	Indirektes Präfix: Steuerzeichenfolge **CS** vor fertigem AT ausgeben	200
P{CS} *	3	Indirektes Postfix: Kette **CS** ausgeben, nachdem AT ausgegeben ist	201
p>K	3	Unterprogramm **#(K** aufrufen, bevor AT ausgegeben wird	200
P>K *	3	Unterprogramm **#(K** aufrufen, nachdem AT ausgegeben ist	201
pX	1	Direktes Präfix: **X** vorn an den AT anfügen	196
PX	1	Direktes Postfix: **X** hinten an AT anfügen	196
pz	3	Indirektes Präfix: Zwischenteil Nummer **z** vor AT ausgeben	200
Pz *	3	Indirektes Postfix: Zwischenteil Nummer **z** nach AT ausgeben	201
R	1	AT rechtsbündig in aktuelle Zeile stellen (geht nur bei **zl>0**)	198
rk,x	1	AT **k**-stellig rechtsbündig machen, links mit **x** auffüllen	198
sk,x	1	AT **k**-stellig linksbündig machen, rechts mit **x** auffüllen	198
tk	1	**k** Zeichen vom Ende des AT entfernen	197
Tk	1	die letzten **k** Zeichen des AT als neuen AT nehmen	197
tX	1	vom AT hinten den mit **X** beginnenden Teil entfernen	197
TX	1	hinteren Teil des AT ab Zeichenfolge **X** als neuen AT nehmen	197
u u[] **U U[]**	1	Entferne die Übergehwörter bzw. zwischen **[]** eingeschlossene Teile; U genauso, aber erstes Wort dann in Großschreibung ändern	198
vk,x	4	Anweisung nur fortsetzen, wenn Zeichen auf Pos. k kleiner als x ist	201
Vk,x	4	Anweisung nur fortsetzen, wenn Zeichen auf Pos. k größer als x ist	201
w	1	AT als Kategorienummer interpretieren, deren Inhalt wird neuer AT	198
x"*OpW***"**	1	Rechenbefehl: AT als Zahl nehmen, Operation *Op* mit Wert *W* ausführen, das Ergebnis wird als neuer AT genommen	209
X *	5	Aus dem Arbeitstext einen Sondereintrag für satzübergreifende Suche machen	202
y0	1	Verhindert die Umcodierung des AT, dann Anweisung fortsetzen	198
y1 / y2	1	Codierung mit **p**- bzw. **q**-Befehlen sofort ausführen, Anweisung fortsetzen	198
y3/y4/y"n"	1	Texel-Ersetzungen ausführen / UTF-8 ignorieren (vb163/164) / bestimmte Ausnahmen	198
Z *	5	Unterdrückt die Ausgabe des AT (gleiche Wirkung wie **e0**)	202

Wichtig deshalb nochmal deutlich: der Originaltext eines Datenfeldes bleibt durch die Manipulationsbefehle unangetastet: das Programm erstellt sich einen **Arbeitstext**, das ist zunächst einfach eine Kopie des Feldtextes, und daran werden dann die Manipulationen ausgeführt. Durch jeden Befehl des Typs 1 **entsteht ein neuer Arbeitstext**, auf den dann der nächste Befehl wirkt. Nach Abarbeitung der Befehlsfolge wird schließlich automatisch der entstandene Arbeitstext ausgegeben, nicht der Originaltext. Dann wird der Arbeitstext sofort wieder vergessen, und mit der nächsten Befehlszeile (für das nächste auszugebende Datenfeld) wird ein neuer angelegt.

Typ 1 : Direkte Änderungen am Arbeitstext

Die wichtigsten zuerst: **direkte Präfixe** bzw. **Postfixe** sind Zeichenfolgen, die mit **p** bzw. **P** direkt angegeben werden, und die das Programm dann sofort vorn bzw. hinten an den Arbeitstext anfügt.
Umgekehrt gibt es alsdann eine Reihe von Befehlen, mit denen man vorn und hinten vom Arbeitstext Teile entfernen kann, um entweder diese Teile selbst oder den Rest als neuen Arbeitstext zu benutzen.

pX **Direktes Präfix** : Zeichenfolge X wird vor die aktuelle Kategorie gesetzt, d.h.: X wird mit dem
mX aktuellen Arbeitstext zu einer Zeichenfolge vereinigt. Der Arbeitstext besteht aus dem Text von #xxf, der zu diesem Zeitpunkt möglicherweise bereits von anderen Manipulationsbefehlen verändert ist.
mX wird nur bei der Abarbeitung von Mehrfachkategorien benutzt: die erste bekommt das mit p, die zweite und weitere das mit m angegebene Präfix. (→ 10.2.6.1 Befehl **++**)
Eine besondere Bedeutung hat das **Index-Präfix**, das sich formal aber nicht von einem "normalen" Präfix unterscheidet: es regelt die Zuordnung eines Schlüssels zu einem Register, z.B. **p"|7"** bedeutet: Einordnung in Register 7. (Siehe "Index-Sonderbefehle" → 10.2.1.3)

PX **Direktes Postfix:** wirkt analog zu **pX**, nur daß die Zeichenfolge X **hinter** den Arbeitstext gehängt wird.
Wenn man beides kombiniert, kann man z.B. ganz leicht etwas in Klammern einschließen:
#85 C p"(" P")" (s.a. *Zusatzmöglichkeiten* unter bX: z.B. p$a oder p"\31a" statt p"▼a"
Der Arbeitstext erhält vorn und hinten eine Klammer, und der Manipulationsbefehl C (siehe unten) bewirkt einen Zeilenvorschub. Ergebnis: der Serientitel wird in Klammern auf eine neue Zeile gesetzt.

bk **Beginn** bei einer bestimmten Position. *Achtung:* Zuerst wird, falls vorher noch nicht erfolgt, die Umcodierung durchgeführt, d.h. die p- bzw. q-Befehle, abhängig vom Zeichen # bzw. ! am Zeilenanfang.

Bk *Beispiel:* vom Erscheinungsjahr sollen nur die letzten 2 Ziffer ausgegeben werden. Wenn ks=5 gesetzt wurde und die Jahreszahl am Anfang des Kategorietextes steht, muß von da aus noch 2 Stellen nach rechts gegangen werden. Die Parameterzeile lautet dann:
`#76 b2 p" (" P") "` oder auch, mit Zwischenteilen: `#76 b2 10 #zz 11`
und im Druckbild erscheint: (82) ,wenn #76 1982 im Datensatz steht. (Dabei ist angenommen, daß man `10=" (" und 11=") "` als Zwischenteile definiert hat; 10 ist hier als Präfix, 11 als Postfix benutzt.
Ist k größer als die Länge der fraglichen Zeichenfolge, so wird bei **bk** weitergemacht, als sei die Kategorie nicht vorhanden, d.h. die Befehlszeile wird abgebrochen, bei **Bk** bleibt der AT unverändert.
Die Zahl **k** muß kleiner als 256 sein! Wenn man aber z.B. eine sehr lange #98 hat, und etwa den Teil ausgeben will, der hinter der Position 500 kommt, kann man schreiben: `#98 b250 b250 ...`
Anfang und Ende des zu druckenden Datenfeldes können auch noch auf andere Arten vorgegeben werden: man kann verlangen, daß das Druckfeld **hinter einer bestimmten Zeichenkombination beginnt** (b-Befehle) bzw. **vor dieser endet** (e-Befehle)

bX **beginne hinter X** : Wenn z.B. der Zusatz zum Sachtitel ("Untertitel") gesondert ausgegeben werden soll, und wenn man, RAK-gerecht, diesen immer mit " : " (Spatium Doppelpunkt Spatium) an den Hauptsachtitel angeschlossen hat, dann muß also die Ausgabe hinter der Zeichenfolge " : " beginnen:
`#20 b" : " 35 #zz 8` oder `#20 b" : " p"Zusatz: " P". - "`
wobei etwa `35="Zusatz: "` und `8=". - "` sein könnte. Aber nicht immer wird es ein einzelnes Zeichen sein, das anzusteuern ist. Es gibt daher noch eine Reihe von

Zusatzmöglichkeiten: (und diese gelten auch für die anderen M-Befehle: `BX, eX, EX, cX, pX, PX, tX, TX`.

- `b"[.,-:;!]"` Die eckigen Klammern bewirken, daß die darin eingeschlossenen Zeichen einzeln gesucht werden. Der Arbeitstext beginnt hinter dem ersten gefundenen Zeichen. Die Nebenwirkung ist, daß `b"["` und `e"["` nicht klappen - aber `b"[]"` und `e"[]"` tun es. Umgekehrt: `b"[]]"` und `e"[]]"` tun es nicht, dafür muß man (evtl. zusätzlich) `b"]"` bzw. `e"]"` schreiben.
- In **X** kann '?' als Maskierungszeichen vorkommen (näheres hinter Typ 5).
- Wenn **X** mit **~** beginnt und aus Kleinbuchstaben besteht, wird Groß/Klein im Arbeitstext ignoriert.
- Nicht eingebbare Zeichen in **X** mit `\nnn` codieren, z.B. `b"\31x"` statt `b"▼x"`.
- *Kurzschreibweise:* Statt `b"▼x"` ist auch `b$x` möglich, statt `b"▼"` auch `b$,` entsprechend für `B, e, E.`
- Mit `b"#uxy"` kann man prüfen, ob der Inhalt der Variable `#uxy` im Arbeitstext vorkommt. Falls ja, wird die Zeile fortgesetzt und der bedingte Sprung ausgeführt, sonst nicht.
- Mit `b">#uxy"` drückt man aus, daß der Arbeitstext größer als der Inhalt von `#uxy` sein soll (nicht zahlenmäßig, dafür gibt es die Rechenbefehle, sondern als Zeichenfolge).

- Wenn eines der Zeichen [< > ~ / " ? als solches gesucht werden soll, also nicht seine Sonderfunktion ausüben soll, muß man es in [ ] einschließen, z.B. `b"[[]"` .
- Wenn der Inhalt von `#uxy` so aussieht: `[abcd]`, also mehrere Zeichen in eckigen Klammern, dann wird mit `b"#uxy"` festgestellt, ob eines der Zeichen `abcd` im Arbeitstext vorkommt.
- Diese Art Bedingungsprüfungen sind auch bei den Befehlen `eX, cX, tX` und `TX` anwendbar.

In jedem dieser Fälle gilt: trifft die Bedingung nicht zu, so wird nichts ausgegeben, d.h. die ganze Befehlszeile, in der **bX** steht, wird ignoriert. Schreibt man statt dessen aber:

BX dann wird die Kategorie komplett ausgegeben; d.h. **BX** wird ignoriert, wenn **X** nicht im AT vorkommt.

ACHTUNG: bei bX, eX, BX und EX immer beachten, ob vorher schon umcodiert wurde (mit `y1` oder `y2`) und deshalb eines der Zeichen in X vielleicht verändert oder beseitigt wurde.

Auch das Ende der Aufbereitung kann durch eine Längenangabe oder durch Zeichenfolge festgelegt werden, letzteres ebenfalls auf zwei Arten:

ek **Ende** der Ausgabe bei einer bestimmten Position = Länge des Arbeitstextes auf k Zeichen setzen.
Um bei dem Beispiel zu bleiben: es könnte ja sein, daß in #76 hinter dem Jahr noch andere Angaben stehen. Um diese auch noch auszublenden, gibt man vorsichtshalber zusätzlich hinter b2 noch e2 an:

`#76 b2 e2 10 #zz 11` **oder** `#76 b2 e2 p"(" P")" #zz 0`

Das e2 bewirkt, daß nach dem zweiten ausgegebenen Zeichen die Ausgabe dieser Kategorie beendet wird. Hinter b und e können beliebige und verschiedene positive Zahlen stehen. Wenn die Zahl hinter e größer ist als die tatsächliche Länge des Arbeitstextes, bleibt dieser unverändert.

Sonderfall: **e0** : der Arbeitstext wird nicht ausgegeben, aber es wird so getan, als ob es so wäre. Nutzeffekt: z.B. eine bestimmte Kategorie, sofern sie vorhanden ist, durch ein immer gleiches Wort (Präfix) ersetzen. Was hinter **e0** noch kommt, wird nicht mehr ausgeführt. Der Parameter **ke** wird allerdings ausgeführt; wird dies nicht gewünscht: **#zz 0** als bedingtes Postfix anhängen.
Z.B. bewirkt `#91 +M e0 #zz 0` , daß ein Sprung nach #-M nur dann erfolgt, wenn #91 im aktuellen Datensatz vorhanden ist, #91 wird jedoch wegen e0 an dieser Stelle nicht ausgegeben. (Der Sonderbefehl `Z` bewirkt dasselbe wie `e0`.)
`#91 p"xyz" e3 #zz 0` bewirkt: es wird "xyz" ausgegeben, wenn #91 vorkommt, aber nicht der Text von #91 selbst. Umgekehrt, also mit
`#91 e0 p"xyz"` , geht es nicht, da hinter e0 nicht weitergemacht wird.

Ek Wirkung wie **ek**, aber es wird **am Ende zusätzlich das letzte Wort beseitigt**, wenn es durch die Verkürzung abgebrochen wurde. Diese Operation kann z.B. für Kopfzeilen sehr nützlich sein, wenn etwa sehr lange Kategorien vorkommen und man abgeschnittene Wörter im Kopf vermeiden möchte.

eX **beende den Text vor X** : Soll z.B. nur der Hauptsachtitel erscheinen, so setzt man für **X** wieder " : "
`#20 e" : "` dann wird **ab X das Ende beseitigt** (auch hier Maskierung mit '?' möglich.)
`#20 e"[.,:;!?]"` beendet den Arbeitstext vor dem ersten Interpunktionszeichen (s.a. **bX**).
Zusatzmöglichkeiten wie bei `bX.`

EX Ein 'E' statt 'e' würde das Ende der Zeichenfolge hinter " : " setzen, d.h. diese Zeichen würden dann stehenbleiben, also mit ausgegeben.

fx **Fx** **Unterdrückung** führender bzw. am Ende stehender Zeichen; 3 Varianten:
fk **Fk** Das Zeichen x bzw. der Code k bzw. jedes der in X vorkommenden Zeichen
fX **FX** wird am Anfang (**f**) bzw. am Ende (**F**) des Arbeitstextes beseitigt.
Das Zeichen x darf keine Ziffer sein, denn diese würde als Zahlenwert eines ASCII-Codes interpretiert. Also würde **f0** nicht gehen, es muß **f"0"** heißen oder **f48**. Dagegen könnte man mit **f$** durchaus führende Dollarzeichen beseitigen, genauso mit **f36** oder **f"$"**.
Die f-Befehle kümmern sich alle um **Einzel**zeichen, für Kombinationen gibt es die T-Befehle:

tk hinten k Zeichen wegnehmen ("trim") ; umgekehrt:
Tk die letzten k Zeichen ("Tail" = Endstück) als Arbeitstext nehmen
(wenn Arbeitstext kürzer als k Zeichen: kein Ergebnis, Abbruch)

tX suche vom Ende des Arbeitstextes aus die Zeichenfolge X und setze den Endpunkt davor (wenn X nicht vorkommt: Arbeitstext unverändert); auch hier ist das Gegenteil möglich:

TX nimm am Ende den mit X beginnenden Teil als Arbeitstext (evtl. noch **bX** ergänzen, um X wegzunehmen). Wenn X nicht auftritt, wird die Anweisung abgebrochen. *Zusatzmöglichkeiten* s. `bX`

,"_X_Y_" Lokale Ersetzung: im Arbeitstext wird überall Zeichenfolge **X** durch Zeichenfolge **Y** ersetzt. Statt '_' kann auch das Komma verwendet werden. Notwendig ist das dann, wenn innerhalb X oder Y das Zeichen '_' vorkommt! Innerhalb X und Y kann man ein nicht darstellbares Zeichen mittels `\nnn` codieren, `nnn`=ASCII-Wert des Zeichens, z.B. `\31` für das Unterfeld-Deieck.

$x oder **▼x** Das Unterfeld **x** wird im Arbeitstext gesucht, sein Inhalt wird zum neuen Arbeitstext. Abbruch, wenn es nicht vorkommt. Dieser Befehl ermöglicht einen sehr einfachen Umgang mit "Subfields". (**▼** und **$** können hier gleichwertig verwendet werden! Das tatsächliche Teilfeld-Steuerzeichen ergibt sich aus der .CFG und könnte sogar noch ein anderes Zeichen sein (→ A.1.3). *Vorteil:* unter UNIX kann man **$** besser eingeben. *Anm.:* Genauso wirkt `b$x e$` oder auch `b"\31x" e"\31"` oder `b"▼x" e"▼"`

~x Das Unterfeld **▼x** wird im Arbeitstext gesucht und entfernt; wenn es mehrfach vorkommt, nur das erste. Kommt es nicht vor, bleibt der AT unverändert. Auf den Bedingten Sprung hat dieser Befehl keine Wirkung.

w Der Arbeitstext wird als Kategorienummer interpretiert, die betreffende Kategorie wird neuer Arbeitstext. Wenn `$x` an die Kategorienummer gehängt ist, dann wird nur der Inhalt von Unterfeld `x` genommen. *Achtung:* Die Kategorienummer kann keine #u-Variable sein.

Beispiel: `#nr p"40 " e3 w` ausgegeben wird der Inhalt von #40
`#nr p"245$b" e5 w` Inhalt des Unterfelds **▼b** von `#245` wird Arbeitstext

Text links/rechts auf feste Länge auffüllen

sk Wenn der Arbeitstext kürzer als k Zeichen ist, wird das Ausgabefeld **mit Leerzeichen** bzw.
sk,x dem Zeichen **x auf die Länge k aufgefüllt**. Diese Operation ist sinnvoll, wenn Tabellen produziert werden sollen oder hinter einem Ausgabefeld ein Zwischenraum gewünscht wird. *Beispiele:*
s12 wenn Text kürzer als 12 Zeichen, rechts mit Leerzeichen auffüllen
s8,- wenn kürzer als 8 Zeichen, rechts mit Bindestrichen auffüllen

rk Wenn der Arbeitsttext kürzer als k Zeichen ist, wird er **nach links mit Leerzeichen auf k**
rk,x **Zeichen verlängert, d.h. in ein Feld der Länge k rechtsbündig** plaziert. Wenn **,x** angegeben ist, werden führende Leerzeichen durch das Zeichen x ersetzt. *Beispiel:*
`r7,0` 7stellig mit führenden Nullen ausgeben
`r10,$` 10stellig mit führenden $-Zeichen
Diese Möglichkeiten sind in Verbindung mit den Rechenbefehlen wichtig.
Um es zusammenzufassen: hinter b, B, e und E kann entweder eine Zahl oder eine in zwei Anführungszeichen eingeschlossene Zeichenfolge stehen (wenn man das zweite Anführungszeichen vergißt, können unerwünschte Ereignisse eintreten), hinter r und s muß eine Zahl stehen.

R (ohne Nummer!) Der aktuelle Arbeitstext wird **an den rechten Rand der Zeile (rechtsbündig)** ausgerückt. Wenn in der aktuellen Zeile nicht mehr genug Platz ist, wird die nächste genommen. Geht nur, wenn **zl>0.**

Nichtsortierzeichen, Umcodierung

u **Übergehwörter herausnehmen**. z.B. **u[]** : Wörter (Zeichenfolgen), die von den Zeichen *x* und *y*,
u*xy* umschlossen sind, werden weggelassen. Wenn *x* und *y* fehlen, wird für *x* das Nichtsortierzeichen genommen (→ Anh.A.1), für *y* das Leerzeichen (bei Nichtsortiermodus n0) bzw. ebenfalls das Nichtsortierzeichen. Mit **u<>** könnte man also bewirken, daß spitz geklammerte Angaben wegfallen. Ein evtl. hinter *y* stehendes Leerzeichen wird auch beseitigt, sonst hätte man ein überzähliges.

U Die Anwendung dieser Befehle empfiehlt sich für Sortierzwecke bei der Kopfkategorie #u1.
U*xy* Bei 'U' statt 'u' wird dann ferner das erste nachfolgende Wort automatisch groß geschrieben.

u¬¬ *Sonderfunktion:* **Nichtsortierzeichen verdoppeln**
Wenn ein Nichtsortierzeichen vor einem Wort steht, wird ein zweites dahinter gesetzt. D.h. der "Nichtsortiermodus" (→ Anh.A.1) wird von 0 auf 1 geändert. *Achtung:* In der .CFG muß zu diesem Zeitpunkt aber `n0` gesetzt sein, hernach muß man es auf `n1` setzen.

y0 **Umcodierung abschalten** : der aktuelle Arbeitstext wird nicht umcodiert, d.h. die p- oder q-Befehle sollen für den aktuellen AT nicht gelten. Umgekehrt:
y1 **Umcodierung erzwingen** : diese Befehle bewirken ein sofortiges Umcodieren. Bei **y1** werden die
y2 p-Befehle, bei **y2** die q-Befehle dazu benutzt. Danach wird die Anweisung normal weiter abgearbeitet. Direkte Prä- und Postfixe, die dann noch folgen, werden nicht mehr umcodiert, was meistens der Sinn dieser Befehle ist.
y3 Nur die Texel-Ersetzungen (eingebettete Zwischenteile) und die V23-Ersetzungen ausführen (s. Vb.163/164)
y4 UTF-8-Codes sollen in die nachfolgende Umcodierung nicht einbezogen werden. *Anders gesagt:* Wenn im AT ein Code >191 kommt, wird dieser und die zugehörigen nachfolgenen 1 oder 2 Bytes nicht umcodiert, wenn danach noch `y1` oder `y2` kommt. ("news 55")

y"n" Vom AT nur die Zeichen bestimmten Typs nehmen, die anderen weglassen
n ist ein Zeichen, dessen Code sich als Summe aus folgenden Werten zusammensetzen kann:
1=Ziffern, 2=Kleinbuchstaben, 4=Großbuchstaben, 8=Sonderzeichen (incl. Spatium)
Das Zeichen n ist dann die Summe aus 48 (="0") + Summe der gewünschten Werte.
Z.B. würde `y"7"` heißen: Alle Buchstaben und Ziffern nehmen, andere Zeichen weglassen.
`y"<"` sagt dagegen: Nur Großbuchstaben und Sonderzeichen nehmen (denn < = 60 = 48+8+4).

Typ 2 : Anwendervariablen

Wichtig: Die Anwendervariablen sind auch in FLEX verfügbar.

=xy... **Anwendervariable vergleichen/gleichsetzen** : Prüfung auf Gleichheit, dann Reaktion.
Der Wert **xy** besteht immer aus 2 Zeichen und steht für eine Sonderkategorie **#uxy**
Dieser Manipulationsbefehl ermöglicht es, einen Arbeitstext unter einem Namen festzuhalten, dann mit dem im nachfolgenden Satz gefundenen, entsprechend aufbereiteten Text zu vergleichen und auf das Vergleichsergebnis zu reagieren. Der als **Anwendervariable** (oder **Nutzervariable)** gespeicherte Kategorie-Inhalt bleibt also erhalten, bis er bei einem weiteren Satz neu besetzt wird. Sehr wichtig ist das z.B. für Rechenoperationen (10.2.6.6).
Vor diesem Befehl kann eine Folge anderer Manipulationsbefehle stehen. Erst das Ergebnis dieser Befehlsfolge wird als Arbeitstext mit der Anwendervariablen verglichen.
Der Befehl kann in drei Formen mit unterschiedlicher Wirkung auftreten:

=xy,z oder **=xy+M** oder **=xy**

`xy` steht für die Anwendervariable. x muß ein **Buchstabe**, y kann ein beliebiges Zeichen sein (Namen mit mehr als 2 Zeichen nicht möglich!),

`z` ist eine Zwischenteilnummer,

`M` ist eine Sprungmarke (beliebiges Zeichen), d.h. bezieht sich auf eine Zeile `#-M`.

Die Wirkung sei an einem Beispiel erklärt: die Ausgabezeile

#85 +S e" ; " =st,17 p'(' #90 4 #zz 1

in einer Parameterdatei würde bedeuten:

- Nimm Kategorie #85 und schneide den mit " ; " beginnenden Teil ab.
- Wenn der so entstandene Arbeitstext (der Serientitel) mit der Anwendervariablen **st** übereinstimmt (d.h. mit dem im vorangegangenen Satz ermittelten Text), gib Zwischenteil 17 aus, **aber sonst nichts**. Der Rest der Zeile (insbes. Prä- und Postfixangaben und der bedingte Sprung +S) bleibt dann unberücksichtigt.
- Wenn der Text **nicht** mit **st** übereinstimmt, ersetze den Inhalt von **st** durch den neuen Text **und** fahre normal fort (Präfix '(' ausgeben, dann den Text, dann die angegebenen bedingten Postfixe für die nachfolgende Kategorie vormerken, dann den Sprung ausführen.
- Wenn wir statt `",17"` jedoch `"+M"` schreiben, wird bei **Gleichheit** der Export an der Sprungmarke `#-M` fortgesetzt - ebenfalls ohne Ausgabe des Textes von #85. Die ganze Zeile bewirkt also dann bei Gleichheit nur eine Verzweigung, bei Ungleichheit das Umkopieren **und** die Ausgabe des Textes **und** den Sprung +S (falls er angegeben ist) zur Marke `#-S`.
- In jedem Fall hat man anschließend den Text von **#85** (ohne den Teil hinter " **;** ") in **#ust**.
- *Sonderfall:* **,0** : der Rest der Zeile wird nicht verarbeitet

Wenn weder `",z"` noch `"+M"` angegeben sind, wird der Arbeitstext in die Anwendervariable `#uxy` kopiert - zum späteren Gebrauch - er wird **aber nicht** ausgegeben. Indirekte Prä- und Postfixe werden jedoch ausgegeben, wenn welche definiert sind.
Wenn **nur** die Zuweisung erfolgen soll, aber garantiert keinerlei Ausgabe: **=xy #zz 0** schreiben. *Beispiel:* **#75 =vl #zz 0** steckt den Verlagsnamen in **#uvl** und sonst nichts, insbes. **ke** wirkt nicht wg. #zz 0

Wenn der Text einer Anwendervariablen als solcher auszugeben ist, muß er stets als Kategorie angesprochen werden. Seine Kategorienummer wäre z.B. `#ust`, man erhält ihn also durch Voranstellen von `#u` (für "user variable"). Das bedeutet: es gibt neben den schon bekannten Sonderkategorien `#u1` und `#u2` noch eine große Zahl (genau 26*62 = 1612) von möglichen Anwendervariablen. Abgelegt werden sie im Hintergrundspeicher. Mit dem Befehl `#au` können Sie sich die aktuellen Anwendervariablen beim Testen im PRESTO-Editor ansehen (aber nicht verändern). Ändern kann man sie nur über drei weitere Manipulationsbefehle:

axy Die Anwendervariable #uxy wird verändert: der Arbeitstext wird vor
Axy bzw. hinter #uxy gesetzt. Sonst passiert nichts, d.h. es wird auch nichts ausgegeben. Mit diesem Mechanismus, mehrfach angewandt, sind sehr lange Anwendervariablen produzierbar. Die Grenze: alle Anwendervariablen zusammen dürfen nicht länger als 10.000 Zeichen werden.
Sonderfall: Wenn y die Tilde (~) ist, wird der AT in die nächste freie Kategorie #uxy kopiert. Wenn z.B. #uZa und #uZb schon besetzt sind, dann kommt #uZc dran, wenn ... AZ~ oder ... =Z~ geschrieben wird.

In ***a99*** kann man sich die Anwendervariablen bequem anschauen und auch verändern: Mit Alt+r (evtl. 2x zu drücken) sieht man sie im Auswahlfeld links, kursiv geschrieben, mit Enter wählt man eine zum Bearbeiten aus.

dxy Löscht die Anwendervariable #uxy (delete). Das kann an jeder beliebigen Stelle geschehen, ohne die Verarbeitung zu beeinflussen. *Ein Beispiel:*

```
#75 dvo P" (" avo      löscht #uvo, hängt " (" an den Text von #75 und speichert das in #uvo
#74 P")" Avo           hängt ")" an den Text von #74 und fügt dies an #uvo hinten an.
```

Diese zwei Zeilen setzen die Kategorien

```
#74 Braunschweig   und   #75 Westermann
```

zusammen zu der Anwendervariablen

```
#uvo Westermann (Braunschweig)
```

die anschließend an beliebiger Stelle für die Ausgabe verwendbar ist.

Sonderfall: Mit dem Befehl

```
#74 p"abc" e3 dwq awq
```

belegt man **#uwq** mit dem Text "abc", falls #74 existiert. Der Befehl **dwq** löscht vorher die Variable #uwq, falls etwas drin steht.

dx~
d~~ *Sonderfall:* Mit **dx~** löscht man alle Variablen #uxy, mit **d~~** löscht man sämtliche #u-Variablen.

Wichtig: Die Anwendervariablen gelten global, d.h. für alle (bei PRESTO bis zu vier) geladenen Parametersets gemeinsam. Die Variablen dienen im Windows-System zugleich der Skriptsprache FLEX.

Als Musterparameter, in denen Gebrauch von Variablen gemacht wird, werden A-0.APR und ALPHA2.APR mitgeliefert:

`A-0.APR` Produktion einer sortierfähigen Datei (aus .ALG oder .ALD-Dateien) mit Personennamen und Satznummern oder Seitenzahlen, parallel zu "fahren" zur Produktion einer sortierten Liste mit einer der Dateien P-*.APR.

`ALPHA2.APR` Produktion des gedruckten Registers aus den mittels A-0.APR hergestellten und
`ALPHA3.APR` anschließend sortierten Daten. ALPHA2 faßt gleiche Namen zusammen, ALPHA3 stellt dann die eigentliche Liste her. Diese zwei Parameterdateien sind unabhängig von den Sortierbegriffen, d.h. man kann sie für beliebige Registerproduktionen verwenden.

Die Stapeldatei ALPHA.BAT koppelt alle Vorgänge hintereinander, so daß die Registerproduktion mit einem einzigen Befehl ausgelöst werden kann. Modellieren Sie Ihre eigene Registerproduktion nach diesem Beispiel, indem Sie die Parameter an Ihre Datenstruktur anpassen.
Anwendervariable können schon beim Start eines Programms vorbesetzt werden (Kap. 12, Option **-U**). Wenn man z.B. SRCH startet, und #ubc soll sofort beim Start den Wert "xyz" haben, würde man schreiben:

```
srch ... -Ubcxyz
```

Tip: Die momentanen Anwendervariablen kann man sich ansehen, indem man im Editor den Befehl `#au` gibt (in ***a99***: Alt+r). Zum Testen, um zu sehen was drinsteht, kann man an jeder Stelle einen Befehl `#uxy` in die Exportparameter einbauen, dann kommt der Inhalt von `#uxy` mit heraus.

Typ 3 : Zeichenausgabe vor/hinter Arbeitstext

pz **Indirektes Präfix** : Das Programm merkt sich die Zwischenteilnummer z bzw. die Steuerzeichen-
p{CS} kette CS und gibt sie aus, bevor der Arbeitstext selbst ausgegeben wird. Für z kann auch ein
p>K Unterprogrammaufruf der Form >K stehen (→ 10.2.6.5) `pz` und `p>K` können mehrfach auftreten.

Unterschiede zu **pX**: (zum genauen Ablauf der Ausgabe siehe 10.2.6.1)

- Ein Zwischenteil kann Steuerzeichen und Strukturbefehle enthalten, dagegen muß X eine in ".." oder '..' eingeschlossene Kette von druckbaren Zeichen sein.
- Das Indirekte Präfix unterliegt nicht der Zeichen-Umcodierung (p- und q-Befehle).
- Bei **pX** werden X und Arbeitstext zu einem neuen Arbeitstext vereinigt, so daß nachfolgende Manipulationsbefehle sich auf diese Gesamtkette beziehen. Insbesondere ist das für **R** und **=** wichtig (s.u.): wenn man zusammengesetzte Dinge rechtsbündig ausgeben oder einer Anwendervariablen zuweisen will. Bei **pz** und `Pz` geht das nicht, denn das Zwischenteil wird **nicht** mit dem Arbeitstext vereinigt, sondern getrennt ausgegeben - der AT selbst verändert sich dabei nicht.
- Indirekte Präfixe sind leichter austauschbar. Bei konsequenter Anwendung kann man die gesamte Zwischenteil-Liste in eine Nachlade-Tabelle legen (.APT-Datei), die in mehreren, evtl. verschiedensprachigen, Versionen gehalten werden kann (→ 10.2, Befehl t)

mz **Mehrfach-Präfix** : Zwischenteil z etc. wird vor jeder weiteren Mehrfachkategorie ausgegeben,
m{CS} nachdem die erste abgearbeitet ist. Wirkt nur dann, wenn statt des bedingten Sprungs der Wiederholungsbefehl `++` gesetzt wurde (→ 10.2.6.1) und die Kategorie mehrfach besetzt ist.
m>K Auch ein UP-Aufruf `>u` ist statt eines Präfixes möglich.
Beispiel:
`#40 ++ p"von: " m"; "`
setzt `"von: "` als Präfix vor den ersten Verf. (#40) und **`"; "`** als Mehrfach-Präfix vor jeden weiteren, also #402, #40a usw. Wenn #40 nicht besetzt ist, werden auch #402 ... nicht ausgegeben!

Pz **Indirektes Postfix**: Zwischenteil z wird im Anschluß an #xxf ausgegeben. (Geeignet für Steuerzeichen!)
P{CS} Die Unterschiede zwischen **Pz** und **PX** sind dieselben wie beim Präfix (siehe vorige Seite).
P>K Reihenfolge der Abarbeitung: 1. direktes, 2. indirektes, 3. bedingtes Postfix.
Die P-Befehle können allesamt nicht mehrfach auftreten.

Für alle indirekten Prä- und Postfixe gilt: bei den Varianten mit { CS } findet keine Umcodierung statt, d.h. innerhalb der Klammern { ... } müssen die wirklich auszugebenden Zeichen stehen.

Typ 4 : Bedingungsprüfungen

[s.a. *Zusatzmöglichkeiten* unter Typ 1, bX]

cX **Check:** der Arbeitstext bleibt unverändert, wenn X darin vorkommt; wenn X nicht vorkommt, wird die Befehlszeile ergebnislos abgebrochen. *Beispiele und Sonderfälle:* (Zahlenvergleiche → 10.2.6.6)

`#20 +m c"Marx" e0`	wenn "Marx" in #20 vorkommt, gehe zur Sprungmarke `#-m`, sonst
`#20 +m c"~marx" e0`	zur nächsten Zeile; das '~' bewirkt, daß es nicht auf Groß-/Kleinschreibung ankommt, also auch "Marx" oder "MARX" gilt.
`#20 +m c"^Marx" e0`	wenn "Marx" in #20 **nicht** vorkommt, gehe zur Sprungmarke `#-m`
`#20 +m c"^~marx" e0`	wenn "Marx" oder "marx" in #20 **nicht** vorkommt, ...
`#20 +m c">abc" e0`	wenn der AT alphanumerisch größer ist als "abc", ...
`#20 +m c">#nnn" e0`	wenn der AT alphanumerisch größer ist als Inhalt von #nnn, ...
`#20 +m c"~>#nnn" e0`	dasselbe, aber groß/klein ist egal. Beides geht auch mit < statt >.
`#20 +m c"#uxy" e0`	wenn #20 den Inhalt von #uxy enthält, ... (bei t3 in CFG: c"#uxyy")
`#20 +m c"[abc]" e0`	wenn eins der drei Zeichen a,b,c in #20 vorkommt, ...
`#20 +m c"^[abc]" e0`	wenn **keins** der Zeichen a,b,c in #20 vorkommt, ...

Zusatzmöglichkeiten wie bei `bX`.

ik,x **Indikatorprüfung**: die Position k (= 1...255) im Kategorietext (#cc) wird verglichen mit dem Zeichen x.
Ik,x Bei Gleichheit (**i**) bzw. Ungleichheit (**I**) wird weitergemacht, andernfalls wird der Arbeitstext nicht ausgegeben, d.h. die laufende Befehlszeile abgebrochen.
Achtung: das erste Zeichen des Kategorietextes hat die Nummer 4 (bei 3stelligen Kategorien 5). Somit kann man mit k=3 bzw. 4 das Indikatorzeichen (die 3. bzw. 4. Stelle der Kategorienummer) prüfen (vgl. "Indikatoraktion" → 10.2.6.4). *Beispiele:*
`i4,A` : nur weitermachen, wenn Arbeitstext mit 'A' beginnt. (2stellige Kategorien)
`I3,t` : nur weiter, wenn das dritte Kategoriezeichen kein 't' ist, also z.B. nicht bei `#30t`

vk,x **Zeichenvergleich**. Das Zeichen an der Position **k** (= 1...255) des Arbeitstextes wird verglichen mit
Vk,x dem Zeichen **x**. Wenn sein ASCII-Code kleiner ist als **x** (bei V: größer), wird in der Zeile weitergearbeitet, andernfalls wird der Arbeitstext nicht ausgegeben, d.h. die Befehlszeile abgebrochen.
Die Numerierung der Zeichen ist genauso wie bei den **i/I**-Befehlen zu verstehen.
Anmerk.: Die i- und v-Befehle prüfen den Inhalt der *Ausgangskategorie* (#cc), nicht den momentanen *Arbeitstext*. Wenn man letzteres will: zuerst eine #u-Variable draus machen und dann diese prüfen.

Beisp.: Ist das erste Kategoriezeichen ein Großbuchstabe: ... **`v4,[ V4,@`** ...
Ist das erste Kategoriezeichen ein Kleinbuchstabe: ... **`v4,{ V4,`` `** ...
Ist das erste Kategoriezeichen eine Ziffer: ... **`v4,: V4,/`** ...
... und wenn es ein 3stelliges Schema ist, muß es immer 5 statt 4 heißen, um das erste Zeichen zu prüfen.

l<k **Längenprüfung**. Ist der AT kürzer als k Zeichen
l>k bzw. länger als k Zeichen, wird die Zeile fortgesetzt, sonst wird sie abgebrochen und nichts ausgegeben.

Typ 5 : Sonderaktionen

C Sonder-Präfix **Zeilenvorschub** ("Carriage Return"). Z.B. bewirkt der Kategoriebefehl
`#87 C p"ISBN "` dasselbe wie `#87 p{ C "ISBN " }` ,
und zwar, daß Kategorie #87 auf neuer Zeile mit "ISBN " davor beginnt.
Genauer: C bewirkt, daß der Wert des Parameters **ze** ausgegeben wird (→ 10.2.3)

N Sonder-Präfix **Seitenvorschub** ("New Page")
Wenn dieser Befehl kommt, führt das Programm auf der Stelle einen Seitenumbruch aus, d.h. es vollführt den Fußabschnitt (→ 10.2.6.5.3), dann gibt es den Parameter **sv** aus, und danach macht es auch noch den Kopfabschnitt, bevor es hinter dem N fortfährt mit der Bearbeitung des aktuellen Arbeitstextes.
Bei Kartendruck wird die aktuelle Karte beendet und eine **Fortsetzungskarte** begonnen. Dabei bleiben die Angaben im Kopf stehen, die zum Zeitpunkt des Sonderbefehls **#wh** vorhanden waren, und der Parameter **fa** wird ausgeführt. Ansonsten wird automatisch eine Fortsetzungskarte gemacht, wenn der Platz nicht mehr reicht; mit diesem Befehl kann man es gezielt veranlassen.
Dieser Befehl ist z.B. auch nützlich, wenn beim Listendruck an bestimmten Stellen eine neue Seite beginnen soll: etwa wenn ein neuer Alphabetabschnitt oder eine neue Sachgruppe beginnt oder wenn eine ganz bestimmte Kategorie besetzt ist oder nicht.

Z Sonderbefehl **Textausgabe unterdrücken** : der Arbeitstext selbst wird nicht ausgegeben, die anderen Teile der Kategoriezeile werden aber ausgeführt: Prä- und Postfixe sowie der bedingte Sprung. Damit kann man Reaktionen auf bestimmte Bedingungen programmieren. Siehe auch `e0` = Textlänge auf 0 setzen - die Wirkung ist gleich. Hinter **Z** muß evtl. noch `#zz 0` stehen, um zu verhinden, daß der Parameter **ke** ausgegeben wird.

l*xy* **Länge** des aktuellen Arbeitstextes in die Variable `#u`*xy* schreiben.

L*xy* **Label ("name") der aktuellen Kategorie** aus CFG in `#uxy` kopieren (→ A.1.2).

M **Arbeitstext als Datenfeld in aktuellen Satz einordnen** ***Globale Manipulation***
Der Arbeitstext muß zu diesem Zeitpunkt mit einer gültigen Kategorienummer beginnen. Dann wird er in den aktuellen Satz als neue Kategorie eingeordnet bzw., wenn darin schon vorhanden, ersetzt.
Gebraucht wird dieser Befehl als Vorbereitung für "Globale Manipulationen". Man legt eine Parameterdatei an, die eine Sprungmarke #-# enthält, und darauf folgend Exportbefehle, die Veränderungen an einem Datensatz vornehmen, indem sie den Befehl M nutzen. Wenn diese Parameterdatei für Export geladen ist (mit `-e`), kann man vom Anzeigemenü (→ 1.5.4) aus mit <Strg>+F10 die Ausführung des Abschnitts #-# starten, also Datensätze und Ergebnismengen manipulieren lassen.
Sonderfall: Kategorie aus dem aktuellen Datensatz löschen. Wenn der Arbeitstext nur die Kategorienummer, keinen Text enthält, wird die betreffende Kategorie aus dem Satz gelöscht.
Beispiel: `#nr p"#71 " e4 M` Kategorie #71 wird beseitigt.
Teilfeld löschen: (siehe Gruppe 1, Befehl **~x**)
Wenn aus Kategorie #245 das Unterfeld `x` zu beseitigen ist, würde man schreiben:
`#245 ~x p"#245 " M` (*Achtung:* Wert von `ks` beachten, d.h. evtl. `b4` o.ä. ergänzen)

X*i* Aus dem Arbeitstext einen **Sondereintrag für satzübergreifende Suche** machen. [neu in V15]
'X' steht hier wirklich für den Buchstaben X.) Diese Operation dient der Lösung des sog. "Schiller-Räuber-Problems" beim Retrieval.
Beispiel: Angenommen, wir haben einen verknüpften Untersatz mit der Kategorie `#09 1234+13` und es handelt sich um Band 13 von Schillers "Gesammelte Werke", betitelt "Die Räuber". Wenn wir folgendes in die Index-Parameter einbauen: (mehr dazu: → 10.2.6.9)

```
ak=09+S
...
#-S
#u1 e"+" X9   #u1 enth. Primärschl. des Hauptsatzes, indexiert in Reg. 9
#+#
```

dann entsteht ein spezieller Registereintrag für diesen Untersatz, aber unter der internen Nummer des Hauptsatzes. Dies ermöglicht beim Retrieval (ab V15) die "Plus-Suche", d.h. den Befehl
`find per &schiller and tit raeuber`. Das '&' vor schiller bewirkt, daß alle Untersätze von Sätzen, die "`schiller`" enthalten, mit in die Verknüpfung einbezogen werden, also auch der Untersatz von Band 13.

? In allen Suchzeichenfolgen X kann das Fragezeichen '?' als **Maskierungszeichen** vorkommen.
Z.B. bewirkt **b"19??"** , daß das Druckfeld genau hinter einer Jahreszahl beginnt.
Soll ein anderes Zeichen als Maskierungszeichen verwendet werden erreicht man dies mit dem Befehl
?=% dadurch wird '%' zum Maskierungszeichen (gültig für die gesamte Laufzeit des Exports) erklärt, und man müßte `b"19%%"` schreiben. - Aber: **#b?=*** macht zwischendurch ad hoc das * zum Maskierungszeichen
In den Befehlen mit 'X' ist auch die **Suche nach einem von mehreren Zeichen** möglich; dies geht genauso wie bei den ak-Befehlen, z.B. **e"[.,:;-/]"** um den Arbeitstext am **ersten** Interpunktionszeichen enden zu lassen, oder **t"[.,]"**, um hinter dem **letzten** Punkt oder Komma abzuschneiden.

! **Testbefehl**. Anwendbar nur in Anzeigeparametern, und nur in PRESTO (nicht in ***a99***)!
Die Verarbeitung hält hier an, der aktuelle Arbeitstext wird am Bildschirm gezeigt, das Programm wartet auf einen Tastendruck. So kann man die Ergebnisse komplizierter Manipulationen beobachten.

Index-Sonderbefehle

Neues Konzept **MultiX** (mehrere Indexdateien): **`h multix`** eingeben

Index-Präfix

Wenn man mehrere Register (Teil-Indizes) anlegen will (Beispiel: CAT.API), braucht man eine Möglichkeit, für jeden Schlüssel seine Zugehörigkeit zu einem bestimmten Register zu regeln. Dazu wird das **Index-Präfix** benutzt. Es besteht aus dem Zeichen '`|`' (senkrechter Strich) und einer Ziffer von 1 bis 9. Für Register 10 bzw. 11: **`|:`** bzw. **`|;`** .
Das Indexpräfix wird wie jedes andere Präfix vor den Arbeitstext gesetzt, also mit dem Manipulationsbefehl p.
Entweder hat man ein Zwischenteil wie z.B. **`37="|5"`** und verwendet es mit dem Befehl **`p37`**, oder man gibt **`p"|5"`** als direktes Präfix.
Beim MultiX-Konzept, ab V27, gibt es ein Präfix `~zi` , das den Schlüssel dem Register *i* einer Indexdatei `cat.azx` zuordnet: neben der früher einzigen Datei `cat.adx` gibt es dann weitere Indexdateien]

Beispiel:

```
ak=4..+N        Personennamen alle bei Sprungmarke #-N verarbeiten
...
#-N
#u1 u p"|7"     Nichtsortierwörter herausnehmen, Zuordnung zu Index 7
#+#
```

Innerhalb eines Abschnitts können **mehrere Schlüssel** aufbereitet werden. Dann muß diese mit dem ASCII-Code 8 trennen. Z.B. kann man die zusätzlichen Schlüssel mit dem Präfix `p{ 8 }` versehen oder den Code 8 mit dem Direktbefehl `#t{ 8}` zwischenfügen. Das Schema sieht so aus:

```
#nnn ... p"|1"
#t{ 8 }
#nnn ... p"|2"
#+#
```

Akronyme

`Kn` Für die Index-Produktion gibt es einen weiteren Manipulationsbefehl zur Herstellung besonderer Schlüsselstrukturen, der sog. **Matchcodes** oder **Akronyme**:
Beispiel: Ein "Titelakronym" soll aus 6 Zeichen bestehen, wobei von den ersten 6 Wörtern der gewählten Kategorie jeweils der erste Buchstabe zu nehmen ist. Das erreicht man mit dem Bearbeitungsbefehl K :
`#20 K1 K1 K1 K1 K1 K1`
Damit werden aus diesen Titeln — diese Schlüssel:

```
#20 Common sense and humour in physics      csahip
#20 Nathan der Weise                        ndweis
```

Der zweite Titel hat weniger als 6 Wörter; der Befehl k statt K würde bewirken, daß der Schlüssel "`ndw `" entstünde, während bei K, wie man sieht, die fehlenden Buchstaben vom letzten Wort ergänzt werden.
Will man die ersten 10 Zeichen aus dem Titel als Schlüssel verwenden so gibt man

`#20 k10`

und erhält "commonsens" bzw "nathanderw", Leerzeichen werden also weggelassen. (Hier wäre das Ergebnis bei K10 dasselbe.)
'K' und 'k' schließen 'u' ein, d.h. Artikel am Anfang, sofern als solche mit dem Nichtsortierzeichen gekennzeichnet, werden weggelassen.
Akronyme sind sinnvoll für Dublettenprüfungen: man lege einen Sonderindex oder eine Abteilung für diese Akronyme an und verwende sie als Primärschlüssel (zuständige ak-Zeile an die erste Stelle rücken).
Anschließend kann man mit dem Programm UPD `-fc` einen Dublettencheck fahren (→ Kap.9).

Kombinationen von Manipulationsbefehlen

Alle Manipulationsbefehle sind miteinander mehrfach kombinierbar: z.B. würde die Befehlskette

`#83` **`b" ; " e", " e10 s10 p"Tagungsort: "`** `#zz 0`

aus der Kongreßkategorie #83 den Tagungsort mit maximal 10 Zeichen herauslösen und kürzere Namen mit Leerzeichen ergänzen. Hier ist der fettgedruckte Teil die Manipulations-Befehlskette, und `p"Tagungsort: "` ist das direkte Präfix. Oder gleichwertig:

`#83` **`b" ; " e", " e10 s10 73`** `#zz 0`

wenn es ein Zwischenteil `73="Tagungsort: "` gibt. Unterschied: das Zwischenteil wird nicht umcodiert!

Wenn also z.B.
`#83 Bundeskongreß für Sportpädagogik ; Garmisch-Partenkirchen, 1983`
bzw.
`#83 Politikwissenschaftliche Arbeitstagung ; Bonn, 1984.10.22-24`
eingegeben wurde, ergibt sich `"Garmisch-P"` bzw. `"Bonn "`
(und im zweiten Fall nicht etwa `"Bonn, 1984"`, da dies durch `e", "` verhindert wird!)
Die Exportprozedur bearbeitet beliebig viele Manipulationsbefehle hintereinander; dabei ist die Reihenfolge wichtig, denn jeder wirkt auf das Ergebnis des vorangegangenen Befehls.
Empfehlung: Befehle mit zahlenmäßiger Längenbegrenzung (`en, bn, rn, sn`) sollten in einer Befehlskette am besten an letzter Stelle stehen. Vor ihrer Ausführung werden automatisch die Zeichenersetzungen erledigt, denn diese können die Länge verändern (z.B. Ersetzung von 'ä' durch 'ae' oder Beseitigung von Sonderzeichen). Wenn also weitere Manipulationen auf einen solchen Befehl folgen, muß man überlegen, ob solche Veränderungen des Textes evtl. stattgefunden haben. Z.B. würde `e"ä"` nicht mehr funktionieren, wenn vorher `e50` stand und 'ä' durch "ae" ersetzt wurde.
Noch ein Beispiel: wenn man in #30 mehrere Notationen stehen hat, jeweils durch `" / "` getrennt, dann würde die Befehlskette

`#30` **`b" / " b" / " b5 e12 ....`**

die dritte Notation herausnehmen und vom 5. Zeichen ab 12 Zeichen drucken. Hier sieht man auch, wie die mehrfache Anwendung desselben Befehls nützlich sein kann. Wenn in #30 nur einmal " / " vorkommt, bricht diese Befehlszeile ab und produziert kein Ergebnis.

Und ein Beispiel für die Behandlung von Kopfkategorien:
Wenn man bei den Namenskategorien fallweise eine Ansetzungsform mit eingibt, die dann im Kopf erscheinen soll,
z.B. `#40 Goethe, Johann Wolfgang von = Goethe, Johann W.`
so könnte man schreiben:
a) für den Kopf: `#40` **`B" = "`** `#zz 4`
b) für den Text: `#40` **`e" = "`** `#69 1 #72 8`

Gleichfalls in den Bereich der Manipulationsbefehle gehören die **Rechenbefehle** und die Befehle zum **Nachladen** anderer Datensätze. Wegen ihrer eigenständigen und umfangreichen Bedeutung werden sie jedoch ausführlich in eigenen Abschnitten behandelt: → 10.2.6.6/7.

10.2.6.4 Steuerbefehle

Zwischen den Anweisungszeilen, die jeweils für die Ausgabe einer bestimmten Kategorie sorgen, können in der Kategorieliste auch **Steuerbefehle** stehen. Diese erweitern die Möglichkeiten der Verarbeitung.
Jeder Steuerbefehl muß auf einer eigenen Zeile stehen und ebenfalls mit '**#**' beginnen.

`#tz` **Unbedingte Texteinfügung**

`#t{CS}` z muß die Nummer eines Zwischenteils sein (→ 10.2.0), CS eine Steuerzeichenfolge. Statt einer Kategorie wird dann an dieser Stelle die betreffende Zeichenfolge ausgegeben. Dies ist nützlich, weil man sonst Textteile nur über die Prä- und Postfixe, abhängig von den vorkommenden Kategorien, einbringen kann - und diese werden nicht ausgegeben, wenn die Kategorie nicht belegt ist.
Beispiele:

`#t4`	Zwischenteil 4 ausgeben
`#t{ C "xyz: " }`	Neue Zeile, dann "xyz: " schreiben.

`#b`*bef* **Basisparameter ändern.** Die Basisparameter (→ 10.2.1-5) sind Konstanten, die während der gesamten Arbeitszeit des Programms gelten, nachdem die Parameter geladen sind. Dieser Befehl ermöglicht es aber, während der Abarbeitung, mittendrin zwischen zwei Exportbefehlen, Basisparameter beliebig oft zu ändern. Ob dies aber sinnvoll ist, muß man sich gut überlegen! Brauchbar ist der Befehl z.B., um den Basisparameter **`ke`** ab- oder wieder einzuschalten: **`#bke=""`** würde ihn abschalten. Sinnvoll wäre auch, zwischendurch ein anderes Maskierungszeichen ein- und wieder abzuschalten: **`#b?=%`** würde das % einschalten, **`#b?=?`** würde wieder das Fragezeichen einschalten (siehe → 10.2.6.3 Typ 3), dasselbe gilt für Umcodierungsbefehle.

#Tk **Umbruchcode** : Hier ist für k nur eine Nummer erlaubt; Zwischenteil k wird nicht ausgegeben, sondern das Programm merkt sich diese Nummer für den Fall, daß während der Ausgabe eines Datenfelds plötzlich ein **Seitenumbruch** fällig wird und der Feldtext dadurch unterbrochen werden muß. Der Seitenumbruch wird dann ausgeführt, und vor dem Weiterdrucken wird Zwischenteil k ausgegeben. Sinnvollerweise enthält k eine Druckersteuerung, mit der die Schriftart der unterbrochenen Kategorie vor dem Weiterdrucken wieder neu eingestellt wird. Es kann sich bei k aber auch um einen Unterprogrammaufruf handeln. *Beispiel:*

```
#T70          Zwischenteil 70 für Seitenumbruch festlegen
#98 ....        wenn während Ausgabe von #98 ein Seitenbruch passiert, 70 ausgeben!
#T0             Dadurch gilt #T70 nur, während #98 ausgegeben wird.
```

#i **Indikator-Aktion** : hiermit kann man auf bestimmte Zeichen an bestimmten Positionen reagieren. Der Befehl bezieht sich immer auf die Kategorie **#cc** (also die zuletzt benutzte, → 10.2.6.2). Zu beachten ist, daß diese innerhalb von Unterprogrammen konstant gleich der zuletzt vor dem Aufruf des UP benutzten Kategorie ist. Für die Überprüfung von Zeichen im aktuellen Arbeitstext gibt es den Manipulationsbefehl "Indikatorprüfung" (→ 10.2.6.3).
Die allgemeine Form der Indikatoraktion lautet:

#ik,x A

mit k = Positionsnummer
x = Zeichen (ein nicht eingebbares Zeichen kann man codieren mit \nnn, z.B. \31 für das Dreieck)
A = Aktion (Zwischenteil-Nummer oder UP-Aufruf)

Wirkung: Aktion A wird ausgeführt, wenn in der aktuellen Kategorie #cc (→ 10.2.6.2) an der Position k das Zeichen x steht. Sonst passiert nichts. Sehr zu empfehlen, wenn eine ganze Serie von Alternativen abzuarbeiten ist.
Die Zeichenzählung beginnt mit der ersten Ziffer der Kategorienummer, das 4. Zeichen ist dann also das erste Textzeichen der Kategorie. Z.B. würde `#i4,3 >U` bedeuten, daß das 4. Zeichen der gerade vorher angesprochenen Kategorie geprüft wird. Ist es das Zeichen '3', dann wird Unterprogramm U ausgeführt. Insbesondere ist der Befehl nützlich, wenn man mit Indikatoren arbeitet, d.h. Steuerzeichen, die immer an einer bestimmten Position der Kategorie stehen.
Wenn man Bedingungen prüfen will, die aus mehr als einem Zeichen bestehen, arbeitet man mit dem Manipulationsbefehl **cX** oder mit "Anwendervariablen" (→ 10.2.6.3).

#wh **Wiederholung des Kopftextes** auf Fortsetzungskarte. Der bis zu diesem Zeitpunkt produzierte Text wird wiederholt (mit Parameter **fa** dahinter, → 10.2.2), wenn eine Fortsetzungskarte gedruckt werden muß. Relevant ist dies nur bei Kartendruck (fl>0).

#q{ "*fragetext*" } [nur in PRESTO und APAC wirksam, nicht in a99 und den anderen Programmen]
Frage an den Benutzer : Der *fragetext* erscheint am Bildschirm über einem Eingabefenster, und das Programm wartet auf eine Benutzereingabe. Erst danach wird die Parameterdatei weiter abgearbeitet. Die Antwort steht dann in `#uxq` und kann beliebig ausgewertet werden.
In *a99* und *alcarta* geht man anders vor, um dem Nutzer eine Frage zu stellen: man nutzt die FLEX-Methode.

Ein besonderer Steuerbefehl ist der **Schaltbefehl**, der mit den Nachladebefehlen in engem Zusammenhang steht. Siehe dazu 10.2.6.7.

10.2.6.5 Sprünge, Unterprogramme

10.2.6.5.1 Sprungbefehle und Sprungmarken

Sprungbefehle der Form "+m" mit einem einzelnen alphanumerischen Zeichen oder Sonderzeichen 'm' können, wie schon beschrieben, in Kopfbefehlen (→ 10.2.1) und als bedingter Sprung in Exportbefehlen vorkommen (→ 10.2.6.1).
Aber es gibt auch eine Art GO TO-Befehl: eine Befehlszeile der Form

#+m in der Kategorieliste heißt **Unbedingter Sprungbefehl**, denn er wird bedingungslos ausgeführt, wenn die Abarbeitung der Kategorieliste an diese Stelle kommt.
Empfehlung: Kleinbuchstaben (im Gegensatz zu den ak-Sprungbefehlen), erlaubt sind aber alle druckbaren Zeichen. Nicht erlaubt sind die ASCII-Werte 0-15, 26, 27,34,("), 35 ('#'),39('), 45 ('-'), 255.

Die **Sprungmarke**, zu der ein Sprungbefehl hinführen soll, muß so aussehen:

#-m

Die Anführungszeichen ' und " sind für m nicht erlaubt, und bestimmte Zeichen haben Sonderwirkung, s.u.
Beispiel: Innerhalb der Kategorieliste könnte diese Sequenz stehen:

```
#39 +y #72 8    heißt: wenn #39 vorkommt, weiter bei Sprungmarke #-y
#40 e" = "          wenn nicht: #40 ausgeben, Präfix " = "
#+w                 Dann Sprung nach #-w
#-x                 Sprungmarke x, zu der von anderswo gesprungen wird
#...                weitere Exportbefehle
#-w                 Sprungmarke w
#...
#-y             Sprungmarke y
#71             hier geht es weiter nach Sprung auf y: #71 wird ausgegeben
```

Eine Sprungmarke beeinflußt selbst den Ablauf nicht, wenn das Programm bei der sequentiellen Bearbeitung auf eine solche Zeile stößt. Das Programm macht dann mit der nächsten Zeile weiter.
Man beachte, daß Sprungmarken nur einzelne Zeichen sein können - im Gegensatz zum Import, wo längere Bezeichnungen erlaubt sind (→ 11.2.3.6). Der Grund ist die Optimierung: beim Export ist Tempo wichtig, denn z.B. die Bildschirmanzeige soll so schnell wie möglich gehen. Leider kann man dadurch aber nur etwa 220 Sprungmarken in einer Parameterdatei haben. Man beachte, daß man bei den bedingten Sprüngen (→ 10.2.6.1) oft ohne Sprungmarke auskommen kann.
Vorsicht ist geboten, wenn man Rücksprünge (Marke oberhalb des Befehls) programmiert. Dadurch entsteht eine Schleife, die nur endet, wenn es einen Aussprung gibt, der auch garantiert erreicht wird.

Sonderfälle von Sprungbefehlen

Die zwei hier beschriebenen Fälle gelten sowohl für bedingte wie für unbedingte Sprünge:

#+# **Ende**

Am Ende der gesamten Kategorieliste braucht kein Endebefehl zu stehen. Wenn aber mittendrin aufgrund einer Bedingung die Ausgabe des aktuellen Satzes enden soll, ist dies der Endebefehl, d.h. die Bearbeitung endet an dieser Stelle, der aufbereitete Satz gilt als fertig und wird ausgegeben. Eine Sprungmarke `#-#` kann deshalb nicht mit einem Sprungbefehl erreicht werden. Statt dessen ist sie reserviert für die "Globale Manipulation" (S.200): sie leitet den dafür vorgesehenen Abschnitt ein.

#+- **Abbruch**

Dieser Sprungbefehl bewirkt einen Abbruch der Verarbeitung. Der Rest des aktuellen Exportsatzes wird nicht ausgegeben, wenn zu dem Zeitpunkt effektiv noch keine Zeichen ausgegeben worden sind. Sinnvoll ist das nur nach einer Bedingungsprüfung (vgl. "Anwendervariable" in 10.2.6.3 und "Indikatoraktion" in 10.2.6.4). Auch das Label `#--` kann deshalb nicht als Sprungmarke vorkommen.
Empfehlung: Wenn bestimmte Bedingungen dazu führen sollen, daß von einem Satz überhaupt nichts ausgegeben werden soll, dann die Bedingungsprüfungen und Abbruchbefehle ganz an den Anfang der Parameterdatei setzen, **und** `as=""` setzen (→10.2.1.1). Die Funktion von `as` können dann geeignet plazierte `#t`-Befehle übernehmen.

Reservierte Sprungmarken

#-1 ... #-9 Wenn **`ic=1`** gesetzt wurde, benutzt das Programm diese Sprungmarken zur Umcodierung der Benutzereingabe in den Registern 1 bis 9 (für 10 ist `#-:` zu verwenden, für 11 `#-;`).

#-0 Hier wird die Kurztitelzeile aufbereitet. Dazu gehört ein Befehl **`ak=zz+0`** .

#-@ Primärschlüssel-Aufbereitung: Für die Identifikationsnummer zu benutzen, wenn man mehrbändige Werke verknüpft speichern will. Dann ist es möglich, in der Titelanzeige mit 'B' und 'b' vor und zurückzuschalten. (Vgl. als Beispiele die entsprechenden Abschnitte in CAT.API)

#-/ Restriktionsschlüssel, mit V15 eingeführt (→ 10.2.9)

#-. sog. SR-Schlüssel für die "Satzübergreifende Suche" (→ 10.2.6.9)

#-` Flip-Aufbereitung: anwendungsspezifische Umschaltungen in der PRESTO-Titelanzeige (10.2.7)

#-# Globale Manipulationen (in Anzeige- oder Exportparametern; (→ 1.5.4, PRESTO-Funktion Strg+F10)

#-- Löschkontrolle (→ 10.2.6.8)

#-_ Nach F4, am Ende einer Ergebnismenge, führt PRESTO diesen Abschnitt aus.

#-(Abschnitt für Anzeige des Internformats in ***a99/alcarta*** (sonst kommt bei F5 nichts)

#- E (Leerzeichen vor dem E!) Endabschnitt für Listenproduktion (→ 10.2.6.5.3); auch bei Beendung von PRESTO wird dieser Abschnitt ausgeführt.

H Hilfsabschnitt für "Programmierte Validierungen" (→ 10.2.8) in den Indexparametern
Hierhin wird nicht mit Sprungbefehl gesprungen, sondern automatisch bei jeder Kategorieeingabe und einigen anderen Ereignissen, z.B. wenn der aktuelle Satz gespeichert werden soll.

10.2.6.5.2 Unterprogramme

Beginn und Ende eines Unterprogramms werden durch öffnende und schließende runde Klammern markiert. Auch diese Zeilen beginnen mit '#'.
An beliebiger Stelle in der Kategorieliste, jedoch **unterhalb des Aufrufs**, kann folgendes stehen:

#(P Beginn des Unterprogramms P
#... Text des Unterprogramms:
#... beliebige Exportbefehle u.a.
#)P Ende des Unterprogramms P

Dabei kann für P jeder Buchstabe und jede Ziffer stehen.

Die Endemarke **#)P** darf mehrfach vorkommen und kann auch oberhalb **#(P** stehen. Das ist nützlich, wenn im UP Sprünge vorkommen.

Das UP wird dann mit seinem "Label" P von beliebiger Stelle aus so aufgerufen:

#>P **Aufruf:** Führe das Unterprogramm P aus
d.h. '**>**' ist das Befehlszeichen für einen Unterprogramm-Aufruf.
Der Aufruf kann nicht nur anstelle eines Exportbefehls stehen, wie hier gezeigt, sondern auch als Manipulationsbefehl (anstatt Präfix) oder anstelle einer Postfix-Nummer.
Wenn man einen UP-Aufruf als Manipulationsbefehl benutzt, bewirkt die Zeile selbst keine Ausgabe; die Kategorie kann aber innerhalb des UP an beliebiger Stelle als Sonderkategorie #cc (→ 10.2.6.2) angesprochen werden.

Hier ein formales Beispiel für den Aufruf von Unterprogrammen aus Exportbefehlen:

```
#xxf +c p>Q #kt1 >R #kt2 >S #zz 1
#ya ...
#yb ...
#yc ...
#-c
#za ...
```

Das bedeutet:

1. Wähle Kategorie #xx mit Folgezeichen f. Wenn sie existiert, führe Unterprogramm Q aus. Darin kann #xxf unter dem Namen #cc angesprochen und ausgegeben werden.
2. Gehe anschließend (ohne #xxf weiter zu bearbeiten oder auszugeben!) zum Sprungziel #-c .
3. Wenn die nächste tatsächlich vorkommende Kategorie (das muß nicht #ya sein, die kann fehlen; es kann #yb oder #yc, #za oder noch eine weitere sein) kleiner oder gleich #kt1 ist, führe Prozedur R aus, wenn sie größer als #kt1, aber kleiner oder gleich #kt2 ist, Prozedur S, sonst gib Zwischenteil 1 aus (wenn die nächste Kategorie größer als #kt2 ist).
4. Wenn Kategorie #xxf aber im aktuellen Satz gar nicht vorkommt, fahre mit der nächsten Zeile (`#ya ...`) fort.

Nochmals: Innerhalb der Unterprogramme #(Q, #(R und #(S kann man auf die zuletzt vor dem Aufruf benutzte Kategorie zurückgreifen: sie steht unter der Sonderkategorie #cc (→ 10.2.6.2). Und zwar enthält sie den unvorbehandelten Text von #xxf. Bei verschachtelten Aufrufen bleibt aber nur der erste Text in #cc erhalten. Andere Texte, die man braucht, kann man vor dem Aufruf in Anwendervariablen unterbringen (→ 10.2.6.2/3).

Statt der bedingten Postfixe kann auch ein direktes oder indirektes Postfix angegeben werden:
#xxf +c p>Q P{ ... } oder **#xxf +c p>Q P>u**

Im Falle **P>u** wird dann das Unterprogramm **#(u** nach der Ausgabe des Kategorietexts ausgeführt.

Beispiel:
In #40, #40a, #40b ... sollen die Namen von Verfassern stehen. In einer Liste sollen sie in der Form "Vorname Name; Vorname Name; Vorname Name ..." erscheinen, also in der nicht-invertierten Form. Das erreicht man so: (wobei 3="; " und 8 =". - ")

```
#40. ++ p>K m>K #40z 3 #zz 8   vor jeder #40x wird Unterprogramm #(K ausgeführt
#+v                            wenn abgearbeitet: weiter bei #-v
#(K                            hier beginnt Unterprogramm K, es invertiert Namen
                               (#cc enthält dann den Text von #40...)
#cc b", " #zz 0                nimm zuerst den Teil hinter dem ", "
#cc e"," p" " #zz 0            dann nimm den Teil vor ", " und setze " " davor ein
#)K                            Ende des UP
#-v                            Sprungmarke: hier geht's weiter, wenn keine #40 mehr
```

Wenn der erste Verfasser anders behandelt werden soll, könnte man hinter p auch ein anderes UP angeben, z.B. `p>E`. In den Parameterdateien D-1.APR und P-KARTE.APR wurde diese Technik auf alle Personennamen angewendet; weitere Unterprogramme erzeugen die Funktionsbezeichnungen.

10.2.6.5.3 Sonderteile: Kopf-, Fuß- und Endabschnitt

Unter dem Ende der Kategorieliste kann man noch zwei separate Listen und einen Sonderabschnitt anfügen, die die Angaben für den Kopf- bzw. Fußabschnitt enthalten. Diese Abschnitte stellen praktisch auch Unterprogramme dar, die aber bei Bedarf automatisch aufgerufen werden. Bei Kartendruck kommen diese Angaben dann oben bzw. unten auf die Karte, bei Listendruck oben bzw. unten auf die Seite (hier sind dann die Sonderkategorien **`#p0..#p2`** sinnvoll). Wenn ohne Seitenumbruch ausgegeben wird (`zm=0`, z.B. ASCII-"download"-Datei), dann werden diese Abschnitte nur am Dateianfang bzw. -ende ausgeführt. Vor allem ist es wichtig, daß man dadurch einen Dateikopf erzeugen kann (s.a. 1 10.2.4, Parameter di), wie es für diverse Fremdsoftware bzw. für die Druckerinitialisierung notwendig ist. Brauchbar sind solche Abschnitte auch für Rechenaktionen: für die Initialisierung der Variablen und die Ausgabe der Ergebnisse.

`H` Hilfsabschnitt, nur in Indexparametern. Für **Programmierte Validierungen** (→ 10.2.8), mit denen jede Dateneingabe sofort überprüft werden kann.

`K` leitet die Liste für den **Kopf**abschnitt (Vorspann) ein und beendet zugleich den Hauptteil, bzw.

`F` die Liste für den **Fuß**abschnitt (Nachspann).
In den **Index**parametern: Abschnitt für beliebige Umwandlung der Variablen **`#u1`**. Der erzeugte Output wird in *a99/alcarta* statt der tatsächlichen Registerzeile gezeigt. Dadurch kann man dem Nutzer vorspiegeln, der Index sähe anders aus als er wirklich aussieht. Was man im einzelnen mit **`#u1`** macht, ist völlig beliebig. Ausgeführt wird dies nur im Moment der Indexanzeige, nicht während der Indexierung! Die Variable **`#u1`** enthält die umzuwandelnde Registerzeile, und zwar die Nummer (`1 ... 9 : ;`) auf der ersten Position vor dem Text. Im Indexfenster kann man mit Alt+u zwischen der normalen und der verwandelten Anzeige hin- und zurückschalten.

Für Listen- und Dateiexport gibt es außerdem noch:

`#- E` Endabschnitt für Listenproduktion (ein Leerzeichen vor dem E!) Wird automatisch am Ende eines Exportlaufs (**Dateiende**) vom Programm SRCH und auch von PRESTO ausgeführt, damit man z.B. Rechenergebnisse ausgeben kann. Wichtig bei `zm>0`, denn der Abschnitt F wird dann an jedem **Seitenende** ausgeführt! In a99 ist es leichter, das Ende eines Exports in FLEX zu programmieren.

`#-_` Diesen Abschnitt führt PRESTO am Ende jeder Ergebnismenge aus, wenn mit **F4** exportiert wurde.

Wenn **K** und **F** beide auftreten, muß **K** zuerst kommen. Es muß außerdem **`ff=2`** gesetzt sein, damit diese Abschnitte wirksam werden (→ 10.2.2). Das Ende dieser Unterprogramme wird automatisch durch den Beginn des anderen bzw. durch das Ende der gesamten Parameterdatei markiert.
Wichtig: Die Sprungmarken der Abschnitte K, F und H sind unabhängig von denen des Hauptteils, d.h. man muß hier keine Rücksicht auf dort schon vergebene Marken nehmen.
Ansonsten können diese Listen alles enthalten, was in der Hauptliste vorkommen kann. Die K-Liste muß bei Kartendruck insbesondere den Befehl #wh enthalten (s.o.), wenn Fortsetzungskarten richtig funktionieren sollen. Jedoch: man benutzt Kopf- und Fußabschnitte in der Regel nur beim Listendruck für Seitenkopf und -Fuß. Beim Kartendruck ist es praktischer, die oft sehr unterschiedlichen Köpfe für verschiedene Eintragungstypen mit Sprungbefehlen von den ak-Zeilen aus zu steuern (→ 10.2.1). In den Parameterdateien P-KARTE.APR und P-NORMAL.APR sind beide Methoden der Kopfgestaltung zu sehen.
Sehr wichtig sind Kopf- und Fußabschnitte auch in Verbindung mit **Rechenbefehlen**, siehe nächster Abschnitt.

In a99/alcarta gibt es die FLEX-Befehle **`export Head`** und **`export Foot`** zur direkten Ausführung der Abschnitte. Gedacht ist das zur Anwendung am Beginn bzw. Ende eines Exportes.

10.2.6.6 Rechenbefehle

Hinweis: Einfacher ist das Rechnen in der FLEX-Sprache von a99/avanti. Mehr dazu: **`h eval`** eingeben.
Gerechnet werden soll mit den Inhalten von Kategorien - mit was sonst? Dabei kann es nötig sein, den in der Kategorie stehenden Text (der dann als Zahl anzusehen ist) noch auf andere Art vor- oder nachzubehandeln. Deshalb lag es nahe, auch die Rechenfunktionen als **Manipulationsbefehle** zu entwickeln (→ Kap.10.2.6.3), damit man alle nötigen Bearbeitungen miteinander kombinieren kann. Aus Gründen der Effizienz wurde für das Rechnen nur ein einziger Befehl geschaffen: der Buchstabe **`x`** wird benutzt, um alle Rechenoperationen einzuleiten. Mehrere x-Befehle können hintereinander ausgeführt werden. Gleich ein Beispiel: eine Kategorie #95 soll "*Lieferant / Preis*" enthalten, und wir wollen den Preis ausgeben, aber vorher sollen 20% abgezogen und das Ergebnis mit 1.14 multipliziert und dann auf 2 Stellen gerundet werden. Das geht so:

```
#95 x"*0.8" x"*1.14" x"r2" p"Endpreis: DM "
```

Wenn in #95 z.B. "Meyer / `DM 531,90" steht, wird "Endpreis: DM 485.09" ausgegeben.`
Die eigentlichen Rechenbefehle müssen also einzeln jeweils hinter einem `x` in Anführungszeichen stehen.
Die allgemeine Form eines Rechenbefehls ist

`x"`*OpWert***`"`**

mit einem **Operatorzeichen** *Op* und dem zu verrechnenden *Wert*. Die Operation wird dann jeweils auf den aktuellen Arbeitstext angewendet, das Ergebnis ist ein neuer Arbeitstext - wie bei den anderen Manipulationsbefehlen. Es hat immer 9 Nachkommastellen (auch bei ganzen Zahlen), aber stets mit Punkt, nicht Komma.
Es macht nichts, wenn der Arbeitstext vor oder hinter der Zahl noch andere Angaben enthält: das Programm schält sich die Zahl heraus (Punkt und Komma sind dabei gleichwertig) und beachtet alles andere nicht. Die Zahl kann negativ sein, dann muß aber hinter dem Minuszeichen (Bindestrich) sofort die erste Ziffer folgen.
Diese Operatorzeichen gibt es:

`+ - * /`	Grundrechenarten
`%`	Divisionsrest (z.B. 25%7 ergibt 4)
`r`	Rundung (Dahinter als *Wert* eine Ziffer für die Anzahl der Stellen, 0 für Rundung auf ganze Zahl)
`=`	Wertzuweisung: der aktuelle Arbeitstext (nicht die Kategorie!) wird durch *Wert* ersetzt (Nicht verwechseln mit dem Manipulationsbefehl **`=xy`**, → 10.2.6.3, der den Inhalt des Arbeitstextes in eine Anwendervariable kopiert)
`> >=` **`< <=`** **`==`**	**Vergleiche**: Diese Befehle verändern den Arbeitstext nicht. Er wird nur mit dem *Wert* verglichen. Wenn die Bedingung zutrifft, wird mit demselben Arbeitstext die Zeile fortgesetzt, wenn nicht, wird sie ergebnislos abgebrochen und zur nächsten übergegangen. (Vergleichen Sie dies mit den Manipulationsbefehlen `bX und cX.`) Insbesondere kann man so auch bedingte Sprünge auslösen. Die Zeile `#77 +M x">=500" e0` bewirkt, daß zur Sprungmarke `#-M` gegangen wird, wenn die Seitenzahl größer oder gleich 500 ist. Dies funktioniert auch, wenn im Datensatz so etwas wie `#77 ca. 600 S.` steht. Römische Zahlen werden nicht erkannt und deshalb übergangen. Die Operation **`==`** prüft die Gleichheit als Gleitkommazahl, nicht als Zeichenfolge.

Als *Wert* kann man drei verschiedene Dinge einsetzen:

Zahl	eine sog. "Gleitkommazahl" mit Dezimalpunkt **oder -komma**. Auch negative Zahlen sind möglich.
Anw.var.	eine Anwendervariable, dann wird deren Inhalt genommen und mit dem Arbeitstext verrechnet, z.B. **`x"+vb"`** (für **`#uvb`** ; wenn diese nicht belegt ist, wird der Wert Null verwendet.)
Kat	eine normale Kategorie mit '#', also z.B. **`x"+#76 "`**

Einige Hinweise auf nützliche Manipulationen:
`e"."` beseitigt den Dezimalteil, **`E".??"`** schneidet hinter der zweiten Nachkommastelle ab (beides ohne Rundung),
`x"r0"` rundet auf eine ganze Zahl, **`x"r2"`** rundet auf zwei Nachkommastellen, und
`x"r2" r7,$` tut dasselbe, macht aus dem Ergebnis aber dann eine siebenstellige, rechtsbündige Zeichenfolge und füllt diese links mit '$' auf.
Ein Beispiel mit Ablaufbeschreibung ("Seitensumme") finden Sie in 10.4.5, dazu gehört die Parameterdatei `R-0.APR,` die Sie als Prototyp für rechnerische Auswertungen von Ergebnismengen ausnutzen können.

10.2.6.7 Satzverknüpfungen ("Nachladen" anderer Sätze beim Export)

Es gilt längst als ausgemacht, daß Datensätze in einer bibliothekarischen Datenbank keine isolierten Einheiten sein sollen, die beziehungslos nebeneinander stehen. Aus der Sicht der Theorie, die mit den Konstrukten "Normdatei" und "Mehrdateisystem" operiert, gab es bis zur Version 11.1 noch mehrere Mängel und daher offene Wünsche:

1. Mehrbändige Werke konnte ***allegro*** zwar schon vor V11 gut speichern und drucken, aber man mußte das gesamte Werk als u.U. riesengroßen und dann unhandlichen Datensatz erfassen. Besser wäre, man könnte wahlweise auch so verfahren, daß man die Bände als eigene Datensätze erfaßt, jedoch mit einer irgendwie gearteten Verbindung zum Datensatz der Hauptaufnahme. Für Druck und Anzeige müßte dann alles wieder zusammenführbar sein. Die Recherchierbarkeit der einzelnen Bände würde sich gegenüber der älteren Lösung in dem Sinne verbessern, daß nur der relevante Teil angezeigt wird.

2. Bestimmte Funktionen ließen sich bis V11.1 mit dem ***allegro***-Konzept gar nicht realisieren, wie z.B. diese: Anlegen einer Stammdatei für Lieferanten - Eingabe eines Lieferantencodes in einen Bestellsatz - Einfügen der Lieferantenadresse in das Bestellformular, d.h. automatisches Zuführen der Angaben aus dem Stammsatz; oder: Anlegen einer Zeitschriften-Stammdatei mit unterschiedlichen Voll- und Kurztitelfassungen je Zeitschrift - Eingabe einer Kurzbezeichnung (wie z.B. CODEN oder ISSN) statt eines vollständigen Zeitschriftentitels bei der Erfassung von Aufsätzen - Einfügen der jeweils gewünschten Titelfassung in die Dokumentaufnahme zum Zeitpunkt der Druckausgabe oder auch Bildschirmanzeige.

3. Ansetzungsformen von Namen, Schlagwörtern u.a. mußten bis V11.1 konsistent in allen Datensätzen gespeichert sein, um eine korrekte Zusammenführung im Index zu gewährleisten. Besonders bei Körperschaftsnamen bedeutet das nicht nur beträchtliche zusätzliche Eingabearbeit (die sich zwar durch die Kopierfunktionen des Editors mindern läßt), sondern auch mehr Speicherbedarf durch die redundante Speicherung. Unangenehme Klimmzüge oder Kompromisse sind dann vonnöten, wenn es um Verweisungsformen geht. Verfährt man so, wie es im → Kap.1.6 angedeutet wird, speichert also im Datensatz nur eine Identifikationsnummer statt z.B. des Körperschaftsnamens, so konnte man noch bis Version 13 für einen Ausdruck den Namen selbst nicht automatisch heranholen und einsetzen lassen - man mußte dann schon die richtige Namensform mit im Satz speichern.

4. Änderungen an Ansetzungsformen mußten in allen betroffenen Datensätzen durchgeführt werden statt an nur einer Stelle, nämlich im Normdatensatz. Zwar kann man mit der "globalen Änderung" recht schnell solche Änderungsaktionen bewältigen, aber integrierte Normdaten sind halt doch noch eleganter!

Ab Version 11.2 sind die Mängel 1. und 2., ab V14 auch 3. und 4. behoben, und zwar durch neue Befehle in der Exportsprache:

- der **Nachladebefehl** (ein Manipulationsbefehl) zum Heranholen eines anderen Satzes, und
- der **Schaltbefehl** (ein Steuerbefehl), der zum Ausgangssatz zurückschaltet.
- besondere Schlüssel, **Ersetzungs-** und **Pseudo**schlüssel, ermöglichen es, daß Nummern automatisch durch Ansetzungsformen ersetzt werden. Dieses geht erst ab V14 (→ 10.2.6.8).

Warum überhaupt ist die Exportsprache der logische Ort für die Satzverknüpfungen? Nun, die oben aufgezählten Probleme haben eins gemeinsam: die Zusammenführung des jeweils Zusammengehörigen muß immer nur zum Zeitpunkt einer Anzeige, Ausgabe oder Indexierung erfolgen, und genau dafür ist ja die Exportsprache zuständig. Ansonsten können alle Daten in der Datenbank stehen, wo sie wollen. Das Zusammenführen geschieht über den Index mit Hilfe von dafür präparierten Schlüsseln und geeigneten Exportbefehlen. Aber: Die Verknüpfungen sind deshalb indirekter Natur, ohne den darauf eingerichteten Index sind sie nicht nutzbar, da es sich nicht um direkte Querverweise zwischen Datensätzen handelt. Der Umgang mit Verknüpfungen ist für den Parametrierer alles andere als trivial, und es bleibt das schwierige Problem der "satzübergreifenden Suche" zu beachten:

5. Bei verknüpfter Speicherung von Haupt- und Unteraufnahme hat man das "Schiller-Räuber-Problem": man findet Schillers "Räuber" nicht durch UND-Verknüpfung, wenn "Schiller" nur in der Hauptaufnahme, die "Räuber" aber nur in einem separat gespeicherten Untersatz stehen. Bei hierarchischer Speicherung ist das kein Problem, denn beides steht im selben Datensatz. Mit V15 wurde auch dieses Problem gelöst (→ 10.2.9).

Die grundlegenden Mechanismen, den man braucht, um die Probleme 1-4 zu lösen, sind plausibel:

1. im Prinzip muß man in jeder Kategorie, wo es logisch sinnvoll ist, statt eines Textes auch einen eindeutigen Schlüsselbegriff unterbringen können, der sich auf einen anderen Datensatz (Normsatz bzw. übergeordneten Titelsatz) bezieht. Dieser Schlüsselbegriff muß einem Indexeintrag entsprechen, der zu dem anderen Satz gehört. Meist ist das eine Identnummer, es kann auch so etwas wie ein "Kürzel" sein (s.u.).

2. Beim Export (wozu auch die Bildschirmanzeige und die Indexierung gehören) muß das Programm mit Hilfe dieses Schlüsselbegriffs entweder
   a) den betreffenden Datensatz über den Index suchen und zusätzlich in den Arbeitsspeicher holen. Anschaulich: der nachgeladene Satz wird oben auf den/die schon vorhandene(n) **gestapelt**. Oder:
   b) den Schlüssel automatisch durch einen Klartext ersetzen (dazu → 10.2.6.8).

3. Anschließend steht im Fall a) der nachgeladene Satz bereit, und jeder seiner Bestandteile kann nun exportiert werden, mit allen Möglichkeiten, die die Exportsprache bietet. Es können auch noch weitere Nachladebefehle kommen, die den Stapel noch vergrößern. Im Fall b) kann man so tun, als gäbe es die Identnummern gar nicht, sondern nur Klartexte.

4. Danach muß man im Fall a) mit einem eigenen Befehl vom nachgeladenen Satz wieder zum Ausgangssatz zurückkehren können, damit man mit dessen Export fortfahren kann. Anschaulich: der Stapel muß, evtl. Stück für Stück, wenn mehr als einmal nachgeladen wurde, wieder abgebaut werden.

5. Diese Vorgänge müssen während des Exports eines Satzes selbstverständlich mehrfach und auch ineinander verschachtelt durchführbar sein, um alle Eventualitäten damit in den Griff zu bekommen.

Anwendungsbeispiel "Zitat"

Nehmen wir als Beispiel die Erfassung von Aufsatzdaten. Die wesentlichen Kategorien sind

```
#20  Aufsatztitel
#40  Verfasser
#70  Quelle (Zeitschriftentitel)    oder   #70 _Kürzel
#704 Band   (das Programm erkennt am '_', daß in #70 ein Kürzel steht und kein
            Titel)
#706 Heft
#708 Seitenzahlen
#70i ISSN
```

Nun würde man gern auf die Eingabe des vollständigen Zeitschriftentitels unter #70 verzichten, muß dann aber auf einen "Stammsatz" für die Zeitschrift zurückgreifen können. Dieser Stammsatz hätte die Kategorien (→ Anhang B)

```
#20  RAK-Ansetzungstitel der Zeitschrift
#8n  Zeitschriftentitel (normierte Ansetzung für die Zitierung)
#8na Kürzel
#8n1 Titel für alternatives Zitierformat 1
#8n2 Titel für Zitierformat 2  usw.
#8nr andere Verweisungsformen des Titels (getrennt durch "; ")
#74  Erscheinungsort
#88  ISSN
```

Die ISSN und das Kürzel sollen im Index 5 registriert sein. In CAT.API ist das z.B. so geregelt.
Unter #8n1, #8n2... kann man mehrere verschiedene Zitiertitel unterbringen. Nehmen wir an, es sollen nun die Aufsatzdaten mit dem Zitiertitel 2 gedruckt werden. Innerhalb der Druckparameterdatei hätten wir zu schreiben:

```
#70 +#8n2 i4,_ e";" |52
#70i |52
#8n2 C p"In: "
#74 p" (" P")"
#<
#704 p", "
#76 p"(" P")"
#706
#708 p", S."
```

Nimm Inhalt von #70; wenn es ein Kürzel ist, lade den zugehörigen Stammsatz. Wenn es keinen gibt: versuche es mit #70i, entnimm aus #8n2 Zitiertitel 2, neue Zeile, "In: " davorsetzen, Erscheinungsort in Klammern dahinter setzen, zurückschalten, d.h. vom Stammsatz wieder zum Titelsatz, restliche Aufsatzdaten ausgeben, so daß die Form Titel, Band(Jahr)Heft, S.nnn-nnn entsteht

Die zwei wichtigen Befehle sind hier der Manipulationsbefehl **|52** , der das Nachladen des Stammsatzes bewirkt, und der Steuerbefehl **#<** , der ihn wieder beseitigt, so daß die Arbeit am Ausgangssatz weitergehen kann.

Nun zur allgemeinen Form der Befehle für das Nachladen von Datensätzen.
Das Laden eines anderen Datensatzes wird durch den Manipulationsbefehl **'|'** veranlaßt. Man muß also zuerst die Kategorie angeben, in der der zu suchende Schlüssel steht. Man sorgt dafür, daß der Arbeitstext genau dem zu suchenden Schlüssel entspricht (auch bzgl. Groß-/Kleinschreibung!). Wenn also z.B. in #70i hinter der ISSN noch etwas anderes stehen könnte, wäre **#70i e9 |52** richtig gewesen (die ISSN hat immer 9 Stellen). Der **Nachladebefehl |52** nimmt dann den Arbeitstext und sucht im Index 5 danach. Bei Erfolg wird der zugehörige Satz gelesen, über den aktuellen Satz gestapelt und selbst zum aktuellen Datensatz gemacht. Nachfolgende Exportbefehle beziehen sich also auf diesen! Die allgemeine Form des Nachladebefehls sieht so aus:

|im Bedeutung: nimm den Arbeitstext als Schlüssel, suche danach im Register i und lade den gefundenen Satz in den Arbeitsspeicher, mache ihn zum aktuellen Satz. Dabei ist

i = Indexnummer (1,..,9, ':' für 10, ';' für 11): in diesem Index soll gesucht werden

m = Moduszíffer (0,..,9), und zwar: gesucht wird immer nach dem aktuellen Arbeitstext, jedoch mit unterschiedlichen Modalitäten:

0 : der erste Satz mit einem genau dem Arbeitstext gleichen Schlüssel
1 : der nächste mit genau gleichem Schlüssel (ein Befehl mit 0 muß vorangehen)
2 : wie 0, aber der Schlüssel kann länger sein als der Arbeitstext, er muß nur am Anfang mit ihm übereinstimmen
3 : wie 1, aber wieder auf den Anfangsteil beschränkt

Falls mehrere Ladebefehle aufeinander folgen, werden die nachgeladenen Sätze jeweils im Arbeitsspeicher aufgestapelt, falls man nicht **#<** gibt. Dies kann zwar erwünscht sein, manchmal jedoch nicht - dann addiert man besser gleich eine 4 auf die Moduszíffer, also

4 : wie 0, aber ein vorher geladener Satz wird beseitigt
5 : wie 1, usw.

aber: 8 : Der Satz wird nicht geladen, nur der erste (8) bzw. nächste (9) gefundene
9 : Schlüssel wird nach **#ux***i* kopiert und kann als Anwendervariable benutzt werden.

Sonderfall: Wenn es ungewiß ist, ob der gesuchte Schlüssel vorkommt, kann man '>' oder '<' vor den Arbeitstext setzen. Dann wird der erste Schlüssel ermittelt, der größer bzw. der letzte, der alphanumerisch kleiner ist als der Arbeitstext.
Achtung: Vor den Nachladebefehl, also direkt vor `|m`, einen Umcodierbefehl einsetzen, also `y1` oder `y2`, wenn in dem gerade aufbereiteten Schlüssel SOnderzeichen vorkommen können, die so nicht im Index stehen, sondern dort umcodiert sind - sonst klappt das Nachladen nicht.

Im Beispielfall geht es nur um einen einzigen Nachladesatz pro Ausgabesatz, dann wäre der Effekt bei **|56** derselbe wie bei **|52**.
Wenn es ungewiß ist, ob ein Stammsatz existiert, kann man durch einen bedingten Sprung auch Fehlerbehandlungen programmieren:

#70c +M |52 suche den Inhalt von #70c im Index 5, wenn gefunden, gehe zu #-M

würde bewirken, daß im Erfolgsfall zur Sprungmarke #-M gegangen wird, bei Mißerfolg aber mit der nächsten Zeile weitergemacht wird. Eine Anweisung mit Nachladebefehl bewirkt in keinem Fall eine Ausgabe, sie dient nur zum Auslösen des Nachladens.

Sonderfälle: **|00** Kurztitelzeile des aktuellen Satzes in die Variable **#ux0** kopieren
|01 Arbeitstext als Satznummer interpretieren und den zugehörigen Satz nachladen

Einschränkung: Das Nachladen ist auch in einer Index-Parameterdatei anwendbar, ABER: die Prozedur des Nachladens setzt ja voraus, daß schon ein Index vorhanden ist. Nur im 2. Durchlauf (siehe Kap. 7) kann deshalb ein Nachladen stattfinden, und auch nur dann, wenn beim ersten Durchlauf die erforderlichen Schlüssel, über die zuzugreifen ist, auch schon erzeugt wurden. Ab Version 14 gibt es für diesen Zweck die wesentlich bequemere Methode, Identnummern direkt und automatisch durch Klartexte ersetzen zu lassen (→ 10.2.6.8).

Der nachgeladene Satz bleibt solange aktuell, bis

- ein neuer Ladebefehl kommt (es hängt von dessen Moduszíffer ab, was dann passiert)
- der Exportsatz beendet wird (durch `#+#` oder `#+-` oder Erreichen des Endes der Kategorieliste)
- ein "Schaltbefehl" kommt:

In der Regel wird man aus dem nachgeladenen Satz eine oder mehrere Kategorien entnehmen, anschließend aber mit dem Ausgangssatz weitermachen wollen (wie im obigen Beispiel). Dann kommt der zweite neue Befehl zum Einsatz, der sog. **Schaltbefehl:**

Schaltbefehl: Es gibt 8 Formen, jede beginnt mit **#<** :

#<+M "zurückschalten" (oder anschaulich: auf dem Stapel 1 Satz hinunergehen) d.h. weitermachen mit dem aufrufenden Satz. Dieser kann seinerseits ein vorher nachgeladener Satz sein! Wenn die Rückschaltung ausgeführt werden konnte, geht es bei der Sprungmarke **#-M** weiter. **+M** darf fehlen, dann geht es mit der nächsten Zeile weiter. *Empfehlung:* Immer **#</**, wenn Rückschaltung nicht gebraucht wird.

#<^M Wirkung von **#<** rückgängig machen, d.h. wieder "hinaufschalten" zum vorher nachgeladenen Satz. Wenn man abwechselnd **#<** und **#<^** gibt, kann man beliebig oft zwischen den momentan geladenen Sätzen hin- und zurückschalten, ohne daß immer neu von der Platte nachgeladen werden muß.

#<<M zurückschalten zum Ausgangssatz, also zum untersten Satz des Stapels, damit man mit diesem weitermachen kann. Danach ist mit **#<^** jedoch (evtl. mehrfach) ein Rückschalten mit **#<^** zu den vorher nachgeladenen Sätzen noch möglich, denn der Stapel bleibt vorhanden.

#</M dasselbe, aber die aufgestapelten Nachladesätze werden nun weggenommen, d.h. anschließend kann man nicht mehr wieder mit **#<^** hinaufschalten.

#<iM Umschalten auf die parallele Datenbank (i=1,2,3), wenn man beim PRESTO-Aufruf 2 oder 3 Datenbanken aktiviert hat (→ Kap.12.1, Option -d). Mit **#<0** wird zur ersten Datenbank geschaltet und zugleich der Zeiger im Index zurückgesetzt, damit man weiterblättern kann. Sprung zu #-M, wenn das Umschalten funktioniert hat.

#<d Bei **hierarchischen Sätzen**: schalte zum nächsten Untersatz (#<d) / zum vorigen Untersatz (#<u) /

#<u zum Hauptsatz (#<U) des hierarchischen Satzes. Dahinter kann jeweils noch ein Zeichen stehen,

#<U z.B. **#<dM** heißt: springe nach **#-M**, wenn der Schaltbefehl ausführbar ist.

Das **M** darf jeweils auch fehlen. Dann fährt das Programm nach Ausführung mit der nachfolgenden Zeile fort. Wenn aber ein **M** gesetzt ist (erlaubt sind alle druckbaren Zeichen), wird nach **erfolgreicher** Ausführung des Schaltbefehls zur Sprungmarke **#-M** gegangen, und nur bei **Mißerfolg** (wenn kein nachgeladener bzw. Untersatz vorhanden war) geht es mit der nachfolgenden Zeile weiter. Folglich kann man auf jede Situation angemessen reagieren.

Anwendungsbeispiel "Mehrbändiges Werk"

Vorab: Die hierarchische Darstellungsweise mehrbändiger Werke kann beibehalten werden. Man kann sogar innerhalb derselben Datenbank wahlweise diese oder die im folgenden beschriebene neue Technik anwenden.

Mit den Lade- und Schaltbefehlen ist eine neuartige Speicherungstechnik für mehrbändige Werke realisierbar. Hauptaufnahme und Bände können getrennte, unabhängig voneinander gespeicherte Datensätze sein. Stößt man beim Zugriff auf einen Band, so soll er zusammen mit der Hauptaufnahme angezeigt werden, findet man letztere, soll das Programm sie mit allen zugehörigen Bänden präsentieren können. Es sind auch mehrere Hierarchiestufen möglich.

Vorbemerkung: Bisher brauchte man nicht unbedingt eine eigenständige, eindeutige Indentifikationsnummer für jeden Datensatz, jetzt aber geht es nicht ohne, denn die besagte Methode ist auf einen zuverlässigen Schlüssel angewiesen. In der Kategorie #00 der Hauptaufnahme muß also, zumindest bei den mehrbändigen Werken, etwas stehen, was eindeutig ist, und man muß es indexieren. (CAT.API macht dies im Index 9.) Es kann sich um eine Zugangsnummer oder Signatur handeln, oder die ID.-Nummer eines Verbundes, oder aber so etwas wie die Library of Congress Card Number. (Statt #00 dürfte es auch jede andere Kategorie sein, denn die Realisierung ist ja vollständig parametriert.)

Statt eine unselbständige Unteraufnahme an die Hauptaufnahme anzuhängen (mit einer Kategorie #01) tut man jetzt so, als erfasse man eine neue Hauptaufnahme. Mit zwei Unterschieden:

- In #00 der Bandaufnahme erfaßt man die Id.-Nr. der Hauptaufnahme **plus** der Angabe, die man sonst in die #01 schriebe, und zwar

  so: **#00** *idnr*+*ordnr*[=*bandnr*]

  idnr Id.Nr. der Hauptaufnahme

  ordnr Ordnungsnummer, sortierfähig (führende Nullen)

  bandnr Bandnummer in Vorlageform, falls abweichend

  z.B.: **#00 89-12345+015=Vol. XV**

- man erfaßt in der Bandaufnahme nur diejenigen Kategorien, die zur Bandstufe gehören, denn alle anderen wird man bei Bedarf aus der Hauptaufnahme entnehmen können, wenn man diese nachlädt.

Ob ein Satz unselbständig ist, kann am '+' innerhalb der #00 erkannt werden. Die Ordnungsnummer *ordnr* kann mehrteilig sein, wenn es sich um hierarchische Unterteilungen von Bänden handelt. Wenn man bisher z.B. #01 2 und #02 A gehabt hat, kann man nun z.B. #00 *idnr*+02+A geben, man trennt also die Stufen durch '+' . Die Gestaltung der sortierfähigen Bandnummern ist freigestellt und muß sich nach den Eigenheiten der vorliegenden Zählung richten. Die maximal mögliche hierarchische Gliederungstiefe ist dabei 12 (nach der alten Methode 6).
Wichtig ist ferner: die Kategorie #00 muß einschließlich der sortierfähigen Bandnummern indexiert sein. Dadurch entsteht im Index eine aufsteigende Folge der Identifikationsnummern: zuerst die der Hauptaufnahme (ohne '+'), dann die der Bände.
In der Hauptaufnahme braucht man außer der Speicherung einer Id.Nr. in #00 keine weiteren Vorkehrungen für die Verknüpfungstechnik zu treffen! Die Feststellung, ob zugehörige Bandaufnahmen existieren, wird mit den oben beschriebenen Befehlen über den Index durchgeführt. Es werden also, um es ganz deutlich zu sagen, keine Satznummern, -adressen oder -schlüssel irgendwelcher Art von untergeordneten Sätzen innerhalb des übergeordneten gespeichert. Die Zusammenführung und die Herstellung der Reihenfolge regeln sich ausschließlich über den Index. Damit ist die Verknüpfungstechnik unempfindlich gegenüber Reorganisationen.
Eine Parameterdatei für Anzeige oder Druck solcher Datensätze sollte nach folgender Vorschrift arbeiten:

1. Prüfe, ob der aktuelle Satz ein unselbständiger ist.
   ja : lade den übergeordneten Satz und mache ihn zum aktuellen Satz, fahre mit 1. fort (dies würde bis zu 12 mal funktionieren, das wäre allerdings sehr extrem)
   nein : fahre mit 2. fort ("aktueller Satz" ist jetzt der zuletzt geladene)

2. Gib den aktuellen Satz aus. Ist noch einer darunter auf dem Stapel?
   ja : nimm den obersten Satz vom Stapel und fahre mit 2. fort
   nein : fahre mit 3. fort (der Ausgangssatz wird nun wieder zum aktuellen Satz!)

3. Prüfe, ob mit dem aktuellen Satz noch ein untergeordneter verknüpft ist. Mit anderen Worten, ob er selbst den Hauptsatz eines verknüpft gespeicherten mehrbändigen Werkes darstellt.
   ja : alle untergeordneten Sätze nacheinander laden und ausgeben. (Dieser Vorgang ist potentiell unbegrenzt, denn die Anzahl der verknüpften Untersätze unterliegt keiner Beschränkung.)
   nein : Ende

Bei normalen einbändigen Werken wird zwangsläufig nur einmal Vorgang 2. ausgeführt.

Die mitgelieferten Parameterdateien D-1.APR für Bildschirmanzeige, P-KARTE für Kartendruck, sowie P-NORMAL und P-EINZEL enthalten diese Vorgänge und auch die unter dem Beispiel "Zitat" beschriebene Methodik. Sie sind ausreichend kommentiert und werden sehr zum Ausprobieren empfohlen.
In den Dateien D-Kxxx.APR und D-K.APT (xxx=RTF, HTM, DOS) findet man die Grundstruktur der Verknüpfungslogik als allgemeines Schema. In der Beispieldatenbank finden Sie im Index 9 eine zweistufige Aufnahme unter dem Schlüssel 654321 . Die Parameterdatei CAT.API und die genannten Exportparameter stellen also schon eine vollständige Implementierung der hier beschriebenen Verknüpfungen dar. Sie enthalten ab V14 auch Möglichkeiten für Personen-, Körperschafts- oder Schlagwortverknüpfungen (10.2.6.8).

Was geschieht beim Retrieval im Falle einer mehrstufigen Hierarchie? Es können drei Fälle vorkommen:

a) man stößt auf eine Aufnahme der obersten Stufe (Hauptaufnahme): der gesamte verknüpfte Komplex wird angezeigt - wie nach der alten Methode
b) ... auf eine Satz einer mittleren Stufe der Hierarchie: es werden hintereinander angezeigt: 1. alle direkt übergeordneten (keine parallel liegenden) Einheiten bis hinauf zur Hauptaufnahme werden angezeigt, 2. die Bandaufnahme selbst, 3. alle darunter hängenden Einheiten.
c) ... auf einen Satz der untersten Stufe: wie bei b), aber der Teil 3. entfällt.

Der wichtigste Fall ist dieser: der -zigste Band eines vielbändigen Werkes wird zusammen mit der Hauptaufnahme angezeigt - nur diese zwei. Das ist wesentlich übersichtlicher als bei der hierarchischen Satzstruktur.

10.2.6.8 Normdaten-Verknüpfungen

Mit Version 14 kam ein neues Verfahren, mit dem die Verwendung von Norm- oder Stammsätzen aller Art stark vereinfacht und wesentlich wirkungsvoller gemacht wurde. Mit einem Satz: Identnummern für Namen, Schlagwörter etc. können automatisch durch Klartexte (Ansetzungsformen) ersetzt werden.

Folgende Ziele wurden verfolgt:

- Die Methode muß mit jedem Kategoriesystem realisierbar sein
- Es muß möglich sein, im selben Feld im einen Fall Identnummern, im andern Fall Klartexte zu verwenden (kein striktes Entweder-Oder). Damit hängt zusammen:
- Die nach älteren Mustern gestrickten Daten sollen nicht verändert werden müssen (also Handlungsbedarf nur dann, wenn die neue Methode angewandt werden soll)
- Existierende Exportparameter müssen ohne Änderung weiter funktionieren, inclusive "Nachladungen"
- Der Aufwand für die Anpassung von vorhandenen Indexparametern an die neue Methode sollte sehr gering sein
- Bei Änderung einer Ansetzungsform im Stammsatz: automatische Änderung der Registereinträge verknüpfter Sätze
- Unterstützung der Erfassung: Stammsatznummern müssen leicht auffindbar und kopierbar sein
- Die bisher bekannten unterschiedlichen Modalitäten des Umgangs mit Identnummern müssen unterstützt werden (z.B. sind die Steuerzeichen verschieden, auch die Methoden der Wiederholbarkeit von Kategorien)
- Es muß parametrierbare Löschkontrollen geben: Stammsatz nicht löschbar, solange Verknüpfungen vorliegen

Wie funktioniert die Lösung?

Die Programme (PRESTO etc.) ersetzen jeweils vor dem Abarbeiten der Index- oder Exportparameter die Identnummern der Stammsätze durch die Klartexte, also die Namen, Schlagwörter etc. Nachladungen sind dann unnötig. Nach dem Abarbeiten wird der Datensatz im Arbeitsspeicher wieder in den Originalzustand versetzt. Die Klartexte werden nicht durch Nachladen der Stammsätze herbeigeschafft, sondern aus einem geeignet konstruierten Register ("Ersetzungsschlüssel", s.u.) - nur so geht es schnell genug. Der Index wird zwar größer, doch er ufert keineswegs aus.
(*Anmerkung:* Die Methode bezieht sich nur auf Stammsätze, nicht z.B. auf Verknüpfungen von mehrbändigen Werken. Dazu bleibt das "Nachladen" die einzige anwendbare Methode, siehe 10.2.6.7.)

Was muß man tun?

Soweit die Forderungen und das Konzept der Lösung, nun zur Realisierung. Was alles ist zu bedenken und zu tun, um die neuen Fähigkeiten der Programme zu nutzen? Damit die Übersicht nicht verlorengeht, müssen ein paar Randbedingungen festgelegt werden:

1. Stammsätze benötigen **Identnummern**. Die Identnummern müssen rein alphanumerisch sein (also nur aus Buchstaben und Ziffern bestehen, ohne Leerzeichen, Punkt, Binde- oder Schrägstriche). Es braucht jedoch keine bestimmte Kategorie für die Identnummern festgelegt zu werden, d.h. Körperschaftsstammsätze können z.B. dafür eine andere verwenden als Schlagwortstammsätze.
2. Die Identnummern müssen **alle im selben Register** stehen, und zwar in der Form `nummer=Klartext`. Hiervon hängt das Funktionieren der Ersetzungen ab. Das Register kann jedoch noch beliebige weitere Einträge enthalten.
3. Die Identnummern müssen **eindeutig** sein: es darf nicht z.B. dieselbe Nummer für ein Schlagwort und eine Körperschaft verwendet werden, obwohl dies ja unterschiedliche Stammsatztypen sind. (Das ist leicht durch ein Präfix wie p / k erreichbar!) Allerdings reicht es aus, die Nummern beim Indexieren durch ein Präfix eindeutig zu machen. Die Länge der Identnummern braucht nicht einheitlich zu sein.
4. Ein bestimmtes Zeichen muß als **Steuerzeichen** festgelegt werden. Das "konsolidierte Format" ($A.CFG) sieht dafür das Zeichen **'_'** vor. Das Pica-System verwendet **'!'**. Dieses Zeichen muß in den Titelsätzen stets **vor** eine Identnummer geschrieben werden, es **kann** (z.B. bei Pica) zusätzlich dahinter geschrieben werden.
   Das Steuerzeichen darf aber auch an anderen Stellen als normales Zeichen verwendet werden, denn wenn es nicht von einer Identnummer gefolgt ist, passiert nichts.
   Will man erreichen, daß für bestimmte Datenelemente keine automatische Ersetzung erfolgt, verwendet man für diese Fälle ein anderes Steuerzeichen, oder man steckt diese Nummern in ein anderes Register! Oder: man erzeugt keinen "Ersetzungsschlüssel" (s.u.) für diese Sätze.

Für Titelsätze braucht man **keine zusätzlichen Kategorien**. Man gibt statt eines Namens oder eines Schlagwortes dessen Stammsatz-Identnummer in derselben Kategorie ein, aber das Steuerzeichen muß davor stehen. In ein und derselben Kategorie kann also wahlweise ein Klartext oder eine Identnummer eingesetzt werden. Das Entscheidende für die Parametrierung ist: man kann so tun, als ob in den Kategorien *immer* Klartext stünde. Man muß die nachfolgend beschriebenen Voraussetzungen schaffen, dann braucht man sich bei der Parametrierung nicht mehr mit den Identnummern zu befassen, denn vor der Abarbeitung der Parameter ersetzen die Programme jeweils alle Identnummern durch die zugehörigen Klartexte. Damit dies funktioniert, muß man in die Index-Parameterdatei (z.B. CAT.API) drei neue Befehle einbauen. Nur **`i5`** muß unbedingt vorkommen, für die anderen beiden gibt es Standardwerte:

`i4=k` Die Ziffer k ist der **Ersetzungsmodus**: Standardwert: **`i4=-1`**
Wenn im Datensatz eine Angabe wie **`_NNN`** steht, dann bewirkt der Wert
0 : Ersatz von **`_NNN`** durch den Klartext, hinter **`_NNN`** folgende Zeichen werden beseitigt
1 : dasselbe, aber wenn hinter **`_NNN`** noch Zeichen folgen, bleiben diese stehen
2 : wie 0, aber das Zeichen '_' bleibt stehen (sinnvoll für Kontrollzwecke)
3 : wie 1, aber das Zeichen '_' bleibt stehen
4 : **`_NNN_`** bleibt stehen, der Klartext wird dahinter eingefügt, nachfolgende Zeichen verschwinden
5 : wie 4, aber nachfolgende Zeichen bleiben stehen.
-1 : Ersetzung inaktiv setzen – es findet keine statt.
Wenn in einer Kategorie mehrere Identnummern vorkommen können, kommen nur 1, 3 oder 5 in Frage.
Wird in einer Exportparameterdatei kein **`i4`** angegeben oder **`i4=-1`**, erfolgt keine Ersetzung.
Dieser Wert läßt sich nachträglich mit **`#bi4=-1`** einschalten und bewirkt, daß ab dem nächsten Satz keine Ersetzungen stattfinden. Wieder einschalten: **`#bi4=1`** o.a.

`i5=x` Das Zeichen x ist das festgelegte **Steuerzeichen** kein Standardwert!
Ohne den Befehl **`i5`** werden keine Ersetzungen durchgeführt. Das bedeutet insbesondere: alle bisherigen Index-Parameterdateien, die ja diesen Befehl nicht enthalten, funktionieren unverändert.

`i5=xx` *(zweimal dasselbe Zeichen)*
Das Steuerzeichen **`x`** muß im Feldtext vor und hinter dem Schlüssel stehen.
Vorteil: Im Schlüssel können auch Sonder- und Leerzeichen vorkommen.

`i6=i` Im Register `i` (mögliche Werte 1 - 11) stehen die **Ersetzungsschlüssel** (s.u.). Standard: **`i6=9`**
Alle für Ersetzungen nötigen Identnummern müssen dann in diesem Register stehen.
Dieses Register muß sich im Hauptindex befinden, also z.B. `cat.adx`, nicht `cat.aex` o.a.

Außerdem muß es bestimmte Typen von Schlüsseln geben, damit die Methode funktioniert. (Im PICA.PPI ist die Parametrierung dieser Schlüssel exemplarisch durchgeführt und kommentiert.)
Damit die Beschreibung nicht zu theoretisch wird, soll alles an konkreten Beispielen erläutert werden.
Hier ist ein Personenstammsatz (wie er im Format $A.CFG aussehen muß):

```
#00 123 oder auch z.B. #00 tucho
#4n Tucholsky, Kurt
#4naTiger, Theobald; Wrobel, Ignaz; Panter, Peter
```

In einem Titelsatz hätte man folglich z.B. dieses Datenfeld für Tucholsky als Verfasser:

`#40 _123` *bzw. z.B.* **`#40 _tucho`** statt **`#40 Tucholsky, Kurt`**

Beim Parametrieren für den Index kann man aber, wie gesagt, so tun, also ob man `#40 Tucholsky, Kurt` hätte.
Für den Stammsatz muß die Index-Parameterdatei folgende Schlüssel liefern, damit das klappt:

```
|9123=|1Tucholsky, Kurt              Ersetzungsschlüssel, 123 ist zugl. Primärschlüssel d. Stammsatzes
|9tucholsky, kurt _123               Übernahmeschlüssel (Erfassungshilfe)
|1panter, peter -> tucholsky, kurt   Verweisungen im Namensregister
|1tiger, theobald -> tucholsky, kurt
|1wrobel, ignaz -> tucholsky, kurt
_1|1tucholsky, kurt                  Pseudoschlüssel für automatische Änderung
```

Wenn hier von "Namen" die Rede ist, gilt dasselbe immer auch für Schlagwörter, Notationen, und andere Datenelemente, für die man Stammsätze nutzen will.
Die Parameter CAT.API enthalten ab V14 die Erzeugung dieser Schlüssel.
Es folgt die Erklärung, was es mit den einzelnen Schlüsseln auf sich hat und wie alles zusammenhängt.

Der **Ersetzungsschlüssel** dient dazu, die beschriebenen Ersetzungen durchzuführen. Bevor die Indexparameter abgearbeitet werden, sieht das Programm nach, ob in dem Datensatz das Zeichen **x** aus dem Befehl **i5=x** vorkommt. Wenn also z.B. **i5=_** ist und in einem Datensatz irgendwo **_123** steht (egal wo!), sucht das Programm im dem Register, das durch i6 festgelegt wurde, in diesem Fall also im Register 9, nach einer Zeile, die mit 123= anfängt. Den **Klartext**, der dahinter steht, hier also Tucholsky, Kurt (in Normalschreibweise, wohlgemerkt), setzt es dann statt der Nummer _123 ein. (Wenn 123 gefunden wird, aber **"=..."** fehlt, passiert nichts!) Die Angabe **|1** ist nur für das Programm INDEX nötig, damit dieses in seinem zweiten Durchgang korrekt arbeiten kann (d.h. die Umcodierung vornehmen kann, siehe unten, **#-1** ist die Sprungmarke dafür). Wenn alle diese Ersetzungen erledigt sind, werden die Indexparameter normal abgearbeitet. Danach werden die Ersetzungen sofort rückgängig gemacht, d.h. der Betrachter merkt von alledem nichts! Nur so ist es aber zu erklären, daß anschließend im Register 1 der Datensatz unter tucholsky, kurt zu finden ist, nicht unter _123 - was bis Version 13 nicht lösbar war.
Der Ersetzungsschlüssel kann zugleich **Primärschlüssel** sein: UPD vergleicht beim Einmischen nur den Teil, der vor dem '=' steht.
Im Falle i5=xx , also z.B. i5=__, muß der Schlüssel auch im Index vorn und hinten markiert sein, z.B. _tucho_, der Ersetzungsschlüssel muß also die Form _tucho_=|2Tucholsky, Kurt haben.

Der **Übernahmeschlüssel** ist als Erfassungshilfe gedacht. Man möchte ja auf bequeme und zuverlässige Art bei der Dateneingabe die Nummer statt des Namens eingeben können, oder besser: kopieren. Dieser Schlüssel muß nicht im Register 9 stehen, man muß nur bei der Erfassung wissen, wo man ihn findet. (Im Namensregister selbst würde er im OPAC womöglich eher irritieren.) Die Erfassung geht so: in dem Moment, wo ein Name einzugeben ist, z.B. in einer Abfragezeile, drückt man F6, das Register erscheint, man schaltet auf das Übernahmeregister um und sucht nach dem Namen. Man setzt den Pfeil auf die Zeile, drückt <Enter>, dann <Strg><Enter> (***a99***: im Indexfenster den Button [Kopie] = ALt+k), und die Nummer steht an der gewünschten Stelle. Wenn allerdings noch kein Stammsatz für den Namen vorhanden ist, müßte man diesen **zuerst** anlegen - oder den Namen zunächst im Klartext erfassen. Nebenbei bemerkt: solange ein Name nur einmal auftritt und keine Verweisungsform benötigt, solange ist ein Stammsatz blanke Umstandskrämerei.
Gelegentlich wird folgender Wunsch geäußert: In dem Moment, wo ein Name einzugeben ist, und wo man (oder das Programm) feststellt, daß es ihn im Katalog noch nicht gibt, muß ein Fenster aufgehen, wo man ihn als Stammsatz erfassen kann. Mit **allegro** geht das vorerst nicht. Es empfiehlt sich folgendes: vor der Erfassung des Titelsatzes kurz nachschauen (es geht ja wirklich schnell), ob die benötigten Namen etc. vorhanden sind; falls einer fehlt und man tatsächlich einen Stammsatz haben will, diesen **zuerst** anlegen, dann erst die Titelerfassung starten. Zuviel der Disziplin?
Möglich ist der Einbau einer Programmierten Validierung (→ 10.2.8), wodurch ein eingegebener Name o.a. am Register geprüft und bei Übereinstimmung eine Meldung ausgegeben wird.

Die **Verweisungen** sind ein wichtiger Grund, warum man überhaupt Stammsätze macht: sie führen im OPAC von der Verweisungsform zur Ansetzungsform und gelten für beliebig viele Titel, obwohl sie nur einmal, eben im Stammsatz, gespeichert sind. Eine neue Verweisung wird nur im Stammsatz ergänzt, an den Titelsätzen ist nichts zu tun. Die Anzahl der Verweisungen und die Art ihrer Darstellung in den Registern haben auf die Funktion der Verknüpfung keinen Einfluß.

Der **Pseudoschlüssel** ist ein sehr wichtiges Hilfsmittel. Er wird nicht einsortiert (daher "Pseudo"), sondern nur für Kontrollzwecke gebraucht: Gesetzt den Fall, man ändert im Stammsatz die Ansetzungsform (**#4n** im Beispiel). Dann müssen ja auch die Titeleinträge, die unter tucholsky, kurt standen, auf die neue Form geändert werden. Das Programm merkt, wenn sich bei einer Korrektur am Stammsatz der Pseudoschlüssel ändert. Dann sucht es die unter der alten Form stehenden Einträge zusammen und überführt sie auf die neue Form. Die Titelsätze werden dazu nicht angefaßt: dort steht ja nur die Nummer, und die ändert sich nicht. (Besser gesagt: sie sollte sich nicht ändern. Man setzt am besten in der .CFG mit **▼P5** den Änderungsschutz auf die Kategorie #00). Wenn man in der Index-Parameterdatei keinen Pseudoschlüssel anlegt, finden derartige automatische Verlagerungen nicht statt.
Der Pseudoschlüssel kann manchmal mit dem **Löschkontrollschlüssel** (s.u.) kombiniert werden. Im Beispiel ist "|1tucholsky, kurt" zugleich der Schlüssel, der vor dem Löschen des Stammsatzes kontrolliert werden muß.
Der Pseudoschlüssel beginnt mit dem Steuerzeichen i5, hier also mit '_'. Die Ziffer '1' dahinter hat eine wichtige Bedeutung: an dieser Stelle kann auch '0' stehen. Die '0' bedeutet, daß nur solche Einträge verlagert werden sollen, die exakt "tucholsky, kurt" lauten. Die '1' dagegen bewirkt, daß alle Einträge verlagert werden, die mit tucholsky, kurt beginnen, also z.B. Schlagwortketten mit diesem Namen als Anfangsteil. Diesen Modus kann man also für die einzelnen Stammsatztypen unterschiedlich festlegen.
Beim Entwurf einer Datenbank sind noch zwei Dinge zu beachten:

1. Je Stammsatz kann man mehr als einen Pseudoschlüssel definieren. Das bedeutet, daß mehrere Eintragungen automatisch verlagert werden können. Würde man z.B. neben dem Namensregister noch ein Verfasser/Titelregister machen, dann kann sich dort die Änderung des Stammsatzes auch auswirken. Dies wird z.B. bei der Parametrierung des Modells ***bolero*** für Musik-Katalogisierung angewandt.

2. *Vorsicht:* Die automatische Verlagerung erfaßt auch solche Einträge, die gar nicht vom Stammsatz herrühren! Wenn man z.B. in manchen Titelsätzen `#40 _123` stehen hat, in manchen anderen aber den Klartext `#40 Tucholsky, Kurt`, dann sehen die Registereinträge für beide gleich aus. Bei einer Änderung der Ansetzungsform im Stammsatz werden dann beide verlagert, denn an das Programm greift dabei nicht auf die Titelsätze zu (das würde zu lange dauern!). Allerdings wird man, wenn man überhaupt mit Stammsätzen arbeitet, solche Mischanwendungen zu vermeiden suchen. Entweder *hat* man für eine Person einen Stammsatz, dann sollte man diesen in jedem Fall verwenden, *oder man hat keinen*, dann kann auch nichts schiefgehen. Mehr Überlegung ist nötig, wenn man den Ersetzungsmodus '1' anwendet. Insbesondere bei Schlagwörtern muß man prüfen, ob dann nicht doch Einträge betroffen werden, die tatsächlich nichts mit dem Stammsatz zu tun haben sondern nur zufällig denselben Anfangsteil besitzen. Will man in ein schon laufendes System sukzessive Stammsätze einführen, sollte man für jeden neuen Stammsatz die evtl. schon vorhandenen Klartexteinträge auf Nummern umstellen.

Diese zwei Einschränkungen sind nicht gravierend. Erstens sind Änderungen der Ansetzungsform in Stammsätzen ihrer Natur nach nicht eben häufig, zweitens werden die wirklich wesentlichen Einträge in jedem Fall geändert, drittens ist durch eine Neu-Indexierung im Notfall alles wieder zu korrigieren.

Löschkontrolle

Sobald man mit Satzverknüpfungen arbeitet, ganz gleich für welche Zwecke, entsteht das Problem der Abhängigkeiten: z.B. können verknüpfte Bandsätze von mehrbändigen Werken nicht mehr ohne den Hauptsatz korrekt verarbeitet (angezeigt oder ausgedruckt) werden. Man wird also verhindern wollen, daß ein Hauptsatz gelöscht wird, solange es noch einen verknüpften Bandsatz gibt. Man wird auch verhindern wollen, daß ein Schlagwortstammsatz gelöscht wird, wenn dieser in irgendeinem Titelsatz benutzt wurde. Die Datenbanktheorie hat hierfür den Fachterminus "referentielle Integrität". Bestehende Beziehungen dürfen nicht unbeabsichtigt zerrissen werden. Wie aber soll man das verhindern, wie kann der Löschbefehl abgefangen werden, wo die Verknüpfungen doch so unterschiedlich aussehen können? PRESTO und seine Derivate, aber auch UPD ab Version 13a enthalten dafür eine Methode, die "Löschkontrolle" heißt.
So installiert man eine Löschkontrolle:
In der Indexparameterdatei richtet man einen Abschnitt mit der Sprungmarke **`#--`** ein. Diese kann nicht normal angesprungen werden, da **`+-`** der Abbruchbefehl ist! Der Abschnitt wird nur dann von den Programmen herangezogen, wenn eine Löschung erfolgen soll. In dem Abschnitt hat zu stehen, welche Schlüssel vor jeder Löschung zu prüfen sind. Das können also auch mehrere sein. Nehmen wir als Beispiel die Verknüpfungen zwischen Haupt- und Untersätzen und die zwischen dem Sammelband und den enthaltenen Aufsätzen, wie sie im A-Format definiert sind. Der bewußte Abschnitt könnte dann so aussehen:

```
#--
!00 p"|9" P"+"
#t { C }      Zeilenvorschub zwischen den zwei Schlüsseln!
!00 p"|9" P"*"
#+#
```

Wenn also ein Datensatz die Kategorie `#00 123456` enthält, werden für die Löschkontrolle daraus die zwei Schlüssel `123456+` und `123456*` gebildet, die beide im Register 9 zu prüfen sind. Werden dort Einträge gefunden, die mit diesen Zeichenkombinationen **beginnen**, wird die Löschung verweigert.
Man beachte den Befehl `#t{ C }`, der normalerweise bei einem Export einen Zeilenvorschub auslöst. Hier trennt er zwei unterschiedliche Schlüssel, die getrennt zu prüfen sind. Ansonsten hat ein solcher Befehl in einer Indexparameterdatei nichts zu suchen. Auf diese Weise können bis zu 32 Löschkontrollschlüssel definiert werden.

Aber wie funktioniert dann das Programm INDEX?

Genau diese Frage werden diejenigen stellen, die seit ***V11*** unzufrieden waren, weil man in der Index-Parameterdatei nicht "nachladen" konnte (→ 10.2.6.7). Es ist natürlich immer noch so: das Programm INDEX kann die oben beschriebenen Ersetzungen nicht vornehmen, weil zu dem Zeitpunkt, wo es die Daten abarbeitet, der Index noch gar nicht vorhanden ist, aus dem es die Klartexte entnehmen müßte. Für das oben benutzte Beispiel bedeutet dies: INDEX erzeugt den Schlüssel **`_123`** für den Titelsatz anstatt **`tucholsky, kurt`**. Es entsteht also zunächst ein Index mit vielen solchen Einträgen. Diese müssen in einem **zweiten Durchlauf** aufgelöst werden. In diesem zweiten Durchlauf schaut INDEX nach, ob Einträge vorhanden sind, die mit '_' beginnen, also z.B. `_123`. Dann sucht es im Register i6 nach `123=`, nimmt den Klartext, den es dahinter findet, führt die Befehle bei Sprungmarke #-A aus, und setzt das Ergebnis in das Register, wo _123 stand, und _123 wird dort gelöscht. Wenn "`123=`" im Register i6 nicht gefunden wird, bleibt _123 unverändert stehen.
Den zweiten Durchlauf startet QRIX automatisch, man kann ihn aber auch von Hand starten:

`INDEX -fa1 -d*datenpfad -eindexparameter/datenpfad -kkonfig`

d.h. INDEX hat eine neue Funktion **`-fa`** erhalten. Sind keine Einträge der bewußten Art vorhanden, geschieht natürlich nichts. Dieser Durchlauf kann auch gestartet werden, wenn die Datenbank schon (im Netz) in Betrieb ist.

Für sehr große Datenbanken ist es von Vorteil, die Datenbank schon wieder freigeben zu können. Der zweite Durchlauf geht allerdings recht schnell, weil die Datensätze selbst nicht nochmals gelesen und ausgewertet werden müssen.
Aus diesem Verfahren ergibt sich eine weitere Einschränkung: wenn man z.B. einen Namen in einer weiteren Index-

eintragung benutzt als zweiten oder weiteren Bestandteil eines Schlüssels, kann INDEX diesen im zweiten Durchlauf nicht finden, und er bleibt als Nummer im Register stehen. Unter PRESTO würden die korrekten Einträge entstehen, da die Ersetzung **vor** dem Abarbeiten der Indexparameter stattfindet. Hier ist zu bemerken, daß mehrteilige Schlüssel eine ***allegro***-Eigenheit sind, die andere Datenbanksysteme überhaupt nicht kennen. Auch die Bildung mehrerer Schlüssel aus einem Datenfeld ist in der Datenbankwelt eine eher seltene Besonderheit.

Es gibt noch eine **Alternative**: INDEX kann in einem ersten Durchlauf mit den Optionen **`-fi0 -@1`** oder auch **`-f70 -@1`** zunächst NUR die Primärschlüssel erstellen, in einem zweiten Durchlauf dann mit **`-fi1 -@2`** alle anderen Schlüssel. Wenn es viele Stammsatzverknüpfungen gibt, ist diese Methode schneller! Und richtig nachladen kann INDEX in diesem zweiten Durchlauf übrigens auch. Mehr zu dem Thema: siehe Kap. 7.1.

Und die Kurzanzeige?
Auch diese kann Elemente enthalten, z.B. den Verfassernamen, die in den Titelsätzen nur aus einer Stammsatznummer bestehen. INDEX korrigiert im zweiten Durchlauf auch die Kurzanzeige, wenn dies der Fall ist. Unter PRESTO ist es ohnehin kein Problem, da vor der Abarbeitung der Index-Parameterdatei, also auch der Kurzanzeige, die Ersetzungen automatisch erfolgen.

SRCH
Nun zur Volltextsuche. Wenn man "`tucho`" als Suchwort eingibt, will man alle Datensätze finden, in denen diese Zeichenfolge vorkommt, auch die, in denen statt dessen "`_123`" steht. Auch dieses funktioniert. SRCH liest bei Version 14, anders als früher, vor der Suchaktion die Index-Parameterdatei und informiert sich darüber, ob derartige Verhältnisse vorliegen können, d.h. es schaut sich die Befehle i5 und i6 an. Und dann tut es folgendes: es liest einen Datensatz ein, kopiert ihn in einen "Suchbereich", nimmt per Index die Ersetzungen vor und ersetzt dann noch mit Hilfe der Tabelle S1.ASP einzelne Zeichen (z.B. Groß- durch Kleinbuchstaben). Somit steht dann im Suchbereich tatsächlich "`tucholsky`". Genauer: dort steht "`_123_tucholsky, kurt`". Das bedeutet: man kann sowohl nach dem Namen wie auch der Nummer suchen. Wenn man z.B. als Suchbegriff "`#4,123`" gibt, werden alle Sätze gefunden, die in einer Kategorie der Gruppe #4 die Ziffernfolge 123 enthalten. Wenn man "`#4,tucho`" gibt, werden alle Sätze gefunden, die in einer Kategorie der Gruppe #4 die Zeichenfolge "tucho" enthalten - auch dann, wenn diese durch Ersetzung erst über den Index hereingeholt wird, denn die Ersetzung passiert vor der Abarbeitung des Suchbefehls.

Für den **Export**, den SRCH evtl. vornehmen soll, ist es freilich noch etwas anders: dafür steht im normalen Arbeitsspeicher der unveränderte Satz (also mit `#40 _123`). Es kann ja sein, daß man genau so und nicht anders exportieren will. Wenn man will, daß die Nummern durch Klartexte ersetzt werden, braucht man nur in der **Export-Parameterdatei** den Befehl **`i4=0`** unterzubringen, das ist alles - Nachladen erübrigt sich. Oder **`i4=4`**, dann stehen Nummer **und** Name für den Export zur Verfügung. Der Wert i4 kann also für den Export anders aussehen als in der Index-Parameterdatei! Alle älteren Exporte funktionieren unverändert, denn sie enthalten keinen Befehl i4.

UPD
Das Programm UPD verfügt ab V14 für die Behandlung der Verknüpfungen über dieselben Funktionen wie PRESTO. Wie oben erwähnt, kann der Ersetzungsschlüssel zugleich als Primärschlüssel dienen. UPD vergleicht ihn nur bis zum Zeichen '=' mit den vorhandenen Schlüsseln. Wenn also ein korrigierter Stammsatz geliefert wird, in dem die Ansetzungsform verändert wurde, kann UPD ihn erstens richtig zuordnen und zweitens die nötigen Veränderungen im betroffenen Register und evtl. in der Kurzanzeige vornehmen.

Die auf PRESTO beruhenden Programme ALFA, APAC, MENUED etc. enthalten ebenfalls die neuen Funktionen.
Noch zwei Kleinigkeiten:

- Wenn man bei PRESTO mit F5 auf Kategorieanzeige umschaltet, sieht man hinter allen Identnummern die Klartexte (so ähnlich wie beim Pica-System). Für den Katalogisierer ist das eine sehr wichtige Hilfe. Wenn man allerdings mit 'E' in die Bearbeitung geht, sieht man nur noch die Identnummern (anders als bei Pica, wo die Klartexte in Rot dahinter stehen.)
- Ein Druck auf ':' führt in der Titel- oder Kategorieanzeige die Ersetzungen durch, die sich durch die Angabe i4 in der Index-Parameterdatei ergeben. Das ist noch eine Kontrollmöglichkeit: man sieht auf diese Weise, wie der Datensatz aussieht, bevor die Indexparameter abgearbeitet werden. Bei F7 wird das alles automatisch ausgeführt (so daß man gleich die richtigen Schlüssel sieht), die Ersetzungen aber im Arbeitsspeicher sofort rückgängig gemacht.

10.2.6.9 Satzübergreifende Suche : Das "Schiller-Räuber"-Problem

Seit es die verknüpften Sätze für mehrbändige Werke gibt (1993, V11.2), ist immer wieder einmal ein Disput über das "Schiller-Räuber-Problem" aufgeflammt. Dabei handelt es sich nicht um ein ***allegro***-spezifisches Problem, sondern alle bekannten Systeme kranken daran (außer MARC-Systemen, weil sie keine Satzverknüpfungen kennen). Keine ganz geringe Herausforderung also, und es war lange Zeit unklar, wie dieses Problem am besten anzupacken wäre. Schließlich und endlich wurde für ***allegro*** eine sinnvolle Lösung gefunden und realisiert. Die folgenden, zur Lösung hinführenden Gedanken sollen zunächst das Problemverständnis fördern und mit instruktiven Beispielen den Lösungsweg begründen.

Was ist überhaupt das Schiller-Räuber-Problem?
Wenn ein mehrteiliges Werk "Schillers Dramen" heißt, der Name "Schiller" aber nur im Hauptsatz und der Titel "Räuber" in einem **verknüpften** Untersatz steht, kann man nichts finden mit einer logischen UND-Verknüpfung zwischen `schiller und raeuber .`
Warum nicht? Zur Bildung von Ergebnismengen (und das gilt nicht nur für ***allegro***) werden aus dem Index nur die Satznummern entnommen und verglichen, die beiden Suchwörter stehen aber in getrennten Sätzen. Das logische UND sieht also zwei ungleiche Nummern und erfaßt sie deshalb nicht.
Eine gängige Abhilfe besteht darin, den Namen des Verfassers auch in jeden Untersatz mit einzutragen. Das erzeugt erstens unerwünschte Redundanz, löst aber zweitens nur einen Teil der Probleme dieses Typs, denn auch für andere Suchbegriffe, nicht nur für Verfasser, wird u.U. eine **satzübergreifende** Suche gewünscht oder stillschweigend erwartet, z.B. für Schlagwörter.

Hier ein anderes Beispiel, damit es noch deutlicher wird (nicht erfunden, dieses Werk gibt es!)
Wir lassen alles weg, was mit dem Problem nichts zu tun hat.

```
Hauptsatz:    Vollmer, Gerhard:
Nr.A          Was können wir wissen?
              Schlagwort: Erkenntnistheorie
```

```
Untersatz:    Band 1
Nr.B          Die Natur der Erkenntnis : Beiträge zur Evolutionären
              Erkenntnistheorie.
              Mitverf.: Lorenz, Konrad.
```

```
Untersatz:    Band 2
Nr.C          Die Erkenntnis der Natur : Beiträge zur modernen
              Naturphilosophie.
              Schlagwort: Naturphilosophie
```

Zur Erklärung:
Die Nummern A, B und C sollen die internen Satznummern der drei Datensätze sein, nicht die Primärschlüssel. Die Ergebnismengen werden stets aus diesen Nummern gebildet, nicht aus Bestandteilen der Datensätze! Man könnte also z.B. haben: A=13245, B=144923, C=55921. Das Programm vergibt sie beim Speichern selbst und man kann sie nicht verändern, daher haben diese Nummern keinen logischen Zusammenhang, und daher kann das Programm von sich aus nicht erkennen, daß ein Zusammenhang zwischen den Sätzen besteht. Aber, wie gesagt, beim Bilden von Ergebnismengen hat das Programm nur diese Nummern und sonst nichts, es schaut dabei nicht in die Sätze hinein (das würde viel zuviel Zeit kosten). Man muß also einen Weg finden, den Zusammenhang auch für das Programm erkennbar zu machen – das ist der programmiertechnische Kern des Problems.

Übliche Retrievalsysteme erbringen in der Regel folgende Resultate, manchmal eben weniger als erwünscht:
(Der Leser überlege für sich selbst, was wohl sinnvoll ist. Wir geben hier bewußt ziemlich viele Beispiele, damit deutlich wird, daß nicht jedes denkbare Ergebnis unbedingt erwünscht ist.)

	Suchfrage:	Resultat:	erwünscht:	
1.	vollmer	A	A,B,C	???
2.	wissen	A	A,B,C	???
3.	erkenntnis	B,C		
4a.	erkenntnis?	A,B,C		
4b.	naturphilosophie	C	A,C	???
4c.	erkenntnisth? AND naturphil?	nichts	C	
5.	vollmer AND wissen	A	A,B,C	???
6.	wissen AND natur	nichts	B,C	???
7.	vollmer AND natur (entspricht schiller-raeuber)	nichts	B,C	
8.	vollmer AND naturphilosophie	nichts	C	
	vollmer AND lorenz	nichts	B	
9.	lorenz AND erkenntnis	B		
10.	lorenz AND wissen	nichts	B	???

(Die Suchfragen sind hier vereinfacht dargestellt. Von der Sache her spielt es keine Rolle, ob die Begriffe in verschiedenen Registern stehen oder im selben. Je nach Syntax des Systems sieht der eigentliche Befehl anders aus, bei *a99/alcarta* ist es wie bei Pica: **`find per vollmer? and tit natur`**. Das '?' ist auch bei Pica das Trunkierungssymbol. Das SR-Problem, nebenbei bemerkt, ist bei Pica nicht gelöst.)
In den Fällen 1, 2, 4b, 5, 6 und 10 kann man wohl nicht generell sagen, daß die erweiterten Ergebnisse in jedem Fall erwünscht sind, sicher trifft das nur für die Fälle 4c, 7 und 8 zu. Es gibt natürlich keinen Algorithmus, der den tatsächlichen Wunsch erschließen kann und in jedem Fall korrekt eine erweiterte Suche (satzübergreifende Suche) durchführt oder nicht. Folglich wird man dem Benutzer auf irgendeine Art eine neue Option "erweiterte Suche" oder "Expansion" zur Wahl stellen müssen und ihm die Sache hinreichend erklären.
Das eigentliche Problem tritt nur bei den UND-Verknüpfungen auf. Die Fälle 1 bis 3 gehören nicht zum Problembereich. Wenn A gefunden wird, hat man implizit auch B und C, da diese dann ohnehin mit angezeigt werden können. Die nachfolgend skizzierte Lösungsidee wird zwar auch die logische NICHT-Verknüpfung mit einbeziehen (siehe unten Beispiele 5,6,10), aber der Leser mag sich selbst überlegen, wie sinnvoll das ist.

Abstrakte Lösungsidee
Es wird ein Steuerzeichen eingeführt, und zwar '&'. Dieses kann vor jeden Suchbegriff einer kombinierten Suche gesetzt werden.
Also z.B. `&vollmer` statt `vollmer.`
Ein so gekennzeichneter Begriff soll "erweitert" (satzübergreifend) gesucht werden. (Man könnte auch von einer "Plus-Suche" sprechen.) Das bedeutet: alle Untersätze werden zu den Ergebnissen hinzugenommen. (Nur die `UNTERsätze`, nicht die ÜBERgeordneten!) Dann würde also im Beispiel `&vollmer` `A,B,C` herauskommen statt nur schlicht `A.` Mit `&lorenz` würde aber unverändert nur B herauskommen. Das erscheint sinnvoll, denn Vollmer hat viel mit B und C zu tun, Lorenz dagegen nicht viel mit A und womöglich gar nichts mit C. Diese Überlegung trifft wohl auch für andere Suchkriterien zu, denn schließlich erfaßt man ja ganz bewußt im Hauptsatz diejenigen Elemente, die auf das Gesamtwerk zutreffen (also AUCH auf die Untersätze), in den Untersätzen aber diejenigen, die NUR für den jeweiligen Untersatz gelten.
Also nochmal ganz klar: die "erweiterte Suche" bezieht zu jedem Datensatz alle **unter**geordneten Sätze mit ein, und dies muß auf jeden Term einer kombinierten Suche einzeln anwendbar sein, sonst können durchaus unerwünschte Ergebnisse entstehen.

Die nachfolgenden Beispiele sind nicht alle sinnvoll, d.h. man würde sie in der Praxis nicht alle anwenden. Das heißt, es wäre wohl nicht zu empfehlen, die Erweiterung grundsätzlich dauerhaft einzuschalten, also als ob automatisch vor jedes Suchwort ein '&' gesetzt würde. Es sollte aber einen "Hauptschalter" geben, der genau dieses bewirkt, wenn es gewünscht wird.

	Suchfrage:	Ergebnisse:
1.	vollmer	A
	&vollmer	A,B,C
2	wissen	A
	&wissen	A,B,C
3.	erkenntnis	B,C
	&erkenntnis	B,C
4.	erkenntnis?	A,B,C
	&erkenntnis?	A,B,C
	erkenntnis? AND naturphil?	C
	erkenntnisth? AND naturphil?	nichts
	&erkenntnisth? AND &naturphil?	C
5.	vollmer AND wissen	A
	&vollmer AND wissen	A
	vollmer AND &wissen	A
	&vollmer AND &wissen	A,B,C
	vollmer NOT wissen	nichts
	&vollmer NOT wissen	B,C
	&vollmer NOT &wissen	nichts
6.	vollmer AND natur	nichts
	&vollmer AND natur	B,C
	vollmer AND &natur	nichts
	&vollmer AND &natur	B,C
	vollmer NOT natur	A
	&vollmer NOT natur	A
7.	wissen AND natur	nichts
	&wissen AND natur	B,C
	wissen AND &natur	nichts
	&wissen AND &natur	B,C
8.	vollmer AND naturphilosophie	nichts
	&vollmer AND naturphil?	C
	vollmer and lorenz	nichts
	&vollmer AND &lorenz	B
9.	lorenz AND erkenntnis	B
	&lorenz AND erkenntnis	B
	lorenz AND &erkenntnis	B
	&lorenz AND &erkenntnis	B
10.	lorenz AND wissen	nichts
	&lorenz AND wissen	nichts
	lorenz AND &wissen	B
	&lorenz AND &wissen	B
	lorenz NOT wissen	B
	lorenz NOT &wissen	nichts

Es wird hieraus deutlich, daß mit der "Plus-Suche" jede gewünschte Erweiterung des Zugriffs differenziert durchgeführt werden *kann*. Voraussetzung: man muß sich klar sein, was man will, und die Plus-Suche *bewußt* einsetzen.
Ob dies verstanden und ausgenutzt werden wird und *wie oft*, das ist eine ganz andere Frage. Man kann auch die Meinung hören, daß die Booleschen Verknüpfungen insgesamt von kaum einem Endbenutzer verstanden werden, wie auch immer man sie verpackt... Andererseits kann man sicher sein, daß die Wünsche nicht verstummen, nachdem dieses Verfahren realisiert ist. Mindestens wird die Frage kommen, wie es denn ist, wenn einer der Beispielsätze, oder alle drei, noch einem anderen oder mehreren anderen Werken untergeordnet sind, und wenn es mehrere Hierarchiestufen gäbe...
Wahr ist aber auch: je größer unsere Datenbanken werden, umso mehr brauchen wir differenzierte Suchmöglichkeiten, auch wenn sie dann nur verhältnismäßig selten zum Einsatz kommen.

Was muß man parametrieren?

Zum Glück wenig. Es genügen ein paar für Kenner sehr einfache Maßnahmen in den Indexparametern.

1. Neuer Befehl `i7`

Mit dem neuen Befehl `i7` wird zum einen das Register festgelegt, wo die notwendigen Hilfsschlüssel anzusiedeln sind, zum andern wird damit dem System signalisiert, **daß** die Datenbank eine Plus-Suche gestattet.

`i7=k` Nummer des SR-Hilfsregisters (1 <= k <= 11) Empfehlung: k= 9 oder 10.

2. SR-Hilfsschlüssel: neuer Manipulationsbefehl `X`

Jeder Untersatz, der bei einer Plus-Suche auffindbar sein soll, muß einen oder mehrere zusätzliche Schlüssel bekommen. Ein solcher "SR-Hilfsschlüssel" muß aus dem Primärschlüssel eines anderen Satzes bestehen (i.d.R. ist das der Hauptsatz eines mehrbändigen Werkes). Es kann zu einem Untersatz mehr als einen Hilfsschlüssel geben, d.h. die Unterordnung unter zwei oder mehr Gesamtwerke kann adäquat berücksichtigt werden.
Ein solcher Hilfsschlüssel wird mit Hilfe des neuen Manipulationsbefehls **`X`** angelegt. Wie das geht, zeigen wir an einem *Beispiel:* Wir gehen vom Konsolidierten Format aus, wo der Primärschlüssel aus #00 gebildet wird und Untersätze in der #00 die IdNummer des Hauptsatzes mit angehängtem '+' enthalten. Die Primärschlüssel stehen im Register 9. Die Hilfsschlüssel sollen aber im Register 8 stehen. Es folgen die notwendigen Zeilen:
(Eine reservierte Sprungmarke gibt es für diesen Zweck übrigens nicht.)

```
i7=8                Die SR-Hilfsschlüssel sollen ins Register 8
...
ak=zz+U             Für jeden Satz wird der Abschnitt #-U angesprungen ...
...
#-U                 ... aber nur bei Untersätzen passiert hier wirklich etwas:
#00 +#00 c"+" e0            Kommt '+' vor, d.h. ist es ein Untersatz?
#+-                 wenn nicht, dann keine Aktion
#00 y0 e"+" X9      SR-Hilfsschlüssel aus Inhalt von #00 per Reg. 9 bilden.
#+#                 (Für Register 10 muß es  X:  heißen; dann aber ':' aus Befehl  i2 herausnehmen!)
```

Und was passiert dann?
Das braucht man nicht unbedingt zu wissen, aber nützlich zum Verständnis dürfte es sein:
Das Programm INDEX bildet zunächst nur einen temporären Schlüssel im Register 1. Wenn man z.B. #00 55555+17 in einem Untersatz hat, sieht der temporäre Schlüssel so aus: \|955555 (Mit F7 in PRESTO und a99 sieht man das). Im zweiten Durchlauf macht INDEX daraus den eigentlichen Hilfsschlüssel. Wenn der Hauptsatz (der mit dem Primärschlüssel 55555 im Register 9) die interne Nummer 123 hat, dann lautet der SR-Hilfsschlüssel |123 und steht im Register 8. Das besagt also: der Untersatz gehört zum Satz mit der internen Nummer 123.
Die Hilfsschlüssel dürfen auch im selben Register stehen wie die Primärschlüssel.

Wenn die Plus-Suche dann ausgelöst wird, spielt sich folgendes ab:
Das Programm nimmt sich die aktuelle Ergebnismenge vor. Diese besteht für das Programm immer nur aus einer Liste von internen Satznummern. (Wohlgemerkt: nur diese sind in dem Moment greifbar, nicht die Primärschlüssel.) Für jede dieser Nummern wird im Register i7 (im Beispiel also Register 8) nachgesehen, ob es Hilfseinträge unter dieser Nummer gibt. Wenn ja, werden die zugehörigen internen Nummern zur Ergebnismenge hinzugenommen.

Daraus ergibt sich: die Plus-Suche muß auf einen Suchbegriff angewendet werden, **bevor** die logische UND-Verknüpfung ausgeführt wird. Nochmals: was ist dafür zu tun?

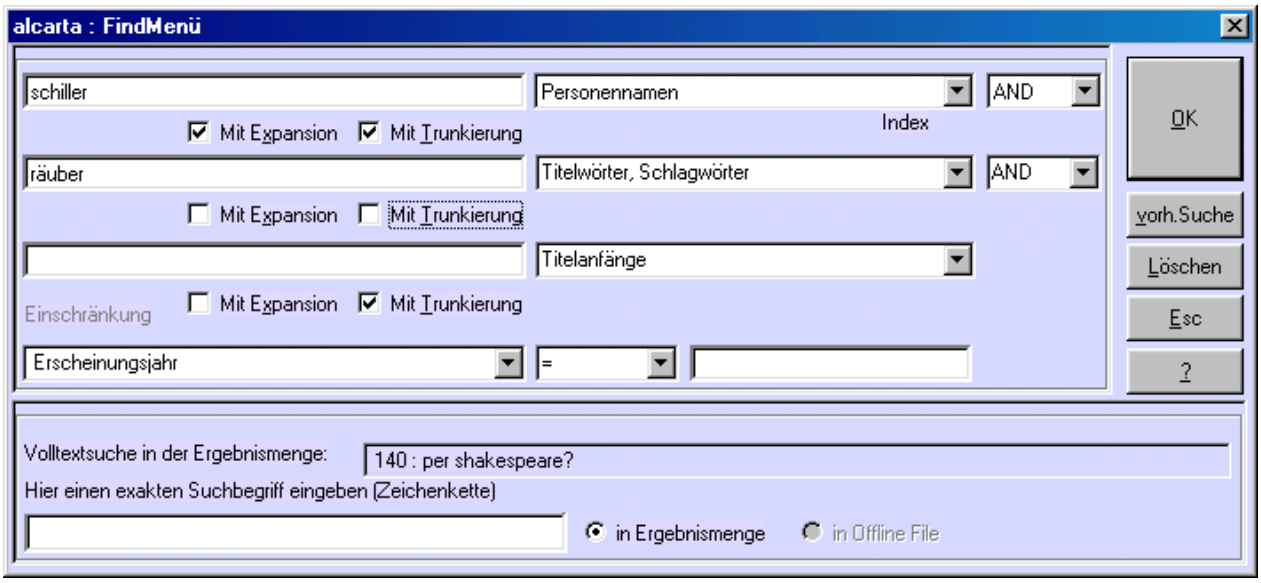
In a99, PRESTO und APAC kann jeweils nur die zum ersten Suchbegriff gehörige Ergebnismenge erweitern. In den Beispielen entspricht das dem Fall `&vollmer AND natur`, also genau dem originalen Schiller-Räuber-Problem.
Man bilde also zuerst die Ergebnismenge zu demjenigen Begriff, der tatsächlich erweitert werden soll (d.h. der möglicherweise in einem Hauptsatz vorkommt, z.B. schiller oder vollmer). Danach Plus-Suche auslösen:
In PRESTO wird '&' gedrückt (auf dem Registerbildschirm).
In APAC ist dafür auf dem ESC-Menü die neue Funktion "Verknüpfungen berücksichtigen" zu finden.
Die Ergebnismenge wird dann etwas größer, aber nur, wenn es zu mindestens einem der Sätze verknüpfte Untersätze gibt, für die ein SR-Hilfsschlüssel gebildet wurde. Nun führt man normal die Verknüpfung mit dem zweiten Begriff aus (also raeuber bzw. natur).

Hauptschalter
Für PRESTO und APAC gibt es auch einen "Hauptschalter": wenn mit der neuen Option **-+** gestartet wird, ist die Plus-Suche fest eingeschaltet. Man überlege, ob dann nicht in vielen Fällen zuviel kommt.

Oberflächenfragen
Ein paar Worte noch zur Präsentation an der Benutzeroberfläche. In der Praxis würde man natürlich nicht vom Endbenutzer verlangen, ein '&' vor jeden zu erweiternden Begriff zu setzen. Das wäre archaisch! Man würde in einem Web-Suchformular zu jedem Suchbegriff einen Schalter setzen, der die Erweiterung für diesen Begriff einschalten würde. Wie für die Restriktionen gilt auch hier: innerhalb der Programme PRESTO und APAC ist eine wirklich benutzerfreundliche, also glasklar verständliche, Implementierung nicht möglich.
In der ***avanti***-Sprache braucht man jedenfalls nur das '&' vor jeden Begriff zu setzen, der satzübergreifend gesucht werden soll (siehe oben unter "Abstrakte Lösungsidee"). Anders als bei PRESTO/APAC kann dies jeder der verknüpften Suchbegriffe sein.
Beispiele für Befehle in ***avanti*** und in ***a99/alcarta***: (s.o. die Beispiele 5 und 8)

```
find per &vollmer? and tit &wissen
find per &vollmer? and &lorenz?
```

Die korrekte Behandlung der Indexparameter für die Hilfsschlüssel ist in die Programme PRESTO, MENUED, ALFA, INDEX und UPD (schon seit V15.0) eingebaut, desgleichen in den ***avanti***-Server und in die Windows-Programme.
Hier sieht man, wie die Funktion in die Programme **a99** und **alcarta** eingebaut ist: Das Kästchen „Mit Expansion“ ist bei „schiller“ angekreuzt, und das bedeutet genau, daß für Schiller die Plus-Suche durchzuführen ist.



Wenn eine Datenbank keine SR-Schlüssel hat, kann der Schalter "Mit Expansion" nicht betätigt werden.

10.2.7 Flips : Querverbindungen, Grafikanbindung etc.

Die nachfolgende Beschreibung ist nur verständlich, wenn man sich in der Export-Parametrierung schon gut auskennt. Für den "Endbenutzer" ist nicht viel zu sagen: Was ein "Flip" ist und was er bewirkt, das wird aus einer Anzeige sofort erkennbar und plausibel. Hier geht es darum, wie man Flips einrichtet. Zunächst für DOS, dann für Windows.
Version 14b erhielt zusätzliche Funktionen, die unter dem Namen "Flip" zusammengefaßt wurden. **Flips** sind Umschaltungen auf andere Sätze oder Registerstellen, die durch Knopfdruck ausgelöst werden können. Zusätze in den Anzeigeparametern (Datei D-1.cPR) ermöglichen es, dem OPAC-Benutzer solche Schaltknöpfe anzubieten, und zwar abhängig vom Inhalt des angezeigten Satzes, also sehr individuell. Es gibt zwei Methoden zum Flippen.

Besonders bei der Gestaltung von OPACs, aber auch sonst, wird man mit Aufgaben konfrontiert, die noch mehr Querverbindungen zwischen Datensätzen verlangen, als es die bisherigen Methoden schon ermöglichen. Seit V11.2 schon kann man bei der Anzeige eines Datensatzes deutliche Hinweise auf andere Datensätze erscheinen lassen oder diese nachladen und gleich mit anzeigen. Das letztere führt manchmal zu weit (komplizierte, überfrachtete Anzeige), das erstere ist zu wenig direkt und verlangt vom Benutzer, ein anderes Register aufzuschlagen und dort einen bestimmten Begriff mit der Hand einzugeben. Seit ***V14b*** kann man folgendes tun: man läßt an beliebigen Stellen innerhalb der Anzeige sog. ***Flips*** erscheinen, z.B. so: (die eckigen Klammern haben als solche keine Funktion!)

`[1]` *Hinweis1* `[2]` *Hinweis2* `[c]` *Hinweis3* ...

Den Ziffern kann man eine auffällige Farbe geben, damit sie optisch hervorstechen (nur unter DOS allerdings).
Eine typische Zeile in den Anzeigeparametern, um so etwas zu erzeugen, sieht z.B. so aus:

```
#31 p{ C "[" t71 "1" t76 "] Werke zum Thema: " }        FLIP 1 anzeigen
```

Dieses muß innerhalb der Anzeigeparameter an einer geeigneten Stelle stehen, am besten unterhalb oder oberhalb des gesamten Datensatzes. Damit überhaupt etwas passiert, muß es eine Anwendervariable namens `#uZ1` geben, in der steht, was bei Druck auf '1' tatsächlich passieren soll. Folgende Zeile würde diese Anwendervariable erzeugen:

`#31 p"?|3" dZ1 aZ1` FLIP 1 generieren, d.h. #uZ1 mit "?|3<Inhalt von #31>" füllen

Diese Zeile steht am besten über oder unter der oben gezeigten Zeile. Wird dann die Ziffer 1 gedrückt, *flippt* das Programm zum ersten betreffenden Datensatz, d.h. holt ihn sofort auf den Schirm, schaltet aber auch das betreffende Register an die bewußte Stelle, so daß man (nach Druck auf <Enter>) dort auch gleich blättern kann. Der neue Datensatz kann seinerseits weitere *Flips* präsentieren, die zu wieder anderen Sätzen hinführen. Mit <Bild↑> kann gleich wieder zum jeweiligen Ausgangssatz zurückgeschaltet werden, natürlich mehrfach hintereinander. Man ist versucht, das Schlagwort "Hypertext" zu bemühen, wobei aber an existierenden Daten noch nicht einmal irgend etwas ergänzt oder verändert werden muß. Für geübte Parametrierer wird die Methode sehr leicht anzuwenden sein: man muß nur die hinter den *Flips* steckenden Registereinträge innerhalb der Anzeigeparameter präparieren und in bestimmte Anwendervariablen speichern, und zwar `#uZ1 ... #uZ9` für die Ziffern 1 bis 9 (ein großes 'Z'!).
Wenn z.B. `#uZ4` den Inhalt `?|1Shakespeare, William` hat, wird beim Druck auf Ziffer '4' zum Register 1 auf `shakespeare, william` geschaltet. Ohne das '?' kommt der erste Datensatz zur Anzeige, der zu dem Registereintrag gehört. Wenn außerdem am Anfang von #uZ4 noch das Präfix "<0" steht, wird auf die andere Datenbank umgeschaltet, falls eine aktiviert wurde. Man bemerkt: hinter Flip [4] steckt also nicht notwendig das Register 4! (Statt Ziffern sind auch alle Buchstaben möglich, hinter denen kein Befehl steckt.) Die präparierten Registereinträge müssen auch nicht exakt der Registerform entsprechen, denn es wird die Umcodierung angewendet, die innerhalb der Indexparameter programmiert ist (Sprungmarken #-1 bis #-9). Klipp und klar: *wenn* in `#uZ4` etwas steht, wird bei Druck auf '4' geflippt, egal was auf dem Schirm steht! Umgekehrt: wenn ein Hinweis [4] in der Anzeige steht, aber `#uZ4` nicht besetzt ist, passiert bei Druck auf '4' nichts. Also: aufpassen beim Parametrieren.
Zum Beispiel könnte man *Flips* für die Schlagwörter oder Systemstellen machen, die bei einem Titel angegeben sind. Der Benutzer kann dann zu der jeweiligen Registerstelle *flippen*, wo diese Schlagwörter etc. stehen, ohne mit der Hand umschalten und eine neue Eingabe machen zu müssen. (Beispieldatei auf Diskette: `D-FLIP.APR`.)
Dieses neue "feature" ist auch in ***allegro-X*** enthalten. Die Grafikversionen von PRESTO und APAC (die Programme PRESTOG.EXE und APACG.EXE) ermöglichen es ferner, hinter einem Flip einen Programmaufruf zu verbergen. Etwas Ähnliches ist in die Windows-Programme eingebaut: mit der FLEX-Technik können beliebige externe Programme gestartet werden.

Die erste Methode, Flips zu programmieren, setzt voraus, daß man u.U. viele Anwendervariablen zu überwachen hat. Wenn die Zahl der Flips von Datensatz zu Datensatz stark schwankt, oder wenn man sich z.B. aufgrund der Datenstruktur einen Automatismus überlegen kann, der alle benötigten Flips in einer Schleife programmieren kann, geht man am besten einen anderen Weg. Dieser eignet sich auch, um Grafik, Sound und andere elektronische Objekte mit einer Datenbank zu verbinden. Ganz allgemein: man kann externe Programme starten.

Zweite Methode
Man benötigt hierzu nur **eine** Anwendervariable, in der die Flips nach folgendem Muster aneinandergereiht sind:

```
#ufv <tr>1FLIP1<tr>2FLIP2<tr>3FL...
```

Die Flipvariable braucht nicht `#ufv` zu heißen! Jeder Flipeintrag beginnt mit einem Trennzeichen und dahinter einem Kennbuchstaben oder -ziffer. Der Kennbuchstabe ist auch beliebig. Als Trennzeichen schlagen wir das Teilfeldtrennzeichen '▼' vor, die Parametrierung, welche bei dieser Methode notwendig ist, vereinfacht sich dadurch.

Man benötigt einen Abschnitt **`#-``** in der **Anzeige**parameterdatei (` = ASCII 96) ; dort muß die Auswertung der Tasten erfolgen. Diese Sprungmarke ist fest für diesen Flipmechanismus reserviert. Man braucht hier keine ak-Zeile, denn der Abschnitt wird nur programmintern angesprungen (ähnlich wie **`#--`**). Zu dem Zeitpunkt stehen automatisch in der Sonderkategorie #u1 die Taste und die Nummer der gerade aktiven Datenbank, z.B.:
`#u1 30`
D.h. Taste [3] wurde gedrückt und aktiv ist Datenbank Nr. 0. Die Datenbanken sind von 0 bis 2 durchnumeriert, wobei die erste Datenbank die Nummer 0 hat.
In diesem Abschnitt wertet man die Taste (und bei Bedarf auch die Datenbanknummer) aus. Man extrahiert aus obiger Anwendervariable den Flip-Befehl, der zu dieser Taste gehört. Dieser Befehl ist das Ergebnis dieses Abschnittes. Das Programm wertet dann das Ergebnis (d.h. den Arbeitstext) aus und reagiert entsprechend.

Ein Beispiel:
In dem Teil für die Anzeige eines Titels findet man diese Befehle (an der Stelle, wo die Flips im Anzeigetext erzeugt werden sollen):

```
...
```
Erzeugen der Flip-Anwendervariablen (hier ist es #utz):
```
#31s p"▼c?|3" Atz        Umschalten auf Reg. 3 Sachschlagwort
#31p p"▼p?|3" Atz        Umschalten auf Reg. 3 Personenschlagwort
#8e p"▼t~view " Atz      Anzeige einer zugeordneten Textdatei (Name in #8e). Hier wird ein Befehl
```
zusammengesetzt: `In #8e` steht der Name eines Textes, der angezeigt werden soll, `"view"` ist der Name des Anzeigeprogramms
```
...
```
Erzeugen des Hinweises in der Anzeige:
```
#31s { "   [" t71 "1" t76 "] mehr zu: " }
#31p { "   [" t71 "2" t76 "] mehr zu: " }
#8e { "   [" t71 "t" t76 "] Text zu: " }
...
```
Auswertung der Tasten:
Das Programm springt automatisch zur Sprungmarke `#-``, wenn auf dem Anzeigebildschirm eine nicht definierte Buchstaben- oder Zifferntaste gedrückt wird. Wenn z.B. Taste [2] gedrückt wurde, entsteht daraus `#u1 20` (s.o.). Programmieren Sie hier einen Sprungverteiler. (Dieser Abschnitt muß an einer Stelle stehen, die beim normalen Durchlauf nicht erreicht wird.) Das Schema sieht so aus:

```
#-`           Welche Taste wurde gedrückt 1,2 oder 3?
#u1 +c i4,1 e0 #zz 0   Taste steht auf erster Stelle im Kategorietext, deshalb i4!
#u1 +p i4,2 e0 #zz 0   bei mehr als zweistelligem Schema muß der i-Befehl
#u1 +t i4,3 e0 #zz 0   angepaßt werden: z.B. i5,1 im dreistelligen Schema
#u1 e0 #zz 0       Alle anderen Tasten dürfen hier keine Reaktion auslösen
#+#
```
Extraktion des entsprechenden Flipbefehls aus `#utz`: (hier zeigt sich der Vorteil von '▼' als Trennzeichen!)
Es entsteht als Ergebnis praktisch der Befehl für das Kernprogramm, wohin es flippen soll:
```
#-c
#utz ▼c          Ergebnis: Arbeitstext ist jetzt der Flipbefehl von #31s
#+#           Rückkehr zum Kernprogramm, dort wird der Sprung ausgeführt
#-p
#utz ▼p
#+#
#-t
#utz ▼t          Teilfeld ▼t enthält den Programmaufruf zum Anzeigen einer Textdatei
#+#
```

Wenn das erste Zeichen des Flips die Tilde '~' ist, wird der Flipbefehl als DOS-Befehl aufgefaßt und dem Betriebssystem zur Ausführung übergeben. Beispiel siehe oben `#8e p"▼t~view " Atz`. Wenn in #8e also ein *dateiname* steht, wird hieraus der DOS-Befehl `view` *dateiname*. Dieser Mechanismus funktioniert bis V14c nur in den grafikfähigen Versionen der Programme, PRESTOG.EXE, APACG.EXE etc. Wenn man aus dem aufgerufenen Programm zurückkehrt, erscheint wieder der Anzeigebildschirm, wo man die bewußte Taste gedrückt hatte.
"Flipfähige" Tasten (Zeichen ohne sonstige Funktion im Anzeigebildschirm) sind die folgenden:

0 – 9 a c e f g i k l m o q t w y z A G K M N O P Q S T U V W X Y Z

Flips In a99 und alcarta

Mit ***alcarta/a99*** kann man alle Flips nach einem einheitlichen Schema anlegen. Der Abschnitt `#-` ` wird dadurch überflüssig. Beispiele sieht man in der Datei D-WRTF.APR für die Anzeige in ***a99*** und ***alcarta***.
Jeder Flip benötigt 3 Dinge:

1. Was man sieht (erscheint z.B. wie ein Netscape-Link, blau und unterstrichen)
2. Die #uY-Variable verbindet 1. mit 3.:
3. Die #uZ-Variable: enthält den eigentlichen Flip-Befehl oder die FLEX-Befehlskette (siehe FLEXGER.RTF)

Diese Variablen müssen nicht ihrem Zweck fest zugeordnet werden wie beim DOS-Verfahren.

Beispiel: Inhalt von #31s (Sachschlagwort) soll in Flips aufgelöst werden. Es können beliebig viele Schlagwörter darin stecken, getrennt durch "; ". Der Flip soll jeweils auf die Registerstelle springen, wo das Schlagwort steht.
Das geht nach diesem Rezept:

Empfohlene Standard-Zwischenteile:

```
69=160 t74 t97      Code 160, blau, Unterstreichung ein
68=t96 t76 160      Unterstreichung aus, schwarz, Code 160
....
#nr dY~ dZ~         alle Flips beseitigen (irgendwo am Anfang der Parameter)
.......
#31s =sw e0         Kopiere #31s nach #usw

#-S   Beginn der Schleife
#usw e";" F" " p"?|3" =Z~      nächste #uZi präparieren
#usw e";" F" " y1 =Y~          zugehörige #uYi (y1 wegen Sonderzeichen!)
#usw C e";" F" " p69 P68       Text ausgeben mit Begrenzern und Attributen
#usw +S b";" f" " =sw          vorderen Teil abschneiden, zurück nach #-S
```

Es entstehen dann #uZ0, #uZ1, ... und #uY0, #uY1, ... Diese gehören paarweise zusammen.
Code 160 ist ein unsichtbares ANSI-Zeichen und dient als Begrenzungszeichen. Statt dessen kann man auch die eckigen Klammern nehmen oder in der INI-Datei per Befehl `FlipMark=...` ein anderes Zeichen dafür einstellen.
Die Zwischenteile haben standardmäßig die Bedeutung (in d-rtf.apt):

74 = blau, 76 = normal (schwarz)
97 = Unterstreichen ein, 96 = Unterstreichen aus

Die Schlagwörter erscheinen also untereinander, blau unterstrichen, und die Begrenzungszeichen ermöglichen, daß das Programm bei Doppelklick oder Markierung die Zeichenfolge exakt vom Bildschirm entnehmen und mit den #uY-Variablen vergleichen kann. Die zugehörige #uZ-Variable wird dann als Befehl ausgeführt.
Folgende Befehle können in einer Z-Variable stecken: (Genau auf die Leerzeichen achten!)

```
f per name...       avanti-Findbefehl
f #nnn              lade Satz nnn (interne Nummer)

j xabc              Springe zu Label #-x in den Anzeigeparametern, in #u1 steht dann abc
h name              Lade Hilfedatei nameGER.RTF
?|iabc              Springe in Register i an die Stelle abc
~prog               Starte das externe Programm prog
```

(es kann für prog ein vollständiger Programmaufruf stehen, auch der Name einer Batchdatei, z.B. cp.bat)

```
x flextext          Hinter x kann eine ganze Befehlskette folgen. Dieses Konzept nennt sich FLEX und wird
X Dateiname         genauer in FLEXGER.RTF dokumentiert. (Befehl "h flex" geben im Schreibfeld von a99)
```

Bei 'X' wird die Befehlskette aus einer Datei des Typs .FLX entnommen.

Wenn ein Flip nicht funktioniert: mit Alt+r in den Reservespeicher schauen, dort sieht man, was in #uYi und #uZi wirklich steht. Insbesondere muß der Inhalt von #uYi exakt mit dem Fliptext in der Anzeige übereinstimmen.

Start von Flips ohne Hyperlink: Die Flips unter #uX0 bis #uX9 können mit der Alt-Taste ausgelöst werden: man drückt Alt+5, um #uX5 auszuführen. (Wenn #uXi nicht existiert, wird #uZi genommen.) Das ist natürlich eher etwas für Eingeweihte, aber diese können sich dann durch Direkteingabe eines Flip-Befehls (besonders mit den FLEX-Erweiterungen) das Leben u.U. sehr erleichtern. Die Flip-Buttons 1-8 unter dem Auswahlfeld lösen ebenfalls diese Funktionen aus; man kann sie per FLEX einstellen, vor allem auch ihre Beschriftung zur Laufzeit verändern (`h xflip`).

10.2.8 Programmierte Validierungen (Eingabeprüfung)

Es gibt eine seit langer Zeit festgelegte Liste von Prüfmethoden für die Dateneingabe. In der Konfigurationsdatei (z.B. $A.CFG) kann man festlegen, ob und welche Prüfung(en) auf eine eingegebene Kategorie angewendet werden sollen (Anhang A.1.2). In der $A.CFG ist z.B. festgelegt, daß bei der Eingabe eines Titels (#20, #22, #23 oder #24) die Prüfung **c** (d.h. "ist das erste Wort ein Artikel?") durchzuführen ist, bei Eingabe von #87 dagegen soll die Prüfziffer der ISBN kontrolliert werden - das wäre Prüfung **g**. Die gewünschte Prüfung wird jeweils sofort, also gleich nach dem Druck auf die <Enter>-Taste, durchgeführt (und nicht erst beim Abspeichern des Satzes).
Mit Version 13 kam die Verbesserung, daß Prüfungen auch auf Teilfelder bezogen werden können. Beispielsweise steht in der Konfiguration $U.CFG für das USMARC-Format die Zeile

```
#245"Title"$M$Aabchnp$Rnp$I01$J0123456789$Cac
```

und hier besagt die Angabe `$Cac`, daß in der Kategorie `#245` das Teilfeld ▼a (das ist der Hauptsachtitel) der Prüfung **c**, also der Artikelprüfung, zu unterziehen ist.

Mit der Ausbreitung von ***allegro***-Anwendungen auf immer mehr Bereiche und immer öfter auch nicht-bibliothekarische Anwendungen entstanden neue und diffizilere Wünsche für Spezialprüfungen, auch "Validierungen" genannt. Die meisten fallen wohl in eine der vier folgenden Gruppen:

1. **Logische Kopplung:** Wenn Kategorie #abc besetzt ist, muß auch #xyz besetzt werden. Oder umgekehrt: wenn #abc besetzt ist, darf #xyz nicht besetzt sein. Oder: ein bestimmter Bearbeiter darf #xyz nicht eingeben.
2. **Numerische Validierung:** Der Inhalt von #abc muß eine Zahl sein und in einem vorgegebenen Bereich liegen. Oder: die Zahl in #abc muß in einem formal bestimmbaren Verhältnis zum Wert in #xyz stehen.
3. **Index-Validierung:** Der Inhalt von #xyz muß im Index i vorkommen. Oder: der Inhalt von #xyz darf im Index i noch nicht vorkommen (Sicherung der Eindeutigkeit).
4. **Formale Validierung**, evtl. mit **automatischer Korrektur:** Die Kategorie #xyz muß ein bestimmtes Zeichen (oder eine Kombination) enthalten, muß mit einem bestimmten Zeichen beginnen oder enden, oder ein bestimmtes Zeichen (oder eine Kombination) darf nicht (am Anfang oder Ende oder überhaupt) vorkommen - wenn es vorkommt, ist es durch ein anderes zu ersetzen oder zu beseitigen.

Die Untersuchung konkreter Beispiele solcher Wünsche ergab, daß nur die Exportsprache das geeignete Mittel zur Realisierung sein konnte, wenn das gesamte Spektrum abgedeckt werden sollte. Der Anwender hätte also seine Prüfroutinen in der Exportsprache zu schreiben. Aber wo? In der Konfigurationsdatei? Das wäre aus programminternen Gründen nicht möglich gewesen. Wegen der sehr wichtigen Anforderungen vom Typ 3 kam nur die Index-Parameterdatei als Ort für die Prüfroutinen in Betracht. Wenn nämlich eine Eingabe am Index abzuprüfen ist, muß dazu oftmals erst eine Umcodierung (groß->klein, Umlaufauflösung etc.) erfolgen, und die dazu nötigen Angaben stehen in der Index-Parameterdatei. Diese ist nun allerdings schon reichlich mit Aufgaben befrachtet. Insbesondere war es eine Befürchtung, daß man endgültig nicht mehr mit den sowieso knappen Sprungmarken auskommen würde, wenn nun noch neue Teile hinzukämen. Dieses Detailproblem konnte jedoch, wie gleich gezeigt wird, entschärft werden.

Wirksam werden die Validierungen in den Programmen PRESTO, *aLFA* und MENUED und auch ***a99***, wenn man Daten eingibt oder korrigiert.
In ***a99*** ist es einfacher: Vor dem Speichern wird der FLEX **`onput.flx`** abgearbeitet; in diesen kann man einbauen, was immer man will.

So programmiert man eigene Validierungen

Grundsätzlich muß dazu in der Indexparameterdatei ein **Hilfsabschnitt** angelegt werden . Dieser Abschnitt muß ganz am Ende stehen, genauer: hinter der letzten Zeile der Kategorieliste. (D.h., dieser Abschnitt ist so etwas ähnliches wie der Kopf- oder Fußabschnitt beim Listendruck, → 10.2.6.5.3, die bei den Indexparametern keinen Sinn machen.) Der Hilfsabschnitt beginnt mit einem **`H`** auf einer eigenen Zeile. Darunter kommt dann eine Kategorieliste, die alles enthalten kann, was die Exportsprache ermöglicht - nur sinnvoll muß es sein, und was das heißt, wird gleich klarer.
Sehr wichtig: Innerhalb des Hilfsabschnitts sind die Sprungmarken unabhängig von denen des Hauptteils, d.h. Sprungbefehle im Hauptteil finden nur innerhalb des Hauptteils, und Sprünge im Hilfsabschnitt finden nur innerhalb des Hilfsabschnitts statt. Das war bis V13 mit den Kopf- und Fußabschnitten nicht der Fall und konnte zu Fehlern führen. Diese ärgerlichen Probleme sind für V14 zugleich auch behoben.
Der Hilfsabschnitt hat folgende Funktion:

- Wenn eine Kategorie eingegeben wird (per Abfrageliste oder in der freien Eingabe), wird sofort nach Druck auf <Enter> der Hilfsabschnitt ausgeführt.
- Die Variable **`#u1`** enthält dann immer die gerade eingegebene Kategorie.
- Wenn ein Ausgabetext produziert wird, dann wird dieser anschließend als Meldung auf der untersten Zeile ausgegeben, und zwar in der Farbe der Fehlermeldungen (→ Anhang A.3). Der Eingabetext wird in jedem Fall akzeptiert, die Meldung soll den Benutzer nur auf Probleme aufmerksam machen. Er wird im eigenen Interesse im Bedarfsfall auf die Meldung reagieren. Allerdings kann mit dem Manipulationsbefehl M die Kategorie sofort wieder gelöscht werden.

· Der Hilfsabschnitt wird aber noch bei anderen Gelegenheiten aufgerufen. Dann steht ein Wert in der Variablen #u2, damit man erkennen kann, welche Situation vorliegt:

`#u2 e` Abfrageliste wurde beendet (letzte Frage beantwortet)

`#u2 s/N` Vor dem Speichern (es wurde F10 gedrückt) / nach Zeitstempel und Nummernvergabe

`#u2 E/C/I` Es wurde soeben 'E' bzw. 'C' bzw. 'I' gedrückt

`#u2 0/1/2/3` Die eingegebene Kategorie steht in #u1 (z.B. `#u1 40 Name, Vorname`) und die Eingabe stammt aus der Abfrageliste (0) bzw. der Direkteingabe (1) bzw. aus der #b-Eingabe (2). Wert 3 tritt *nach* dem Einordnen einer Kategorie auf.

Im Hilfsabschnitt muß in der Regel zuerst geprüft werden, was für eine Kategorie denn nun gerade eingegeben wurde, denn die Prüfroutine beginnt, wie gesagt, jedesmal am Anfang des Hilfsabschnitts.

Folgende Einzelheiten sind noch zu beachten:

Fälle 0,1,2

Die Kategorie ist zu diesem Zeitpunkt noch nicht im Aufnahmespeicher. Wenn sie dort mit einer Veränderung landen soll, muß man im Hilfsabschnitt einen output produzieren; dieser wird dann als Kategorie übernommen, anstelle der Eingabe. Der output muß die Kategorienummer mit enthalten! Also kann man auch eine andere Kategorienummer einsetzen, wenn das so gewünscht wird.

Achtung: mit extrem langen Kategorien kann man das nicht machen; die Grenze für den output ist 300 Zeichen.

Wenn man keinen output produziert, oder kein Hilfsabschnitt vorhanden ist, wird die Kategorie unverändert übernommen (auch wenn sie sehr lang ist). Statt einen output zu produzieren, kann man eine Kategorienummer vor den Text von #u1 setzen und diese Kategorie dann mit M einordnen lassen. Das geht bei beliebiger Länge.

Fall e

ist geeignet, um fehlende Kategorien gleich am Ende der Abfrage zu entdecken. Z.B. kann man eine fehlende Kategorie dann mit einer Standardangabe belegen. Oder mit "????", das wird dann gleich anschließend sichtbar, da ja der Satz nach Schluß der Abfrageliste in Kategorieform angezeigt wird. Wenn die Abfrage mit 'x' verlassen wird, greift dieser Fall nicht! Wichtiger ist sicher auch deshalb der Fall 's'.

Fälle e, s und E

Wenn man in diesen Fällen einen output produziert, wird er als Fehlermeldung angezeigt. Im Falle s wird das Speichern nicht ausgeführt, denn man wird einen Fehler noch korrigieren werden sollen, bevor man speichert. Hier können also fehlende Kategorien oder Teilfelder angemahnt werden. Natürlich kann man für die beiden Fälle e und s denselben Abschnitt abarbeiten lassen.

Fälle E und C

sind geeignet, um vor dem Bearbeiten schon gewisse Veränderungen an einzelnen Kategorien vorzunehmen, die man im Abschnitt s auch wieder rückgängig machen kann. Z.B. kann man ein bestimmtes Teilfeld aus einer Kategorie in eine andere (Hilfs-)katagorie schreiben lassen. Diese läßt sich dann leichter bearbeiten. Hinterher (Fall s) hängt man diese Kategorie eieder als Teilfeld an die Ausgangskategorie und beseitigt die Hilfskategorie.

Oder: man "hinterlegt" eine Kategorie, die nicht verändert werden soll, als Kopie in einer Anwendervariablen und sortiert sie hinterher vor dem Speichern (Fall s) wieder ein

Mit konkreten Beispielen zu den vier Typen von Prüfungen wird nun gezeigt, wie man Validierungen programmieren kann. Das recht komplizierte Beispiel zum Typ 2 stammt aus der Musik-Parametrierung (s.S. 5).

Beisp. zu 1: Die Kategorien #40 und #60 dürfen nicht gleichzeitig besetzt sein.

Beisp. zu 2: Kategorie #77z soll die Gesamt-Spielzeit eines Musikstücks enthalten. Die einzelnen Teile (Sätze) des Stücks sind in der Kategorie #81m erfaßt, und zwar in der Form

`#81m1: Satzbezeichn.1 MM:SS¶2: Satzbez.2 MM:SS¶...¶Satzbez.N MM:SS.`

Daher soll das Programm den Inhalt von #77z aus der Summe der Spielzeiten der einzelnen Teile in #81m errechnen und dann die #77z automatisch erzeugen (!)

Beisp. zu 3: Wenn die Kategorie (egal welche) ein Stammsatz-Kürzel (oder eine Identnummer) enthält statt eines Klartextes, soll das Programm prüfen, ob dies ein gültiges Kürzel ist, d.h. ob es im Index 9 zu finden ist. Wenn umgekehrt ein neuer Stammsatz eingegeben wird, soll das Programm prüfen, ob das neue Kürzel in #00 noch nicht im Register 9 vorkommt. Dort sind die Kürzel mit nachfolgenden '=' eingetragen.

In diesem Fall soll die Annahme verweigert, d.h. die Kategorie #00 nicht abgespeichert werden.

Außerdem soll das Programm noch prüfen, ob die ISBN #87 im Register 9 schon vorkommt. Dort stehen die ISBNs ohne Prüfziffer und mit einem 'i' vorweg! Und der Titel in #20 soll im Register 4 gesucht werden: Wenn er vorkommt, Kontrollmeldung ausgeben.

Beisp. zu 4: Kategorie #95c soll eine Zahl enthalten, und zwar mit Dezimalpunkt statt Komma, keine weiteren alphanumerischen Zeichen. Wenn es eine ganze Zahl ist, soll kein Punkt dahinter stehen. Die Zahl soll ferner positiv sein.

Es folgt der Hilfsabschnitt, der alle diese Prüfungen durchführt:

```
H    Hier fängt der Hilfsabschnitt an.  Zuerst eine Verteilerliste für die zu prüfenden Kategorien.
     Die Angabe e0 ist nötig, damit diese Zeilen keine Ausgabe produzieren:
     (Jede produzierte Ausgabe erscheint unten auf der letzten Zeile!)
#u2 +- c"s" e0         F10 gedrückt? Dann hier nichts machen. Sonst Eingabe prüfen:
#u1 +4 i1,4 i2,0 i3,   e0  ist es #40 ? Dann #-4
#u1 +6 i1,6 i2,0 i3,   e0  ist es #60 ? Dann #-6
#u1 +8 i1,8 i2,1 i3,m e0  ist es #81m? Dann #-8
#u1 +0 i4,_ e0       Erstes Textzeichen ist '_'? Dann #-0
#u1 +A i1,8 i2,7 e0     ist es #87? Dann #-A
#u1 +C i1,9 i2,0 e0     ist es #90? Dann #-C
#u1 +D i1,9 i2,5 i3,c e0  ist es #95c? Dann #-D
#u1 +K i1,0 i2,0 e0     ist es #00? Dann #-K
#+#            andere Kategorie? Dann nichts machen.
#-4
#60 +Z e0       #40 wurde gerade eingegeben. Ist #60 besetzt? Dann #-Z
#+#
#-6
#40 +Z e0       #60 wurde gerade eingegeben. Ist #40 besetzt? Dann #-Z
#+#
#-Z
#t{ "#40 und #60 sind zugleich besetzt, das darf nicht sein!" }
#+#
#-8
#u1 dsb dzt asb Hilfsvariable #uzt löschen, #usb mit Inhalt von #u1 füllen
#-Y                   d.h. der Inhalt von #u1 steht jetzt in #usb
#usb e"¶" T" " c":" e":" x"*60" x"+zt" =zt Minuten -> Sekunden, auf #uzt addieren
#usb e"¶" T":" x"+zt" =zt           Sekunden dazurechnen
#usb +Y b"¶" =sb       Schleife: nächste Satzangabe fängt hinter ¶ an. Kein ¶ mehr? Dann
#uzt x"/60" e"." r2 dgt agt    Sekunden->Minuten, in #ugt ablegen
#uzt x"%60" e"." r2 p":" Agt   Rest-Sekunden mit ':' anhängen. #ugt enthält nun das Ergebnis.
#ugt p"#77z" M                 Ergebnis aus #ugt in #77z überführen
#+#
#-0
#u1 +# y0 b"_" e"[ ,;:]" P"=" |98   Eingegebenes Kürzel in Register 9 suchen
#t{ "Kürzel kommt noch nicht vor" } meldung, wenn nicht gefunden
#+#
#-A    #87
!u1 +a e11 p"i" |98          kommt ISBN schon im Register 9 vor?
#+#                          nein, dann nichts machen
#-a
#t{ "ISBN ist bereits vorhanden!" 7 }    die 7 loest den Piepston aus
#+#
#-D      Zahl isolieren, Nullen und Punkt hinten abschneiden, in #95c einsetzen
#u1 x"*1" F"0." p"#95c" M (mit x"*1" entsteht die reine Zahl! Sie hat immer einen Punkt.)
#95c +# x">0" e0
#t{ "Unzulässiger Zahlenwert!" }
#+#
#-K                          Kürzel schon vergeben?
!u1 +k P"=" |98 e0          Es müßte mit '=' dahinter in Register 9 stehen
#+#      trifft nicht zu, Ende
#-k      trifft zu:
#t{ "Dieses Kürzel existiert schon! " 7 }
#u1 y0 p"#00 " e4 M       #00 gleich wieder löschen! (M-Befehl mit leerer #00 ausführen)
```

10.2.9 Restriktionen (Einschränkungen von Ergebnismengen)

Kataloge wachsen mit den Beständen, und so wachsen zwangsläufig auch die Ergebnismengen, die man beim Zugriff erhält. Der häufigste Wunsch ist es dann vermutlich, den Zugriff auf die neueste Literatur zu begrenzen. Genau das konnte man bis V14 mit ***allegro*** kaum machen. Zwar kann man die Erscheinungsjahre indexieren (bei CAT.API geschieht dies im Register 6), aber eine Begrenzung der Ergebnisse mit Bedingungen wie "nach 1998" oder "vor 1850" ist damit nicht erreichbar. Weitere Wünsche sind Einschränkungen nach Sprache oder Publikationstyp, falls man diese erfaßt hat. Auch solche Elemente kann man indexieren, aber die Ergebnismengen unter solchen Einträgen werden zu groß, sobald man eine Datenbank mit einigen -zigtausend Datensätzen hat. Andere Systeme kennen Suchbeschränkungen schon lange. An der Endbenutzer-Oberfläche haben die Systeme in der Regel zusätzliche Schalter, Auswahllisten oder Eingabefelder für Sprache, Typ oder Erscheinungsjahr. Es gibt keine allgemein übliche Bezeichnung für solche Funktionen. Z.B. wird von "Sekundäraspekten" gesprochen, im Englischen nennt man derlei Elemente meistens "qualifier" oder "limitation", manchmal findet man auch recht unanschauliche Namen wie z. B. bei Pica "ADI" (**A**dditional **D**iscriminative **I**nformation). Um einen möglichst knappen, prägnanten und unverwechselbaren Begriff zu haben, wurde für ***allegro*** der Terminus **Restriktionen** gewählt.
Um Restriktionen endlich auch unter ***allegro*** möglich zu machen, mußte ein neues Konzept her, denn in die bisherige Indextechnik ließen sich solche Funktionen nicht effizient einbauen. Das neue Konzept sollte natürlich einerseits flexibel sein, also unter jeder Konfiguration für jeden gewünschten Zweck parametrierbar, andererseits sollte der Aufwand für den Anwender so gering wie möglich sein.
Die unter diesen Vorgaben realisierte Lösung ist eine Ergänzung, keine Änderung der Indexierungstechnik. An bereits existierenden Indexparametern braucht nichts verändert zu werden, bestehende Indexdateien funktionieren unverändert weiter. Es kommen nur ein paar neue Befehle und eine neue Datei hinzu. Diese erhält den Typ .RES (für **Res**triktionen), also z.B. CAT.RES. Im Prinzip wird sie genauso erzeugt wie die Kurztiteldatei vom Typ .STL.
Die Grundidee: man hinterlegt in der Datei .RES für jeden Datensatz ein paar Bytes für die Restriktionen, die es geben soll. Zum Beispiel: 4 Bytes für das Erscheinungsjahr + 3 Bytes für die Sprache + 1 Byte für den Publikationstyp. (Die Gesamtzahl Bytes, hier also 7, multipliziert mit der Anzahl Datensätze ergibt die Größe der Datei.) Das Programm (APAC, PRESTO, AVANTI etc.) kann hernach diese Bytes benutzen, um für jeden Satz festzustellen, ob er zur Ergebnismenge gehören soll oder nicht. Es gibt damit einen sehr erwünschten Zusatzeffekt: wenn die Restriktion bereits eingeschaltet wird, bevor eine Ergebnismenge gebildet wird, dann wird diese von vornherein kleiner. Somit kann eine Suche auch dann noch Erfolg haben, wenn es im Index z.B. unter einem Stichwort viele -zigtausend Einträge gibt. Allerdings kann die Bildung der Ergebnismenge in solchen Fällen etliche Sekunden dauern, weil ja für jeden Eintrag zusätzliche Vergleiche nötig sind. Ohne eingeschaltete Restriktion ändert sich am Zeitverhalten der Zugriffe nichts.

Was muß man tun?
Ganz genau wie für die .STL-Datei muß man bestimmte Zusätze in die **Index**parameter einbauen, um die Einträge für die .RES-Datei zu erzeugen:

`ir=k` **Länge des Restriktionsschlüssels** (entsprechend `i0 bei .STL`)
Wenn `ir` gesetzt ist, wird automatisch eine Datei *dbn*.RES angelegt, mit k Bytes je Satz.

`ak=zz+/` ak-Befehl für den "Restriktionsschlüssel" (entspr. `ak=zz+0 bei .STL`)

`#-/` Abschnitt für die Erstellung des Restriktionsschlüssels. Dieser muß das Präfix `"|/" erhalten.`
`...` (entspr. `#-0` und Präfix `"|0" bei .STL`) In diesem Abschnitt muß dann stehen, wie die Restriktionsangaben zu bilden sind.

Für die Benutzung (APAC, ***avanti, alcarta***) kommen noch eine oder mehrere Hilfszeilen in folgender Form hinzu (ebenfalls in der .API):

```
R XXX mp "Text zu dieser Restriktion?Hilfszeile dazu"
  XXX                    symbolischer Name der Restriktion, z.B. ERJ für Ersch.Jahr
      m                  Modus r, R, s oder S (d.h. die zugehörigen Werte sind in .RES bzw. .STL)
       p                 Position 1..72 (in .RES bzw. .STL), wo die Werte stehen
         "Text?Hilfe"    wird z.B. von APAC, a99 und alcarta für das Menü benutzt
```

Beispiel:

```
R ERJ r1 "Erscheinungsjahr?Einschränkung auf einen Zeitraum"
R TYP r5 "Publikationstyp?Art des Dokuments"
```

Das bedeutet: man hat im .RES-Schlüssel auf Position 1 das Erscheinungsjahr, symbolischer Name soll ERJ sein. An der Position 5 steht ein Code für den Typ des Dokuments, symbolischer Name TYP.

Die R-Befehle werden in APAC bzw. *a99/alcarta* zu einem Auswahlmenü verarbeitet (s.u.). Dem *avanti*-Server ermöglichen sie das Abarbeiten von Befehlen wie `find ... erj >1990` , denn dem Server muß ja gesagt werden, was "erj" bedeutet. Für sich genommen (und für PRESTO) bewirken die R-Zeilen aber gar nichts: die Struktur der .RES (und evtl. .STL) muß damit übereinstimmen, das ist das Entscheidende, sonst kann die Sache nicht funktionieren. Hauptsächlich kommt es also auf den Abschnitt unter **`#-/`** an. Unter PRESTO kann der Eingeweihte auch Restriktionen nutzen (Positionen in der STL) , für die es keine R-Zeilen gibt.
Die Modusbuchstaben 'R' und 'S' haben eine etwas andere Wirkung als 'r' und 's', siehe unten.

Es gibt sogar die **Alternative**, sich keine *dbn*.RES zu leisten, sondern nur Teile innerhalb der bestehenden *dbn*.STL für diesen Zweck zu nutzen. Dann sind keine zusätzlichen Maßnahmen nötig, nur die R-Zeilen, und zwar mit einem 's' oder 'S' statt 'r' oder 'R' an der Position m. Die .STL muß dann natürlich geeignete Bestandteile haben, die immer an derselben Position stehen. Jede Position von 1 bis 72 ist dafür nutzbar.
Nachteil: Zugriffe auf *dbn*.STL sind viel langsamer auf *dbn*.RES. (bis ca. Faktor 10)
Beides ist kombinierbar, d.h. es kann R-Zeilen mit 'r' oder 'R' und andere mit 's' oder 'S' geben. Der OPAC-Benutzer muß den Zusammenhang und die Positionen nicht kennen - dafür sind ja die R-Zeilen gedacht.

Zusatzfunktion für INDEX

Der Aufruf **`INDEX -fx ...`** produziert die .RES-Datei (analog zu **`-fs`** für die .STL)
Wenn der Index erneuert wird (mit **`-fi, -f7`** oder **`-fn`**), wird die .RES automatisch mit erzeugt, genau wie auch die .STL. (Leider konnte man hierfür nicht `-fr` nehmen, weil dies schon für das Entlüften belegt ist.)
Im ***CockPit*** von V15 gibt es einen eigenen Menüpunkt `"x = Restriktionsdatei erneuern"` unter "organisieren". Hat man die oben beschriebenen Vorkehrungen getroffen, braucht man nur noch diesen Punkt anzuwählen.

Nutzung in PRESTO

Im Gegensatz zu APAC oder *alcarta* hat PRESTO keine Menüunterstützung. Die Nutzung der Restriktionen unter PRESTO ist daher nicht benutzerfreundlich, jedoch möglich. Man muß die Positionen der .RES oder .STL und ihre Bedeutung kennen. (Viel einfacher machen es die Windows-Programme ***a99/alcarta***, siehe übernächste Seite.)
Auf dem Registerbildschirm von PRESTO drückt man :

F9 Dann wird die gewünschte Restriktion abgefragt. Die Frage lautet
`Pos >=< Value?`
Man muß den Wunsch dann explizit eingeben, d.h. eine Position, einen Operator und einen Wert, z.B.

`r1>1990`	`Vergleich ab Position 1 in .RES, Wert soll >1990 sein`
`s58<1950`	`Vergleich ab Pos. 58 in .STL, Wert soll < 1950 sein`
`s58=1997`	`Auf Position 58 in der .STL soll 1997 stehen-`
`s68!XYZ`	`Auf Position 68 in der .STL soll` *`nicht`* `XYZ stehen`
0	Rücksetzen (Restriktion aufheben)

Der versierte PRESTO-Anwender kann hier auch s-Positionen nutzen, die nicht in einer R-Zeile definiert sind!
Der durchgeführte Vergleich ist immer ein *Zeichen*vergleich, kein *Zahlen*vergleich, auch wenn auf der fraglichen Position Zahlen stehen. Die Anzahl der zu vergleichenden Zeichen ergibt sich aus der Eingabe, z.B. 4 wenn man **`r1>1990`** eingegeben hat, d.h. nachfolgende Zeichen in der .RES oder .STL haben keinen Einfluß.
Nachdem mit F9 eine Restriktion gesetzt wurde, wirkt sie sich auf alle nachfolgenden Funktionen (+ / -) zur Ergebnismengenbildung aus. Wenn man also '/' drückt, werden nur diejenigen Sätze in die Ergebnismenge genommen, die der Restriktion genügen. Für den Anzeigebildschirm gilt das nicht, und zwar aus gutem Grund: wenn man einen einzelnen Satz auf dem Schirm hat, entscheidet man sich wohl ganz bewußt dafür, ihn in die Ergebnismenge aufzunehmen, und das sollte das Programm dann nicht verhindern.
Zu einem Zeitpunkt kann immer nur eine Restriktion gesetzt sein.
Wenn nichts gesetzt oder die Setzung mit '0' wieder aufgehoben ist, funktioniert alles normal wie gewohnt.
Wenn man eine Ergebnismenge gebildet hat, kann nachträglich eine Restriktion durchgeführt werden. Sie muß dann, nach der Setzung mit **F9**, durch **Strg+F9** auf die bestehende Ergebnismenge angewendet werden. So kann man nacheinander die Ergebnismenge durch zwei oder mehr verschiedene Restriktionen verkleinern, d.h. jeweils mit **F9** und **Strg+F9** unterschiedliche Aktionen durchführen.
Hilfestellung: In PRESTO erhält man auf dem Anzeigeschirm mit Alt+F7 die vorhandenen Restriktionen angezeigt: es

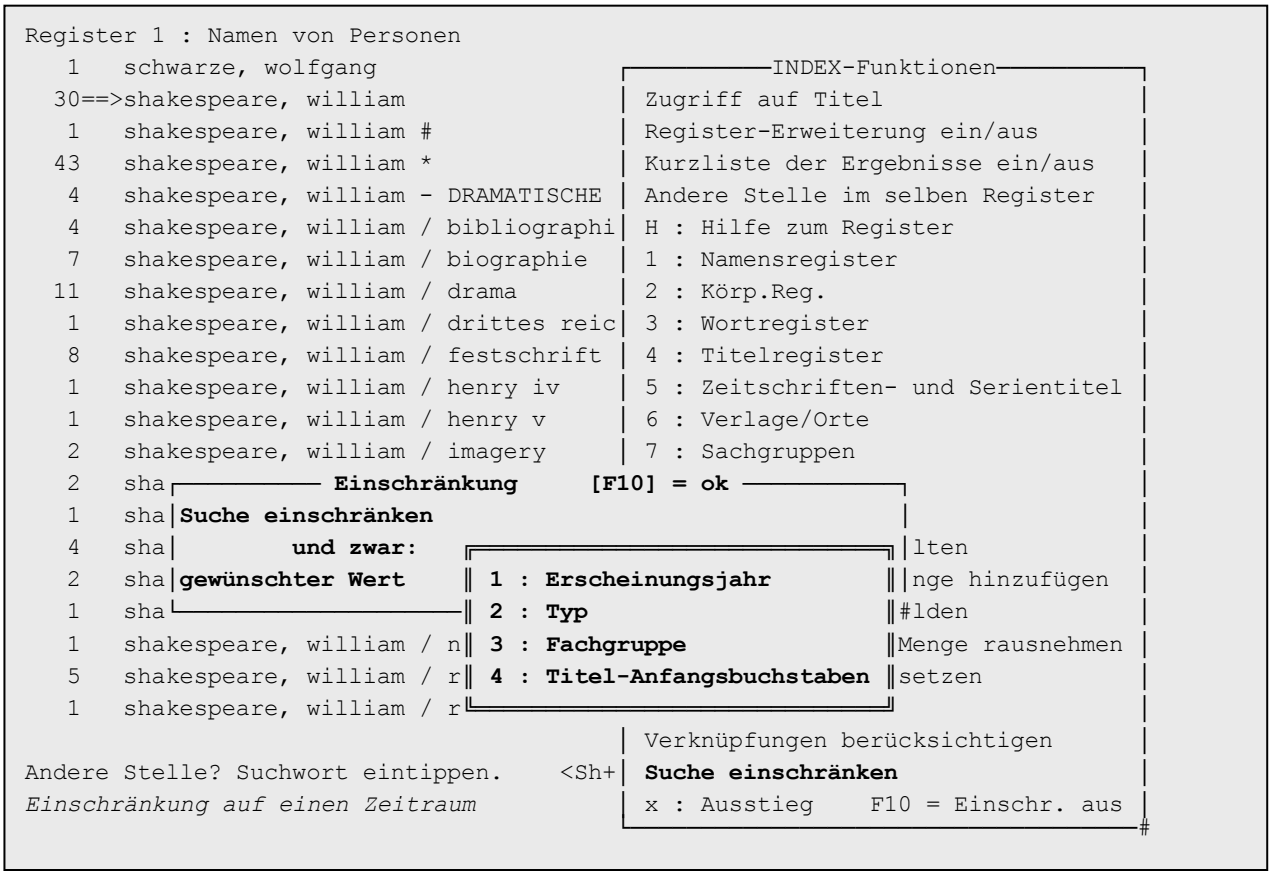
erscheinen die R-Zeilen aus den Indexparametern.

Und wie sieht es im DOS-OPAC aus?
Das Programm APAC macht es dem Nutzer leicht: es erscheint ein Menüpunkt "`Einschränkungen`" im Indexmenü (Achtung: die Dateien UIFA... wurden dafür um einige Zeilen ergänzt). Man bekommt ein Eingabeformular mit drei Zeilen: Auswahl der Restriktion, des Operators, und Eingabe des gewünschten Wertes. Die ersten zwei erlauben per Hilfemenü nur die sinnvollen Werte (die sich aus den R-Befehlen ergeben).

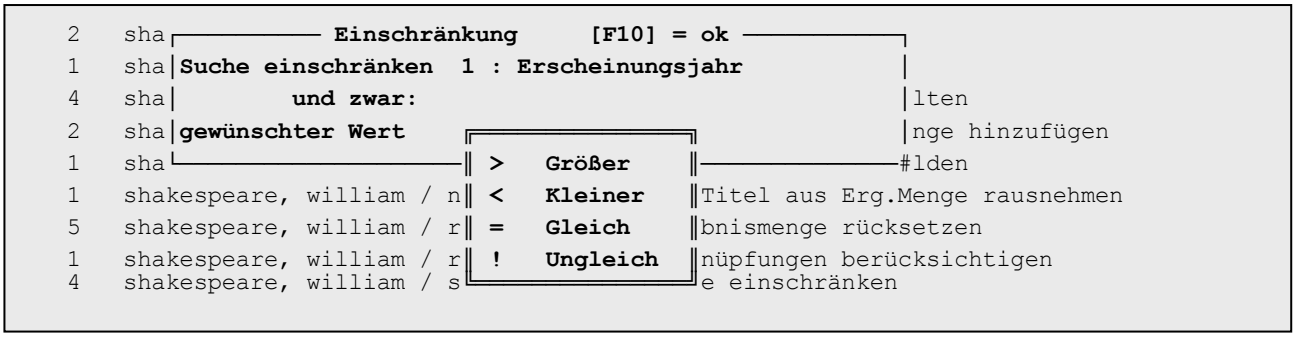
Angenommen, wir haben vier Restriktionen und dafür in der CAT.API diese Hilfezeilen:

```
R ERJ r1 "Erscheinungsjahr?Einschränkung auf einen Zeitraum"
R TYP r5 "Typ?z.B. ho=Dissertation af=Aufsatzsammlung"
R FGR r7 "Fachgruppe?z.B. ma=Mathematik, py=Physik"
R TAB s1 "Titel-Anfangsbuchstabe?z.B. G, wenn die Titel mit G anfangen sollen"
```

APAC liefert uns dann auf dem Registerbildschirm über den Menüpunkt "`Suche einschränken`" dieses Bild:



In der Menüzeile "**`und zwar:`**" hat man die Wahl zwischen den vier Vergleichsfunktionen **`< > !`** und **`=`**.
Wenn man "Erscheinungsjahr" gewählt hat, sieht das dann so aus:



Unter "**`gewünschter Wert`**" gibt man schließlich den Begriff ein, der für die Restriktion verwendet werden soll, also

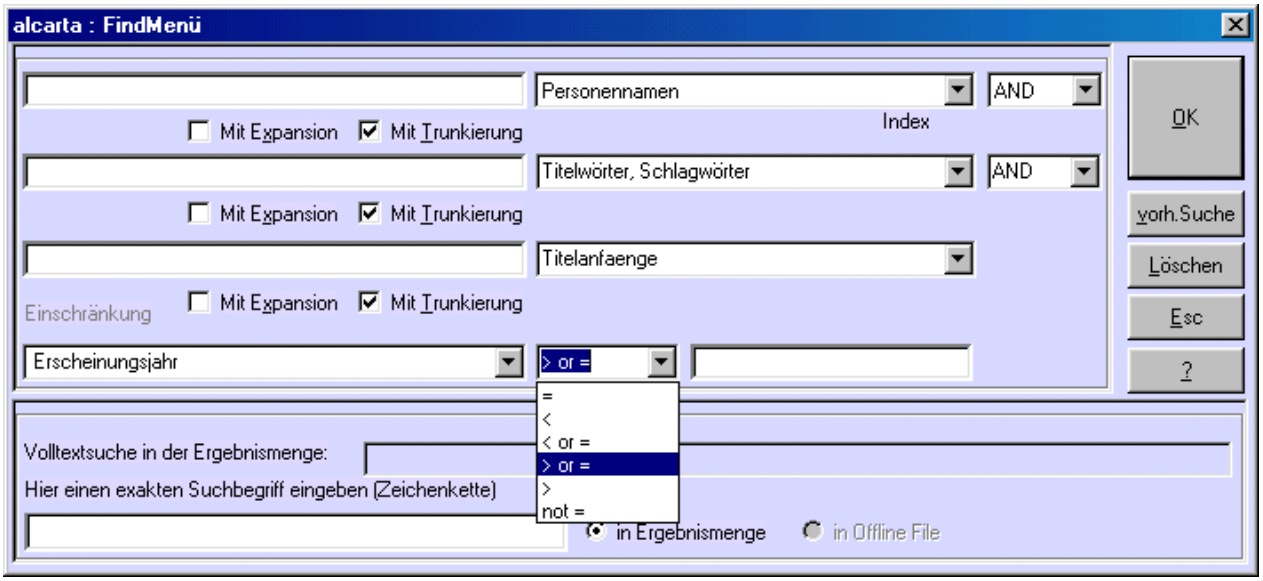
z.B. eine Jahreszahl. Mit [F10] wird die Restriktion dann dem Programm übergeben.
Wenn der Leuchtbalken auf `"gewünschter Wert"` steht, sieht man unten einen Hinweis `" >>> F1 = Hilfe <<<".` Damit aber Hilfe kommt, muß man eine Hilfeseite haben. Diese Seiten müssen **`HRES1`**, **`HRES2`** usw. heißen, d.h. für jede der Restriktionen kann man eine spezifische Hilfeseite einrichten. Am besten legt man diese Seiten in das Datenverzeichnis, denn in der Regel werden sie datenbankspezifisch sein.

Besonderheiten
Wenn zum Zeitpunkt dieser Menüauswahl schon eine Ergebnismenge besteht, wird diese gleich anschließend der gewählten Restriktion unterzogen, also verkleinert. Vermutlich ist dies genau das, was der OPAC-Benutzer erwartet. Der PRESTO-Benutzer muß **Strg+F9** drücken, um die Restriktion ausführen zu lassen. Die eingeschaltete Restriktion wirkt sich dann aber auf alle nachfolgende Operationen aus, bis man eine andere einstellt oder sie vom Indexmenü mit **F10** wieder aufhebt. In der vorletzten Zeile unten rechts wird die eingestellte Restriktion angezeigt, auch unter PRESTO.
Der Systemverwalter hat die Möglichkeit, die Restriktionen auf unterschiedliche Art in APAC zu implementieren:

Die Restriktionen mit 'R' oder 'S' (statt 'r' oder 's') haben eine etwas andere Wirkung: VOR Anwendung der Restriktion wird die Ergebnismenge expandiert (siehe "Satzübergreifende Suche" in Kap.10.2.6.9), d.h. als ob man zuerst '&' drücken würde. Außerdem wird die Restriktion nach einmaliger Anwendung wieder abgeschaltet. Beim OPAC muß man ja bedenken, daß ein Nutzer jederzeit das Gerät verlassen kann. Der nächste kommt und bemerkt gar nicht die eingeschaltete Restriktion, aber wundert sich dann natürlich über die Ergebnisse, die er bekommt oder eben nicht bekommt.

Abschalten der Restriktion
Auf dem ESC-Menü des Registerschirms gibt es den Hinweis: `F10 : Einschr. aus.`
Zusätzlich kann die eingeschaltete Restriktion in APAC und PRESTO auch mit der Rücktaste ⟵ abgeschaltet werden.

a99 und *alcarta*
Die Restriktionen findet man auf dem Find-Menü (Fernglas) wieder, unterhalb der Zeilen für die Suchbegriffe:



Tip: Wenn schon eine Ergebnismenge besteht, kann man sie auf diesem Wege verkleinern: man gibt NUR eine Angabe zur Restriktion und keine Suchworte, dann wird die bestehende Ergebnismenge entsprechend verkleinert.

10.3 Testen und Optimieren

10.3.1 Direkte Änderung von Parametern

Wenn man eine Parameterdatei neu erstellt, gelingt es höchst selten, sofort das richtige Ergebnis zu erzielen. Deshalb wurde eine Möglichkeit geschaffen, die Parameter während des Testens vom Editor aus zu ändern. Bis auf die Exportbefehle in der Kategorieliste (→ 10.2.6.1) können alle Parameter über den Befehl #p geändert werden: Soll z.B. eine andere Zeilenänge getestet werden, so gebe man einfach den Befehl #pzl=60, um die Zeilenlänge z.B. auf 60 Zeichen umzustellen. Will man Zwischenteil 26 ändern: #p26=... eingeben mit der gewünschten Steuerzeichenfolge hinter dem '=' .
Möchte man eine Druckercodierung ändern: z.B. definiert #pp Ä "Ae" die Ersatzdarstellung "Ae" für das Ä.
Allgemein: man schreibt im Erfassungsmodus hinter das #p die Parameterangabe in der oben in diesem Kapitel beschriebenen Form und gibt dies als Befehl ein. Sofort wird der betreffende Parameter geändert (allerdings nur im Arbeitsspeicher für die Dauer der Sitzung, nicht in der Parameterdatei!) und kann mit dem Befehl #d getestet werden. Wenn man auf diese Weise die optimalen Werte ermittelt hat, muß man schließlich noch die Parameterdatei selbst entsprechend verändern.
Nicht möglich ist die direkte Änderung der ak-Liste (→ 10.2.1) und der Kategorieliste (→ 10.2.6). Eingabe entsprechender Befehle würde eine Ergänzung (Verlängerung) dieser Listen bewirken, keine Änderung.

10.3.2 Mehrere Parameterdateien gleichzeitig

Es ist möglich, bis zu 3 Parameterdateien (Sätze), im Programm SRCH sogar 4, gleichzeitig zu laden und abwechselnd zu benutzen.
Automatisch wird immer D-1.cPR geladen, wenn man nicht per Option -p eine andere wählt (s. Kap.12). Im Editor gibt man zum Laden weiterer Parameter den Befehl #p, wählt dann den Unterpunkt 'e' und kreuzt in der Auswahlliste den Namen, z.B. P-NORMAL.APR, der nachzuladenden Datei an. Wenn mehr als eine Parameterdatei geladen ist, schaltet man zwischen denselben um, indem man **F2** oder den Befehl #d (ohne Kategorienummer) gibt und auf dem dann erscheinenden Menü die betreffende Ziffer wählt. Der nächste Druckbefehl, z.B. #d40 , benutzt dann die ausgewählten Parameter. Benutzen Sie den Parameter **dn** (→ 10.2.4) als Gedächtnisstütze.
Diese Möglichkeit ist günstig, wenn man z.B. mehrere Versionen einer Parameterdatei nebeneinander testen will. Man macht auch davon Gebrauch, wenn das Suchprogramm (→ Kap.8) mehrere Exporte mit unterschiedlicher Struktur gleichzeitig erstellen soll. Die Option -e ist deshalb bei diesem Programm wiederholbar (→ Kap.12).
Die Befehle zur direkten Änderung von Parametern (siehe oben) wirken sich jeweils nur auf das gerade angewählte Set (die gerade aktive Parameterdatei) aus.

10.3.3 Export-Parameterdatei visuell testen

0. Nehmen wir an, die zu testende Datei XYZ.APR befindet sich auf dem Programmverzeichnis. Es wird vom ***CockPit*** aus gestartet und die Datenbank CAT auf C:\ALLEGRO\KATALOG benutzt.
1. Beliebigen Datensatz aufblättern, auf dem ANZEIGEbildschirm **F2 F2 F2** drücken. Dann hat man die Auswahlliste der Parameterdateien vor sich, die auf dem Programmverzeichnis stehen.
2. Die Datei XYZ.APR mit '+' markieren, <Enter> drücken: jetzt wird sie geladen und sofort als Anzeige-Parameterdatei benutzt, auch wenn sie gar nicht für eine Bildschirmanzeige gedacht ist! D.h. der aktuelle Datensatz wird gleich mit Hilfe dieser Parameterdatei aufbereitet und neu angezeigt. (Wenn nicht: in CP.OPT ist im Abschnitt für PRESTO eine Eintragung für eine Druck-Parameterdatei vorhanden; diese Zeile beginnt mit q und entspricht der Aufruf-Option -q von PRESTO. Man macht die q-Zeile ungültig, dann geht es.)
3. Verschiedene Arten von Datensätzen aufsuchen und die Anzeigeform begutachten. Abweichungen von der gewünschten Form notieren.
4. Zum ***CockPit*** zurück, dann über **µd e** die getestete Datei wieder anwählen und bearbeiten. Änderungen ausführen. Zurück zur Datenbankbenutzung → 1.

Für Experten: die "Allers'sche Testschleife"
Dies ist eine noch schnellere, aber auch sehr flexible Methode für intensives Testen, benannt nach ihrem Erfinder Heinrich Allers.
Man stellt eine Batchdatei TESTAPR.BAT nach folgendem Schema zusammen:

```
:START
presto -f1 -a3 -n1 -S -dC:\ALLEGRO\KATALOG -p %1
pause
x %1.APR
goto START
```

und gibt dann den Befehl `testapr xyz`
In dieser schlichten Endlosschleife wird jeweils die Datenbank gestartet, und zwar mit XYZ.APR als Anzeige-Parameterdatei (statt der standardmäßigen D-1.APR), so daß man diese dann gar nicht erst mit **F2 ...** laden muß. Verläßt man die Datenbank, wird der Editor aufgerufen, um die Parameterdatei sofort wieder bearbeiten zu können. Der Befehl `pause` ermöglicht einen Abbruch der Endlosschleife mit Strg+c.
Wenn man in den EDITORzustand geht, kann man mit Befehl #a die Anwendervariablen kontrollieren.
Die Testschleife läßt sich universell für alle .APR-Dateien verwenden. Der Aufruf `testapr abc` ermöglicht jederzeit den Test jeder beliebigen Parameterdatei ABC.APR.

"Merseburger Testschleife"
Mit PRESTO gibt es eine noch schnellere Testmethode: in der Titelanzeige **F2 F2** geben (evtl. nochmal **F2**), die zu testende Parameterdatei mit dem Cursor anwählen, **F10** drücken: dann wird der X-Editor aufgerufen und die Parameterdatei zur Bearbeitung geladen (→ Anh.D, in 2 Minuten lernbar)! Man ändert etwas, speichert mit <Esc> q s, gelangt zurück in die Auswahlliste, drückt '+' <Enter>, und die Parameterdatei wird geladen und für die Anzeige benutzt, man sieht also das Resultat der Änderungen. Das läßt sich beliebig wiederholen.
Aber: beim PRESTO-Aufruf darf weder -q noch -e als Option gegeben werden, sonst klappt es nicht.

Soll der Test mit SRCH laufen, ersetzt man den PRESTO-Aufruf durch einen SRCH-Aufruf:

```
:START
srch -f4 -n1 -dC:\ALLEGRO\KATALOG -e %1/testlist -s0 -m0 -v0
v testlist                (Ergebnisdatei testlist anzeigen)
pause
x %1.APR
goto START
```

Durch `v testlist` wird dann die Ergebnisdatei angezeigt, bevor man wieder in die Bearbeitung der Parameterdatei kommt. Ein Trick: mit Druck auf 'x' kann man jederzeit den SRCH-Lauf abbrechen. Man kommt in den EDITOR und kann sich den zuletzt geladenen Datensatz ansehen. Mit Befehl #Da sieht man das Exportergebnis auf dem Bildschirm. Mit dem Befehl #a kann man evtl. vorhandene Anwendervariablen kontrollieren, die von den Exportparametern erzeugt werden! Mit #e bricht man den SRCH-Lauf ab und die Schleife geht weiter.

So testet man eine Index-Parameterdatei:
Das ist sehr einfach; man braucht auf keinen Fall erst neu zu indexieren, sondern:

0. Nehmen wir an, ABC.API liegt auf dem Programmverzeichnis und soll getestet werden. Die aktuelle Parameterdatei unserer Datenbank heißt CAT.API (d.h. wir haben eine Datenbank namens CAT) und liegt auf dem Verzeichnis C:\ALLEGRO\KATALOG.

1. CAT.API umbenennen z.B. in CAT.XXX, ABC.API umbenennen in CAT.API. Anschließend "denkt" dann PRESTO, dieses sei die richtige Index-Parameterdatei und benutzt sie.

2. Datenbank starten, Datensätze aufblättern, jeweils **F7** drücken. Dann sieht man die Liste der durch ABC.API erzeugten Schlüssel. Abweichungen von der gewünschten Form notieren.

3. Mit **F8** zum ***CockPit*** zurück, dann über **`µd i`** die getestete Datei wieder anwählen und bearbeiten. Änderungen ausführen. Zurück zur Datenbankbenutzung → 2.

Um das ***CockPit*** zu umgehen, damit's noch etwas schneller wird, kann man sich auch hier eine "Allers'sche Testschleife" anlegen, nennen wir sie TESTAPI.BAT :

```
:START
presto -f1 -a3 -n1 -S -dC:\ALLEGRO\KATALOG    (Datenbank wird aufgerufen, Test
mit F7)
pause
x CAT.API           (Parameterdatei bearbeiten)
goto START
```

Dieses kleine Programm schaltet ständig zwischen der Benutzung der Datenbank und der Bearbeitung der CAT.API hin und her. Man steigt aus der Schleife aus, indem man nach Verlassen der Datenbank <Strg>+c drückt.
Man muß auch hierfür vorher die Parameterdateien umbenennen, wie oben unter 1. beschrieben, denn PRESTO setzt voraus, daß der Name der Index-Parameterdatei identisch mit dem der Datenbank ist. Meistens wird man aber Veränderungen an der schon existierenden Parameterdatei einer Datenbank testen wollen, und dann entfällt dieser Schritt.

10.3.4 Export- und Index-Parameter optimieren

Kann man Zugriffe und Funktionen schneller machen? Wie spart man Speicherplatz? Wie kann ich als Systemverwalter Zeit sparen und die Übersicht behalten?
Je größer und zahlreicher die Datenbanken werden, umso drängender werden diese Fragen. Aus den vorliegenden Erfahrungen ergeben sich eine Reihe von praktischen Tips.

A. Zeitökonomie des Systemverwalters
Die eigene Zeit sollte und wird ein Systemverwalter im Zweifel höher bewerten als die Zeit, die seine Maschinen für ihre Aufgaben benötigen. Mühe darauf zu verwenden, daß ein Computer Zeit spart, scheint vielleicht manchem schon pervers. Jedoch - beliebigen "Overhead" in Kauf zu nehmen, nur um sich jedes Nachdenken (und jede Lernanstrengung) zu ersparen, das ist das andere Extrem und führt zu grober Ineffizienz und zu einem Schießen mit Kanonen auf Mücken. Solches passiert laufend in der Windows-Welt.
Hat man mehrere Datenbanken zu verwalten, sollte man danach streben, die Unterschiede gering zu halten. Erstens behält man besser die Übersicht, zweitens kann man die zeitaufwendigen Parameterdateien mehrfach verwenden. Also: gleiches Schema, gleiche Feldstrukturen für möglichst alle Datenbanken anstreben. Die konsolidierte Konfiguration $A.CFG deckt so ziemlich alles ab, was man für bibliographische Datenbanken braucht...
Insbesondere die Exportparameterdateien vermehren sich leicht. Damit die Übersicht nicht verloren geht, teilt man sie am besten in Typen ein, wie es Kap. 10.0 empfiehlt, und man orientiert sich für ihre Gestaltung an den Prototypen. Beim Zugriff über ***CockPit*** hat man so immer alles hübsch beisammen. Abschnitte, die sich in mehreren Parameterdateien wiederholen, packt man in .APT-Dateien und lädt diese bei Bedarf an den richtigen Stellen hinzu (→ 10.2 : Nachladebefehl **`t`**) .

Auch Batchdateien sind vermehrungsfreudig. Wenn mehrere Produktionsläufe nach demselben Schema ablaufen, nur mit unterschiedlichen Parameterdateien, kann man u.U. mit einer einzigen Batchdatei auskommen und immer vorher die jeweils gewünschten Parameterdateien auf einen immer gleichen Namen umkopieren. So macht es ***CockPit*** für die Listenproduktionen. Es kommt daher mit nur vier Batchdateien aus: `SR-LIST.BAT` und `PR-LIST.BAT`, `QUEX.BAT` und `QUANT.BAT`. Mehr dazu im Kap.6.

B. Zeitbedarf des Rechners (für das Speichern und Indexieren)

Das A und O einer schnellen Datenbank ist die Index-Parameterdatei. Durch Optimierung sind manchmal erstaunliche Steigerungen erreichbar. Folgende Tips, nach Wirksamkeit geordnet, helfen dabei:

- Globale Ersetzungen vermeiden. Besonders: Einzelzeichen immer mit p- und q-Befehlen umcodieren, denn z.B. der Befehl **`p ä "ae"`** ist erheblich schneller als **`_ä_ae_`** !
- Anzahl der ak-Befehle gering halten. Oft kann man mehrere zu einem "Sprungverteiler" zusammenfassen. Wenn man z.B. `ak=10+A 11+B 12+C` hat, kann man das ersetzen durch

  `ak=1.+A`

  und dann in der Kategorieliste:

  `#-A` #ch ist die momentan vom ak-Befehl ausgewählte Kategorie. Ihr zweites zeichen wird untersucht:
  `#ch +B i2,1 e0` (wenn es #11 ist, gehe nach #-B)
  `#ch +C i2,2 e0` (wenn #12, dann #-C)
  `#ch +- I2,0 e0` (wenn es nicht die #10 ist, keine Aktion)
  `...` (wenn es #10 ist, geht es also hier weiter)
  `#+#`
  `#-B` (hier wird #11 behandelt) ...
- Zum selben Thema kann man noch etwas anderes tun: gleichartige Kategorien zu einer Anwendervariable zusammenfassen. Sind z.B. #130, #240, #245, und #730 alle in Stichwörter zu zerlegen, also auf dieselbe Art zu indexieren, braucht man nicht vier ak-Befehle dafür, sondern man kann mit einem auskommen: in dem Abschnitt, wo der Primärschlüssel bearbeitet wird, fügt man noch einige Zeilen hinzu, um die genannten Kategorien zusammenzufassen:

`#nr dts e0`	Die Hilfsvariable #uts wird gelöscht
`#130 ats`	der Inhalt von #130 wird hineinkopiert
`#245 p" " Ats`	#245 wird, mit Leerzeichen davor, hinten angehängt
`#240 p" " Ats`	dasselbe passiert mit #240 und #730
`#730 p" " Ats`	

  Dann genügt `ak=uts" "+C` , um alle vier zu indexieren, denn sie stehen nun alle hintereinander, mit Leerzeichen getrennt, in #uts. Dieser Befehl kommt erst an die Reihe, nachdem der Primärschlüssel schon bearbeitet wurde, also steht die #uts dann auch bereit. [Bis Version 12.2 konnte man keine #u-Variablen in ak-Befehlen ansprechen!]
- Mit **einem** `ak`-Befehl kann man **mehrere** Schlüssel erzeugen: dazu muß man die einzelnen Schlüssel mit dem Code 8 trennen, also z.B. den Befehl **`#t{ 8 }`** dazwischen setzen. Dann aber kein Tabulatorbefehl & möglich.
- Innerhalb der Kategorieliste ersetzt man die **bedingten** Sprünge `+M` durch Sprünge der Form `+#nnn` , evtl. mit einer sonst nicht benutzten Kategorienummer, die dann nur als Sprungmarke dient. In der CAT.API können Sie dieses Verfahren ausführlich studieren. Dort wird meist #39x als Sprungmarke verwendet, mit unterschiedlichen Buchstaben 'x' . (Aber: **un**bedingte Sprünge `#+#nnn` statt **`#+M`** gehen noch nicht!)
- Die meistgebrauchten Abschnitte der Kategorieliste nach oben verlagern. Wenn also z.B. `#-C` der Abschnitt für die Titelstichwörter ist, setze man diesen Abschnitt an den Anfang der Kategorieliste.

C. Speicherbedarf

Man hört zuweilen die Ansicht, man brauche hier nicht mehr über Einsparungen nachzugrübeln, weil Platten fast nichts mehr kosten und schon nächste Woche wahrscheinlich noch billiger werden. Bibliotheksbestände wachsen aber unablässig, und eine Art Parkinsonsches Gesetz besagt, daß jede Platte voll wird, und zwar schneller, als man denkt. Nun geht ***allegro*** mit dem Speicherplatz schon "von Natur aus" sparsam um, und vom ***CockPit*** aus hat man bequeme Funktionen zum Wiedergewinnen von Platz (**`µr o c`** und **`µr o R`**), aber ein paar kleine Maßnahmen können weitere Einsparungen bewirken:

- Beim Datenbank-Neuentwurf nicht unnötig viele Kategorienummern definieren, sondern zusammengehörige Elemente in eine Kategorie packen, mit Teilfeldtrennung durch das Zeichen ▼
- Die Anzahl der Füllzeichen nicht zu hoch ansetzen - oder auf Null setzen und dann immer mal "entlüften".
- Die Länge der .STL-Einträge evtl. herabsetzen (Befehl **`i0=72`** setzt sie auf 72 Zeichen je Datensatz).

10.4 Beispiele: Prototypen

Zu den in 10.0 beschriebenen 10 Typen von Exporten werden ausführlich kommentierte Beispieldateien mitgeliefert. Sie finden diese nach der Installation

a) vom ***CockPit*** aus : im Menü "Dateien / Export-Parameter" unter dem Kennbuchstaben des Typs
(Tastenfolge: z.B. **µd e <umschalten> s** für den Typ S. Die Prototyp-Datei S-0 steht ganz oben.)

b) In a99 gibt es das Parameter-Bearbeitungsmenü : **h param** eingeben, dann "Exportparameter" wählen. Man sieht alle vorhandenen Parameterdateien auf den relevanten Verzeichnissen.

Ein vollständiger Abdruck aller Prototyp-Dateien hier im Handbuch ist aus Platzgründen nicht zu vertreten. Um aber für das theoretische Studium denen, die die Software selbst nicht besitzen, Material an die Hand zu geben, finden Sie im folgenden 5 der Prototypen, und zwar zu den Typen DISPLAY, SORT, PRINT, EXTERN und RESULT, jeweils mit Hinweisen zu ihrem Einsatz. Außerdem INDX.API als knappes, aber komplettes Beispiel einer INDEX-Parameterdatei.

10.4.1 Typ DISPLAY : Grundmuster

Es ist möglich, für die drei Plattformen DOS, Windows und HTML eine einzige, einheitliche Anzeigeparameterdatei zu erstellen. Dazu werden die Vorlagen **D-S.APT** und **D-K.APT** mitgeliefert. Das zweite braucht man nur, wenn es verknüpfte Sätze gibt. Diese Datei ist die einzige, in die man eingreifen muß. Sie enthält ausreichende Kommentare. Für die drei Plattformen wählt man als eigentliche Parameter dann die Datei D-SDOS, D-SRTF, D-SHTM bzw. D-KDOS, D-KRTF und D-KHTM, in die man aber nicht einzugreifen braucht. Wenn man nicht mit A.CFG arbeitet, kopiert man sich diese Dateien um auf z.B. D-S.xPT und D-SRTF.xPR usw.

D-0.APR : einfachstes Beispiel

Diese Datei zeigt die charakteristischen Eigenschaften, die eine für die Bildschirmanzeige geeignete Parameterdatei haben muß. Sie läßt sich leicht ausbauen. Erheblich umfangreicher ist D-1.APR, da sie alle Satztypen berücksichtigt und außerdem die Möglichkeiten der Satzverknüpfungen nutzt. Weil viele Finessen und Spezialfälle eingebaut sind, ist sie schwer überschaubar. Die vergleichsweise primitive D-0.APR eignet sich viel besser als Grundlage für den Einstieg.
Für die Kommentare in den Beispielen wurde eine andere, kleinere Schriftart gewählt, damit sich Parameter und Kommentare besser voneinander abheben.

```
  Parameterdatei für Bildschirmanzeige UND Ausgabe in Form von Katalogkarten
  Prototyp für den Exporttyp DISPLAY
pn="DISPLAY Prototyp"
```

```
  --------- Konstanten ------------------------------------------
zl=65             Zeilenlänge
fl=14             Anzahl Zeilen je Karte
fm=2              Modus: Einzelkarten für alle Bände (1 = keine Einzelkarten)
fb=13 10 10 10    Formular-Bruch: 3 Zeilenvorschübe zwischen zwei Karten
tPRINTER          den vom CockPit gewählten Druckertreiber laden, er heißt PRINTER.APT (= Kopie eines P-*.APT)
                  (enthält die Zwischenteile 79-99 für die Schriftarten)

  ======== Kopfbefehle ==========================================
                  (jede der folgenden Kategorien erzeugt einen Ausgabesatz, falls sie existiert)
  Haupteintragung:
ak=zz+H           (diese Zeile wird IMMER ausgeführt, bei Sprungmarke #-H beginnt der Aufbau der "Einheitskarte"
  Zusatzeinträge: (Nebeneintragungen)
ak=40."; "+V      alle Verfasser kriegen eine
                  (#40, #402..., #40a..., auch #40 Name; Name; Name... ist möglich)
ak=85.+G          Serientitel (#85, #85a, #85b...)
  ================================================================

  ....... Zwischenteil-Definitionen: (nach Bedarf erweitern) .......
4=C  Zeilenvorschub  (die Zwischenteile 79-99 stecken in PRINTER.APT !
  ................................................................
```

```
  ******* Kategorieliste ****************************************
                = Hauptteil der Parameterdatei
#hi +Z e0       Unteraufnahme -> Sprungmarke #-Z
                die Untersätze werden nicht über ak-Befehle gesteuert, sondern die Verarbeitung beginnt am Anfang der
                Kategorieliste, nachdem die Hauptaufnahme verarbeitet ist. Mit diesem Befehl kann man zu einem
                Abschnitt springen, der die Aufbereitung der Untersätze regelt, hier #-Z (#hi existiert nur, wenn ein
                Untersatz verarbeitet wird)
#-V  Verfasser-Einträge
#t99            Fettschrift ein
#u1 +- i3,      wenn #40, dann kein zusätzlicher Zettel, denn #40 bekommt bereits den Einheitszettel bei #-H
#u1             der durch ak=40.";"+V jeweils ausgewählte Name
#t98            Fett aus
#90 p"Signatur: " R         Signatur rechtsbündig
#t4             Zeilenvorschub
#+h             weiter wie bei der Haupteintragung

#-G  Serienzettel
#t99
#u1 e" ; "      zuerst nur der Serientitel
#u1 b" ; " p"-- " C         dann die Zählung auf neuer Zeile, mit "--" davor
#t98
#90 p"Signatur: " R         Signatur rechtsbündig
#t4
#+h

#-H  Hauptzettel-Produktion beginnt hier (Einheitskarte)
#90 "Signatur: " R Signatur rechtsbündig
#t4

#-h
#40 P":" P4     "Name:"   dann neue Zeile (Postfix P4)
#20             Titel
#39 " / "       Verf.Angabe
#-A             Einsprung hier von der Bandbearbeitung aus (s.u.)
#71 ". - "      Ausgabe
#74 ". - "      Ersch.Ort mit ". - " angeschlossen
  #74 C "   "   wenn Ort auf neuer Zeile eingerückt erscheinen soll
#75 ": "        Verlag mit ": " dahinter
#76 ", "        Jahr mit ", " angehängt
#77 ". - "      Umfang mit ". - " abgetrennt
#85 C p"(" P")" Serie auf neuer Zeile in Klammern
#83 C "[Kongr.:] "
#87 C e11 p"ISBN "
#+#             Ende

#-Z  Unteraufnahme beginnt hier

#hi1 C C P". "  Bandnummer, 2x Zeilenvorschub
#t{ #250 }      Einrückung an der Position hinter der Bandnummer festsetzen
#90 R           Signatur rechtsbündig
#19             Bandbezeichnung
#20             Bandtitel
#41 " / " P" [Hrsg.]"
#+A             weiter wie bei der HA

  +++++++ Codierungen +++++++++++++++++++++++++++++++++++++++++++++
p ¬ 1           Nichtsortierzeichen nicht mit anzeigen
p @ 1           Nicht-Stopzeichen auch nicht
```

10.4.2 Typ SORT : S-0.APR

Das Herstellen sortierter Listen ist ein mindestens dreistufiger Vorgang (genaueres siehe Kap. 6). Der erste Schritt ist immer die Produktion einer **sortierfähigen** Datei. Diese hat die Struktur einer ***allegro***-Grunddatei (→ 0.3), aber jedem Satz geht eine besondere Kategorie voran: **#u1** , das Sortierfeld. Es kann noch **#u2** hinzutreten und eine Druckform des Sortierfeldes enthalten; **#u1** eignet sich nicht für den Ausdruck, weil es computergerecht sortierfähig sein muß, und das bedeutet, daß es vollständig in Kleinbuchstaben, mit aufgelösten Umlauten und weggelassenen Akzenten geschrieben ist. Wenn **#u2** dann den Sortierkopf in druckfähiger Form enthält, kann für den (nach dem Sortieren) dritten Schritt, die Druckformatierung, die Kategorie **#u2** als Kopf benutzt werden. Der Vorteil ist: die Parameterdatei für die Druckgestaltung (Typ PRINT) kann unabhängig davon gestaltet werden, welche Köpfe konkret vorliegen, ob es also eine Schlagwort-, Titel- oder Verfasserliste oder noch etwas anderes ist.
Die mitgelieferten Dateien **S-*.APR** und **P-*.APR** sind alle auf diese Weise gestaltet und können daher beliebig miteinander kombiniert werden. Im Kap. 6 ist beschrieben, wie die konkrete Produktion von Listen vom ***CockPit*** noch zusätzlich unterstützt wird.
Hier folgt nun der Prototyp **S-0.APR** für die Erstellung einer sortierfähigen Datei mit Kategorien **#u1** und **#u2**. Nach diesem "Strickmuster" arbeiten alle Dateien dieses Typs.

```
  Sortierfähige .ALG-Datei erstellen  Beispiel: Person/Jahr
  Prototyp für den Exporttyp SORT

pn="Prototyp SORTiervorbereitung"

 Ziel: eine .ALG-Datei mit Sortierkopf und Druckkopf. Struktur:
 #u1 sortierkopf: Verfasser (#40) oder Urheber (#60) oder Sachtitel (#20)¶Jahr
 #u2 Druckform derselben Angaben
 #... die gesamte Aufnahme in .ALG-Struktur

 anschliessend mit ASORT zu sortieren und mit einer der PRINT-Param. auszudrucken.

  --------- Konstanten ----------------------------------------

zl=0              Zeilenlänge unbegrenzt (kein Umbruch)
ks=1              Beginn des Ausgabetextes beim ersten Zeichen hinter #
ke=0              Kategorie-Ende = Code 0 (wegen .ALG-Struktur)
                  (unten steht  "#zz 0" in jeder Zeile, die NICHT automatisch diese Null am Ende bekommen soll.)
as=""             Aufnahme-Start: keine Angabe (wird mit #t gemacht)
ae=13 10          Aufn.Ende: 13 10  (nötig für das Sortieren)

tS                lädt die Zeichenumwandlungstabelle S.APT für das Sortieren
dx=1              an sich nicht nötig, aber zum Testen nützlich (mit ATEST S-0 kann man am Bildschirm sehen, wie die
                  Ergebnisse, d.h. die Sortiersätze, aussehen)

am=1              maximal 1 Ausgabesatz
  ------------------------------------------------------------

  ========= Kopfbefehle ====================================
  ================== werden nicht gebraucht ==================

  ********* Kategorieliste ************************************

#hi +b e0 #zz 0  wenn es eine Unteraufnahme ist: Sprung nach #-b
                  (wird dann an die bereits ausgegebene Hauptaufnahme angehängt.)
```

```
   Normale Sätze gehen hier los
#-a
                    Umwandlung der !-Felder über Alternativtabelle:
                    der Sortierbegriff wird zusammengesetzt und als #u1 ausgegeben
                    (Name Sachtitel bzw. Verfasser Sammlungsvermerk)

#t{ 1 }             1 = Startcode für "Hauptaufnahme"
!40 +s e"[;=]" b3 u F" " p"u1 " #zz 0  Verfasserwerk
                    Kopfkategorie als Sortierfeld, Ende bei ";" oder "="
                    b3 heisst: Beginn beim 4. Zeichen = 1. Textzeichen
                    (wegen ks=1 würde sonst die Kategorienummer mit ausgegeben,
                    (bei 3stelligem Schema muß es b4 heißen)
!60 +s e"[;=]" b3 u F" " p"u1 " #zz 0  Urheberwerk
!20 e" : " b3 u u[] p"u1 " #zz 0     Sachtitelwerk

#-s
!76 b"1" e3 p"¶" #zz 0
                    zweites Element: "¶Jahr"  (¶ sorgt für richtige Sortierung: kürzere Titel vor längeren)

#t{ 0 }             als Abschluß von #u1 (ASCII-Code 0 ausgeben)
                    Ergebnis:  #u1 name¶erscheinungsjahr\0

   Die Druckform des Sortierfeldes: Produktion als #u2
   (genauso wie #u1, nur ohne Umcodierung, dann für den Druck verwertbar!)

#40 +d e"[;=]" b3 u F" " p"u2 " #zz 0  Verfasserwerk
                    Kopfkategorie als Sortierfeld, Ende bei ";" oder "="
                    b3 heißt: Beginn beim 4. Zeichen = 1. Textzeichen
                    (wegen ks=1 würde sonst die Kategorienummer mit ausgegeben,
                    (bei 3stelligem Schema muß es b4 heißen)
#60 +d e"[;=]" b3 u F" " p"u2 " #zz 0  Urheberwerk
#20 e" : " b3 u u[] p"u2 " #zz 0                    Sachtitelwerk

#-d
#76 b"1" e3 p"¶ #zz 0
                    zweites Element: "¶Jahr"  (¶ ermöglicht, es für den Druck ausseinanderzunehmen)

#t{ 0 }             als Abschluß von #u2
                    Ergebnis:  #u1 Name, Vorname¶Erscheinungsjahr\0
#-P
##                  jetzt alle Kategorien hintereinander (Pauschalexport)
 tSELECT            statt dessen, wenn nur eine bestimmte Kategorieauswahl in
                    die Sortierdatei gelangen soll (Auswahl über den CockPit-
                    Menüpunkt "Optionen / Vollständigkeit")
#nr b3 p"99z"       und die interne Datenbank-Satznummer am Schluß als #99z
#+#   Ende der Hauptaufnahme

#-b                 Sprungmarke b: hier geht es bei untergeordneten Sätzen los
#t { h0 }           Hierarchiecode (= 2 für Stufe #01 usw.)
#+P                 Zum Pauschalexport, wie bei Hauptaufnahme
```

10.4.3 Typ PRINT : P-0.APR

Das Drucken erfordert einen **Zeilenumbruch** und evtl. auch einen **Seitenumbruch**. Für diese Aufgaben gibt es Parameter, die sonst nicht gebraucht werden.

```
 Allgemeines Listenformat mit oder ohne Seitenumbruch                    (verkürzt, siehe Datei P-0.APR)
 P-0.APR      940602
 --------- Konstanten -------------------------------------------
zl=72                Zeilenlänge
zi=5                 Einrückung ab 2. Zeile des Satzes
fl=0                 Formularlänge 0, d.h.: kein Formulardruck
ae=13 10             Zeilenvorschub am Aufnahmeende
zm=62                max Zeilen je Seite, oder: (für Nachbearbeitung mit Textprogramm)
  zm=0               dann kein Seitenumbruch, sondern fortlaufende Ausgabe
z1=0                 Titel am Seitenende nicht unterbrechen (bei zm=0 unwirksam)

tPRINTER             Druckertreiber laden  (PRINTER.APT)
                     Entsteht bei Auswahl über CockPit µo d  als Kopie einer der Dateien P-*.APT, z.B. P-EPSON.APT.
                     PRINTER.APT kann auch die oben gesetzten Werte ändern!
 ..... ZWISCHENTEILE .....................................
4=C                  Zeilenvorschub
13=C C               2x Zeilenvorschub
20="--  "
                     die Nummern 79-99 sind in PRINTER.APT definiert, z.B: 99 für Fett ein, 98 für Fett aus
 ********* Kategorieliste ***********************************
 Das folgende ist zu ersetzen durch die eigene Gestaltung

#t{ &0 }             an den Zeilenanfang gehen (wichtig bei Untersätzen, damit sie wieder am Anfang der Zeile beginnen)
#hi1 +#20 p"  " P". "   wenn's ein Untersatz ist: Bandbezeichnung (#hi wird aus #01...#05 gebildet)
#t99                 fett ein
#u2 e": " =ti,20       wenn vorher derselbe Name, "-- " statt dessen
#u2 +#20 e0          wenn es eine Sortierdatei ist: #u2 = Sortierbegriff
#40 P": "            sonst wird #40 zuerst ausgegeben
#20 p98
#74 p". - "
#75 p": "
#76 p", "
#77 p". - "
#85 C p"(" P")"
#t88                 kursiv ein
#90 e"▼" R           Signatur rechtsbündig
#t87                 kursiv aus
#t4                  Zeilenvorschub am Ende eines Satzes
#+#                  Export des Satzes beendet

#- E                 Endabschnitt: wird ganz am Ende der Liste ausgeführt
.....                Text des Endabschnitts
K                    Kopfabschnitt (Ausführung am Seitenanfang)
.....                (Hier kann ein eigener Abschnitt folgen, s. Datei P-0.APR)
F                    Fußabschnitt (Ausführung am Seitenende)
.....                (Hier kann ein eigener Abschnitt folgen, s. Datei P-0.APR)
```

10.4.4 Typ EXTERN : E-0.APR

Um ***allegro***-Daten mit anderen Programmen verarbeiten zu können, muß man sie in der Regel in eine Form bringen, die **Feldbezeichnungen** und Zeilenumbrüche enthält. Dieser Prototyp zeigt, welche Einstellungen dazu nötig sind:
(Weiteres wichtiges Beispiel: E-W.APR für extern editierbare, mit ***avanti*** und VPW wiedereinlesbare Dateien)

Ausgabe als ASCII-Datei mit Kategorie-Nummern und Zeilenumbruch
Prototyp für den Exporttyp EXTERN

```
pn="EXTERN Prototyp"
```

Gibt Daten als ASCII-Datein aus, jede Kategorie mit Nummer auf neuer Zeile,
längere Kategorien werden bei 66 Zeichen umgebrochen und auf der Fortsetzungszeile eingerückt.

```
  --------- Konstanten ------------------------------------------
```

```
zl=66              Zeilenlänge
zi=4               4 Zeichen Einrückung bei Folgezeilen
 zi=5              bei 3stelligem Schema  (einzige Änderung!!!)
zm=0               kein Seitenumbruch (fortlaufende Ausgabe)
ae=13 10           Zeilenvorschub am Aufnahme-Ende
fl=0               Listenstruktur (keine Karten)
ks=0               Startpos. 0  (= Zeichen '#') innerhalb jeder Kategorie
  ks=1   setzen, wenn '#' wegfallen soll)
  ks=4   wenn nur der Text, nicht die Kategorienummer auszugeben ist (ks=5 bei 3stell. Schema)
ke=C               Kategorie-Ende: Zeilenvorschub
dx=1               damit man beim Testen auch die Steuerzeichen auf dem
                   Bildschirm sieht. (Ansonsten nicht nötig)
  ------------------------------------------------------------
```

```
  ********* Kategorieliste ***************************************
                   Entweder:
##                 sämtliche Kategorien ausgeben in interner Reihenfolge

                   falls nicht alle gebraucht oder gewünscht werden:
                   Weglassungen möglich: Kategorienummern mit / hier anfügen, z.B.
  /89  dann wird #89 weggelassen
  /9..  dann wird die ganze Gruppe #9 weggelassen

                   oder: falls nur ganz bestimmte gebraucht werden: hier eine Liste der auszugebenden Kategorien eintragen
                   (dann  ##  beseitigen!)
    z.B.
  #20              Hat man oben ks=4 (bzw. ks=5), dann muß man hier geeignete Präfixe
  #40              vor die Kategorien setzen lassen. Das könnte z.B. so aussehen:
  #74              #40 "AU: "   bzw.   #20 "TI: "
  #75              Je nach Länge dieser Präfixe müßte man oben den Wert ki heraufsetzen.
  #87              damit Fortsetzungszeilen bei längeren Kategorien richtig eingerückt werden.
  #90
                   dann werden nur diese Kategorien ausgegeben
  ************************************************************
```

Code-Umwandlungen: p-Befehle bei Bedarf hier anfügen oder mit Befehl t hinzuladen

10.4.5 Typ RESULT : R-0.APR

Dieses Beispiel zeigt zugleich das Verfahren, wie man rechnerische **Auswertungen** durchführt. Ergebnismengen oder auch eine gesamte Datenbank werden, um z.B. eine Summierung mit Durchschnittsberechnung vorzunehmen, durchgearbeitet wie bei einem Export, nur daß die Exportparameterdatei die einzelnen Datensätze nicht in eine Ausgabedatei schreibt, sondern lediglich Berechnungen anstellt. Erstens kommen also **Rechenbefehle** (→ 10.2.6.6) zum Einsatz. Zweitens braucht man Anwendervariablen (→ 10.2.6.3), um Zahlen von einem Datensatz zum nächsten aufbewahren und immer weiter verrechnen zu können. Am Ende dann wird auch noch etwas ausgegeben, aber kein Datensatz, sondern eben die in den Anwendervariablen erzeugten Rechenwerte. Wie das im einzelnen aussieht, zeigt der Prototyp R-0.APR, der die Seitenzahlen und Erscheinungsjahre aller Bücher summiert und am Ende die Durchschnittswerte ausrechnet. Unsinn? In den Beispieldaten, und an denen können Sie es ja zunächst nur ausprobieren, sind keine anderen Zahlenangaben zu finden. Die Übertragung auf andere Kategorien, die sich für sinnvolle Berechnungen eignen, fällt mit Hilfe dieses Prototyps nicht schwer.

```
  Summierung und Durchschnittsbildung von Kategorieinhalten
  Prototyp für den Exporttyp RESULT

pn="Seitensumme, Prototyp RESULT"
```

Gibt nichts aus, sondern addiert nur. Erst der Fußabschnitt führt eine Ausgabe durch, siehe unten

```
  --------- Konstanten -----------------------------------------
ff=2              um Kopf- und Fußabschnitte zu aktivieren
zm=0              damit der Fußabschnitt erst ganz am Ende ausgeführt wird
  ----------------------------------------------------------------
```

Es sollen die Seitenzahlen addiert werden, wobei Fälle mit weniger als 10 Seiten nicht mitgerechnet, sondern getrennt gezählt werden (z.B. kann dabei sowas wie #77 Vol.1-5 vorkommen, was also keine Seitenzahl ist! Unteraufnahmen werden mit verarbeitet!)
Ferner soll die kleinste und die grösste Seitenzahl ermittelt und der Durchschnitt berechnet werden.
Die Erscheinungsjahre sollen ebenfalls addiert und ihr Durchschnitt ausgegeben werden.

```
  !!!!!!!!!!!!!!!!!!!!   Anpassungen für andere Zwecke : !!!!!!!!!!!!!!!!!!
  !  Setzen Sie für #76 und #77 die Kategorien ein, die Sie addieren wollen.!
  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

ak-Befehle: keine

```
  ************* Kategorieliste ***********************************
```

```
#76 x">1449" x"+ej" =ej
#76 x">1449" x"=1" x"+nj" =nj
```

wenn >1449, dann Ersch.Jahr addieren, #uej = #uej + #76
und #unj um 1 erhöhen #unj = #unj + 1 (nur dann, wenn #76 existiert!)

Der Befehl Z bewirkt, daß die Zeilen jeweils nur ausgeführt werden, aber keine Ausgabe bewirken. D.h. es werden nur die Anwendervariablen bedient, die Zwischenergebnisse aber nicht ausgedruckt.
(Z macht dasselbe wie e0)

```
#77 +M x"<10" Z  Sprung nach #-M, wenn Sonderfall #77 < 10 (Z muß sein, weil sonst der Inhalt von #77 ausgegeben
                 wird, wenn der Vergleich zutrifft; Z unterdrückt das. Bei den folgenden Anweisungen ist Z nicht
                 nötig, denn der Befehl '=' macht nur eine Zuweisung und beendet dann die Anweisung ohne
                 Ausgabe des Textes)
#77 x"+su" =su   Seitenzahl addieren:    #usu = #usu + #76 ,
#unr x"+1" =nr   #unr um 1 erhöhen
#77 x">gw" =gw   wenn größer als bisheriger größter Wert:   #ugw=#77
#77 x"<kw" =kw   wenn kleiner als bisheriger kleinster Wert: #ukw=#77
#+#

#-M              Zählung der Sonderfälle
#uk1 x"+1" =k1   #uk1 um 1 erhöhen
#+#
```

`********** **Kopf- und Fußabschnitte** **********`

Kopfteil: Initialisierung der Variablen

#dt wird als nur Hilfskategorie benutzt, weil sie immer existiert

```
K
#dt x"=0" =su        #usu für Summe der Seitenzahlen
```

das bedeutet: nimm den Inhalt vom #dt als Variable, weise ihr den Wert 0 zu und speichere diesen in #usu (etwas indirekt, aber nur so geht es)

Variablen:

```
#dt x"=0" =nr        #unr für Zähler der tatsächlich addierten Fälle
#dt x"=0" =ej        #uej für Summe der Ersch.jahre
#dt x"=0" =nj        #unj für Anzahl der addierten Erscheinungsjahre
#dt x"=0" =gw        #ugw für größten Wert
#dt x"=99999" =kw            #ukw für kleinsten Wert
#dt x"=0" =k1        #uk1 für Anzahl der Fälle kleiner als 10
```

`**********`

Fußteil: Ausgabe der Ergebnisse

(Kontrolle mit Befehl #a im Editor möglich!)

```
F
#t{ "Ergebnisse: " }
#usu C e"." r7 p"Summe:  "              e"." schneidet den Dezimalteil ab
#unr C e"." r7 p"Anzahl: "       r7 füllt links mit Blanks auf 7 Stellen

#ugw C C x"*1" e"." r6 p"größte:  "    x"*1" wird gemacht, weil die Werte
#ukw C x"*1" e"." r6 p"kleinste:"      noch Buchstaben enthalten könnten, die werden dadurch beseitigt
#uk1 C e"." r6 p"unter 10:"
#usu C C x"/nr" x"r2" r9 p"Durchschnitt: "
```

nimm Inhalt von #usu, teile durch #unr, runde auf 2 Stellen

```
#uej C C x"/nj" e"." r6 "durchschn. Ersch.Jahr: "
```

So arbeitet man damit

Jetzt zeigen wir noch, wie man mit so einer Parameterdatei umgeht, d.h. wie man denn die Berechnung wirklich durchführt. Am schnellsten läßt es sich so ausprobieren:
Geben Sie auf dem Verzeichnis C:\ALLEGRO den Befehl

```
srch -f4 -dbeispiel -er-0/liste -v0 -m0 -s"berlin/muenchen"
```

Es erscheint die Liste der CAT_*.ALD-Dateien. Setzen Sie '+' vor CAT_1 und CAT_2 und drücken Sie <Enter>. Nun lehnen Sie sich bequem zurück. Bald darauf kommt die Vollzugsmeldung. Mit dem Befehl

```
v liste
```

sehen Sie die Ergebnisse. Der Suchbegriff **`-s"berlin/muenchen"`** kann durch jeden anderen ersetzt werden. In diesem Fall wurden also diejenigen Bücher summiert, in deren Datensätzen die Wörter Berlin oder München vorkommen. So sieht die Ergebnisliste aus:

```
Ergebnisse:
Summe:    17253
Anzahl:      52

größte:    1218
kleinste:    46
unter 10:     6

Durchschnitt:    331.79

durchschn. Ersch.Jahr:   1971
```

Für Auswertungen, die man regelmäßig durchführen will, stellt man ein Batchfile zusammen, dessen Kern ein SRCH-Aufruf wie der gerade gezeigte ist. Angebunden an das ***CockPit*** reduziert sich dann die Auswertung auf die Wahl eines Menüpunktes.

Und so macht man es im Schnellzugriff:

1. Parameterdatei laden: mit <Shift+F4> im Anzeigemenü sucht man sich die R-0.APR und läßt sie laden, als Exportdatei gibt man NUL, weil man keine Ausgabedatei zu schreiben hat.
2. Ergebnismenge bilden, die verrechnet werden soll
3. Von irgendeiner Titelanzeige aus **F4** geben, um die Ergebnismenge durchrechnen zu lassen.
4. Wenn "erledigt" kommt: **F2 3 F** eingeben: Ergebnisse werden angezeigt.
9. Mit **F2 3 K** setzt man die Werte wieder auf 0 (Kopfabschnitt wird durchgearbeitet), und ein neuer Durchlauf mit anderer Ergebnismenge kann beginnen.

Wenn Sie nun eigene Auswertungen programmieren wollen, gehen Sie so vor:

1. Vom ***CockPit***-Menü "Dateien / Export-Parameter" suchen Sie sich die Datei R-0.APR.
2. Das Untermenü hat einen Punkt "neue Version erstellen". Diesen wählen Sie an.
3. Die Frage nach dem Namen für die neue Datei beantworten Sie mit einem neuen Namen, der mit **`R-`** beginnen sollte, damit die Zugehörigkeit zum Typ RESULT immer sofort ersichtlich ist.
4. Eine Kopie von R-0.APR wird zum Bearbeiten vorgelegt. Wenn es nur um Summierungen und Durchschnittsbildungen, aber mit anderen Kategorien geht, ist nicht viel zu tun. Abweichende Aufgaben erfordern evtl. einige Kenntnisse des Parametrierens, besonders der Rechenbefehle (→ 10.2.6.6).
5. Die neue Parameterdatei kann nun auf dieselbe Weise wie oben gezeigt benutzt werden.

Hinweis: Für ***a99*** gibt es als Teil des ORDA-Paketes auch Beispiele, wie man mit der FLEX-Technik Ergebnismengen durchrechnen kann. Man startet die DEMO-Datenbank und gibt im Schreibfeld den Befehl `h orda` . Es erscheint ein Menü, auf dem man "Kontingent-Berechnung" (kont.flx) oder "Bestellungen eines Lieferanten" (lief.flx) wählt.

10.4.6 Typ INDEX : INDX.API

Diese Beispieldatei zeigt nur die Befehle, die nötig sind, um alle Personen aus den Kategorien #40 bis #59 in Register 1 und alle Stichwörter aus #20 in Index 3 zu ordnen. Die neuen Möglichkeiten von V14 sind hier noch berücksichtigt (→ 10.2.6.8). Man findet dafür Beispiele in CAT.API und PICA.PPI.

```
pn="Prototyp einer Indexparameterdatei"
```
die folgenden Zeilen gelten für jede Index-Parameterdatei, nicht ändern!
```
  -------------- Konstanten ------------------------------------------------
zl=0                kein Zeilenumbruch
zm=0                keine Seiteneinteilung
ad=0                Wenn keiner der "Köpfe" (ak) vorkommt, dann keine Eintragung
ag=0                Hierarchische Untereinheiten ebenfalls für Index behandeln
  -------------- spezifische Index-Parameter -------------------------------
il=150              maximale Schlüssellänge (72 ist optimal und deshalb default)
i0=72               Kurzregister von 72 Zeichen Länge gewünscht (.STL-Datei)

i2="¶<> "           Indexeintragungen sollen nicht mit einem dieser Zeichen anfangen
i3=".,:]- "         diese Zeichen sollen am Ende eines Schlüssels beseitigt werden
```
Namen der Register / Überschrift der Titelanzeige
```
|1="1 : Namen von Personen , Titel (Kreuzregister)"
|3="3 : Wörter aus Titeln, Schlagwörter"
|a="             Katalog der 'allegro'-Demo-Bibliothek                    "
```
und für die Kurzanzeige: (s.u. Abschnitt #-0)
```
|<="·····Titel·····································Verf.···········Signatur"

   ===== Kopfbefehle: (= die für Index-Eintragungen heranzuziehenden Kategorien)
ak=4."; "+A 5."; "+A        alle Personennamen bei "; " zerlegen
```
Sachtitel: Wörter auseinandernehmen bei Leerzeichen und Apostroph
```
ak=20" "+C
```
Kurztitelregister
```
ak=zz+0             (Standard-Befehlszeile für diesen Zweck!)
   ============= Ende der  Kopfbefehle ====================================
   ************* Hauptteil: Kategorieliste *********************************

#-A                 hier beginnt die Bearbeitung für die #4. und #5.-Kategorien
!u1 e" = "          hinter " = " folgt evtl. eine Verweisungsform: mit e" = " wird sie abgeschnitten
#+#                 d.h. z.B. #40 Ansetzungsform = Verweisungsform

#-C                 Sachtitelwörter
!u1 p"|3"           sind in Reg. 3 einzutragen
#+#

#-0                 Erzeugung des Kurzregisters (die Sprungmarke dafür muß #-0 sein!)
#20 e" : " U f'"' y0 e55 "|0"
#40 +#76 { &47 } u y0 "/" e11
#90 { &66 } y0
#+#
```
so entstehen Einträge der Form
```
  "Titel                                          /Verfasser         Signatur"

   ++++++++ Codierungsbefehle ++++++++++++++++++++++++++++++++++++
p...                hier folgt die Liste der Zeichenumwandlungsbefehle
q...                (groß in klein, Umlautauflösung etc., hier nicht ausgeführt)
tswl1               Die Stopwortliste  SWL1.APT  wird geladen
```

11 Import-Parameter

Online: `h ac11`

11.0 Zum Konzept

Der Ausdruck **Import** steht im ***allegro***-System für die Umwandlung andersartig strukturierter Daten in eine ***allegro***-Grundstruktur.
Wie so etwas abläuft, wird in Kap. 5 beschrieben. Lesen Sie zum besseren Verständnis den Abschnitt 5.0 vor der Lektüre dieses Kapitels.

Dieses Kapitel führt in die **Importsprache** ein, mit der man Fremddaten von fast beliebiger Struktur beschreiben und ihre Überführung in eine ***allegro*-Grunddatei** (Typ .ALG) programmieren kann. Eine Grunddatei läßt sich anschließend, also in einem zweiten Schritt, mit dem Programm INDEX oder UPD in eine Datenbank umwandeln oder in eine vorhandene Datenbank einmischen.
Legionen von Programmierern haben sich für gleiche oder sehr ähnliche Anwendungen die unterschiedlichsten Datenstrukturen ausgedacht. Genau aus diesem Grund ist eine universelle Importsprache nötig, weil man sonst für jede Umwandlung ein eigenes Programm in C, Pascal oder einer anderen "Hochsprache" schreiben müßte. Eine solche universelle Sprache muß weitaus mehr leisten als nur Datenfelder umgruppieren und anders numerieren. Dieses Kapitel beschreibt ausführlich, welche Einzelaufgaben anfallen können und wie man sie mit ***allegro*** löst.
In der ***allegro***-Importsprache als Spezialsprache erstellt man Umwandlungsprogramme, hat man sich einmal eingearbeitet, in Bruchteilen der Zeit, die man etwa für ein C- oder BASIC- oder Perl-Programm bräuchte. Auch ad-hoc-Umwandlungen für Einmal-Aktionen, die sonst am Aufwand scheitern würden, sind daher realisierbar. Allerdings ist es alternativ auch möglich, Umwandlungen mit der Skriptsprache FLEX durchzuführen. Dies geschieht schon in einigen wichtigen Fällen, z.B. mit `dnb.flx` für Daten aus dem DNB-Portal (`X dnb` eingeben in ***a99***) und `srugbv.flx` für GBV-Daten (`X srugbv` eingeben).

Nach Untersuchung einer größeren Zahl von Datenformaten stellte sich heraus, daß man alle überhaupt relevanten Formate in nur vier Typen einteilen kann: (Beispiele siehe 11.1)

A **feste Satz- und Feldlänge**: alle Datensätze haben dieselbe Länge und sind an festgelegten Stellen in Felder untergliedert. Zu diesem Typ gehören die meisten kommerziellen Datenbanksysteme. Steuerzeichen für Feld- und Satzbegrenzungen gibt es in der Regel nicht, denn sie wären redundant.

B **feste Anzahl Felder mit variabler Länge**: die einzelnen Felder sind zwar (meist in Grenzen) variabel lang, aber es gibt eine feste Anzahl von Feldern, die durch ein bestimmtes Steuerzeichen (sog. "delimiter") oder eine Zeichenkombination getrennt sind. Oft ist dies das TAB-Zeichen (ASCII-Code 9). Bei diesem Typ haben die Felder meist keine Kategoriekennungen, sondern die Bedeutung ergibt sich aus ihrer Position innerhalb der Feldkette.

C **variable Anzahl Felder mit variabler Länge**: dieser Typ, zu dem auch ***allegro***-Daten gehören, hat in der Regel eindeutige Feldkennungen, damit die einzelnen Datenfelder überhaupt identifizierbar sind. Es gibt Steuerzeichen oder Zeichenkombinationen, die als Satz- und Feldbegrenzungen fungieren. Jede Datenquelle hat hierbei ihre Besonderheiten. Die sog. "Download"-Dateien von Datenbank-Hosts und von den meisten CD-ROM-Datenbanken fallen unter diesen Typ, ebenfalls Daten, die man mit "Copy&Paste" aus WWW-Anzeigen herauskopiert, z.B. bei dem gern genutzten ZACK-System (zugänglich über den *allegro*-Web-Server).

D **Satzstruktur mit "directory"**: nach einer internationalen Norm (ISO 2709 bzw. DIN 1506) werden bibliographische Austauschdaten auf Magnetband mit dieser Struktur geliefert. MARC- und MAB1-Daten sind oft, aber nicht immer, in dieser Form anzutreffen. Die Norm läßt noch einige Freiheitsgrade, die in der Praxis auch ausgenutzt werden. Die Feldkennungen stehen in einem "Inhaltsverzeichnis", die Feldinhalte davon getrennt in einem variablen Teil. Es kann, muß aber nicht, zusätzliche Feld- und Satzbegrenzungszeichen geben. Ansonsten liegt hier ebenfalls eine variable Anzahl Felder mit variabler Länge vor.
Das Format MAB2, das Anfang 1996 eingeführt wurde, wird von der Deutschen Bibliothek nicht mehr in ISO-Struktur verbreitet. Es gehört damit zum Typ C.

Für den ***allegro***-Anwender können noch zwei Sonderfälle wichtig werden:

F ***allegro***-Daten. Damit sind .ALG-Dateien gemeint (→ 0.3), aber auch andere Dateien, in denen die korrekten Kategorienummern bereits als Feldbezeichnungen enthalten sind. Es kann vorkommen, daß ein Anwender seine eigenen Daten importieren will, um etwa die Fähigkeiten der Importsprache für gewisse Umwandlungen auszunutzen. Deshalb gibt es auch für diesen Datentyp noch besondere Möglichkeiten (→11.2.3.1)

X **XML**-Daten: Diese fallen unter den Typ C! Die FLEX-Sprache hat komfortable Funktionen zur Extraktion von Elementen aus XML-Daten. Mehr dazu: **`h xcstring`** eingeben in ***a99***, oder http://www.allegro-c.de/axml.htm.

Datenbanken, die intern nicht einem dieser Typen entsprechen, können in aller Regel eine Ausgabeform ("download" oder "export") produzieren, die unter Typ B oder C fällt. Aus MySQL z.B. produziert man eine Ausgabe vom Typ B mit dem Befehl `INTO OUTFILE '/tmp/datei.txt'`, angehängt an ein SELECT-Statement. Die Felder sind dann durch TAB getrennt.

11.0.1 Struktur von *allegro*-Dateien

allegro-Daten sind vergleichsweise einfach strukturiert (wenn man vom Index absieht):

- Hauptsätze in Grunddateien (Typ .*d*LG) und Datenbank-Dateien (Typ .*d*LD) beginnen mit dem Code 1 (01h), Untersätze mit 2...6 (02h...06h). In Datenbank-Dateien steht hinter dem Code 1 eine 4 Byte lange Satznummer, ansonsten sind beide Strukturen gleich. Ein Hauptsatz und beliebig viele Untersätze bilden eine Satzgruppe. Ein gelöschter Satz beginnt mit Code 9 statt 1, der Rest ist gleich.
- die Datenfelder (Kategorien) beginnen mit festgelegten Kennungen aus zwei oder drei Ziffern oder Buchstaben und enden mit dem Code Null (00h). Am Bildschirm erscheint vor der Kategoriekennung das Zeichen '#', das aber nicht mit gespeichert wird. (Das anwendungsspezifische Kategorienschema steht von den Daten getrennt in einer Datei *c*.CFG (Standard: A.CFG). Darin sind die Nummern der erlaubten Felder, deren Eigenschaften und auch ihre interne Reihenfolge festgelegt. Beschreibung → Anhang A.)
- Die Kategorien sind als fortlaufende Zeichenfolgen gespeichert, also ohne Zwischenräume und ohne Zeilenumbrüche.
- am Ende einer Satzgruppe, hinter dem abschließenden Code 0 des letzten Feldes, stehen die Codes 13 10 (0Dh 0Ah). Notwendig sind sie nur in Grunddateien (.*d*LG) und nur zum Sortieren.

(Mehr zu diesem Thema: siehe auch → 0.2.2)
Will man irgendwelche Daten in diese Form bringen, muß man also ein bestimmtes Kategorienschema zugrundelegen (beschrieben in einer Konfigurationsdatei *c*.CFG), in welches man die Daten überführen will. Man erstellt dann eine **Import-Parameterdatei**, welche den Dateityp .*d*IM bekommt. Dadurch bleibt immer erkennbar, vor allem auch vom Programm aus, zu welchem Schema *c* die Parameter gehören: .AIM gehört zu A.CFG, .UIM zu U.CFG usw.
Der Name der Parameterdatei ist ansonsten beliebig.

Wer selbst programmiert (in Basic, C, Pascal o.a.) und seine Daten gut kennt, kann die hier beschriebene Form auch selbst herstellen, um sich die Einarbeitung in die Importsprache zu ersparen. Wenn die Einzelheiten dann genau stimmen, kann man eine solche Datei unmittelbar indexieren oder per UPD in eine Datenbank einmischen.
Wenn man die Daten mit ***a99*** einlesen und einspeisen will, kann man auch die leichter zu erstellende Struktur des Typs .ADT wählen (→ Kap.0.2.2). ***a99*** kann so eine Datei über das Menü "Datei / Weitere Offline-Datei laden" einlesen und mit dem FLEX-Befehl **update** auch genauso einspeisen, wie das DOS-Programm UPD es machen würde.
Die Reihenfolge der Kategorien ist dabei nicht zwingend vorgeschrieben, nur die #00 muß am Anfang eines Satzes stehen. (Hierarchische Untersätze müssen mit #01,..,#06 beginnen.) Die Programme ordnen beim Einlesen dann die Kategorien innerhalb eines Satzes in der Reihenfolge, die in der CFG vorgegeben ist.

11.0.2 Inhalt einer Import-Parameterdatei

Eine **Import-Parameterdatei** enthält folgende Bestandteile (in dieser Reihenfolge):

a) eine Anzahl **Basis-Parameter**, die man auch Rand- oder Nebenbedingungen nennen könnte: Steuerzeichen von Satzanfang- und -ende sowie Feldende (dabei kann es sich um beliebige Zeichenkombinationen handeln). Diese Angaben sind vom Typ der Fremddaten abhängig. (→ 11.2.1)

b) evtl. Befehle für **globale Ersetzungen** von Zeichenfolgen, die an einem Fremdsatz vorzunehmen sind, bevor er umgewandelt wird, (→ 11.2.2.2)

c) **Zeichenumwandlungsbefehle** unterschiedlicher Art: z.B. muß man gelegentlich von EBCDIC nach ASCII umcodieren. Oder man hat etwa in MARC-Daten Codierungen für Sonderzeichen (speziell für Umlaute), die aus jeweils einem Steuerzeichen (Protyp) und einem Textzeichen bestehen, z.B. "Φu" für 'ü'. Diese Kombinationen lassen sich in die Einzelzeichen umsetzen. Andere Zeichenmanipulationen können über die Exportfunktion vorgenommen werden. Man muß die eingelesenen Daten ohnehin über eine Export-Parameterdatei wieder ausgeben lassen (siehe Kap.5), daher kann man die Umwandlungsfunktionen der Export-Schnittstelle mitbenutzen. Im allgemeinen wird man I-1.cPR als Export-Parameterdatei nehmen, um die Fremddaten im ***allegro***-Schema abzuspeichern. Dann müßte man die entsprechenden Zeichen-Umwandlungsbefehle in die Datei I-1.cPR einfügen. (→ 11.2.2)

d) eine **Liste der *allegro***-Kategorien, die aus dem vorliegenden Material gebildet werden sollen.
Diese Kategorien können in beliebiger Reihenfolge aufgeführt werden. (→ 11.2.3)
Zu jeder der Kategorien muß bzw. kann es **Befehle** geben, die dem Programm sagen:

d.1. wo der **Anfang** der betr. Kategorie im Fremddatensatz zu finden ist (dabei gibt es mehrere Modalitäten, je nach Typ); dieses macht man mit einem **Positionierungsbefehl**, (→ 11.2.3.2)
Es ist dabei auch möglich, auf eine Fremdkategorie mehrfach zurückzugreifen oder eine nochmals aufzugreifen, die bereits umgewandelt wurde oder die als Variable im Hintergrundspeicher steht.

d.2. wie das **Ende** bzw. die **Länge** des Kategorietextes zu bestimmen ist, (→ 11.2.3.3)

d.3. ob **lokale Ersetzungen** im Kategorietext durchzuführen sind. (→ 11.2.3.4)
Auch eine Umsetzung von Feldinhalten mittels **Konkordanzlisten** ist möglich.

d.4. welche sonstigen **Bearbeitungen** am Kategorietext vorzunehmen sind. (→ 11.2.3.4)
Hierbei gibt es eine Anzahl von Spezialfunktionen.

d.5. Ferner gibt es **Vergleichs-** und **Sprungbefehle**, die bei Vorliegen oder (→ 11.2.3.5/.6)
Nichtvorliegen bestimmter Bedingungen Verzweigungen im Ablauf auslösen.

d.6. Zwei Möglichkeiten gibt es für **mehrfach besetzte** Kategorien: (→ 11.2.3.6)
solche können in jeweils eigene ***allegro***-Kategorien oder gesammelt in eine einzige überführt werden, z.B. getrennt durch ';'.

d.7. **Hierarchisch gegliederte Datensätze** (→ 11.2.3.4)
können in ebensolche ***allegro***-Sätze überführt werden.

e) **Kommentare**: Jede mit Leerzeichen beginnende Zeile wird als Kommentar behandelt.
Zwei Leerzeichen hintereinander in einer Zeile bewirken, daß der Rest der Zeile ignoriert wird.

11.1 Wie macht man eine Import-Parameterdatei?

Eine Import-Parameterdatei kann in gleicher Weise wie die Export-Parameter (→ 10.1) vom ***CockPit*** aus (über **µd**) oder mit jedem Texteditor als Textdatei erstellt werden. Der Name muß von der allgemeinen Form *iname.c*IM sein, wobei *iname* ein beliebiger Name Ihrer Wahl ist, *c* der Kennbuchstabe Ihres Kategoriesystems. Standard wäre also .AIM. An der Dateikennung .cIM erkennt die Software, um was es sich handelt. So wird z.B. verhindert, daß man eine .PIM-Parameterdatei benutzt, wenn man gerade mit dem Kategoriesystem A.CFG arbeitet.
In ***a99*** kann man mit `h param` das Parameter-Hauptmenü aufrufen und dann **AIM : Import-Parameter** wählen.

Wo gehört die Parameterdatei hin? Sie darf auf dem Datenverzeichnis stehen, wo die Quelldaten untergebracht sind (d.h. die zu konvertierenden Daten; dort sucht das Programm zuerst danach), aber auch auf dem Programmverzeichnis (dort liegt sie im Normalfall), drittens auf dem Startverzeichnis, von wo aus IMPORT aufgerufen wird.

Auch für den Import kann man sich, wie beim Export, an Beispiele halten. Zur Orientierung können folgende mitgelieferten und kommentierten Parameterdateien für die einzelnen Typen dienen:

A	**BAS.AIM**	BAS-Format → A.CFG Die einzelnen Felder haben feste Positionen in Sätzen mit fester Länge.
B	**DBIII.AIM**	dBASE ASCII-Datei → A.CFG dBASE kann mit dem COPY-Befehl sog. "ASCII files" herstellen. Dabei sind dann die Datenfelder auf ihre tatsächliche Länge beschnitten und durch Begrenzungszeichen ("delimiter") eingeschlossen. Jeder Satz hat die gleiche Anzahl Felder. Unbesetzte Felder bestehen nur aus zwei Begrenzungszeichen. *Tip:* Mit dem Hilfsprogramm DB2DELIM kann man dBase-Daten vorbehandeln und die Details der Struktur herausbekommen.
C	**MEDIC.AIM**	MEDLARS download → A.CFG Ein typischer Fall für Dateien, die man aus Recherchesitzungen erhalten kann. Der Satzanfang ist an einer bestimmten Zeichenkombination erkennbar, die Felder haben festgelegte Bezeichnungen. Zeilenumbrüche innerhalb von Feldern müssen beseitigt werden.
	PICA.AIM	Pica3-Format (vom GBV und anderen Pica-Verbünden) → A.CFG
	DBCD.AIM	CD-ROM-Daten der Deutschen Bibliothek (MAB-90) → A.CFG (s. 11.3)
	MAB2.AIM	MAB2 → A.CFG. Beides MAB-Formate, aber ohne Directory.
	MABDISK.AIM	Hiermit können auch ZDB-Daten umgewandelt werden. MABDISK ist für das ältere MAB1 gedacht, und zwar die Disketten-Variante.
	Z39.AIM	Für MAB-Daten, die man aus dem virtuellen Katalog ZACK gewonnen hat.
D	**OCLC.AIM**	MARC21 → A.CFG Das MARC21-Format z.B. von OCLC ist, wie MAB1, nach ISO 2709 konstruiert, hat aber ansonsten mit MAB1 wenig gemeinsam. Diese Parameterdatei funktioniert auch mit Daten des RLIN-Verbundes und der Library of Congress.
	BIEL.AIM	Bielefeld MAB → A.CFG (nur noch historisch?) Der Bielefelder CD-ROM-Katalog produzierte als Export ein weitgehend normgerechtes MAB1-Format mit Directory.
F	**ALG.AIM**	***allegro***-Grunddatei (Typ .ALG) → A.CFG
	KAT.AIM	ASCII-Datei mit Kategorienummern → A.CFG Dateien, die bereits das richtige Kategoriesystem haben, können sehr bequem
	KAT00.AIM	importiert werden, wobei man noch beliebige Vorbearbeitungen einbauen kann. KAT00.AIM ist für den Re-Import von Exporten mit E-W.APR gedacht.

11.2 Die Import-Sprache

Konventionen für Bezeichnungen

Für die nachfolgende Befehlsübersicht gelten diese Vereinbarungen:

n	steht für eine ganze Zahl >= 0
k	steht für eine ganze Zahl zwischen 0 und 255
x	steht für ein druckbares Zeichen (Buchstabe, Ziffer, Sonderzeichen)
CS	steht für einen beliebigen Character String, wie in 10.2.0 beschrieben, wobei hier aber Strukturbefehle und Wiederholungsfaktoren nicht sinnvoll und deshalb unzulässig sind.

Wenn rechts ein <Normalwert> (engl. default) angegeben ist, wird dieser vom Programm automatisch angenommen, wenn der betreffende Parameter nicht ausdrücklich gesetzt wird.

Achten Sie auf Groß-/Kleinschreibung bei den Befehlen! Die im folgenden gezeigte Schreibweise ist verbindlich.
Zeilen in einer Import-Parameterdatei dürfen nicht länger als 240 Zeichen sein.

11.2.1 Basis-Parameter

Wie beim Export gibt es eine Anzahl von Werten, die sich auf globale Eigenschaften der Struktur beziehen. Auch hier werden diese Werte mit zweibuchstabigen Bezeichnungen benannt.
Zuerst muß das Programm wissen, wie es die Fremddatensätze einzulesen hat. Die folgenden Angaben legen den Typ (A...D) fest und teilen dem Programm die zum exakten Einlesen nötigen Werte mit.

Typ A : Feste Satzlänge

rl=n **Record Länge** <0>

Es werden jeweils n Zeichen eingelesen und dann die Kategorieliste (→ 11.2.3) durchgearbeitet. Dort muß dann zur Positionierung mit den Befehlen **j** und **l** gearbeitet werden (→ 11.2.3.2/.3).

rl > 0 bedeutet für das Programm: die Daten sind vom Typ A

fh=n Länge des **Dateikopfes** (File Header) <0>

Diese Anzahl Zeichen am Dateianfang wird übergangen. Bei Sätzen mit fester Länge kommt es vor, daß vor dem ersten Satz noch einige für ***allegro*** unwichtige Steuerdaten des jeweiligen Quellsystems stehen.
Wenn der Dateikopf keine feste Länge hat, verwendet man den Parameter rs zum Einlesen (s.u.), z.B. **rs=0 13 10 32** bei Original-dBASE-Dateien.
Verwendbar ist **fh** auch bei den anderen Typen. Im Einzelfall läßt sich mit diesem Parameter der Datenbeginn individuell einstellen. Siehe auch: **rs** (Typ C).

Typ B : Sätze mit fester Feldanzahl

Das sind die Fälle, wo die einzelnen Datenfelder zwar variabel lang sind, jeder Datensatz aber dieselbe Anzahl Felder hat. Die Bedeutung jedes Feldes ergibt sich dann aus seiner Position in der Reihenfolge der Felder. Meistens gibt es bei solchen Formaten keine Feldkennungen, an denen man sich orientieren könnte. Dann benötigt man diese zwei Parameter:

fx=n **Feldanzahl** je Datensatz <0>

Hier ist die Anzahl der Felder je Fremddatensatz anzugeben.
In der Kategorieliste kann jedes Feld dann mit seiner Nummer über den Befehl F aufgerufen werden (→ 11.2.3.2): das erste mit **F0**, das zweite mit **F1** usw..
fx > 0 bedeutet für das Programm: die Daten sind vom Typ B

fe=CS **Feldende** <0>

Das Feldende kann als beliebige Zeichenfolge angegeben werden, bis zu 25 Zeichen. Sehr häufig ist der Fall
fe=13 10 (Wagenrücklauf, Zeilenvorschub).
Die Zeichenkombination **fe** wird vom Programm beim Einlesen der Daten benutzt: es liest jeweils **fx** Zeichenfolgen, die mit **fe** enden, aus der Quelldatei und behandelt diese dann als einen Datensatz.

Hinweis: Es ist auch möglich, zusätzlich den Parameter `re` zu benutzen (→Typ C), wenn es eine Zeichenkombination gibt, die eindeutig das Satzende markiert. Der Positionierungsbefehl **F** (→ 11.2.3.2) funktioniert dann trotzdem.

dBASE kann (mit dem Befehl `copy delimited...`) ASCII-files produzieren, wobei der "Feldtrenner" wählbar ist. Alternativ kann man dBase-Dateien des Typs .DBF mit einem Hilfsprogramm namens DB2DELIM.EXE in eine ASCII-Datei mit Feldtrennern verwandeln. Wenn die Datei z.B. TITEL.DBF heißt, gibt man

```
db2delim titel 64.
```

Es entsteht TITEL.ASC. In diesem Fall wurde '@' als Feldtrenner gewählt. Die Anzahl der @ je Satz ist immer gleich, und zwar in diesem Fall 17. Jeder Satz bildet eine lange Zeile, denn nur am Satzende kommen die Codes 13 10 als Satztrennung:

```
313 134@113@fi@Eco, Umberto@@@Semiotik - Entwurf einer Theorie der Zeichen
(A Theory of Semiotics, dt.)@München@Fink@1987@(Supplemente ; 5)@10205/88@D
eskriptor@@,,F,F,@11@Kommentar@
45 361@38@@Hood, Christopher (Hg)@Schuppert, Gunnar Folke (Hg)@@Verselbstä
ndigte Verwaltungseinheiten in Westeuropa : d. Erfüllung öffentl. Aufgaben
durch Para-Government Organizations (PGOs)@Baden-Baden@Nomos Verl.-Ges.@198
8@@11638@@@,,F,F,@11@@
```

Die Parameterdatei DBIII.AIM, mit der man dieses importieren kann, beginnt so:

```
re=13 10      Am Satzende stehen die Codes 13 10
fe="@"        @ ist das Feldende
fx=17         Es werden 17 Felder je Satz, getrennt durch @, eingelesen

#20      Sachtitel
f 6      steht in Feld 6 (hinter dem sechsten '@')
N        Artikel am Anfang mit Nichtsortierzeichen markieren  ...
```

ACHTUNG: Wenn am Ende des letzten Feldes kein Feldtrenner steht, geht es nur so:

```
fx=17
fe=0
y .64 0       Setzt den Feldtrenner @ beim Einlesen schon auf 0 (deshalb klappt dann fe=0 )
y .10 256     Weg mit Code 10
y .13 0       und 13 wird dann ebenfalls Feldtrenner
```

Typ C : Variable Sätze

Bei vollständig variablen Sätzen handelt es sich entweder um die sehr elaborierten Formen des Typs D, oder aber es gibt wohldefinierte Zeichen(kombinationen), an denen Satz- und Feldende eindeutig erkennbar sind. Welche Zeichen dafür verwendet wurden, entnehme man der Dokumentation der Fremddaten oder (falls keine vorhanden) man analysiere die Struktur selbst mit üblichen Hilfsmitteln, die eine Untersuchung auf Byte-Ebene ermöglichen, zur Not mit dem MS-DOS-Befehl DEBUG (Beispiel → 11.3).

`rs=CS` **Record Start** (bis zu 25 Zeichen) <"">

wird nur benötigt, wenn am Dateianfang vor dem Beginn des ersten Datensatzes noch andere Angaben stehen können. Das ist z.B. meistens bei "download"-Daten der Fall.
rs kann auch bei den anderen Typen verwendet werden. Wenn zusätzlich fh (s.o.) gegeben wurde, liest das Programm zuerst fh Zeichen und sucht dann nach der Zeichenfolge rs.
rs wird nur zur Suche nach dem Anfang des ersten Datensatz benutzt! Der Anfang des nächsten ergibt sich automatisch aus:

`re=CS` **Record End** (bis zu 25 Zeichen) <"">

die hier angegegene Zeichenfolge CS wird benutzt, um beim Einlesen eines Datensatzes von der Fremddatei dessen Ende festzustellen. Der Lesevorgang wird dann unterbrochen und es erfolgt die Umwandlung des gelesenen Satzes.
Oft ist das Satzende nicht durch charakteristische Zeichen erkennbar, sondern der Satzanfang. Das tut nichts zur Sache, man nimmt dann die Anfangszeichen, und es funktioniert. rs und re dürfen also gleich sein.
Beim Einlesen wird hinter dieser Zeichenkombination Schluß gemacht. re selbst gehört also noch zum eingelesenen Text. Darauf muß man achten, wenn es sich in Wirklichkeit um den Satzanfang handelt.

Wenn `re` angegeben wird, liegt für das Programm der Typ C vor.
Wenn es **Untersätze** gibt, verwende man für re eine Zeichenfolge, mit der Haupt- und Untersätze getrennt eingelesen werden, damit man sie als getrennte Datensätze behandeln kann. Die Untersätze werden durch den Befehl h (→ 11.2.3.4) an den Hauptsatz gekoppelt.

>>> Achtung: wenn **`re`** nicht ganz genau stimmt, also CS gar nicht vorkommt, kommt eine Fehlermeldung
"text too long, check your 're' command"
denn das Programm findet dann kein Satzende.

Das Feldende wird bei variablen Sätzen

entweder mit dem e-Befehl individuell für jedes Feld bestimmt (→ 11.2.3.3),

oder man benutzt **`fe`** wie beim Typ B (z.B. `fe=13 10`)

oder man bringt durch eine globale Ersetzung (→ 11.2.2.2) den Code 0 an die Feldende-Positionen
z.B. **`_ 13 10`**
`_ 0` "Carriage Return/Line Feed" ersetzen durch Code 0

Es ist auch möglich, daß man die zweite oder dritte Methode anwendet, um sich nicht bei jedem Datenfeld ausdrücklich um das Ende kümmern zu müssen, daß man aber bei einzelnen Feldern den Endebefehl einsetzt, um Feldbestandteile "herauszupräparieren".
Wenn es ein einzelnes Zeichen X gibt, das immer als Feldende (und sonst nirgends) vorkommt, kann man setzen:
`y X 0`
und hat dann auch in diesem Fall den Vorteil, daß bei den einzelnen Feldern das Ende automatisch feststeht.

Typ D : Datensätze nach ISO-Norm

Nur bei Formaten, die dem ISO-Standard folgen, (nicht alle MAB- oder MARC-Daten sind so organisiert - "download"-Daten sind allerdings meistens Typ C!) muß man die folgenden Parameter verwenden. Es gibt in diesen Formaten keine Begrenzungszeichen, sondern die Längen der Sätze sowie die Kategorienumern, Längen und Positionen der Felder sind als Dezimalzahlen gespeichert, für jede Kategorie 12 Byte: 3 Byte Kategorienummer, 4 Byte Länge, 5 Byte Position innerhalb des Satzes. Man erkennt daher solche Formate sofort an einer langen Folge von Ziffern am Satzanfang.

Nur das echte Magnetbandformat MAB1 hatte ein solches Directory, das Format MAB-Diskette dagegen nicht. MAB2 kennt überhaupt kein Directory mehr. Das echte USMARC, z.B. von OCLC, hat dieses noch immer.

Manche Ausprägungen von ISO-Formaten, z.B. das Original MAB1, haben Feldtrennzeichen und **zusätzlich** Satztrennzeichen; dann können Sie sich aussuchen, ob Sie nach Typ C oder D verfahren.

`al=k` **Adressen-Länge** in Bytes <5>

Dieser Wert gibt an, wieviele Bytes die Längenangabe hat.
Für ISO-Formate ist al=5 der normale Wert.
Wenn ein Wert al>0 angegeben wird, wird vom Programm aus angenommen, daß Typ D oder E vorliegt: die ersten al Bytes werden gelesen, die Satzlänge daraus bestimmt (siehe am), und dann diese Anzahl Bytes aus der Fremddatei eingelesen.

al > 0 bedeutet für das Programm: die Daten sind vom Typ D

`am=k` **Adressen-Modus** <1>

zulässig sind nur 0 = hexadezimal, 1 = dezimal
die in al angegebene Anzahl Bytes werden entweder als hexadezimale oder dezimale Zahl interpretiert. Nur bei einem Format „Bibliofile“ gab es mal `am=0`, bei ISO-Formaten isr stets `am=1`, z.B. bei den LC- und OCLC-MARC21-Daten.

`fd=n` **field directory position** : Anfangsposition des **Directory** <24>

innerhalb des Datensatzes. Bei MAB1 war die Position fd=192,
bei MARC ist es i.d.R. `fd=24` (vgl. OCLC.AIM).
Dieser Wert muß exakt stimmen, sonst geht nichts. Die Befehle D und T (s.u.) benutzen den Wert `fd` als Anfangspunkt für die Suche nach einer Kategorienummer.

Wichtig: Die einzelnen Felder werden bei diesen Formaten mit den Positionierungsbefehlen `D und d` (und nur für "Bibliofile" [obsolet] mit `T und t`) angesprochen (→ 11.2.3.2). Die Positionierungsbefehle **`s`** und **`g`** können nur ausnahmsweise benutzt werden, wenn Inhalte mit Zeichenkombinationen eingegrenzt sind (→ 11.2.2.2). Bei den echten MARC-Formaten stehen die Kategorienummern nicht vor den Feldinhalten, sondern nur im Directory! Längst antiquiert, aber LC-Daten sind immer noch so, DNB-Daten dagegen sind keine ISO-Daten mehr mit dieser Besonderheit.

Hilfswerte für die variablen Satzstrukturen

Die folgenden Werte erleichtern in bestimmten Situationen die Verarbeitung.

`fs=k` **Feld-Start-Position** <0>

Wenn der Feldinhalt nicht unmittelbar hinter der Feldkennung anfängt, sondern mehrere Positionen dahinter, kann man die Anzahl der hinter der Kennung zu übergehenden Bytes in **`fs`** angeben.
Man müßte sonst bei jedem Feld den Befehl **`}`** benutzen (→ 11.2.3.3), um die Position des Textzeigers zu verschieben.
Bei Bibliofile gilt z.B. `fs=5`.

`fc=k` **Feldlängen-Korrektur** <0>

Die Länge des Datenfeldes bestimmt sich normalerweise aus dem Parameter fe (Typ B und C) oder automatisch durch den Endecode 0 oder den Befehl l (Typen A - C).
Bei Formaten wie Bibliofile, die sich am ISO-Standard orientieren, ergibt sich aber die Feldlänge (wie die Satzlänge) aus binär codierten Angaben hinter der Feldkennung. In der Länge ist dann i.a. die gesamte Länge einschließlich eventueller Indikatoren und sonstiger Angaben am Anfang des Feldes enthalten. Dann wird die mit fc angegebene Zahl von der ermittelten Länge abgezogen.

`fm=x` **Mehrfachfeld-Kennzeichnung** <ú>

Das hier angegebene Zeichen wird als Folgezeichen benutzt, um die ***allegro***-Kategorienummer eines Mehrfachfeldes zu ergänzen wenn das betr. Feld tatsächlich mehrfach vorkommt.
Siehe dazu unten den Befehl **`m`** in der Kategorieliste (→ 11.2.3.6).
Der Code ú bedeutet, daß IMPORT die Mehrfachcodes aus der Angabe **`▼M`** in der .CFG entnehmen soll. Wenn dort also z.B. `#90"Signatur"▼Mabcdefghijk` steht, erhält die erste #90 das Folgezeichen 'a', die zweite 'b' usw. Wenn 'k' überschritten wird, fallen weitere Exemplare von #90 alldings weg.
Ist dagegen `fm=2` gesetzt, würde IMPORT die Felder #902, #903 usw. bilden. Sind '2' und '3' etc. nicht in der Angabe **`▼M`** für #90 enthalten, entfallen sie.

`fr=k` **Feld-Repetierfaktor** <1>

Wenn z.B. das Feld "Schlagwort" mehr als zweimal vorkommt, und man hat das erste in #30 umgesetzt, dann wird das zweite mit Hilfe von fm in #301 überführt, das dritte ergibt sich durch Addition von fr, also erhält man #302, wenn fr=1 ist.
fr gilt global; wenn man bei einem bestimmten Feld eine andere Regelung wünscht, kann man den Faktor beim Befehl **`m`** (→ 11.2.3.6) angeben.

11.2.2 Globale Konvertierungsbefehle

Das Importprogramm kann 1:1-Zeichenersetzungen vornehmen, Doppelkombinationen aus einem Protyp und einem Grundzeichen, sowie einen globalen Austausch ganzer Zeichenfolgen ausführen. Weitere Umwandlungen kann man über die Exportfunktionen (s.o., Einbau in die Datei I-1.cPR) nachschalten. Die Befehle lauten:

11.2.2.1 Zeichenumwandlungen (Umcodierungen)

`y x k`
`y .m k`

Wirkung: bereits beim Einlesen wird das Zeichen x durch den Code k (dezimal anzugeben) ersetzt bzw. der Code m durch k (beides dezimal angegeben). Notwendig ist dies z.B. beim Einlesen von EBCDIC-Daten. Der Befehl y bietet ansonsten dieselben Möglichkeiten wie der Befehl p bei den Export-Parametern (s. Kap.10.2.4), bis auf die Ersetzung von Einzelzeichen durch Zeichenfolgen.
Sonderfall: k = 256 : Zeichen x soll beim Einlesen übergangen werden, also entfallen.

Die **Ausführung** erfolgt schon direkt beim Einlesen, vor allen anderen Aktionen.
Empfehlung: Prophylaktisch setze man

`y .1 32`	Code 1 soll durch Leerzeichen ersetzt werden, `desgl. 2 - 9`
`[usw. bis]`	Die umgewandelten Daten können sonst Probleme machen.
`y .9 32`	Mit 256 statt 32 kann man sie auch ersatzlos beseitigen!

Damit werden die Codes 1 bis 9 im Fremdtext durch Leerzeichen ersetzt, wenn sie vorkommen.

11.2.2.2 Globale Ersetzungen

Sehr häufig ist es sinnvoll, vor Beginn der Umwandlung im zu bearbeitenden Datensatz Ersetzungen bestimmter Zeichenfolgen vorzunehmen. Dafür gibt es den "globalen" Ersetzungsbefehl. ("Global" deshalb, weil er sich auf den gesamten Datensatz bezieht. Ersetzungen in einzelnen Feldern sind bei dem das Feld betreffenden Abschnitt anzugeben (→ 11.2.3.4 "Lokale Ersetzungen"); Ersetzungen auf Zeichenebene, d.h. Ersetzung einzelner Zeichen durch andere, können über die Export-Schnittstelle behandelt werden , siehe dazu Kap.10.2.4.
Der Befehl hat immer zwei Teile:

`_CS1` CS1 ist die zu ersetzende Zeichenfolge,
`_CS2` CS2 die dafür einzusetzende.

Jede dieser Zeichenfolgen kann aus einer Mischung von Steuer- und Textzeichen bestehen.
Beispiel:

```
_ 13 10 "    "
_ " "
```

Hierdurch wird die Folge "Wagenrücklauf Zeilenvorschub 5 Leerzeichen" ersetzt durch ein Leerzeichen. Solche Ersetzungen sind oft bei "download"-Daten nützlich um einen Zeilenumbruch zu beseitigen, der in ***allegro***-Daten stören würde. Zu beachten ist, daß der Befehl hier aus zwei Zeilen bestehen muß, anders als bei den Export-Parametern! Dies ist nötig, weil hier in CS1 und CS2 **alle** Codes, also auch Steuerzeichen, der Code 0, und das Zeichen '_' selbst enthalten sein können.
Die **Ausführung** erfolgt nach dem Einlesen, wenn die y-Befehle bereits erledigt sind. D.h. man muß die durch y umgewandelten Zeichen in den CS-Ketten schon in der umgewandelten Form berücksichtigen. Diese Art Ersetzungsbefehl kann aber auch innerhalb einzelner Kategorieabschnitte ("Paragraphen") zum Einsatz kommen und wirkt dann nur auf die betreffende Kategorie als "Lokale Ersetzung" (→ 11.2.3.4). Alle global gemeinten Ersetzungsbefehle müssen in der Datei oberhalb der Kategorieliste stehen.

Einschränkung: in echten ISO-Formaten (`al>0`, z.B. MARC21) müssen CS1 und CS2 bei **globalen** Ersetzungen gleich lang sein, weil sonst die Adressen nicht mehr stimmen; bei **lokalen** Ersetzungen funktionieren dagegen beliebige Längen.

?=X **Maskierungszeichen umdefinieren** (engl. "wildcard") <?>

Es kommt häufig vor, daß in Ersetzungsbefehlen oder in Positionierungsbefehlen (→ 11.2.3.2) bestimmte Zeichen variieren können, also nicht festliegen. Diese Zeichen muß man in der Suchzeichenfolge CS1 oder CS "maskieren" können.
Standardmäßig wird '?' als Maskierungszeichen benutzt. Kommt es in einer Suchzeichenfolge CS selber als echtes Zeichen vor, dann muß man ein anderes Zeichen wählen. Man gibt dazu diesen Befehl und setzt für 'X' ein Zeichen ein, das in keinem Ersetzungsbefehl vorkommen wird, denn es gilt dann für den gesamten Import und kann nicht zwischendurch geändert werden.
Für lokale Ersetzungen gilt das Zeichen auch (→ 11.2.3.4).

11.2.2.3 Protyp-Ersetzungen

Doppelkombinationen aus einem "Protyp"-Zeichen und einem Ersatzzeichen können durch Einzelzeichen ersetzt werden. Die Befehlsform ist:

`p C abcde vwxyz`

wobei folgendes gilt:

C	Protypzeichen
abcde	Zeichen, die hinter dem Protyp vorkommen können
vwxyz	Zeichen, die als Ersatz eingesetzt werden sollen (gleiche Anzahl wie in abcde)

Ergebnis: **Ca** wird durch **v** ersetzt, **Cb** durch **w** usw.
Beispiele sind in manchen MARC-Daten zu finden; z.B. für das Protypzeichen Φ:
`p Φ aoueiyAOU äöüëï_ÄÖÜ` also Φa durch ä ersetzen usw.
Achtung: Für jedes Zeichen C nur eine solche Zeile!

Ausführung: Diese Ersetzungen werden erst im Anschluß an alle Vorbearbeitungsbefehle der Kategorieliste (→ 11.2.3) ausgeführt, d.h. für die dort benutzten Such- und Ersetzungsbefehle gelten die Protypzeichen noch.

11.2.2.4 Sonderersetzungen

Für Spezialfälle wurden folgende Befehle geschaffen, die alle drei global wirken:

P Punkt direkt vor Buchstaben ersetzen durch "**.** " (Punkt Leerzeichen)

S Mehrfach-Leerzeichen beseitigen

$$ sog. "Dollar-Verweisungen" auflösen. In alten PICA-Daten oder auch SWD-Daten (dort "Montagebefehle" genannt) kamen solche Dinge vor. Hinter einem Dollarzeichen steht eine Kategorienummer. Wurde dieser (global wirkende) Befehl gegeben, ersetzt das Programm alle solchen Verweise durch den zugehörigen Kategorietext, löst also die Verweise auf. Das erste Dollarzeichen ist der Befehl, das zweite das tatsächlich in den Fremddaten benutzte Zeichen.
Wenn z.B. in einem Kategorietext irgendwo "$3000" vorkommt, wird dies durch den Inhalt der Fremdkategorie 3000 ersetzt. Voraussetzung: Das Feldende der Fremdkategorien muß durch den Code 0 markiert sein. (Normalerweise erreicht man das durch eine globale Ersetzung.)

11.2.3 Die Kategorieliste - das Kernprogramm

Den Hauptteil einer Import-Parameterdatei bildet die **Liste der Kategorien**. Diese stellt sozusagen ein Programm dar, geschrieben in einer speziellen Sprache, eben der Importsprache.

Die Kategorieliste besteht aus **Paragraphen**.
Jeder Paragraph enthält, in dieser Reihenfolge, (*Achtung:* 1. und 2. dürfen nicht fehlen)

1. eine **Kategorie-Angabe** = Angabe einer erlaubten ***allegro***-Kategorie (→ 11.2.3.1)
2. einen **Positionierungsbefehl,** um den Kategorie-Anfang festzulegen, (→ 11.2.3.2)
3. evtl. eine oder mehrere **Aktionen**: (→ 11.2.3.3-6)
   - Befehle zur Präzisierung von **Textanfang** und **Textlänge** (Kategorie-Ende)
   - **Vorbearbeitungs-**, **Vergleichs**- und **Sprungbefehle.**
4. evtl. weitere Positionierungsbefehle plus zugehörige Aktionen, wenn die Kategorie aus mehreren Fremdfeldern zusammengesetzt werden soll; d.h. 2. und 3. können sich innerhalb eines Paragraphen wiederholen

Die Paragraphen müssen nicht in der richtigen Reihenfolge der ***allegro***-Kategorien aufeinander folgen, die stellt das Programm selbst her: die vom Paragraphen gelieferte, umgewandelte Kategorie wird sofort im Arbeitsspeicher in den aktuellen Datensatz eingeordnet.
Alle genannten Elemente müssen, anders als beim Export, jeweils auf einer **eigenen Zeile** stehen!
Überall können, wie beim Export, Leerzeilen und mit Doppelleerzeichen beginnende Kommentare vorkommen.

11.2.3.1 Kategorie-Angabe

`#xxf` Angabe einer ***allegro***-Kategorie

Die Kategorie-Angabe sagt dem Programm: "importiere das, was in den folgenden Zeilen beschrieben wird, in die Kategorie #xxf".

f kann entfallen, dann wird dafür ein Leerzeichen angenommen.

Bei einem dreistelligen Kategorienschema wie U.CFG muß es `#xxxf` heißen.

Wenn f das Zeichen '.' ist, wird der anschließend aufbereitete Text unmittelbar hinter #xx geschrieben, sonst wird bei fehlendem f, wie gesagt, ein Leerzeichen hinter #xx gesetzt.
#xxf ist zugleich der Startbefehl eines Paragraphen *und* das Ende des vorangehenden Paragraphen. Es muß keine Leerzeile zwischen zwei Paragraphen stehen, jedoch ist dies übersichtlicher. Wenn das Programm bei der Abarbeitung der Kategorieliste auf ein '#' am Anfang der nächsten Zeile stößt, führt es noch die Protyp-Ersetzungen durch (11.2.2.3), schreibt den zu diesem Zeitpunkt ermittelten und bearbeiteten Textteil in die Kategorie #xxf und ordnet diese ein zwischen die anderen schon im Arbeitsspeicher stehenden, vorher importierten Kategorien. Deshalb kommt es auf die Reihenfolge der Paragraphen nicht an. Wenn allerdings zwei Paragraphen mit gleicher Kategorie-Angabe beginnen, bleibt nur die zuletzt importierte übrig. Es mag Fälle geben, wo man dies nutzen kann.
Sonderfälle: Wenn **f** das Zeichen **~** ist, wird die nächstfolgende Wiederholungskennung benutzt, wie es sich aus der Deskriptorzeile in der CFG ergibt (→ Anh.A.1.2), also z.B. #402, wenn #40 schon besetzt ist, #403, wenn #402 schon existiert, usw.
Wenn **`#xxf`** keine erlaubte Kategorie ist, wird das Ergebnis des Paragraphen nicht gespeichert. Dies läßt sich als Trick ausnutzen, wenn ein Paragraph nur Prüfungen und Sprünge machen soll.
Wenn **`#xxf`** = `#00` oder `#01` ist, dann wird ein neuer Datensatz bzw. Untersatz angefangen; d.h. #00 muß normalerweise den ersten Paragraphen der Kategorieliste bilden.

**Pauschal-Import**

Wenn Daten bereits mit den richtigen Kategorienummern vorliegen (Typ F), vereinfacht sich das Importieren ganz erheblich. Man braucht dann nämlich nicht jede Kategorie einzeln explizit mit einem eigenen Importbefehl herüberzuholen, sondern der simple Befehl **##** transportiert den gesamten Datensatz vom Einlesebereich in den ***allegro***-Arbeitsspeicher. Dabei ordnet das Programm die Kategorien, wenn nötig, in der Reihenfolge, die in der .CFG-Datei vorgegeben ist.
Ein Trick: Wenn bestimmte Kategorien nicht mit importiert werden sollen, nehme man vorher deren Nummern aus der Kategorieliste der .CFG heraus, das Programm nimmt sie dann nicht.
Die folgenden Voraussetzungen müssen **alle** gegeben sein, damit der Befehl anwendbar ist:

- Es gibt eine eindeutige Trennung zwischen den Datensätzen, die mit dem Befehl **re** anzugeben ist (es handelt sich also nicht um ein ISO-Format)
- der Startpunkt des ersten Datensatzes der Datei (die erste Ziffer der ersten Kategorienummer) muß mit dem Befehl **fh** oder **rs** ermittelbar sein,
- die Kategorien besitzen eine eindeutige Feldendekennung, die mit **fe** anzugeben ist,
- die Einzelkategorien müssen als fortlaufende Zeichenfolgen vorliegen. Eventuell vorhandene Zeilenumbrüche müssen durch einen globalen Ersetzungsbefehl beseitigt werden
- jede Kategorie beginnt mit einer gültigen Kategorienummer (die in der .CFG vorkommt); es kann, aber muß nicht, ein '#' vor der Kategorienummer stehen. Kategorien, die sich wiederholen, erhalten automatisch eine Mehrfachkennung gemäß CFG (z.B. #402, #403, ...)

Ab Version 13: Auf den Befehl **##** kann ein Paragraph mit mehreren Befehlen folgen, der dann für jede übernommene Kategorie ausgeführt wird. So kann man eine pauschale Vorbehandlung jeder einzelnen Kategorie vornehmen.
Anschließend können noch weitere Paragraphen für einzelne Kategorien folgen, wobei man auch auf die bereits umgewandelten Kategorien zurückgreifen kann (mit dem **k**-Befehl, s. dort).
Wenn das Ausgangsformat USMARC u.ä. ist: **M "???"** statt **##** (siehe 11.2.3.2)

Zu den folgenden Sonderfällen werden die Import-Parameterdateien mitgeliefert, die man nicht nur für A.CFG verwenden kann:

.ALG-Dateien (siehe mitgelieferte Datei ALG.AIM)

`fh=1`	das erste Zeichen der Datei muß übergangen werden
`re=0 13 10 1`	eindeutige Satzende-Zeichenfolge
`y .2 256`	die Codes 02...06 sind beim Einlesen auszublenden
`...`	
`y .6 256`	(das Importprogramm erkennt an #01 ... #05, daß es sich um Untersätze handelt)
`##`	**Pauschal-Import**

ASCII-Dateien erzeugt durch Export mit E-1.APR (siehe KAT.AIM, KAT00.AIM, → 0.2.2)
Wenn mehrfach besetzte Kategorien keine Folgeziffern haben, werden automatisch welche vergeben, beginnend mit **fm** (→ 11.2.1). Hier ist angenommen, daß lange Kategorien einen Zeilenumbruch aufweisen (Unterbrechung nur bei Leerzeichen), wobei Fortsetzungszeilen mit 4 Leerzeichen beginnen.

`re=13 10 13 10`	Satzende: 2 Zeilenvorschübe
`fe=13 10`	Feldende: ein Zeilenvorschub
`_ 13 10 "  "`	Hiermit werden die beschriebenen
`_ " "`	Umbrüche langer Kategorien vorab beseitigt
`##`	**Pauschal-Import**

Wozu aber, wird man vielleicht fragen, sollen .ALG- oder .ALD-Dateien importiert werden, denn sie haben doch schon die richtige Struktur? Der Vorteil liegt in der Möglichkeit, globale Veränderungen am Datenbestand vornehmen zu können. Zu der Grundstruktur, die in den Parameterdateien ALG.AIM etc. vorgegeben ist, können noch weitere Befehle nach Bedarf hinzukommen. Und wenn auf den Befehl **##** noch weitere Import-Paragraphen folgen, dann ersetzen die so definierten Kategorien die in dem Datensatz eventuell schon vorhandenen Kategorien.

Wir geben hier Beispiele für einige exemplarisch wichtige Fälle:
(zu den Befehlen **s** und **j** kommen wir im Abschnitt 11.2.3.2)
Man nimmt für die folgenden 4 Aufgaben die **ALG.AIM** und macht daran jeweils die beschriebenen Änderungen. Am leichtesten geht es über ***CockPit*** mit der Funktion
µd <Import-Param> <neue Version erstellen>.

1. **Teilung oder Zusammenlegung** bestimmter Kategorien
   wenn z.B. die Kategorie #85 in der Form "Serientitel ; Zählung" erfaßt ist, und man möchte beide Elemente aufteilen auf #85 und #85z, so würde man schreiben:

```
##              Pauschalimport
#85             Kategorie #85 anlegen (ersetzt dann die schon vorhandene!)
s 0 "85 "       suche diese Kombination im Quelltext (Code 0 = Feldende des vorangehenden Feldes!)
e " ; "         beende den Text bei " ; "
#85z  beginne #85z (als neue Kategorie für die Zählung)
s 0 "85 "       gehe im Quelltext zur selben Position wie vorher
b " ; "         aber nun beginne hinter " ; "
```

   Entsprechendes könnte man sich für "#20 Sachtitel : Zusatz" vorstellen.
   Das Umgekehrte: man hat #85 und #85z vorliegen und möchte beide zu #85 zusammenfügen mit " ; " als Trennung:

```
_ 0 "85z"       vor dem Pauschalimport wird #85z "unleserlich" gemacht, indem man 85 durch
_ 0 "xyz"       xyz ersetzt, was keine nach A.CFG zulässige Nummer ist.
##    Pauschalimport
#85             beginne Kategorie #85
s 0 "85 "
W " ; "         Spatium-Semikolon-Spatium dahintersetzen
g 0 "xyz"       Im Quelltext ist Inhalt von #85z hinter dieser Zeichenfolge zu finden
```

2. **Änderung von Kategorienummern**
   wenn beispielsweise aus #84 die Kategorie #70 gemacht werden soll:

```
_ 0 "84 "       unmittelbar vor der Kategorienummer steht in der .ALG-
_ 0 "70 "       Datei der Code 0, daher ist dieser Ersetzungsbefehl eindeutig
##              alles importieren, dabei werden die Kategorien auch neu sortiert
```

3. **Löschen einer schon importierten Kategorie**
   Gelegentlich erstellt man eine Kategorie für Hilfszwecke und will sie hinterher nicht mit abspeichern. Nach folgendem Rezept kann man sie aus dem aktuellen Satz wieder löschen: Wenn #99 eine solche Kategorie ist, schreibt man diesen Paragraphen hin:

```
#99             Tue so, als ob erneut die #99 erstellt werden soll,
k99             Benutze die vorhandene #99 als Arbeitstext
w" " setze ein Leerzeichen davor
l 0             und übernehme Null Zeichen aus der alten #99. Dann wird sie gelöscht.
```

4. **Erzeugung neuer Kategorien**

Nehmen wir an, eine neue Kategorie "#3g Gattung" soll eingerichtet werden. Zunächst muß man sie in A.CFG als erlaubte Kategorie eintragen (→ Anh.A). Wenn irgendwo im Fremdsatz die Zeichenfolge "Biograph" oder "biograph" vorkommt, soll in "#3g BIO" entstehen, wenn irgendwo "Tagung", "Kongr" oder "Konferenz" vorkommt, soll "#3g CONF" entstehen etc. Man könnte schreiben:

```
#3g             lege Kategorie #3g an
j0              Positionierung (nur pro forma, weil jeder Paragraph mit einem Positionierungsbefehl
                anfangen muß, → 11.2.3.2)
C"Biograph"
w "BIO" 0
C"biograph"
w "BIO" 0
C"Konferenz"
w "CONF" 0
C"Kongre"
w "CONF" 0
C"Conferenc"
w "CONF" 0
C"Tagung"
w "CONF" 0
```

Ohne die '0' wäre kein Endezeichen da und es stünden dann unvorhersehbare Zeichen hinter den eingesetzten Zeichen.

5. **Umwandlung über Konkordanzliste** (→ 11.2.3.4)

Dieses Konzept läßt sich bei allen normierten Datenelementen anwenden, deren Inhalt von einer festgelegten Liste bestimmt wird. Als Beispiel seien Sprach- oder Ländercodes genannt.
Man könnte schreiben, wenn #37c den Ländercode enthalten soll:

```
...
_ 0 "37c"
_ 0 "xyz"       Kat. #37c ungültig machen, damit sie beim Pauschalimport nicht durchkommt
...
#37c
s 0 "xyz"
@A
##

@@A             Konkordanzliste beginnt: hier Umsetzung für Ländercodes
"de"            de wird ersetzt durch GER
"GER"
"us"            us wird ersetzt durch USA
"USA"
....
@               Ende der Konkordanzliste
```

11.2.3.2 Positionierung : Ermittlung des Textbeginns

Der **erste Befehl** eines Paragraphen, auf der ersten Zeile gleich nach der Kategorie-Angabe, muß ein **Positionierungsbefehl** sein. Damit wird der sog. **Textzeiger** positioniert (d.h. der Anfang des **Arbeitstextes** festgelegt), und zwar auf den Anfang der gewünschten Kategorie im Fremdtext. Anschaulich sollte man sich einen Zeiger vorstellen, der vom Positionierungsbefehl genau auf das erste Zeichen des gesuchten Textteils gesetzt wird. Mit den nachfolgenden Befehlen kann dieser Textzeiger unter Umständen noch verschoben werden, z.B. auf den Beginn eines Teilfeldes. Die **Textlänge** wird zunächst automatisch bestimmt: das Fremdfeld endet bei dem Code 0 bzw. bei der Zeichenkombination **`fe`**, falls diese definiert ist (→ 11.2.1). Die Textlänge läßt sich außerdem noch durch einige Befehle verändern wie z.B. **`l n`** oder **`e CS`**.

Es gibt die folgenden Positionierungsbefehle, die zum Teil nur bei bestimmten Dateitypen sinnvoll sind:

`s CS` **suche** die Zeichenfolge CS im gesamten Fremdsatz

`S CS` (anwendbar bei allen Typen, allerdings bei A und D meist nicht sinnvoll)

Im Text des eingelesenen Satzes wird die Zeichenfolge CS gesucht. Kommt CS nicht vor, so wird der Rest des Paragraphen übergangen (d.h. #xxf entfällt) und mit dem nächsten Paragraphen fortgefahren. Wenn CS vorkommt, wird der Beginn des Kategorietextes für #xxf unmittelbar hinter dem letzten Zeichen von CS angenommen. Man sagt auch, der **Textzeiger** wird auf das erste Zeichen hinter CS gestellt.

Die Zeichenfolge CS (→ 10.2.0) kann Steuerzeichen und Text enthalten, sie kann auch z.B. so aussehen: **`s 0 "xyz"`** oder **`s 13 10 "xyz"`** (was nicht selten sehr nützlich ist).

Das große S bewirkt, daß beim Suchen der Unterschied von Groß- und Kleinbuchstaben nicht beachtet wird (bis auf Umlaute!). In CS müssen dann ausschließlich Kleinbuchstaben stehen.

Wenn der Wert **`fs`** größer als Null ist, wird der Textzeiger noch um **`fs`** Stellen vorgeschoben (→ 11.2.1).

Wenn keine weiteren Befehle kommen, d.h. der Paragraph nur diesen Positionierungsbefehl enthält, wird der Text in die Kategorie #xxf überführt und der nächste Paragraph bearbeitet. Die Textlänge ergibt sich automatisch aus der Zeichenfolge **`fe`**, die von der Position des Textzeigers aus gesucht wird und selbst nicht mit zum Text gehört.

Wenn **`fe`** fehlt, wird der Code 0 als Feldende benutzt!

Wichtig:

`s` und die anderen Positionierungsbefehle können in einem Paragraphen mehrfach vorkommen. Dann wird jeweils der bereits ermittelte Textabschnitt, der aktuelle Arbeitstext, in einen **Transferbereich** übertragen (hinten angehängt). Dieser Transferbereich ist also die noch nicht fertige Kategorie **`#xxf`**. Der neue Positionierungsbefehl legt dann einen neuen Arbeitstext an, und nachfolgende Manipulationen beziehen sich nur auf diesen, nicht auf den Transferbereich. (Vgl. dazu in 11.2.3.4 die Beschreibungen der Befehle **`w, W, Kxy`** und **`Yxy`**. Also kann man ***allegro***-Kategorien aus mehreren Bestandteilen eines Fremdsatzes zusammensetzen, indem man in einem Paragraphen mehrere Positionierungen macht.

Vorsicht: Man **vermeide** in `CS` die Form `s "xyz?"` , also '?' am Ende von CS.
Statt dessen schreibe man `s "xyz"` und dann `} 1` in der nächsten Zeile.

Hinweis: Soll nur das **Vorkommen** von CS geprüft werden: siehe → 11.2.3.5, C und Z.

`g CS` **"go to"** : suche CS vom aktuellen Textzeiger aus.

`G CS` D.h., die Suche nach CS geht nicht vom Anfang des Fremdtextes aus, sondern von der Stelle, die mit dem letzten s-Befehl gefunden wurde. (Natürlich kann dieses CS ein anderes sein als beim s-Befehl.) Dadurch ist es möglich, eine nachfolgend auftretende Mehrfachkategorie gezielt auszuwerten. Alternativ könnte man im Anschluß an den s-Befehl auch den Befehl x geben (→ 11.2.3.4), um das Feld im Fremdtext ungültig zu machen. Dann ist 's' statt 'g' benutzbar.
Für das große G gilt entsprechend dasselbe wie oben für das S.

`F n` **Feld n auswählen** (n = 0, 1, ..., fx)

Nur bei Formaten mit fester Feldanzahl je Satz anwendbar (Typ B, fx>0). Der Textzeiger wird auf den Anfang des Feldes Nummer n gestellt (Leerzeichen hinter F !) Die Zählung beginnt mit 0 für das erste Feld, d.h. der Befehl **`F 1`** positioniert dann den Zeiger auf den Anfang des zweiten Feldes!

D "nnn" **sucht im ISO-Directory die Feldnummer nnn** (immer dreistellig!).
Der Textanfang und seine Länge werden aus der Directory-Eintragung entnommen. Anwendbar nur bei Typ D.

D0 "nnn" macht dasselbe, jedoch wird anschließend die Kategorie "nnn" im Directory ungültig gemacht, kann also bei nachfolgenden Positionierungsbefehlen nicht erneut gefunden werden. Das kann wichtig sein, wenn Felder mehrfach besetzt sein können.

d "nnn" wirkt wie D, gesucht wird jedoch von der Position ab, wo der letzte D-Befehl endete. Dadurch ist es möglich, eine nachfolgend auftretende Mehrfachkategorie gezielt auszuwerten. Wenn aber vorher **D0** benutzt wurde, bewirkt **D "nnn"** dasselbe.

D "???" sucht nach der ersten noch nicht ungültig gemachten Kategorie. (Geht nur bei ISO-Formaten!)
Wenn man z.B. am Ende der Kategorieliste schreibt:

```
#99a
D0 "???"
m
```

dann werden alle nicht ausgewerteten Kategorien unter #99a, #99b ... aufgeführt, und zwar wird ihre Originalnummer in Klammern dem Text vorangestellt. Man kann dadurch genau analysieren, was die Fremdsätze so alles an noch nicht berücksichtigten (vielleicht sogar nicht dokumentierten oder fehlerhaft erfaßten) Dingen enthalten.

M "???" (Pauschalimport für MARC-Formate (Typ D)) : macht dasselbe, verwendet aber im Importsatz dieselbe Kategorie wie im Fremdsatz.

j n **Sprung (jump)** auf die Position n (n=0 : erstes Byte des Fremdsatzes)
Besonders bei Formaten mit fester Satzlänge (Typ A), die dann in der Regel auch feste Feldlängen haben, bestimmt sich ein Feld durch seine Position relativ zum Satzanfang. Mit **j 0** springt man auf das erste Zeichen des Satzes.
Dieser Sprungbefehl ist auch geeignet, wenn aus einem Feld fester Länge am Anfang eines Satzes mit variabler Länge ein bestimmter Abschnitt gebraucht wird, z.B. die ISBN oder eine Jahres-, Sprach- oder Länderangabe, wie bei den MARC-Formaten.
Die Feldlänge muß anschließend mit einem der Textausschnitt-Befehle bestimmt werden (meistens wohl mit dem Befehl 'l', → 11.2.3.3)

kxxf Kategorie #xxf des aktuellen Satzes als Arbeitstext benutzen. Der Textzeiger wird statt auf eine Position im Fremdsatz auf die Kategorie #xxf des aktuellen, gerade in Bearbeitung befindlichen Satzes gesetzt. #xxf kann jede zulässige Kategorie sein, aber auch eine der Sonderkategorien (z.B. #dts) sowie eine Anwendervariable `#uxy` (→ 11.2.3.4). Dann müßte man `kdts` bzw. `kuxy` schreiben. Abbruch des Paragr., wenn `#kkf` nicht vorhanden.

Spezialbefehl für Typ "Bibliofile": [obsolet, das Produkt existiert nicht mehr]

T k m **suche das "tag" k*256 + m**
Im "Bibliofile"-Format sind die Feldnummern (dort "tags" genannt) binär in 2 Byte verschlüsselt. Um das Feld 245 = Sachtitel zu finden, muß man bei "Bibliofile" folglich T 0 245 geben, für Feld 700 = Personenname für Nebeneintragung: T 2 188 (denn 2*256+188 = 700). Das Programm bestimmt die Länge dann aus den zwei Bytes, die hinter dem tag stehen, addiert ks auf den Textzeiger und subtrahiert fc von der Länge. Diese etwas komplizierte Mechanik war bislang nur beim "Bibliofile"-Format nötig, alle anderen sind etwas einfacher.

t k m suche "tag" k*256 + m von der vorigen Position aus
Ähnlich wie 'g' im Verhältnis zu 's' sucht dieser Befehl nicht vom Anfang, sondern von der mit dem letzten T gefundenen Stelle.

11.2.3.3 Textausschnitt bestimmen = Kategorie-Beginn und Ende festlegen

Nichts zu tun braucht man, wenn am Feldende der Code 0 steht oder mit den Positionierungsbefehlen D, T oder F gearbeitet wird und der gesamte Text des so gefundenen Fremdfeldes übernommen werden soll. Daher ist es günstig und wird empfohlen, durch einen globalen Ersetzungsbefehl die Feldenden in den Code 0 zu verwandeln (→ 11.2.2.2).
Es kommt jedoch oft vor, daß man nur einen Teil des Textes in eine Kategorie überführen will. Dafür gibt es folgende Möglichkeiten, die kombinierbar sind, d.h. es können mehrere Befehle aufeinander folgen (jeweils auf neuer Zeile!).
Durch Kombination (evtl. Wiederholung) der folgenden Befehle lassen sich beliebige Teile eines Textfeldes isolieren:

`e CS` **Textende festlegen** (In CS dürfen, auch bei den folgenden Befehlen, Maskierungszeichen vorkommen)
Das Ende des Textes ist das letzte Zeichen vor der Zeichenfolge CS.
Achtung: **`e 20`** bedeutet: Ende bei Code 20, nicht: Ende nach 20 Zeichen (wie beim Export, vgl. 10.2.6.3) Für letzteres muß man hier **`l 20`** geben, siehe unten.

`b CS` **Textbeginn versetzen**
`B CS` Der Textanfang wird innerhalb des ermittelten Feldes auf die **Position hinter CS** gesetzt. Wenn CS nicht im Fremdfeld vorkommt, wird der Paragraph ergebnislos beendet (bei 'b') bzw. das Fremdfeld wird vollständig übernommen (bei 'B'), d.h. der Befehl B CS hat dann keine Auswirkung auf die Feldbegrenzung; der Paragraph wird weiter abgearbeitet.
Wenn man zweimal, in zwei Zeilen untereinander, denselben Befehl **`B CS`** gibt, wird der Textzeiger hinter das zweite **`CS`** im Fremdfeld gesetzt, usw.
Achtung: **`b 20`** bedeutet: Beginn bei Code 20, nicht: Beginn auf Position 20 (wie beim Export, vgl. 10.2.6.3). Für letzteres muß man hier **`} 20`** geben, siehe unten.

`< CS` **Vom Ende her suchen** : CS wird im aktuellen Fremdfeld von hinten gesucht. Wenn es gefunden wird, geht der Textzeiger hinter CS, sonst bleibt er unverändert. Sinnvoll ist das, wenn CS in der Fremdkategorie mehrfach vorkommen kann, wie z.B. Interpunktionszeichen, und man den Teil braucht, der hinter dem letzten Code steht.

`{ k` Textzeiger k Zeichen **nach links bzw. rechts** bewegen.
`} k` Die Länge wird dabei automatisch so angepaßt, daß das vorher ermittelte Ende nicht verschoben wird (0 < k < 256). Wenn das Fremdfeld bei } kürzer als k ist, entsteht nichts.

`l n` Länge direkt festlegen
Die Textlänge beträgt genau n Zeichen, ungeachtet der vorher durchgeführten Operationen. Wenn das Feld vorher schon kürzer war als n Zeichen, ändert sich nichts. (l >= 0)

Sonderfall: Wenn man **`l 0`** gibt und anschließend einen w-Befehl, kann man eine Kategorie mit einem festen Text erzeugen, die nur besetzt wird, wenn ein bestimmtes, vorher mit einem geeigneten Positionierungsbefehl ermitteltes Fremdfeld vorkommt. *Beispiel:* wenn in den Fremddaten vom Typ C ein Feld "SO: " vorkommt, soll "#97 ZS" in den Datensatz eingetragen werden. Angenommen ist hier, daß das Feldende vorher mit globaler Ersetzung auf Code 0 gesetzt wurde:

```
#97                 Dokumenttyp
s 0 "SO: "          kommt das Feld SO im Fremdtext vor? (dann muß es
                    so beginnen: nach einem Feldende (0) kommt "SO: " )
l 0                 setzt Länge auf 0
w " ZS"             schreibe " ZS" in #97
#32  ...            wenn "SO" nicht vorkommt, geht's gleich hier weiter
```

`A` **Feldlänge nach der SOIF-Methode bestimmen.** (Harvest-Daten) Es muß ein s-Befehl vorangehen, der den Anfangspunkt bestimmt. Die Länge steht bei SOIF-Daten dahinter in { ... }.

11.2.3.4 Manipulationsbefehle und Unterprogramme

An einem Datenfeld, dessen Beginn und Länge mit den vorgenannten Befehlen bestimmt wurde, können gewisse Bearbeitungen vorgenommen werden, bevor es tatsächlich abgespeichert (d.h. in den ***allegro***-Datensatz übernommen) wird.
Zwischen den Zeilen können, wie beim Export, Leerzeilen und mit Leerzeichen beginnende Kommentarzeilen stehen.
Wichtiger Unterschied zur Exportsprache: **jeder Befehl** steht auf einer **eigenen Zeile**.

_ CS1 **Lokale Ersetzung, mehrfache Ausführung**
_ CS2 wirkt wie die globale Ersetzung (→ 11.2.2), aber nur in der aktuellen Kategorie. Jedes vorkommende CS1 wird durch CS2 ersetzt.
Sonderfall: wenn das letzte Zeichen von CS2 der Code 0 ist, wird zugleich automatisch das Feldende auf diese Position gesetzt, der hinter CS1 stehende Teil also abgeschnitten.

% CS1 **Lokale Ersetzung, einfach**
_ CS2 wie oben, jedoch wird nur das erste CS1 in der aktuellen Kategorie durch CS2 ersetzt.
Sonderfall: Wenn das letzte Zeichen von CS2 der Code 0 ist, wird automatisch das Feldende auf diese Position gesetzt. (Dieser Befehl kann nicht global angewandt werden.)
Es kann mehrere lokale Ersetzungsbefehle hintereinander geben. Die Anzahl ist nicht begrenzt. Man muß nur darauf achten, daß jeder Befehl aus zwei Zeilen besteht.

i k **Ignoriere führende** Nullen (k=48), Leerzeichen (k=32) etc.
Wenn der ASCI-Code k ein- oder mehrfach am Anfang des aktuellen Feldes steht, wird er weggelassen, d.h. der Textzeiger wird hinter diese Zeichen gestellt. k ist eine Dezimalzahl.

. **Punkt am Ende nicht beseitigen**
Am Ende eines Datenfeldes werden normalerweise die Interpunktionszeichen ".,:;" sowie Leerzeichen beseitigt. Der Punkt am Ende bleibt stehen, wenn man diesen Befehl gibt. (Z.B. wendet man dies an bei Feldern, die meist am Ende eine Abkürzung enthalten. Wenn am Ende des Feldes eine einbuchstabige Abkürzung steht, wird der Punkt nicht entfernt.)

: **Interpunktion am Ende nicht beseitigen**
Alle evtl. am Ende des Feldes stehenden Interpunktionszeichen bleiben stehen.

N **Nichtsortierwort am Anfang kennzeichnen**
Das erste Wort des Textfeldes wird mit der Liste der Artikel verglichen. Wenn es darin vorkommt, wird es als Nichtsortierwort gekennzeichnet. (Vgl. dazu die Befehle n und N in der Konfigurationsdatei, Anh.A.) Die Anwendung erfolgt nicht auf den momentanen Arbeitstext, sondern auf das fertige Datenfeld, wenn der Paragraph abgearbeitet ist.

Write-Befehle, Hintergrundspeicherbefehle
w CS **Präfix** : CS wird sofort in den Transferbereich geschrieben, der Arbeitstext bleibt unverändert
W CS **Postfix** : der Arbeitstext wird in den Transferbereich geschrieben, CS dahinter. Nach einem W-Befehl ist also der Arbeitstext nicht mehr für Änderungen verfügbar!
Die Zeichenfolge CS wird in die ***allegro***-Kategorie eingefügt, und zwar bei 'w' vor bzw. bei 'W' hinter den importierten Text. Vgl. dazu den bei Befehl 'l' beschriebenen Sonderfall.

Kxy **Arbeitstext zwischenspeichern** : der aktuelle Arbeitstext wird in die Anwendervariable **#uxy** übertragen. Mit **kuxy** (!) kann man wieder darauf zugreifen (→ 10.2.6.3; 11.2.3.2).

Yxy **Transferbereich zwischenspeichern** : Innerhalb eines Paragraphen können mehrere Positionierungsbefehle vorkommen. Jeder neue solche Befehl bewirkt, daß der Arbeitstext in einen anderen Bereich kopiert wird, den Transferbereich. Dieser enthält praktisch den schon fertiggestellten Teil der neuen Kategorie. Der Befehl **W CS** (s.o.) bewirkt ebenfalls, daß der Arbeitstext an den Transferbereich gehängt wird (und **CS** selbst hintendran). Der Befehl **Yxy** kopiert den Inhalt dieser noch unfertigen Kategorie in die Anwendervariable **#uxy**.

v a=x **Hilfsvariable** a auf den Wert x setzen; a = Buchstabe oder Ziffer, x = beliebiges Zeichen
Alle Buchstaben und Ziffern kann man als Namen für Hilfsvariablen benutzen und ihnen beliebige Zeichen als Werte zuweisen. An anderer Stelle kann man diese Variablen mit dem Befehl **V a=y** wieder auswerten (→ 11.2.3.5). Besonders wichtig: die Werte bleiben auch für nachfolgende Sätze erhalten.

h **Hierarchie-Ebene erhöhen** (sehr selten nötig)
Haupt- und Untersätze laufen als getrennte Sätze durch die Verarbeitung, wenn **re** auch zwischen Haupt- und Untersatz steht (→ 11.2.1, Typ C)
Wenn eine Bedingungsprüfung ergeben hat, daß der aktuelle Satz ein Untersatz des vorangehenden ist, erhöht dieser Befehl die Hierarchiekennung des Satzes, d.h. er macht #01 aus #00 etc. Wichtig: vorher muß die Kategorie #00 eingerichtet werden, sonst würde z.B. #21 aus #20 gemacht, wenn zufällig #20 die erste Kategorie des aktuellen Satzes ist. Wenn man also mit hierarchischen Sätzen arbeitet, sollte man in der Kategorieliste immer mit #00 beginnen.

H **Neuen Datensatz beginnen**
Der aktuelle Satz wird beendet, die aktuelle Kategorie kommt bereits in den nächsten Datensatz, der nun neu angelegt wird. Mit diesem Befehl kann man erreichen, daß aus einem Fremdsatz mehrere Datensätze gemacht werden. Es wird ein Hauptsatz sein, falls man nicht als erstes dann eine Kategorie #01, #02 etc. erzeugt.
Andere Möglichkeit: man legt innerhalb der Bearbeitung eines Satzes eine neue Kategorie `#00` an. (D.h. die erste in der CFG definierte Kategorie.)

U **Zum vorigen Datensatz zurückkehren**
Vom aktuellen Satz wird zum vorangegangenen zurückgekehrt, die aktuelle Kategorie wird dort einsortiert. Mit dem Befehl **H** (im nachfolgenden Paragraphen!) kann man erneut zurückschalten. (Bis V15.0 wurde bei Vorliegen dieses Befehls immer intern auf manuellen Modus geschaltet, als ob man `-m0` gegeben hätte. Dieses Problem wurde beseitigt.)

x **Feld im Fremdsatz ungültig machen**
Das mit dem letzten Positionierungsbefehl angesteuerte Feld im Fremddatensatz wird dort ungültig gemacht, so daß es mit einem nachfolgenden Positionierungsbefehl nicht mehr gefunden wird.
Sinnvoll ist dies, wenn bestimmte Feldkennungen mehrfach vorkommen können, aber nicht alle unter derselben ***allegro***-Kategorie mit Mehrfachbesetzung untergebracht werden sollen, also der Befehl m nicht anwendbar ist.

@X **Umsetzung per Konkordanzliste**

Zusätzlich zu den globalen und lokalen (kategoriespezifischen) Ersetzungen gibt es die Möglichkeit, umfangreichere Umsetzungen mit Hilfe von Konkordanzlisten vorzunehmen.

Die **lokalen** Ersetzungen werden in der gesamten Kategorie, nicht nur am Anfang, ausgeführt.

Allzuviele **globale** Ersetzungen verlangsamen den Ablauf. Man sollte sie nur anwenden, wenn die zu ersetzenden Zeichenfolgen innerhalb des Datensatzes an beliebigen Stellen stehen können. Darunter fallen Zeilenbegrenzungen (die aus mehr als einem Zeichen bestehen) und Einrückungen umbrochener Zeilen.

Eine **Konkordanzliste** ermöglicht die Zuordnung und den Austausch von Kategorieinhalten durch andere Inhalte, und zwar mit hoher Geschwindigkeit.

Dazu folgendes Beispiel, das in der Import-Parameterdatei MARCBF.AIM enthalten ist:
Die Library of Congress Classification ist in der MARC-Kategorie 050 zu finden. Für die Datenbankstruktur CAT (→ 1.0) sollen die Hauptgruppen dieser Klassifikation in die SWD-Sachgruppen der Deutschen Bibliographie umgesetzt werden. Das leistet dieser Importparagraph: (für das Bibliofile-MARC)

Befehle	Kommentare
`#30a`	"allegro"-Kategorie #30a soll das Element aufnehmen:
`T 0 50`	finde die MARC-Kategorie ("tag") 050
	(z.B. für das OCLC-MARC müßte es hier D "050" heißen,
	für das BNB-MARC von der CD-ROM dagegen s "050" ...)
`@A`	Konkordanzliste A zur Umsetzung benutzen (s.u.)
`m"; "`	kann mehrfach auftreten: Elemente mit "; " trennen.

und dazu gehört die Konkordanzliste A, die am Ende der Import-Parameterdatei steht und so beginnt und endet:

`@@A`	Beginn Liste A
`"A"`	wenn der Kategorieinhalt mit "A" beginnt, wird dieses
`"01" 0`	durch die Zeichenfolge "01" und Byte 00 (als Abschluß) ersetzt.
`"BF10"`	"BF10" wird durch "11p" 0 ersetzt.
`"11p" 0`	
`"BF"`	andere mit "BF" beginnende Klassifikationen durch "11" 0
`"11" 0`	
`"BA"`	
`"10" 0`	
... usw. usf. ...	
`"Z2"`	electronic publishing
`"28" 0`	wird zu 28 = Informatik geschlagen
`"Z"`	Buchwesen wird ansonsten Gruppe 02 zugeordnet
`"02" 0`	
`@`	Ende der Liste

Regeln für Konkordanzlisten:

- Es kann mehrere solche Listen geben. Jede kann in mehr als einer Kategorie zur Anwendung kommen.
- Die Listen stehen am Ende der Import-Parameterdatei, hinter dem letzten Paragraphen der Kategorieliste.
- Jede Liste beginnt mit @@X (mit einem Großbuchstaben X) und endet mit @
- Die Liste besteht aus Paaren von Zeichenfolgen, die in der üblichen CS-Notation anzugeben sind (druckbare Zeichen in `"..."`, Steuerzeichen als Dezimalzahlen). Die jeweils erste Zeichenfolge des Paares wird durch die zweite ersetzt, aber nur, wenn sie am Anfang des vorher aufbereiteten Kategorietextes steht. Der Rest des Textes bleibt erhalten, es sei denn, die zweite Zeichenfolge endet mit 0 (siehe oben!).
  Im Prinzip ist also die Syntax dieselbe wie bei den anderen Ersetzungen, nur ohne das Zeichen '_' am Anfang. Alle Listen einer Parameterdatei können zusammen 1.000 solche Paare von Zeichenfolgen enthalten. Es müssen keine Leerzeilen zwischen den Paaren stehen, doch verbessert das die Übersicht.
- Die Länge einer Zeichenfolge ist auf 255 Zeichen begrenzt, die Gesamtlänge der Konkordanzen dadurch, daß alle Parameter zusammen (Import und Export) in einen Bereich von 64K passen müssen.
- Kommentare können, wie sonst auch, durch mindestens zwei Leerzeichen abgesetzt oder auf eigenen Zeilen, die mit Leerzeichen beginnen, zwischengefügt werden.
- Der eigentliche Austausch wird durch den Befehl @X vorgenommen, der nach eventuellen Manipulationsbefehlen eingesetzt wird (s.o.). Bei diesem Vorgang wird die Liste von oben nach unten abgearbeitet. Deshalb hat man die Möglichkeit, wie man oben sieht, z.B. "Z2" durch "28" zu ersetzen, und danach dann alle anderen mit "Z" beginnenden Zeichenfolgen durch "02". D.h.: die Liste ist nicht alphabetisch zu ordnen, sondern so, daß die Abarbeitung sinnvolle Resultate liefert.

un Standard-Unterprogramm n ausführen

Es gibt mehrere vorgefertigte **Standard-Unterprogramme** für besondere Aufgaben:

n = 1 Jahreszahl aus dem Text herauslösen. Im Textfeld wird nach einer Jahreszahl gesucht und nur diese dann in das aktuelle ***allegro***-Feld übernommen.

2 Römische Zahlen in Großbuchstaben umsetzen: die Buchstaben `ivxlcdm` werden in Großbuchstaben umgesetzt. (Anwendbar z.B. bei den Seitenzahlen aus MARC-Daten)

3 Namen in RAK-WB-Form ändern. Der zweite Vorname wird auf die Initiale gekürzt, weitere Vornamen weggelassen (nicht mehr zu empfehlen, die Regel wurde abgeschafft).

4 Nur die erste Initiale wird übernommen, der Rest und alle anderen Vornamen entfallen.

5 alle Vornamen auf Initialen kürzen

6 Bindestriche in ISBN einsetzen (in MARC-Daten ist die ISBN oft ohne Bindestriche angegeben). Bei mit '9' beginnenden ISBNs gibt es aber Ungenauigkeiten.

7 MARC subfield codes durch `"--"` ersetzen.

8 MARC subfield codes umsetzen: aus `"▼x"` wird `" x) "`

9 Nichtsortierkennzeichnung gemäß MAB (^Der ^Name der Rose) ersetzen durch Kennzeichnung gemäß NMN (¬Der Name der Rose).

10 Zusatz zum Sachtitel groß schreiben. Der erste Buchstabe hinter der Zeichenkombination " : " wird in einen Großbuchstaben umgewandelt.

11 Jedes Wort des Datenfeldes mit großem Anfangsbuchstaben versehen. Wenn die Fremddaten z.B. alle in Großbuchstaben sind, setzt man vorher mit dem Befehl

`y A/Z a`

alles in Kleinschreibung um und wendet dann u11 bei jedem Datenfeld an, wo es sein soll.

12 Umsetzung Groß->klein auf Feldebene. Wenn z.B., wie bei "Current Contents on Diskette", ein einzelnes Feld in Großbuchstaben ankommt, kombiniert man (auf zwei Zeilen hintereinander, u12 u11, dann wird es wie gewünscht in Normalschrift umgesetzt. Umlaute werden berücksichtigt, andere Zeichen (z.B. akzentuierte Buchstaben) jedoch nicht.

Vor der Verwendung von 3 - 5 wird ausdrücklich gewarnt: erstens hat man die RAK-WB-Form der Personennamen so gut wie abgeschafft, zweitens, wenn man es denn will, kann man diese Form, z.B. für einen Zetteldruck, auch beim Export herstellen. Wenn man schon vollständige Namen bekommt (z.B. von der DB, die nie gekürzt hat), warum sollte man sie dann verstümmeln?

>P Eigenes Unterprogramm ausführen

Hat man irgendwo in einem anderen Paragraphen (weiter **oben** in der Liste) eine Zeilengruppe mit dem Buchstaben P markiert (siehe unten), kann diese Zeilengruppe von jedem anderen Paragraphen aus aufgerufen werden. Sie wird dann ausgeführt und auf den aktuellen Arbeitstext angewendet, als wäre sie Teil desselben Paragraphen. P kann jedes beliebige Zeichen sein, d.h. man kann sehr viele Unterprogramme machen. Hier jedoch, im Gegensatz zu den Sprungmarken, kann nur ein einzelnes Zeichen als Name benutzt werden. Außerdem sind **keine Verschachtelungen** möglich, d.h. kein Aufruf eines UP aus einem anderen heraus.

(P Unterprogramm P beginnt
)P Unterprogramm P endet

Diese Befehle müssen innerhalb desselben Paragraphen stehen. Sie stören den Ablauf selbst nicht, bilden aber eine **Zeilengruppe**, die von anderswo (weiter **unten**) als Unterprogramm aufrufbar ist. Diese zwei Befehle müssen jeweils auf eigener Zeile stehen. (Siehe auch: Vergleichsbefehle, 11.2.3.5)

Steuerbefehle : Vergleiche und Sprünge

Außer denjenigen Befehlen, die sich direkt auf einzelne Kategorien beziehen, gibt es auch noch einige Befehle zur Ablaufsteuerung. Hierdurch erst wird die Importsprache tatsächlich zu einer speziellen Programmiersprache.

Wichtig: Diese Befehle können nur **innerhalb** von Paragraphen vorkommen, **nicht dazwischen**. Also z.B. nicht sofort hinter einer Sprungmarke: dort kann nur eine Kategorie-Angabe stehen. Auch nicht sofort unter einer Kategorie-Angabe, dort muß zuerst ein Positionierungsbefehl stehen.

11.2.3.5 Vergleichsbefehle

Alle Vergleichsbefehle haben dieselbe Wirkungsweise: Die einem Vergleichsbefehl folgende Zeile des Paragraphen wird nur ausgeführt, wenn die Bedingung erfüllt ist, sonst wird sie übergangen. Wenn die betr. Zeile ein Sprungbefehl ist, ergibt sich dadurch auf einfache Weise ein bedingter Sprung.
Textzeiger und Textlänge bleiben unverändert.
Umwandlungen durch y-Befehl und globale Ersetzungen sind bei den Vergleichen zu berücksichtigen, denn sie werden vorher, gleich nach dem Einlesen, ausgeführt.

Sonderfälle:

- Wenn auf den Vergleichsbefehl ein lokaler Ersetzungsbefehl folgt, der ja immer aus zwei Zeilen besteht (→ 11.2.3.4), wird das Programm im Falle eines negativen Vergleichsergebnisses diese Zeilen beide übergehen. Also kann man sehr leicht Ersetzungen von Vergleichen abhängig machen.

- Wenn die dem Vergleichsbefehl folgende Zeile nur aus "(P" oder "(" besteht, werden alle nachfolgenden Zeilen übergangen, bis eine kommt, die mit ")" anfängt. Also kann man Befehle ganz einfach zu **Zeilengruppen** zusammenfassen (aber immer nur innerhalb eines Paragraphen!), die als Ganzes ausgeführt oder übergangen werden. Wichtig ist, daß die Klammer jeweils am Zeilenanfang stehen muß und dahinter höchstens noch ein Kommentar. Die Klammern bilden demnach eigene Befehlszeilen. Ist ein Buchstabe oder anderes Zeichen hinter der Klammer, kann diese Zeilengruppe von weiter unten als Unterprogramm aufgerufen werden (siehe oben).

`= CS` **Prüfen auf Gleichheit**
`= #nnn` Mit Beginn an der Stelle des vorher ermittelten Textzeigers wird der Text mit CS verglichen. Bei Gleichheit wird die nachfolgende Zeile(ngruppe) ausgeführt, sonst wird sie übergangen. Statt einer Zeichenfolge CS kann auch eine Kategorie oder eine #u-Variable angegeben werden. Deren Inhalt wird dann zum Vergleich benutzt.

`L n CS` **Vergleich an anderer Position**
`R n CS` Der Vergleich wird n Zeichen links bzw. rechts vom Textzeiger ausgeführt

`c CS` Ist **CS im Fremdfeld enthalten**?
Wenn die Zeichenfolge CS in der aktuellen Fremdkategorie vorkommt, wird die nächste Zeile(ngruppe) ausgeführt, sonst übergangen.
(In CS dürfen, auch bei den folgenden Befehlen, Maskierungszeichen vorkommen)

`z CS` dasselbe, aber Groß-/Kleinschreibung werden ignoriert (CS in Kleinbuchstaben angeben)

`C CS` Ist CS **im gesamten Datensatz enthalten**?
Wenn die Zeichenfolge CS irgendwo im eingelesenen Satz vorkommt, wird die nächste Zeile(ngruppe) ausgeführt. (s.a. → 11.2.3.2)

`Z CS` dasselbe, aber Groß-/Kleinschreibung werden ignoriert (CS in Kleinbuchstaben angeben)

`V a=y` **Hilfsvariable prüfen** : wenn Hilfsvariable a den Wert y hat, nächste Zeile(ngruppe) ausführen. Die Hilfsvariable a muß vorher mit einem Wert besetzt worden sein (→ 11.2.3.4).

11.2.3.6 Sprungbefehle

/ **Nächste Zeile(ngruppe) bedingungslos übergehen**
Sinnvoll ist dies nur hinter einem Vergleichsbefehl: die Vergleichslogik wird dadurch umgekehrt. Wenn eine Zeilengruppe folgt (runde Klammern am Anfang und Ende) wird sie als Ganzes übergangen (→ 11.2.3.5 unter "Sonderfälle").

+LABEL **Feld abschließen, dann Sprung**
Das Textfeld wird mit der vorher ermittelten Länge in die ***allegro***-Kategorie #xxf übertragen, dann sucht das Programm die Sprungmarke `-LABEL` (s.u.) und macht mit dem darauf folgenden Paragraphen weiter. Hier kann `LABEL` ein beliebig langes "Wort" aus Buchstaben und Ziffern sein, Sonder- und Leerzeichen sind nicht erlaubt.
(Man beachte den Unterschied zum Exportprogramm, wo leider nur Einzelzeichen als Sprungmarken erlaubt sind.)

qLABEL **Sprung ohne Speicherung** ("quit")
Hinter einem Vergleichsbefehl kann es sinnvoll sein, die Bearbeitung des Feldes abzubrechen, ohne es zu speichern. Dazu dient dieser Befehl. Der Paragraph wird ohne Wirkung beendet, #xxf wird nicht gespeichert, ansonsten wirkt qLABEL wie +LABEL.

Wenn LABEL nicht existiert, wirkt der Sprungbefehl in beiden Fällen als **Endebefehl**, d.h. der aktuelle Satz gilt als fertiggestellt, die Bearbeitung des nächsten beginnt.

>P **Unterprogramm P ausführen** (→ 11.2.3.4)

-LABEL **Sprungmarke**
unmittelbar vor einem Paragraph, d.h. vor einem Kategoriebefehl (nicht innerhalb eines Paragraphen) kann eine Sprungmarke stehen. Sie beeinflußt selbst den Ablauf nicht, sondern wird mittels der Befehle +LABEL und qLABEL angesprungen. Rücksprünge sind möglich (Schleifengefahr!). Wenn -LABEL mehrfach vorkommt, wirkt nur das erste.
Wenn das Programm während der normalen Abarbeitung auf eine Sprungmarke stößt, wird sie ignoriert.

mk **Mehrfachbesetzung** - Kategorie-Wiederholung
Wenn **mk** am Ende eines Paragraphen steht, wird dieser mehrfach durchlaufen. Der Positionierungsbefehl wird dann wiederholt und die Kategorie `#xxf` mehrfach besetzt, wenn die gesuchte Zeichenfolge mehrfach vorkommt. Wenn eine Ziffer **k** angegeben ist, wird sie auf **f** addiert für jede weitere Kategorie. Wenn **f** fehlt, wird bei der zweiten Kategorie der Wert **fm** eingesetzt, bei jeder weiteren wird **k** bzw. **fr** (wenn **k** fehlt) addiert (→ 11.2.1).
Andere Möglichkeit: Verwendung von ~ als Steuerzeichen hinter der Feldnummer (→ 11.2.3.1)

m"xyz" **Mehrfachbesetzung** - Wiederholung innerhalb Kategorie
mehrfach besetzte Felder werden alle in dieselbe ***allegro***-Kategorie #xxf überführt, getrennt jeweils durch die Zeichenfolge **xyz**. Ein Beispiel wäre **m"; "** .
Zwischen m und "..." ist kein Leerzeichen erlaubt.

Zu beachten ist, daß alle diese Sprungbefehle nur innerhalb eines Paragraphen vorkommen dürfen. Insofern sind es immer bedingte Sprünge: sie funktionieren nur, wenn eine Kategorieangabe *und* eine Positionierung vorangeht. Bei **q** kann die Positionierung entfallen:
Einen **unbedingten Sprung** zur Marke **-LABEL** konstruieren Sie am besten so:

#99 Kategorieangabe pro forma
qLABEL Absprung nach **-LABEL** (ohne daß eine `#99` erzeugt wird)

11.3 Beispiel einer Umwandlung mit *allegro*

DB-MAB-90 → Konsolidiertes Format A.CFG

Hier ist eine "download"-Datei abgedruckt, die zwei Datensätze aus der Deutschen Nationalbibliographie enthält. Das hier gezeigte Format ist das DB-MAB-90 der CD-ROM-Version. Es gehört nicht gerade zu den einfachsten der zur Zeit beobachtbaren Datenformate. (Vor 1993 wurde jedoch ein noch komplizierteres, sog. "Datenbankformat" angeboten.) Mitgeliefert wird ab Jan.93 eine Datei DBDISK.AIM, die das Format "MAB-Diskette" umwandelt und auch für ZDB-Daten geeignet ist. Dieses wird inzwischen von Der DB als Quasi-Standard für den Austausch per Diskette gehandelt, obgleich es einige Sonderzeichen gibt, z.B. die ostsprachigen Akzentbuchstaben, die darin nicht übermittelt werden!
Die hier gezeigten Daten müssen jedoch mit DBCD.AIM umgewandelt werden, denn MAB-Diskette ist mit DB-MAB-90 nicht identisch! Die Kategorienummern stimmen zwar überein, aber ansonsten gibt es viele Unterschiede. Seit 1996 werden außerdem die DNB-Daten wahlweise im Format MAB2 angeboten. Dafür gibt es eine Importparameterdatei MAB2.AIM. Ab 2000 wird nur noch MAB2 ausgeliefert.
Zunächst sehen Sie hier die Daten in genau der Form, die man von der CD-ROM bekommt.
Die Zeilen sind durch die Zeilenvorschub-Steuerzeichen "Carriage Return" und "Line Feed" (Codes 13 und 10 bzw. hexadezimal 0D und 0A) getrennt. Dies nutzt man beim "allegro"-Import aus.

Ein Datensatz beginnt bzw. endet immer mit der Zeichenkombination **" ### "**, in der Importsprache schreibt sich das als `re="  ###  " .`

```
Deutsche Nationalbibliographie 2.0
(C) 1992 Buchhändler-Vereinigung GmbH

   ###  nt81113000086.022994.71mth292--------
        3-7723-6903-0 bs 1z DEz 171986
_  001  86.022994.7
_  100  Plate, Jürgen
a  104  Wittstock, Paul
_  331  Pascal: Einführung - Programmentwicklung - Strukturen
_  335  e. Arbeitsbuch mit zahlr. Programmen, Übungen u.
        Aufgaben                                  Man beachte hier den Zeilenumbruch
_  359  Jürgen Plate ; Paul Wittstock
_  403  3., neu bearb. u. erw. Aufl.
_  410  München
_  412  Franzis
_  425  1986
_  433  426 S.
_  434  graph. Darst.
_  435  23 cm
b  451  Franzis-Computer-Praxis
a  540  ISBN 3-7723-6903-0 kart. : DM 58.00
a  544  D 86/8601
_  568  86,N10,0131
_  574  86,A20,0576
_  700  28
s  902    ( 30m )  PASCAL
```

```
          Zwischen den Datensätzen stehen die Codes 13 10 13 10, d.h. zwei Zeilenvorschübe. Das
          interessiert uns jedoch nicht. Sie werden vom IMPORT-Programm ausgefiltert. Der nächste Satz
          beginnt wieder mit "   ###  " :

     ###  nt14783000085.054847.01mth292--------
          3-88180-112-X bs 1z DEz 371985
_    001  85.054847.0
b    100  Schwartz, Frank  |[Hrsg.]|
b    104  Panknin, Walter  |[Mitverf.]|
_    331  Flugmodell & [und] Computer
_    335  26 Basicprogramme für C 64, VC 20, Spectrum, ZX 81, TI
          99/4A, Atari 400, Video Genie I, Video Genie II,
          Alphatronic PC, PC 1251
_    359  Frank Schwartz (Hrsg.). Von Walter Panknin ...
_    410  Baden-Baden
_    412  Verlag für Technik u. Handwerk
_    425  1985
_    433  104 S.
_    434  graph. Darst.
_    435  30 cm
_    451  Modell-Technik-Berater ; 13
r    453  55.097638.8
c    454  Modell-Technik-Berater
_    455  13
a    540  ISBN 3-88180-112-X kart. (Pr. nicht mitget.)
a    544  D 85b/8285
_    568  85,N26,0120
_    574  86,A41,0637
_    700  28
_    700  43
s    902    ( 36 )  Flugmodell
s    902  1 ( 30 )  Mikrocomputer
s    902  2 ( 30m )  BASIC
f    902  11 Programm
_    903  2314 3214
```

Diese Darstellung, die man z.B. mit dem MS-DOS-Befehl TYPE am Bildschirm sehen würde, liefert noch nicht alle Angaben, die man für eine Umwandlung braucht. Dazu muß man auch die Steuerzeichen sehen können, die zwischen Zeilen und Sätzen stehen. Es gibt zwei Methoden, wie man wirklich alles sichtbar machen kann:

-- Vom ***CockPit*** aus kann man mit **`µd s <Auswahl> <diagnost>`** eine Anzeige bekommen, die alle Zeichen mit ihren Dezimalwerten zeigt. Man geht über den Menüpunkt "Sämtliche Dateien", wählt die Fremddatei aus und aktiviert dann den Punkt "diagnost". In der dann erscheinenden Anzeige bekommt man die ersten 80 Zeilen der Datei zu sehen und kann darin blättern. Es stehen immer die echten Zeichen und ihre Dezimalwerte untereinander.

-- Ein Hilfsprogramm wie HE.EXE (ein sog. "Hex-Editor") zeigt alle Zeichen an, und zwar als Hexadezimale ASCII-Codes und zusaetzlich die druckbaren Zeichen in lesbarer Form. Das sieht dann so aus, wie Sie es auf der nächsten Seite sehen:

Nehmen wir an, die oben abgedruckten Daten stehen in einer Datei namens EXPORT.DOC. Die folgenden zwei Befehle produzieren dann auf dem Bildschirm ein genaues Speicherabbild (mit Hexadezimaldarstellung links und ASCII-Zeichen rechts): (einige markante Stellen sind unterstrichen)

```
he        (Aufruf des Programme HE.EXE)
file: export.doc     (Eingabe des Dateinamens)
    (Anzeige des ersten Teils der Datei, die ersten 256 Zeichen)
export.doc                                Seite:    0 Byte:   0    Schreibschutz
0D 0A 0D 44 65 75 74 73 63 68 65 20 4E 61 74 69    ...Deutsche Nati
6F 6E 61 6C 62 69 62 6C 69 6F 67 72 61 70 68 69    onalbibliographi
65 20 32 2E 30 20 0D 0A 0D 28 43 29 20 31 39 39    e 2.0 ...(C) 199
32 20 42 75 63 68 68 84 6E 64 6C 65 72 2D 56 65    2 Buchhändler-Ve
72 65 69 6E 69 67 75 6E 67 20 47 6D 62 48 20 0D    reinigung GmbH .
0A 0D 0A 20 20 20 23 23 23 20 20 6E 74 38 31 31    ...   ###   nt811
31 33 30 30 30 30 38 36 2E 30 32 32 39 39 34 2E    13000086.022994.
37 31 6D 74 68 32 39 32 2D 2D 2D 2D 2D 2D 2D 2D    71mth292--------
20 0D 0A 20 20 20 20 20 20 20 20 33 2D 37 37 32     ..        3-772
33 2D 36 39 30 33 2D 30 20 62 73 20 31 7A 20 44    3-6903-0 bs 1z D
45 7A 20 31 37 31 39 38 36 20 20 20 20 20 20 20    Ez 171986
20 20 20 20 20 0D 0A 5F 20 20 30 30 31 20 20 38         ..   001  8
36 2E 30 32 32 39 39 34 2E 37 0D 0A 5F 20 20 31    6.022994.7..__ 1
30 30 20 20 50 6C 61 74 65 2C 20 4A 81 72 67 65    00  Plate, Jürge
6E 0D 0A 61 20 20 31 30 34 20 20 57 69 74 74 73    n..a  104  Witts
74 6F 63 6B 2C 20 50 61 75 6C 0D 0A 5F 20 20 33    tock, Paul..__ 3

^F : Suchen            ^L : Suche fortsetzen                    (^F heißt Strg+F)
^P : Seite setzen      ^U : Schreibschutz umschalten
^V : Modus umschalten

ESC: Programm beenden                                  (mit ESC Ausstieg aus HE)
         (Taste <Bild↓> drücken, dann kommt der nächste Abschnitt von 256 Zeichen:)

33 31 20 20 50 61 73 63 61 6C 3A 20 45 69 6E 66    31  Pascal: Einf
81 68 72 75 6E 67 20 2D 20 50 72 6F 67 72 61 6D    ührung - Program
6D 65 6E 74 77 69 63 6B 6C 75 6E 67 20 2D 20 53    mentwicklung - S
74 72 75 6B 74 75 72 65 6E 0D 0A 5F 20 20 33 33    trukturen.._  33
35 20 20 65 2E 20 41 72 62 65 69 74 73 62 75 63    5  e. Arbeitsbuc
68 20 6D 69 74 20 7A 61 68 6C 72 2E 20 50 72 6F    h mit zahlr. Pro
67 72 61 6D 6D 65 6E 2C 20 9A 62 75 6E 67 65 6E    grammen, Übungen
20 75 2E 0D 0A 20 20 20 20 20 20 20 20 41 75 66     u..        Auf  Zeilenumbruch
67 61 62 65 6E 0D 0A 5F 20 20 33 35 39 20 20 4A    gaben.._  359  J
81 72 67 65 6E 20 50 6C 61 74 65 20 3B 20 50 61    ürgen Plate ; Pa
75 6C 20 57 69 74 74 73 74 6F 63 6B 0D 0A 5F 20    ul Wittstock.._
20 34 30 33 20 20 33 2E 2C 20 6E 65 75 20 62 65     403  3., neu be
61 72 62 2E 20 75 2E 20 65 72 77 2E 20 41 75 66    arb. u. erw. Auf
6C 2E 0D 0A 5F 20 20 34 31 30 20 20 4D 81 6E 63    l..._  410  Münc
68 65 6E 0D 0A 5F 20 20 34 31 32 20 20 46 72 61    hen.._  412  Fra
6E 7A 69 73 0D 0A 5F 20 20 34 32 35 20 20 31 39    nzis.._  425  19

^F : Suchen            ^L : Suche fortsetzen
^P : Seite setzen      ^U : Schreibschutz umschalten
^V : Modus umschalten

ESC: Programm beenden
```

Zur Verdeutlichung unterstrichen sind die Zeichenfolgen für Satzanfang und -ende (###) und die Anfänge zweier Kategorien (100 und 331). Das Feldende (0D 0A) erkennt man nur korrekt in der Hexadezimal-Darstellung.

Nur eine solche Anzeige erlaubt unter anderem die genaue Untersuchung aller Steuerzeichen, die mit dem TYPE-Befehl oder auch mit dem X-Editor nicht sichtbar werden. Diejenigen Stellen sind unterstrichen, an denen eindeutig Satzanfang und Feldende zu erkennen sind. Der Zeilenumbruch im Sachtitelzusatz (Kat. 335), ebenfalls unterstrichen, muß z.B. beseitigt werden.

allegro-C : Import-Parameterdatei DBCD.AIM ***gekürzt!***

```
  DB-CD-MAB-Format --> A.CFG  einschl. mehrbändigen Werken
  DBCD.AIM   950703   korr. 961101

  zu kombinieren mit den Ausgabeparametern  I-1.APR  !!!
  zwecks Erstellung von korrekten "allegro"-Daten, d.h. Start z.B. mit
  IMPORT -f5 -idbcd -ei-1/dbneu.alg ...

re="   ###  "
rs="   ###  "

  Vor der Bearbeitung eines Datensatzes folgende Ersetzungen: (Abarbeitung in dieser Reihenfolge!)
_ 13 10 "         "          ersetze Zeilenende/8 Blank
_ " "                        durch nichts

_ 13 10          ersetze sonstige Zeilenvorschübe
_ 0              durch Code 0  (wirkt dann automatisch als Kategorie-Ende)

_ " ¬  "         Weitere globale Ersetzungen
_ " "
_

_ "]¬"
_ "]"

_ 0 "_  9??  3? "   Ansetzungskette!
_ " /¯"

y | 170

  Hier beginnen die Feldbeschreibungen: ("Paragraphen")
    jeweils zuerst die allegro-Kategorienummer, dann ein Suchbefehl,
    (z.B. s 0 "335 " heisst: suche die Zeichenfolge 0 "335 ")
    dann kommen evtl. Vorbearbeitungsbefehle
 ....

  #00  aktivieren, wenn DB-Nr als IdNr benutzt werden soll, dann nächste Zeile rausnehmen
#89D   DB-Ident-Nr. (bei Hauptaufnahmen)
s 0 "?  001  "
w"d"    'd' als Kennzeichen vor die Nummer setzen

#09
s 0 "?  010  "

#20    Hauptsachtitel: 331
s 0 "?  331  "
n     Nichtsortierwort am Anfang markieren
C 0 "?  333  "
(
s 0 "?  333  "
w" ▼ "
)
s 0 "?  335  "
w " : "
```

```
#21      Sammlungsvermerk
s 0 "?  300? "

#22      Einheitssachtitel
s 0 "?  304  "
n
 .....  weitere Kategorien

#330     Komplizierte Aufbereitung der Schlagwortketten
s 0 "?  902  "
(S
Ksw
{ 8
l1
Kin
l0
W "▼"
kuin
kusw
B")  "
i49     Formschlagw. beginnt mit "11 "
i32
i49
i32
)S
m""

#330
k330
s 0 "?  903  "
w"P"

#331          2. Schlagwortkette
 ... #339   wie #330
 ...

 ....   viele weitere Kategorien
#99
s 0 "?  089"     nicht bei Untersaetzen
q59
q

#30a
s 0 "?  700  "
@C      Umwandeln mit Konkordanztabelle @@C (siehe unten)
m" "

  ....  es kommen noch weitere Kategorien und Konkordanztabellen
```

Ergebnisse der Umwandlung:

```
#20 Pascal: Einführung - Programmentwicklung - Strukturen : e.
    Arbeitsbuch mit zahlr. Programmen, Übungen u. Aufgaben
#30aCS
#31sPASCAL
#330▼sPASCAL
#39 Jürgen Plate ; Paul Wittstock
#40 Plate, Jürgen
#402Wittstock, Paul
#71 3., neu bearb. u. erw. Aufl.
#74 München
#75 Franzis
#76 1986
#77 426 S. : graph. Darst. ; 23 cm
#85 Franzis-Computer-Praxis
#87 3-7723-6903-0
#89D86.022994.7; 86/8601

#20 Flugmodell & [und] Computer : 26 Basicprogramme für C 64, VC
    20, Spectrum, ZX 81, TI 99/4A, Atari 400, Video Genie I, Video
    Genie II, Alphatronic PC, PC 1251
#30aCS KW
#31sFlugmodell; Mikrocomputer; BASIC
#330▼sFlugmodell▼sMikrocomputer▼sBASIC▼fProgramm▼P2314 3214
#39 Frank Schwartz (Hrsg.). Von Walter Panknin ..
#41 Schwartz, Frank ¬[Hrsg.]
#42 Panknin, Walter ¬[Mitverf.]
#74 Baden-Baden
#75 Verlag für Technik u. Handwerk
#76 1985
#77 104 S. : graph. Darst. ; 30 cm
#85 Modell-Technik-Berater ; 13
#87 3-88180-112-X
#89D85.054847.0; 85b/8285
```

Der Zeilenumbruch wurde hier nur der besseren Lesbarkeit wegen eingeführt. In den "allegro"-Dateien sind die Kategorien fortlaufend gespeichert. Dazwischen befindet sich nur ein Code 0 als Feldende.

12 Programm-Aufrufe (Batch)

Online: `h ac12`

12.0 Konzept

Der Anwender kann die verschiedenen Programme (→ 0.4), aus denen ***allegro*** besteht, direkt aufrufen, mit Optionen versehen und somit in größere Abläufe einbauen.

Sinn der Sache: Wenn man solche Aufrufe in MS-DOS-"Batchfiles" (unter UNIX in "Shell-Scripts") unterbringt, lassen sich umfangreiche Routinevorgänge mit einem einzigen Befehl auslösen. Solche Verarbeitungen sind z.B.:

- Selektion von Daten nach bestimmten Kriterien und anschließende Produktion einer sortierten Liste dieser Daten (siehe u.a. die Stapel SR-LIST.BAT, PR-LIST.BAT, ALPHA.BAT, QUEX.BAT, QUANT.BAT)
- Import-Bearbeitung neuer Fremddaten und Einfügen in eine vorhandene Datenbank (z.B. DBDISK.BAT und update.job)

Hinweis: Das **Windows**-Programm ***a99*** startet man so: `a99` *inidatei*. In der *inidatei* stehen, in Windows-üblicher Form, ähnliche Angaben, wie man sie den DOS-Programmen mit den nachfolgend beschriebenen Optionen übergibt. Siehe dazu die Datei **`a99.ini`**. Darin sind alle Möglichkeiten dokumentiert, und es wird auch auf den Zusammenhang mit den DOS-Optionen hingewiesen. Das Programm ***alcarta*** wird genauso aufgerufen.

Die einzelnen Programme (siehe Liste → 0.4) sind mit folgenden Namen aufzurufen:

Name	Progr.Nummer	Aufgabe des Programms
`acp`	**`c`**	***CockPit*: Start- und Steuerprogramm für das DOS-Gesamtsystem**
`presto/apac`	**`1`**	**Schnellzugriff (Datenbankbenutzung und Bearbeitung) / OPAC**
`acon`	**`a`**	**Vielzweck-Konsolprogramm, auch f. avanti-Web-Anbindung**
`srch`	**`4 / 6`**	**Volltext-Suche und Selektion, ferner Export-Produktion von Dateien, Listen, Karten etc.**
`import`	**`5`**	**Import von Fremddaten**
`index`	**`7`**	**Datenbank-Generierung (für PRESTO)**
`qrix`	**`q`**	**Quick Index Production und Index-Management**
`asort`	**`8`**	**alphanumerisches Sortieren**
`upd`	**`9`**	**Datenbank-Management**

Ein Programmaufruf sieht so aus:

`programm [ -x1 y1 -x2 y2 ... -xn yn ]`

dabei ist ***`programm`*** einer der Programmnamen, ***`xi`*** jeweils ein Befehlsbuchstabe (Option) und ***`yi`*** der Optionswert (auch "Argument" genannt), wobei nicht jede Option immer eine solche Wertangabe braucht (**`-S`** z.B. nicht).
Zwischen Befehlsbuchstabe aund Optionswert muß kein, darf aber ein Leerzeichen stehen.

Beispiel:

```
presto -a2 -n14 -d demo -S
```

startet den Schnellzugriff auf das Unterverzeichnis `DEMO.` Neue Daten werden in Datei 14 gespeichert. ***CockPit*** produziert solche Aufrufe automatisch, z.B. aus dem Menü "Funktionen".

Ein Tip: in Batchfiles, die unbewacht (etwa nachts) laufen sollen, sorgfältig prüfen, ob keine Option vergessen wurde oder so gesetzt ist, daß das Programm noch eine Frage stellen muß. Sonst wartet es endlos auf eine Antwort ...

Beachten Sie folgende Regeln (die im übrigen recht genau den UNIX-Konventionen für die Befehlssyntax entsprechen):

- zwei Optionen sind durch mindestens ein Leerzeichen zu trennen, d.h. vor jedem Bindestrich **muß** ein Leerzeichen stehen (wenn er nicht zu einem Namen gehört, z.B. `S-PJ.APR`)
- zwischen `'-'` und Befehlsbuchstabe **darf kein** Leerzeichen stehen
- zwischen Befehlsbuchstabe und Argument **kann** ein Leerzeichen stehen, muß aber nicht
- Anzahl und Reihenfolge der Optionen sind beliebig
  (die Programme fragen alle fehlenden Angaben noch ab bzw. setzen Standardwerte)
- jede Option außer -e und -U kann nur einmal vorkommen; bei Mehrfachangabe wirkt dann nur die erste.
  Nur bei PRESTO kann **`-d`** bis zu dreimal vorkommen, und zwar wenn man zwei oder drei Datenbanken gleichzeitig benutzen will. <Alt>+a ermöglicht dann das Umschalten zwischen den Banken.
- die Befehlsbuchstaben sind exakt so anzugeben, als Kleinbuchstabe oder Großbuchstabe, wie sie hier in der Liste stehen. Nur Programm-, Datei- und Verzeichnisnamen können unter MS-DOS wahlweise groß oder klein geschrieben werden, unter UNIX muß man auch dabei die richtige Schreibung beachten
- ein Dateiname (siehe -d und -e) wird so ausgewertet: bei fehlender Laufwerksangabe wird 'C:' ergänzt; wenn der Name nicht mit '\' beginnt, wird die Datei im "allegro"-Programmverzeichnis bzw. einem Unterverzeichnis desselben gesucht. (Gilt nicht für ASORT, siehe Kap.8.)

Hier ein weiteres Beispiel für einen Programmaufruf:

```
srch -f4 -ka -d*mdat\*.al? -e kurzlist/mdat\namelist -m0 -s goethe
```

Bedeutung: das Suchprogramm (Funktion 4) soll sich alle Datendateien (d.h. .ALD und .ALG) vornehmen, die auf dem Unterverzeichnis `MDAT` (Unterverzeichnis des "allegro"-Verzeichnisses) liegen, Datensätze heraussuchen, in denen "goethe" vorkommt und im selben Unterverzeichnis eine Datei NAMELIST produzieren mit dem Export-Parametersatz `KURZLIST.APR,` manuelles Eingreifen wird nicht gewünscht `(-m0).` Wenn NAMELIST schon existiert, wird sie gelöscht `(sonst` '+' statt '/' geben, dann werden die Daten hinten angehängt; bei UNIX '=' statt '/').

Sechs Optionen können unter DOS auch durch **Umgebungsvariable** (**environment variables**) ersetzt werden:

Option	**Bedeutung:**
`-b`	Datenbankname (= nur der Name der Index-Parameterdatei)
`-d`	Datenverzeichnis (Pfad, wo die Datenbank liegt)
`-k`	Konfiguration (Name der CFG-Datei)
`-k1`	Konfigurationsbuchstabe (1. Buchstabe des CFG-Namens)
`-l`	Sprache der Menütexte
`-P`	Name des Programmpfades (wenn nicht C:\ALLEGRO)

Das sind Werte, die man dem Betriebssystem mit dem SET-Befehl mitteilt und die es sich in einer Tabelle merkt, auf die dann die Programme zugreifen können. So könnte man z.B. durch den Befehl

`set -k=px`

festlegen, daß die nachfolgenden Programme mit dem Kategorienschema PX.CFG arbeiten sollen. Grundsätzlich gilt aber: **Optionen haben Vorrang vor Umgebungsvariablen**. Wenn also Option **`-kaps`** gegeben wird, dann nimmt das Programm auf jeden Fall `APS.CFG.` Der Sinn dieser Sache: mit den Umgebungsvariablen kann man gewisse Grundparameter für alle nachfolgenden Aufrufe festlegen. ***CockPit*** setzt diese Werte, bevor es ein Batchprogramm startet. Im Einzelfall kann bei einem konkreten Programmaufruf dann immer noch eine abweichende Option angegeben werden.
Es folgt die Liste der existierenden Optionen. Zum Teil sind diese auf einzelne Programme beschränkt und auch nur für diese sinnvoll:

12.1 Liste der Optionen (DOS und UNIX)

Die Programme QRIX und ASORT haben noch mehr Optionen, die hier nicht erwähnt sind (→Kap.7.5 u. 8).
Zu beachten ist die DOS-Grenze für die Länge eines Programmaufrufs: 128 Zeichen. Das Programm ***a99*** kennt keine Optionen; es erfährt seine Einstellungen aus der INI-Datei, die ihm als Argument mitgegeben wird, sowie evtl. aus einem `_start.flx`.

Option	anwendbar bei Programm	Bedeutung
	1:presto+apac 4/6:srch 5:import 7:index 8:asort 9:upd a:acon c:*CockPit* q:qrix	

`-+` 1 **"Hauptschalter" für die satzübergreifende Suche** [nicht bei acon]

Wenn diese Option gesetzt ist, werden bei der Ergebnismengenbildung automatisch immer alle untergeordneten, verknüpften Sätze mit hinzugenommen. (→ 10.2.9)

`-@` 7 **Primärschlüssel / sonstige Schlüssel erzeugen**

Wenn man mit Stammsatz-Ersetzungen arbeitet, gibt es die Möglichkeit, zwei INDEX-Durchläufe zu veranstalten. Die Aufrufe müssen so aussehen: (Genaueres: Kap.7)

`index -fx0 -@1 ...` nur Primärschlüssel erzeugen (x = 7, n oder i)
`index -fi1 -@2 ...` sonstige Schlüssel erzeugen.

`-a` 1 **Berechtigungsstufe für Schreibzugriff** (access)

Es gibt vier Stufen:

`-a0` (Standard) Keine Schreibberechtigung, nur Lesezugriff
`-a1` Schreibzugriff nur auf die mit Option -n angegebene Dateinummer
`-a2` Volle Schreibberechtigung für alle Dateien.
`-a3` zusätzlich Berechtigung für globale Änderungen und Löschungen (→ 1.5, **F10**), manuelle Indexänderungen (→ 1.4, **Entf** und **Einfg**) und Löschung trotz Löschkontrolle.

Wenn eine zweite (und dritte) Option **`-d`** gegeben wird, um parallel in zwei (oder drei) Datenbanken arbeiten zu können, muß man für die zweite (und dritte) eine entsprechende Ziffer ergänzen, z.B. **`-a320`**.

`-b` 1, 7, 9, a **Datenbankname** (unter UNIX immer, unter DOS nicht immer nötig)

Nur der Name der Index-Parameterdatei ist hier anzugeben, z.B. `-bcat`
Bei UPD wichtig, wenn mehr als eine Datenbank auf einem Verzeichnis liegt.

`-b` 4, a **Datenbankpfad\Datenbankname für Stammsätze**

Wenn das Programm SRCH bei einer Exportaufgabe Sätze nachladen soll, muß es wissen, aus welcher Datenbank es diese Sätze holen soll (→ 10.2.6.7). Entfallen kann **`-b`**, wenn es dieselbe Datenbank ist, in der SRCH sucht. Ansonsten gibt man z.B.
`-b f:\allegro\bib\cat` für die Datenbank CAT auf F:\ALLEGRO\BIB
Netzwerk: Es muß Schreibberechtigung auf dem Datenbankpfad bestehen.

`-d` 1-7, 9,a,c,q **Datenquelle** : wo sind die zu verarbeitenden Daten? (UNIX: '/' statt '\')

`-d verz\` Unterverzeichnis VERZ\ des aktuellen Verzeichnisses, aus dem der Aufruf erfolgt. Ein Verzeichnisname ist immer mit dem '\' abzuschließen! (Unter UNIX muß es '/' sein)

`-d verz\buch` alle Dateien BUCH* innerhalb VERZ\ werden zur Auswahl angezeigt bzw. Datenbank BUCH wird gewählt, wenn auf VERZ\ mehrere Datenbanken liegen.

`-d*verz\dd*` alle mit DD beginnenden Dateien innerhalb VERZ\ ; der '*' bewirkt, daß automatisch **alle** betroffenen Dateien verarbeitet werden, sonst erscheint die Liste der zutreffenden Namen, wie gewohnt, zur Auswahl.

Wenn -d nicht angegeben ist, entnimmt das Programm den Datenpfad aus der Umgebungsvariablen -d . Fehlt auch diese, gilt das aktuelle Arbeitsverzeichnis.
Option **`-d`** kann bei Programm 1 zwei- oder dreimal gegeben werden, wobei dann jeweils auch der Datenbankname anzugeben ist (z.B. **`-dkatalog\kat`**). Dann hat man mit <Alt>+a die Möglichkeit, zwischen den Datenbanken umzuschalten.

Option	anwendbar bei Programm	Bedeutung

1:presto+apac 4/6:srch 5:import 7:index 8:asort 9:upd c:*CockPit* q:qrix

`-e` 1, 4-7,9 Export-Parameter/Exportdatei

Die vorgeschriebene Form ist **`-e`** *param/outfile* oder **`-e`** *param+outfile*

mit *param* Name einer Parameterdatei (.cPR bzw. bei INDEX .cPI)
Der Name *param* kann eine Pfadangabe enthalten (ab V15).
outfile Dateiname für Ausgabedaten;
bei INDEX: nur der Name des Datenverzeichnisses

Wichtig: Ab V34 kann statt **/** auch **=** stehen; unter UNIX/Linux war das schon immer so.

Die zwei Angaben sind zu trennen durch: (***Achtung*** unter **UNIX**: stets **'='** statt **'/'**)

'/' : Ausgabedatei *outfile* neu anlegen (überschreiben, falls bereits vorhanden), bzw.

'+' : Daten an *outfile* anhängen, falls diese Datei schon existiert

Beispiele:

`-e p2+liste.txt` Ausgabe auf LISTE.TXT mit P2.APR als Parameterdatei ('+' bewirkt: Daten werden angehängt, falls Datei liste.txt vorh.)

`-e p-card/karte` Ausgabe auf die Datei KARTE. Diese kann nach der Sitzung mit **`print karte`** als Ganzes ausgedruckt werden.

`-e cat/bibl` Indexproduktion (Funktion INDEX -f7) mit Param. CAT.API; CAT.ADX wird auf ALLEGRO\BIBL\ erzeugt.

Wenn *param* fehlt (**/** oder **=** bzw. **+** darf nicht fehlen), erscheint dafür eine Auswahlliste.
Der Dateiname *outfile* ist ggfls. einschließlich eines Verzeichnisnamens anzugeben.
Der Dateiname **PRN** bewirkt Ausgabe auf dem Drucker, **CON** auf dem Bildschirm.
Bei Programm 4 können bis zu vier Exportbefehle stehen, bei den anderen bis zu drei. In einem Durchgang kann man also mehrere Ausgabedateien erzeugen.
Wenn *outfile* fehlt, wird der gewünschte Name abgefragt.
Bei APAC funktioniert der Export NUR dann, wenn eine Option `-e` angegeben ist oder mindestens `-a1`. Dann taucht "Daten exportieren" auf dem Menü auf.

`-f` 4-7, 9, q Funktionswahl

Einige der Programme haben neben ihrer Grundfunktion noch Zusatzfunktionen. Mehr darüber steht in den jeweiligen Kapiteln (besonders 7 u. 9). Für die Grundfunktion gibt man hinter -f die Programmziffer an, die Zusatzfunktionen haben andere Kennungen, und zwar gibt es folgende:

```
srch    -f4   Selektion mit oder ohne Export
        -f6   Export

import  -f5   Konvertierung von Fremddaten

index   -f7   Datenbank neu generieren incl. Index-Erstellung
        -fn      dasselbe, aber die Satznummern bleiben erhalten
        -fi   Index erneuern  (→ Kap.7.1)
        -ft / -fs / -fx    .TBL-Datei / .STL / .RES erneuern
        -fr   Dateien "entlüften"

qrix    -fq   Index aus Zwischendateien erstellen  (→ Kap.7.5)
        -fd   Index ausdrucken/anzeigen
        -fc   Index kompaktieren, evtl. Teilindex beseitigen

upd     -f9   .LOG-Datei einspielen  (→ Kap.9.1)
        -fmxy .cLG-Datei einspielen ("mischen" oder "merge")
        -fc   .cLG-Datei prüfen ("check")
```

Ohne -f erscheint (bei DOS, nicht UNIX) jeweils ein Zwischenmenü, wo man die gewünschte Funktion dann mit der Hand einzugeben hat.

Option	anwendbar bei Programm	Bedeutung

1:presto+apac 4/6:srch 5:import 7:index 8:asort 9:upd a:acon c:*CockPit* q:qrix

-F 9

Update ohne Verzögerung (Fast)

Bei Mehrplatzbetrieb (ohne Option -S) wartet UPD zwischen zwei Datensätzen immer eine Sekunde, weil sonst die evtl. gleichzeitig aktiven Mitarbeiter keine Speicherung mehr vornehmen könnten, denn UPD würde den Index fast ständig blockieren. Mit Option -F schaltet man diese Verzögerung ab.

Setzt man aber eine Ziffer hinter das F, z.B. `-F3`, so wird dies als Sekundenzahl gedeutet und nach jedem Datensatz diese Anzahl Sekunden gewartet. Das ist sinnvoll, wenn der UPD-Vorgang im Netz den allgemeinen Betrieb möglichst wenig behindern soll.

4 V14-Ersetzungen nicht durchführen (falls das nicht gebraucht wird). SRCH arbeitet dann schneller.

-I 1, 9

Index-Parameter

falls für Sonderzwecke eine andere als die zur Datenbank gehörige Parameterdatei benutzt werden soll.

-i 1, 9

Index-Parameter

Wenn PRESTO zur Erzeugung einer neuen Datenbank gestartet wird, gibt man mit dieser Option den Namen der zu benutzenden Index-Parameterdatei an. Das geschieht z.B. in der Batchdatei CREATE.BAT (ab V14c nicht mehr vorhanden). Dieser Name wird zugleich Datenbankname.

Achtung: Bei INDEX ist hierfür **`-e`** zu benutzen!

Beim Programm UPD wird **`-i`** *dbname* nur verwendet, wenn eine Datenbank mit dem Programm UPD neu aufgebaut werden soll (→ Kap.9.4).

-i 5

Import-Parameter

Hinter -i braucht nur der Name der Import-Parameterdatei (vollständiger Pfadname, wenn sie woanders liegt) angegeben zu werden, z.B.

`-ioclc` Parametersatz für Import von USMARC-Daten (in Verbindung mit -ka wird oclc.aim, bei -ks würde oclc.sim geladen. Das Programm sucht danach auf dem Datenverzeichnis, wenn ohne Erfolg, dann auf dem Programmverzeichnis. Für die Ausgabe wird man in der Regel dann den Parametersatz I-1.APR wählen (per Option -e i-1/...), aber auch jeder andere passende Satz ist benutzbar (oder zusätzlich einsetzbar).

-j a

Jobdatei-Name

Soll `acon` das Skript `abc.job` ausführen, gibt man ihm die Option `-jabc` mit.

-K q

maximale Schlüssellänge des Index

Nur bei `qrix -fq0` anzuwenden, damit das Programm Bescheid weiß. In der Regel wird QRIX für diese Funktion automatisch von INDEX aufgerufen, so daß man es nicht wissen muß (→ Kap.7.5), oder es entnimmt den Wert selber aus der IndexParameterdatei, wenn es diese findet.

-k 1-7, 9, a, c

Konfiguration

Hiermit sagt man dem aufgerufenen Programm, mit welchem Kategorienschema es arbeiten soll. Wenn Option -k fehlt, entnimmt das Programm die Angabe aus der Umgebungsvariable -k , die man evtl. mit dem Befehl **`set -k=x`** definiert hatte. Wenn auch diese nicht vorhanden ist, wird standardmäßig A.CFG als Konfigurationsdatei geladen (ab V13: $A.CFG, aber trotzdem **`-ka`** angeben). *Beispiel:*

```
-ku   :   Wahl des USMARC-Schemas ($U.CFG)
```

-l 1-7, 9, a, c

Sprache der Menütexte (language)

Das aufgerufene Programm benutzt die UIF-Dateien und Hilfeseiten mit der Sprachkennung, die hinter -l angegeben wird. (→ 0.3, 0.6)

Standard: **`-l ger`**

Wenn die Option -l fehlt, wird die Umgebungsvariable -l benutzt, falls sie definiert wurde. So kann man mit dem Befehl **`set -l = eng`** die Sprache auf Englisch einstellen, ohne dies jedem aufgerufenen Programm per Option -l mitteilen zu müssen.

Option	anwendbar bei Programm	Bedeutung
	1:presto+apac 4/6:srch 5:import 7:index 8:asort 9:upd a:acon c:*CockPit* q:qrix	

-L 1, 9, a

Sicherungsprotokoll abschalten / umlenken

Normalerweise schreiben PRESTO und UPD alle erfaßten und korrigierten Daten zusätzlich in die Datei *dbn*.LOG, um eine Restaurierung zu ermöglichen. Mit `-L kann man` das abschalten (was für den Normalfall nicht empfohlen wird). Wenn hinter **-L** ein *dateiname* angegeben ist, wird diese Datei statt *dbn*.LOG verwendet.

-M c

Monochrom-Darstellung (nur bei *CockPit* anwendbar)

Die Farbkombinationen des ***CockPit*** sind nicht ähnlich variabel wie bei den anderen Programmen (→ Anhang A, w-Befehle). Auf monochromen Bildschirmen, besonders vom VGA-Typ, empfiehlt sich der Einbau von -M in CP.BAT (→ 0.11).

1 Bei **apac** bewirkt **-M**, daß nach dem Start als erstes das **Esc-Menü** erscheint.

-m 1, 4-7

manuelles Eingreifen (APAC: `-m` bewirkt, daß man nicht aussteigen kann) [obsolet]

`-m1 : soll erlaubt sein`
`-m0 : soll nicht erlaubt sein` (unter UNIX nur dieses, und Strg+c = Abbruch)
Laufende Vorgänge können mit Druck auf 'x' jederzeit unterbrochen werden. (in V34 nicht mehr)
Wenn `-m1` gegeben wurde, werden die zu dem Zeitpunkt im Arbeitsspeicher stehenden Datensätze zur Bearbeitung bereitgestellt, d.h. man gelangt in den Editor, so daß also Korrekturen vor der Ausgabe auf die mit -e angegebene Datei erfolgen können. Aus dem Bearbeitungsmodus springt man mit dem Befehl `#r` wieder heraus, und der Vorgang läuft weiter.
Bei INDEX bewirkt `-m1,` daß nach dem Erstellen der vorsortierten Zwischendateien (→ Kap.7) eine Unterbrechung möglich ist, um evtl. durch Umkopieren oder Löschen einiger Dateien Platz zu schaffen für den Vorgang der Indexerstellung.
Wenn bei INDEX -m0 gegeben wurde, kommt die Frage "Start? j/n", damit man direkt vor dem Start noch eine Chance zum Abbruch hat. In Batchfiles: besser `-m0` setzen.

-N 1, 9

Speichermodus für neue Datensätze

`-N0` Keine Leersätze ersetzen (keine Verzögerung beim Speichern)
`-N1` Alle Leersätze ausnutzen, egal in welcher Datei (d.h. -n ohne Wirkung)
`-N2` (default) Neue Sätze nur in der durch -n vorgegebenen Datei speichern.

Bei `-N2` geht das Speichern auffällig langsam, wenn viele gelöschte Sätze existieren. Dies kann man in a99 auf dem ORG-Menü mit "ungenutzten Platz prüfen" feststellen (per ***CockPit*** mit **`ur o u`**). Es bietet sich an, die Datenbank dann gelegentlich zu "entlüften" (**`ur o R`**).

-n 1, 7, 9

Dateinummer für neue Daten

nur bei Schnellzugriff und Indexierung (→ Kap.1). Hinter -n muß eine Zahl zwischen 1 und 255 stehen. Wenn die Datenbank z.B. DOK heißt, werden neu eingegebene Daten in die Datei DOK_8.ALD gespeichert, wenn man -n8 angegeben hat. Bei INDEX bedeutet -n0, daß die Nummern der neu zu indexierenden Dateien erhalten bleiben sollen.

-O 1

Bearbeiterkennung (Operator): Bei Export als Sonderkat. `#op` verwendbar. Der Wert hinter `-O` wird an die Zeitstempel angehängt (→ Anh.A, Befehle `cn und ce`).

-o c

Vorgabendatei für *CockPit* wählen (nur DOS)

Normalerweise ist DEFAULT.OPT die Datei, aus der ***CockPit*** seine Vorgaben entnimmt. Mit `-o` kann man eine andere Datei dafür angeben.

-P 1 - 9, a, c

Programmpfad : Laufwerk und Verzeichnis, wo sich die Programme befinden; Als Standard wird C:\ALLEGRO angenommen (UNIX: /allegro)

-p 1

Anzeigeparameter

Standardmäßig wird `d-1.apr` als Anzeigeformat geladen. Mit **`-p display`** veranlaßt man, daß `display.apr` genommen wird.

Option	anwendbar bei Programm	Bedeutung 1:presto 4/6:srch 5:import 7:index 8:asort 9:upd c:*CockPit* q:qrix

-Q q

QRIX-Protokolldatei

Wenn das Protokoll nicht PROTOQ heißen soll, gibt man **-Q***dateiname.*

-q 1

Druckparameter

Wenn man beim Schnellzugriff ein vom Anzeigeformat (**-p...**) verschiedenes zum Drucken (mit F2) wählen möchte. Zusätzlich kann auch noch mit Option -e ein Exportformat geladen sein, d.h. Anzeige, Druck und Export können alle drei verschieden strukturiert sein. Das Exportformat kann überdies im Schnellzugriff während einer Sitzung geändert werden (→ 1.5, Funktion Shift+F4).

-R 9

Alle UPD-Aktionen aufzeichnen [obsolet, nicht mehr abschaltbar]

Die Meldungen, die sonst während des Durchlaufs am Bildschirm erscheinen, werden alle zusätzlich in die Datei UPRO geschrieben, sonst nur die Statistik.

-R3 9

Zusätzlich alle ungültigen Kategorien aufzeichnen [obsolet]

Wenn Datenfelder vorkommen, deren Kategorienummer nicht der CFG entspricht. Mit -R2 werden NUR diese ungültigen Kategorien aufgezeichnet.

-r 1 - 7

Wiederholung (repeat, nur DOS) [obsolet]

Nach dem ersten Durchlauf kommt man in ein Untermenü, von dem aus das Programm sofort wieder gestartet werden kann, ohne neu geladen zu werden. Ohne -r wird das Programm nach Ablauf verlassen.

Bei PRESTO bedeutet dies, daß man über das Zwischenmenü schneller auf andere Datenbanken umsteigen kann als durch Neuaufruf des Programms vom Hauptmenü aus. Diese Datenbanken müssen auf demselben Unterverzeichnis liegen und auf derselben Konfiguration beruhen.

-S 1, 9, c

Einzelplatzbetrieb [obsolet]

(Single user) : Standardmäßig arbeiten PRESTO und UPD im Mehrplatzbetrieb. Auf Einzelplatzsystemen kann das funktionieren (wenn vorher der MS-DOS-Befehl SHARE gegeben wurde). Wenn nicht, ist -S nötig.

Ab V15 ist **-S entbehrlich:** PRESTO und UPD merken selbst, daß Einzelplatzmodus vorliegt.

-s

Suchbegriff

1

-s xyz mit beliebiger Zeichenfolge xyz bewirkt, daß der Index für die erste Anzeige an der Stelle abc aufgeblättert wird. Sonst erscheint eine zufällige Stelle.

-s"|4m" wenn z.B. der Indexeinstieg im Register 4 beim Buchstaben m sein soll.

4, 5

Der für eine Selektion im Such- oder Importprogramm zu benutzende Suchbegriff kann hierdurch dem Programm von außen mitgegeben werden (sonst wird er wie gewohnt abgefragt). *Beispiel:*

```
-s (#76XG1982)+(#74Xlondon)
```

Selektiert werden dann die Sätze mit Erscheinungsdatum ab 1983 und Erscheinungsort London. (Man beachte das 'X' anstelle des Leerzeichens. dieses ist nötig, weil ein Optionsargument kein Leerzeichen enthalten darf. Ferner sind statt '<' und '>' die Buchstaben 'S' bzw. 'G' zu verwenden, weil sowohl unter MS-DOS als auch UNIX die Zeichen '<' und '>' in Programmaufrufen eine Sonderfunktion haben: sie lenken die Ein- bzw. Ausgabe des Programms auf eine Datei um.)

q

-s abc mit beliebiger Zeichenfolge abc bewirkt, daß nur der Abschnitt ab **abc** des Index bearbeitet wird. Der Endpunkt wird durch das große S bestimmt:

-S xyz . In Kombination mit **-w** oder **-W** und **-x** kann QRIX einen ganz bestimmten Indexabschnitt zur Anzeige oder Ausgabe bringen (Funktion -fd).

-t 1

Phrasendatei automatisch laden

-t xyz lädt die Phrasendatei xyz.APH sofort nach dem Start. Sonst müßte man sie vom Editor aus mit Befehl #p laden (→ 3.3).

-T 7, 9, q

Testmodus einschalten

Programm gibt einige Testmeldungen aus, zur besseren Kontrolle des Ablaufs.

Option	anwendbar bei Programm	Bedeutung
	1:presto 4/6:srch 5:import 7:index 8:asort 9:upd c:*CockPit* q:qrix	

-U 1-9 **Uservariablen für Export vorbesetzen** [→ 10.2.6.2/.3]
Hinter -U muß eine Angabe *xyTEXT* folgen. *TEXT* wird in die Variable `#uxy` kopiert. Diese Option ist wiederholbar, d.h. man kann mehrere Variablen vorbesetzen. Aber:
Achtung: bei DOS wird leicht die maximale Befehlslänge von 128 Zeichen überschritten!

-u 9 **Update-Datei**
Hinter -u ist der Name der einzuspeisenden Datei anzugeben. Wenn es eine .cLG-Datei ist, werden neue und vorhandene Datensätze bei gleichem Primärschlüssel gemischt, wenn es eine .LOG-Datei ist, muß die Datenbank in dem Moment aus einer Sicherungskopie bestehen, die genau vor Beginn der .LOG-Datei gezogen wurde (→ 0.7).

-u*i* 8 **Sortierfelder beseitigen**
i=1: Nur #u1, i=2: #u1 und #u2, i=3: auch den Code 01 am Satzanfang beseitigen.

-V 1 **Zeilenanzahl für Registeranzeige** (besonders für APAC; Wert <23)
Standard: ist `-V22. Wenn es auf 24-Zeilen-Terminals laufen soll, setzt man -V21 .`

-v 4, 5 **Anzeige der Datensätze** (view)
`-v1 : gewünscht`
`-v0 : nicht gewünscht`
Soll der Selektions- bzw. Importvorgang ohne visuelle Beobachtung (und dann auch schneller!) ablaufen, so gibt man -v0. Zum Testen ist natürlich -v1 nützlich.

-w 7, q **Auswahl der zu bearbeitenden** bzw. der

-W 7, q **wegzulassenden Register** : vgl. dazu die Ausführungen in Kap. 7.5

-X 7 **Umfang der Datenbank** [wenn nicht genau bekannt: größere Zahl angeben]
Hinter -X gibt man die Anzahl der Datensätze, aus der die neue Datenbank bestehen wird. Notwendig nur bei den Funktionen -fr und -fi, wenn die .TBL-Datei ebenfalls zu erneuern ist. Dadurch werden Fehler durch korrumpierte Datensätze ausgeschlossen.

-x 1, q **Schwelle für Indexanzeige**
Bei **`-x200`** zeigt PRESTO nur bis zu 200 Treffer im Index an. Zeilen mit höherer Trefferzahl werden mit **`>200`** gekennzeichnet. Das Programm QRIX reagiert genau umgekehrt: Nur Einträge mit mehr als 200 Treffern werden angezeigt.

7 **Unterbrechung vor Aufruf von QRIX**
INDEX ruft nach Abschluß seiner Arbeit automatisch QRIX auf (falls nötig), um die entstandenen Zwischendateien zu mischen. Bei `-x` unterbleibt dies, aber man wird informiert, wie QRIX in dem Falle aufzurufen ist, damit es den Indexaufbau vollendet.

-x 9 **Name der UPD-Protokolldatei** (Standard: UPRO)
Das Protokoll wird in die hinter `-x` angegebene Datei geschrieben

-y 1, 7, 9 **Pfad der Indexdatei**
Wenn aufgrund von Platzproblemen der Index und die Kurztiteldatei einer Datenbank (.cDX und .STL) auf einem anderen Pfad oder Laufwerk liegen, gebe man mit Option `-y` die komplette Pfadangabe dem Programm mit: z.B. `-y e:\abc`

-z 1, 7, 9 **Maximalwert für Dateigröße**
Das Indexprogramm erstellt Dateien (Typ .cLD) bis zu 16 MB Größe oder (ab V20) einem Vielfachen davon (→ Kap.10.2.1.3, Parameter **ii**). Wird dieser Wert erreicht, beginnt es automatisch mit einer neuen. Der Grund ist, daß die einzelne Datenbankdatei nicht größer sein darf. Da jedoch eine Datenbank bis zu 255 solche Dateien haben kann, liegt hierin keine Einschränkung der Datenbankgröße. (Der Index umfaßt stets alle zugehörigen Dateien.) [Ganz früher setzte man z.B. **`-z1400000`**, wenn bei Erreichen von 1.400.000 Byte eine neue Datei anfangen sollte. Dann paßte hinterher jede .ALD-Datei noch auf eine Diskette!]

12.2 Beispiele für Aufrufe

Das System der Programmaufrufe ermöglicht es, das ***CockPit*** um eigene Routinen zu ergänzen, oder sogar für eine spezifische Anwendung eine eigene "Benutzeroberfläche" maßzuschneidern. Wegen der umfassenden Parametrierung der Programme kann kein für alle Anwender gleichermaßen geeignetes Schema erstellt werden, vielmehr sollen die hier gezeigten Beispiele Anregungen für die Ausgestaltung der eigenen Oberfläche geben. Studieren Sie die Batchfiles selbst und ändern Sie alles für Ihre Zwecke beliebig ab.Die meisten Batchdateien werden für das Produzieren von Listen geschrieben, weil dabei in der Regel mehrere Programmläufe kombiniert werden müssen. Mitgeliefert werden SR-LIST.BAT, PR-LIST.BAT, QUEX.BAT, QUANT.BAT (siehe Kap. 6). Wir drucken hier nur SR-LIST.BAT vollständig als Beispiel ab:

1. Volltext-Selektion und sortierte Ausgabe

`sr-list`

Die per ***CockPit*** vorgewählte Datenbasis wird durchsucht (der Suchbegriff wird dann vom Programm noch abgefragt), die Ergebnisse sortierfähig (mit Parameter SORT.APR) auf eine Zwischendatei geschrieben, diese dann sortiert und mit einem SRCH-Aufruf schließlich mit Seitenumbruch (Parameter PRINT.APR) als fertige Liste auf eine Datei geschrieben, die dann mit PRINT ausgegeben oder vorher noch editiert werden kann. Und so sieht die Batchdatei `SR-LIST.BAT` aus:

```
rem Allgemeine Stapeldatei für Listenproduktion, vom CockPit zu starten
rem nach Auswahl der Sortiervorbereitung und Druckaufbereitung
rem Volltextsuche, Sortieren, Aufbereiten
rem alle Dateien werden auf's aktuelle Verzeichnis geschrieben!
del liste
srch -m0 -e SORT/uuu.%-K1%lg -f4
asort uuu.%-K%lg sss.%-K1%lg
del uuu.%-K1%lg
srch -m0 -e PRINT/LISTE -f4 -dsss -v0 -b%-D%\%-B% -s0
del sss.%-K1%lg
echo Produktion beendet, Ergebnis = LISTE
echo Liste kann nach dem nächsten Tastendruck betrachtet werden
pause
v LISTE
```

2. Selektion per Schnellzugriff und sortierte Ausgabe (DOS)

`pr-list`

Die per ***CockPit*** vorgewählte Datenbasis wird zunächst von PRESTO zur Benutzung bereitgestellt. Die erste Zeile von `PR-LIST.BAT sieht so aus:`

```
presto -f1 -e sort/uuu.%-K%lg
```

`...`

der Rest entspricht `SR-LIST.BAT.`

Man stellt Ergebnismengen zusammen und exportiert sie mit F4. Dabei werden die Ergebnisse sortierfähig auf eine Zwischendatei UUU.ALG geschrieben. Wenn man schließlich aus PRESTO aussteigt, läuft der Rest genauso wie oben bei **`sr-list`** beschrieben.

Wählen Sie vom ***CockPit*** aus (**`µo 1 bzw. 3`**) eine andere Sortier- bzw. Druck-Parameterdatei, um die Ordnung bzw. Druckgestaltung zu verändern. Diese Dateien werden auf SORT.APR bzw. PRINT.APR kopiert.

3. Umwandlung von Fremddaten und Einspeisen in eigene Datenbank

Nehmen wir an, eine Datei EXPORT.TXT soll vom Laufwerk A: gelesen, umgewandelt und in Datei 8 unserer Datenbank CAT (also CAT_8.ALD) auf dem Verzeichnis KATALOG eingespeist werden. EXPORT.DOC sei eine VLB-Download-Datei, und wir haben dafür die Import-Parameterdatei VLB.AIM auf unserem Programmverzeichnis. Der folgende Zweizeiler leistet die Arbeit:

```
import -f5 -ivlb -d a:export.txt -ka -s0 -m0 -v0 -e i-1/hhhhh.alg
upd -fm01 -dhhhhh.alg -dkatalog\cat -n8 -ka
```

Läßt man eine der Optionen weg, wird die betreffende Angabe abgefragt.

4. Zwei Hilfsprogramme für Stapeldateien

Zur Standardlieferung des DOS-Systems gehören zwei kleine Hilfsprogramme, die zum Erstellen von interaktiven Batchdateien benötigt werden. (DOS selbst ist hier erstaunlich defizitär.)

GK.COM

Hiermit kann man in einer Batchdatei eine Tastaturabfrage vornehmen. Das Strickmuster dafür sieht so aus:

```
echo Drücken Sie a = Vorgang1   b = Vorgang2   c = Vorgang3
gk abc                                  es geht nur weiter, wenn a,b oder c gedrückt wird
if errorlevel 3 goto vorg3   c löst den ERRORLEVEL 3 aus
if errorlevel 2 goto vorg2   b den Wert 2, a den Wert 1. Man muß die Werte in dieser
[hier folgt Vorgang1]                          Reihenfolge abarbeiten!
:vorg2
[hier folgt Vorgang2]
usw.
```

ACREPLY.COM (DOS)

Dieses Programm wartet auf eine Eingabe, die mit <Enter> abgeschlossen wird. Es erzeugt eine kleine Batchdatei (nur eine Zeile), die man anschließend aufrufen kann.
Strickmuster:

```
echo Auf welchem Verzeichnis liegt Ihre Datenbank?      Frage an den Benutzer
acreply verz 13 setv.bat        Auf Eingabe von maximal 12 (!) Zeichen warten
                    Es entsteht setv.bat mit Inhalt   set verz=eingabe
call setv.bat       Den set-Befehl ausführen lassen
presto -d%verz% -n1 -a3 -S   Hier wird die Eingabe benutzt, die jetzt in   verz steht
```

Wenn man keine Länge angibt, wird 128 genommen, wenn man den Dateinamen (hier setv.bat) nicht angibt, wird ACANTW.BAT genommen.

Solche Hilfsprogramme mußten nur für DOS erstellt werden. Unter UNIX gibt es in der Shell-Sprache genügend Möglichkeiten, Antworten vom Benutzer entgegenzunehmen und in Variablen abzulegen.

Anhang A Konfiguration

A.0 Vorbemerkungen

a99: geben Sie **h aca-1** ein

Eine Datenbanksoftware benötigt Angaben, wie der Inhalt einer konkreten Datenbank beschaffen ist. Zum Beispiel muß die Software die verwendeten Feldbezeichnungen (= Kategorienummern) kennen, um damit arbeiten zu können. ***allegro*** entnimmt solche Angaben aus einer **Konfigurationsdatei** **c.CFG**, die oft einfach nur **Konfiguration** **c** genannt wird. Die Programme lesen diese Datei beim Start und wissen dann über die fragliche Datenbank Bescheid. Alle Parameterdateien (Export und Import) setzen jeweils eine bestimmte Konfiguration voraus und funktionieren nur mit dieser zusammen. Z.B.: Parameterdateien des Typs `.MPR` gehören zur Konfiguration `M.CFG` usw.

Wer braucht dieses Kapitel? Auf jeden Fall, wer eine ganz neue Datenbank gestalten will, aber auch, wer existierende Datenbanken ausbauen, also neue Felder definieren und Abfragelisten erweitern will. Wenn man eine ganz neue Datenbank zu gestalten hat, kann man mit dem Hilfsprogramm PRONTO.BAT (→Kap.1.1.2) beginnen. Damit wird eine Konfiguration, aber auch Anzeige und Indexparameter, in einem Dialog erstellt. Anschließend kann man sie weiter verbessern.
Anmerkung: Bei anderen Datenbanksystemen redet man statt von einer Konfiguration oft von einer **Datenbankdefinition**. Eine solche enthält aber meist auch schon Angaben über die Indexierung. Diese sind bei ***allegro*** völlig getrennt von der Konfiguration: die **Indexdefinition** steht in einer Index-Parameterdatei (→ 0.3 Typ *i*PI, auch → 10.)
Der ***allegro***-Anwender kann eine vorhandene Konfiguration benutzen, diese abwandeln oder für seine Zwecke eine ganz neue erstellen. (***CockPit*** : **µo k** <Auswahl> <neue Version erstellen>)
Mitgeliefert werden folgende Konfigurationen: (die mit dem '$' funktionieren nur ab Version 13)

$A	Standard (abgedruckt in A.4; eine Weiterentwicklung des ehemaligen Niedersächsischen Schemas ("NMN"), Kurzbeschreibung der Kategorien siehe Anhang B) Vollständige Beschreibung : http://www.allegro-c.de/doku/form2004/
$D	MAB (das Austauschformat des deutschen Bibliothekswesens)
N	sog. **Neutralformat** (siehe http://www.allegro-c.de/neutral/) Dazu gibt es ein eigenes Installationspaket, in dem alles ist, was man braucht.
$P	Pica-Schema (sog. Pica3-System des Niedersächsischen Verbundes)
$U	US-MARC (das weltweit meistverwendete Schema)

Hinweis: Nur die Parameter zu N.CFG sind eingerichtet auf die Verwendung des Windows-Zeichencodes (ISO-8859 mit allegro-Zusatzcodes, sog. "OstWest"-Zeichensatz in Windows-Version, s. Anh. E) statt DOS-ASCII. Beide Zeichensätze sind 100% kompatibel, d.h. ineinander umwandelbar. Aber man kann N-Datenbanken weniger gut mit dem DOS-Programm PRESTO bearbeiten!

Sie vergleichen am besten die folgende Beschreibung mit den kommentierten Dateien $A.CFG, $U.CFG oder $P.CFG, in denen alle Einzelheiten vorkommen.
Die folgenden Abschnitte behandeln alle **Konfigurationsbefehle**, die in einer .CFG-Datei auftreten können.
Die Groß-/Kleinschreibung ist bei den Befehlen genau zu beachten.
Die angegebenen Standardwerte werden automatisch wirksam, wenn man den betreffenden Befehl wegläßt.
Die Buchstaben am rechten Rand besagen, ob man den jeweiligen Wert bei einer schon laufenden Datenbank ändern kann:

Ä Änderung ohne Bedenken möglich	**Ä**
V Vorsicht! (Knowhow nötig; u.U. könnten nach einer Änderung Maßnahmen wie Neu-Indexierung fällig sein)	**V**
N Nicht ändern, nachdem die Datenbank angelegt ist (Gefahr für die Funktion der Datenbank.)	**N**

Kommentare in der CFG

Zeilen, die mit **Leerzeichen** beginnen, gelten als Kommentarzeilen und werden vom Programm nicht beachtet. Zwei oder mehr Leerzeichen hintereinander *innerhalb* einer Zeile leiten ebenfalls einen Kommentar ein! Man beachte das besonders bei der Abfrageliste: nur zwischen Anführungszeichen dürfen Mehrfach-Leerzeichen *vorkommen*.

A.1 Konfigurationsbefehle

Jeder Befehl besteht aus einem Befehls**zeichen**, das am Zeilenanfang stehen muß, und dem Befehls**inhalt**, der ohne Zwischenraum darauf folgt. Manche Befehle haben Ziffern oder Zahlen, andere haben Text als Inhalt.

A.1.1 Festlegung der Struktur der Datenfelder

Jedes Datenfeld (= Kategorie) einer Datenbank ist eine Zeichenfolge. Sie beginnt mit einer Kategorienummer (engl. "tag") und enthält einen beliebig langen Text (ohne Zeilentrennungen), der stets auf einer bestimmten Position hinter der Kategorienummer beginnt. Als Standard wird der einfachste Fall angenommen: die Kategorienummern sind zweistellig, der Text beginnt immer auf der vierten Stelle, zwischen Nummer und Text ist ein Leerzeichen. Wenn die Kategorie mehrfach auftreten kann, steht an der dritten Stelle ein unterscheidendes Zeichen (Wiederholungszeichen).
Liegt dieser Normalfall nicht vor, muß man mit zwei Befehlen festlegen, wie lang die Kategorienummern sind und an welcher Position der Text beginnt. Diese somit sehr wichtigen zwei Befehle stellen wir an den Anfang:

t*ziffer* Breite der Kategorienummern **N**

Möglich sind hier für ***ziffer*** die Werte 2 bis 6.
Wenn **t** fehlt, nehmen die Programme an, daß es eine .CFG "alter Art" ist (→ A.1.2, A.5).
Standard: **t2**

k*pos* Position des Kategorie-Textanfangs **N**

Das Zeichen '#' steht auf Position 0. Will man mit einem 3stelligen Kategorienschema arbeiten und hinter der Kategorienummer noch einen Indikator unterbringen (wie bei MAB) oder sogar zwei (wie bei MARC), sind hier `k6` bzw. `k7` sinnvoll, ohne Indikator wäre `k5` zu schreiben. Der `k`-Wert muß größer sein als der `t`-Wert.
Standard: **k4**

Beispiel: (trifft zu auf US-MARC, siehe $U.CFG)

t3 3stellige Kategorienummern
k7 Text beginnt auf Position 7
Jedes Datenfeld wird dadurch auf diese Struktur festgelegt:

```
#nnnwijTEXT...
 nnn                 Kategorienummer (3stellig)
    w                Wiederholungskennung (normal: Leerzeichen)
     ij              Indikatoren (können dem Feld eine Zusatzbedeutung geben)
       TEXT          Kategorie-Inhalt (beginnt auf der 7. Stelle)
```

Man erkennt: die Anzahl 2 der Indikatoren ergibt sich aus den Befehlen **t3** und **k7** automatisch.
Im Standardfall **t2** und **k4** gibt es keine Indikatoren, nur die Wiederholungskennung auf Position 3.

A.1.2 Kategoriedeskriptoren

Die Befehle t und k legen nur die allgemeine Form fest. Darauf muß eine genaue Aufstellung aller zulässigen Feldbezeichnungen (= Kategorienummern) folgen. Jede Kategorienummer muß durch eine Zeile beschrieben werden, die mit einem **#** beginnt. Eine solche Zeile wird **Kategoriedeskriptor** genannt, weil sie die Eigenschaften einer Kategorie beschreibt.
Achtung: **m-Befehle müssen vor die erste solche Zeile!** (→ A.1.3)

#... Kategoriedeskriptor

Die Reihenfolge dieser Deskriptoren, nicht die numerische Folge der Kategorienummern, legt die interne Speicherung in den Datensätzen fest. Sie können später jederzeit (auch im laufenden Betrieb!) neue Nummern zwischenfügen (d.h. neue Felder definieren!) oder auch (ab V14c) die Reihenfolge ändern.
Hier als erstes Beispiel eine typische Zeile aus der Konfiguration $U.CFG für USMARC:

```
#245"Title statement"$M$Aabchnp$Rnp$I01$J0123456789$Cac
```

Diese Zeile teilt dem Programm alle wesentlichen Eigenschaften der Kategorie #245 mit.

Zunächst aber noch ein kurzer Rückblick: bis Version 12.2 sah in der .CFG-Datei eine Kategorie-Definitionszeile so aus:

`#20c21a22c23c24c25a27a28a30a31a32a33a39a40d41d42d43d`

Damit wurde ausgedrückt, daß **`#20, #21, #22, ..., #43`** erlaubte Kategorienummern und in dieser Reihenfolge zu speichern seien, daß #20 der Prüfroutine c zu unterwerfen sei (Artikel am Anfang), #21 aber keiner Prüfung, #40 hingegen der Prüfroutine d (ob "Komma Leerzeichen" vorkommt). Weitere Einzelheiten konnte man nicht festlegen. Jede Kategorie war automatisch beliebig mehrfach belegbar durch schlichtes Hinzufügen eines unterscheidenden dritten Zeichens hinter der Kategorienummer. So wurden auch #20a, #20ö, #20§ usw. vom Programm ohne weiteres akzeptiert. Daher konnte man für ein dreistelliges Kategoriesystem wie MAB oder MARC die dritte Stelle nicht festlegen, und eine dreistellige Feldnummer war als solche nicht wiederholbar. Man konnte #10y oder #33R eingeben, was ja nun wirklich keine MAB-Kategorien sind, aber das Programm konnte man nicht bewegen, daran Anstoß zu nehmen. Ab Version 13 ist genau dieses und noch manches andere möglich. (→ A.5 : Umstieg auf Version 13)

Ab Version 13 kann es **für jede Kategorie eine eigene Definitionszeile** geben, während bisher eine einzige Zeile gleich ein Dutzend und mehr Kategorien (17 Stück im obigen Beispiel) definieren konnte. Eine Definitionszeile der neuen Art, auch **Felddeskriptor** oder **Kategoriedeskriptor** genannt, muß formal so aussehen (wenn man alle Möglichkeiten nutzt):

`#`*`tag`***`"`***`name`***`"$M`***`mmm`***`$A`***`aaa`***`$N`***`nnn`***`$R`***`rrr`***`$C`***`scscsc`***`$I`***`iii`***`$J`***`jjj`***`$P`***`p`***`$F"`***`fff`***`"`**

Auf den ersten Blick ist das eine verwirrende Fülle von Angaben. Jeder Teil außer *#tag* kann entfallen, manche werden nur selten gebraucht.

Die *kursiv* gedruckten Teile sind variabel, d.h. vom Anwender einzusetzen, die anderen haben festgelegte Bedeutungen als Definitionsteile. Das **#** am Zeilenanfang leitet, wie früher, die Deskriptorzeile ein. Darauf folgen:

tag Kategorienummer ("Kategorienummer" heißt auf Englisch "tag"). Es muß keine Nummer sein, nur für die erste Stelle wird empfohlen, daß es eine Ziffer ist. Die Reihenfolge der Deskriptoren, nicht deren ASCII-Werte, bestimmt die Reihenfolge der Abspeicherung. Die nachfolgenden Angaben beschreiben alle Eigenschaften:

"*name*" Frei wählbare Bezeichnung für die Kategorie. Keine Längenbeschränkung. Für DOS-Version ohne Funktion, bis auf den Pauschal-Exportbefehl #L (→ 10.2.6.1). Bei a99 hat diese Angabe mehrere Funktionen, z.B. auch für den dort möglichen XML-Export: es werden dann XML-Tags daraus gemacht.

`$M`*mmm* Erlaubte **M**ehrfachkennungen; hier sind alle Zeichen aufzuführen, die als Wiederholungszeichen hinter der Kategorienummer auftreten dürfen. Steht hinter **`$M`** kein weiteres Zeichen, sondern gleich z.B. **`$A`**, dann ist die Kategorie nicht wiederholbar. Fehlt dagegen **`$M`***mmm*, ist die Kategorie beliebig wiederholbar! (Die Angabe wird von den Programmen INDEX und UPD nicht beachtet, nur von PRESTO, APAC, a99/alcarta)

`$A`*aaa* **A**ngebbare Teilfelder. Vor allem für MARC-Formate relevant. Die mit *aaa* angegebenen Zeichen und keine anderen sind als Teilfeldkennungen in der Kategorie verwendbar. Fehlt die Angabe, ist jedes Teilfeld erlaubt.

`$N`*nnn* **N**otwendige Teilfelder. Diese müssen vorkommen, wenn die Kategorie überhaupt besetzt wird. Fehlt `$N`*nnn*, sind alle Teilfelder optional, d.h. keins muß vorkommen.

`$R`*rrr* **R**epetierbare (mehrfach erlaubte) Teilfelder. Nur die hiermit definierten Kennungen dürfen mehrfach als Teilfeld innerhalb der Kategorie vorkommen. Fehlt `$R`*rrr*, dürfen alle mehrfach vorkommen. Steht hinter **`$R`** nichts (d.h. fehlt *rrr*), darf jedes höchstens einmal vorkommen.

`$C`*scscsc* Prüfmethoden (**C**hecks). Dies sind Paare aus je einem Teilfeldzeichen *s* und einem Prüfbuchstaben *c*. Kommt das Teilfeld $*s* vor, wird es der Prüfung *c* unterworfen. Die Liste der zulässigen Prüfbuchstaben folgt unten.

`$I`*iii* **I**ndikator 1: erlaubte Zeichen. Das zweite Zeichen hinter der Kategorienummer (das erste ist ja die Mehrfachkennung) wird als **Indikator** angesehen und muß in der Liste *iii* vorkommen. Das Leerzeichen ist immer erlaubt. Fehlt $*Iiii*, sind alle Zeichen erlaubt. MAB-Formate haben einen Indikator (`t3/k6`).

`$J`*jjj* Indikator 2: erlaubte Zeichen. Das dritte Zeichen hinter der Kategorienummer wird als zweiter Indikator angesehen und entsprechend geprüft. MARC-Formate haben zwei Indikatoren (`t3/k7`).

`$P`*p* Eigenschaftszahl. Eine Zahl zwischen 0 und 255. Wenn diese Angabe fehlt, wird als Standardwert 7 angenommen. Die Zahl errechnet sich als Summe aus folgenden Eigenschaften, die ein Feld haben oder nicht haben kann: (im Standardfall **`$`**P7 sind also die ersten drei Eigenschaften gegeben, bei `$P15` alle vier)
1 : Feld ist **eingebbar** (zugleich muß 4 gesetzt sein, d.h. es muß `$P5` heißen)
2 : Feld ist **korrigierbar** (zugleich muß 4 gesetzt sein, d.h. es muß `$P7` oder `$P6` heißen)
4 : Feld ist **sichtbar** (wird normalerweise angezeigt)
8 : **Mehrfach-Leerzeichen** sind erlaubt (normalerweise werden sie entfernt)
Folglich würde die Angabe **`$P5`** bedeuten: das Feld ist eingebbar, aber nicht mehr änderbar.

`$F"`*fff*`"` **F**ormale Maske für feste Feldinhalte. Damit kann man festlegen, wie ein Feldinhalt formal zusammengesetzt sein kann. *fff* muß nur dann in Anführungszeichen stehen, wenn darin Mehrfach-Leerzeichen vorkommen.

Die letzte Angabe, **$F***fff* , ist vor allem gedacht zur genaueren Definition von Feldern fester Länge. *fff* besteht aus einer beliebigen Anzahl von Zeichen, die den zulässigen Inhalt der einzelnen Positionen des Feldes charakterisieren, und zwar wie folgt:

A	Buchstabe oder Leerzeichen
N	Buchstabe, Ziffer oder Leerzeichen
9	Ziffer (0 - 9)
#	Ziffer, Leerzeichen, oder '+' oder '-'
L	'1' oder '0' (logisches "Richtig" / "Falsch")
X	jedes beliebige Zeichen darf statt X auf dieser Position stehen
c	auf dieser Position darf nur das Zeichen c stehen (c=beliebiges ASCII-Zeichen außer AN9#LX)

Nehmen wir als Beispiel die Signatur. Angenommen, #90 ist unsere Signaturkategorie, und unsere Signaturen bestehen aus zwei Buchstaben, einem Leerzeichen und vier Ziffern. Dann wäre zu schreiben:

```
#90"Signatur"$FAA 9999
```

In der Praxis dürfte gerade bei Signaturen die Methode meist nicht anwendbar sein, denn feste Strukturen gibt es dafür normalerweise nicht. Und Ausnahmen oder Alternativen läßt eine solche Definition nicht zu. Wenn es aber z.B. so ist, daß mindestens die ersten zwei Zeichen Buchstaben sein müssen, könnte man immerhin schreiben **$FAA** . Denn **hinter** den festgelegten Positionen kann immer noch beliebiger weiterer Text eingegeben werden, der dann nicht geprüft wird.

Die Prüfroutinen, auf die man in der Angabe **$C***cscscsc* zurückgreifen kann, sind unten aufgelistet. Die Vorteile gegenüber den Versionen vor V13: ein Feld kann mehreren Prüfungen unterzogen werden, und jede Prüfung kann auf ein Teilfeld beschränkt werden. Hinter dem 'C' stehen, wie gesagt, Zeichen**paare**. Das erste muß immer ein Teilfeldcode sein, der in der betreffenden Kategorie vorkommen kann, das zweite ein Prüfbuchstabe. Wenn das erste ein Leerzeichen ist, wird die Prüfung auf die gesamte Kategorie angewendet. Wenn $C... fehlt, wird das Datenfeld ohne Prüfung akzeptiert.
Spezialfall: ist **s** ein Leerzeichen, wird die Prüfroutine **c** auf jedes Teilfeld angewendet.

b: reserviert für die Gruppe #0 (→ A.4, Beispiel A.CFG)

c: Das erste Wort wird geprüft. Wenn es ein Artikel ist, fragt das Programm, ob er ordnen soll, wenn nicht: Einfügen des "Nichtsortierzeichens" (siehe Befehl N) vor den Artikel (Modus n0) bzw. Einschluß des Artikels in zwei solche Zeichen (Modus n1). Die Liste der Artikel ist unter dem Befehl **d** (siehe unten) anzugeben.

d: Prüfung, ob **", "** im Eingabetext vorkommt (für Personennamen sinnvoll)

e: Prüfung, ob im Text eine Jahreszahl zwischen 1449 und dem aktuellen Jahr +2 vorkommt

f: Prüfung, ob **" ; "** im Eingabetext vorkommt.

g: Berechnung der Prüfziffer in der ISBN

h: dasselbe für die ISSN. Geprüft wird nur, wenn in der Kategorie ein Bindestrich vorkommt und 4 Zeichen davor eine Ziffer steht. Ansonsten wird die Kategorie ungeprüft übernommen.

t: Tagesdatum: die Eingabe muß von der Form `jjjj.mm.tt` sein (sortierfähig!)

Ab V14 gibt es darüber hinaus die Möglichkeit, selbst Eingabeprüfungen zu programmieren (auch "Programmierte Validierungen" genannt). Dazu ist ein Abschnitt in den Indexparametern einzurichten (→ 10.2.8).

Beispiele für Kategoriedeskriptoren:

MAB: `#100"Verfasser"$M$Ibcef$C d`
Die Kategorie #100 enthält den Verfasser. Sie ist nicht wiederholbar (der 2. Verfasser kommt in #104). Als Indikator kann einer der Werte bcef auftreten. Der Feldinhalt als Ganzes wird nach Methode d geprüft.

MARC: `#100"Main Entry Person"$M$Aabcdefgklnpqtu46$Rcnq$I0123$J01$Cad`
Die Kategorie #100 mit der Bezeichnung "Main Entry Person" ist nicht wiederholbar; sie kann die Teilfelder abcdefgklnpqtu46 haben, dabei dürfen c, n und q mehrfach auftreten. Der erste Indikator kann einer der Werte 0123 sein, der zweite 0 oder 1. Das Teilfeld a wird nach Methode d geprüft.

A.1.3 Sonstige Konfigurationsbefehle

$*Code* **Unterfeld-Trennzeichen** **N**

Hier ist der Code anzugeben, der als Steuerzeichen für die Einleitung von Unterfeldern benutzt wird. So würde `etwa` **$**`36` das Dollarzeichen (ASCII-Code 36) dafür festlegen, d.h. man hätte z.B. `$a` statt ▼`a`. Ein Manipulationsbefehl (→10.2.6.3) beim Export (egal ob $a oder ▼`a`) sucht im Arbeitstext nach dem mit diesem Befehl angegebenen *Code*. Eingabe des Zeichens ▼ geht unter DOS mit Strg+-, Windows (deutsch): AltGr+2 (die '2' oben links) oder Alt+31
Standard (z.B. in `$a.cfg`) ist **$31**, also das Zeichen ▼

cn **Kategorie für Neu-Erfassungsdatum festlegen** **V**

ce **Kategorie für Korrekturdatum festlegen** (Datum der letzten Korrektur)

Hinter cn bzw. ce gibt man die Nummer der Kategorie, in der das betreffende Datum registriert werden soll, z.B. `cn99n` und `cn99e` . Wahlweise kann cn oder ce (oder beide) fehlen, wenn die Datumsspeicherung nicht erwünscht ist. Es ist zweckmäßig, dann in der Kategorieliste `#99...$P5` zu setzen, dann sind nachträgliche Änderungen von Hand ausgeschlossen. Wird beim Aufruf Option **-O** gegeben ("Operator", → Kap.12.1), setzt das Programm jeweils den Bearbeitercode hinter das Datum. (S.a. Befehl D)

cg **Kategorie für automatische Nummernvergabe festlegen** **Ä**

Hinter `cg` gibt man die Kategorienummer `xyz` an, in die das Programm automatisch eine laufende Nummer eintragen soll. Ist kein `cg`-Befehl vorhanden, erfolgt **keine** Nummernvergabe. Meist wird `cg00` gesetzt.
Beispiel: **cg15z** . Dies bedeutet: das Programm schaut jedesmal vor dem Abspeichern eines Satzes nach, ob darin die Kategorie **#15z** besetzt ist. Trifft dies zu **und** kommt ein Fragezeichen darin vor, wird an dessen Stelle die nächste noch nicht vergebene Nummer eingesetzt; siehe unten. Wenn `#xyz` aber im Datensatz nicht vorkommt, wird der Befehl **ci** zur Nummernvergabe herangezogen: (Den ersten Satz von Hand numerieren!)

Hinweis: *a99* ermöglicht eine flexiblere Nummernvergabe per FLEX: `h nummer` eingeben

ci **Standardform der zu vergebenden Nummer** **Ä**

Hinter **ci** muß eine Angabe der Form **Rabc?kABC** stehen. Diese Angabe legt fest, wie im Normalfall die zu vergebende Nummer strukturiert und in welchem Register sie indexiert sein soll. D.h. die Indexparameter müssen aus `#xyz` einen Schlüssel in dieser Gestalt machen.
Die Bedeutung der Einzelteile:

R Nummer des Registers (1-9 bzw. **:** für Register 10 oder **;** für Register 11)

abc beliebige Zeichen, die vor der Nummer stehen sollen, genau in der Schreibweise der Registereinträge (kann entfallen, dann steht die Nummer unmittelbar als solche im Register.)

?k an dieser Position soll die Nummer stehen (die eigentliche Vergabe erfolgt erst im Augenblick des Speicherns!). Wenn hinter **?** eine Ziffer **k** steht, wird die Nummer auf soviele Stellen mit führenden Nullen aufgefüllt. Wenn **k** fehlt, bekommt die Nummer keine führenden Nullen. Ein **?** muß vorkommen, sonst hat diese Sache keinen Sinn.

ABC beliebige Zeichen, die hinter der Nummer stehen sollen (kann entfallen, selten sinnvoll)

Nochmals: der Befehl **ci** tritt **nur** in Kraft, wenn die durch **cg** benannte Kategorie **nicht** besetzt ist. Wenn diese belegt ist, hängt es von ihrem Inhalt ab, was dann passiert. Der Inhalt kann so strukturiert sein:

`#xzy` **|R#nnfabc?kABC**

Die einzelnen Teile können jeweils fehlen; sie bedeuten dasselbe wie oben bei **ci** angegeben; zusätzlich kann hier aber die Angabe **#nnf** vorhanden sein: dann wird die zu ermittelnde Nummer in **#nnf** und nicht in der durch `cg` benannten Kategorie `#xyz` gespeichert.
Der Sinn dieses komplizierten Arrangements: `cg` sagt dem Programm nur, daß die Kategorie `#xzy` eine Anweisung enthält, auf welche Weise beim Abspeichern des Satzes eine Nummer zu vergeben ist. So ist es auch möglich, unterschiedlichen Satztypen ganz verschieden strukturierte und indexierte Nummern zu verpassen, indem man in dem jeweiligen Abschnitt der Abfrageliste (→ A.2) eine Kategorie **#xzy** mit jeweils geeignetem Inhalt nach dem angegebenen Muster unterbringt.
Das Verfahren ist z.B. auch für eine automatische Signaturvergabe anwendbar (aber nicht für Signatur UND Inventarnummer zugleich). Wenn die Signatur im Register 8 indexiert ist, kann man setzen

ci8a?5
cg90

und in #90 schreibt man jeweils nur die Signaturgruppe und ein '?' an der Stelle, wo die laufende Nummer hin muß, also z.B. `#90 Mpg ?` , dann wird im Register 8 die letzte Nummer unter Mpg gesucht und das Fragezeichen durch die nächsthöhere ersetzt. Das passiert während des Speichervorgangs, wo das Register blockiert ist. Doppelte Vergabe einer Nummer im Mehrplatzbetrieb wird so ausgeschlossen. (Die Angabe a?5 bleibt ohne Wirkung, wenn die #90 immer ein ? enthält!)

CU **Intern wird Unicode verwendet (D = DOS, W = Win)** [ab V33.2]

Dies ist ein Signal für's Programm bei der Anwendung der Tabelle `o.apt`: UTF-Codes, die mit >191 beginnen (2 Byte) bzw. >224 (3 Byte), werden nicht umgewandelt. (Codes D und W haben keine Wirkung.)

d **Artikelliste festlegen** **V**

Die hinter 'd' angegebenen Wörter (in Kleinbuchstaben zu schreiben) werden bei Kategorien mit dem Prüfbuchstaben 'c' oder 'f' automatisch geprüft. Es kann mehrere mit 'd' beginnende Zeilen geben, das Programm faßt sie alle zusammen. Die folgende international gebräuchliche und den RAK entsprechende Liste ist in der Konfiguration A.CFG standardmäßig enthalten:

```
d a al an as az bir das dat de dei dem den der des det di die dos een eene
d egy ei ein eine eines einer einem einen eit el en et ett eyn eyne gl gli
d ha hai he heis hen hena henas het hith hin hinar hinir hinn ho hoi i il
d ka ke l la las le les lo los mia n na nji o os s t ta the to um uma un
d una une uno y yr
```

Anm.: Bis Version 12.1 war die Liste in die Programme eingebaut und brauchte nicht in der .CFG zu stehen. Jetzt muß sie drinstehen, wenn man Wert auf Artikelprüfung legt. Wenn die Artikelliste nicht in der .CFG steht, wird außerdem nach einer Datei ARTIKEL.CFG gesucht und diese zusätzlich gelesen. Die mitgelieferte ARTIKEL.CFG enthält genau die oben angegebene Liste. Wird sie nicht gefunden, gelten Artikel eben als ganz normale Wörter.

D **Datumslänge** **V**

Die Datumsangaben (siehe Befehle `ce` und `cn`) werden auf eine bestimmte Zeichenanzahl gekürzt. D8 liefert nur das Datum, D4 nur das Jahr. Das so gekürzte Datum wird auch in die Phrase "Leertaste" kopiert.

Standard: **`D17`** (gesamtes Datum mit Uhrzeit) Dieser Wert wird empfohlen.

F **Füllzeichen-Code** **N**

Hinter F gibt man den Code des gewünschten Füllzeichens an. Nachträgliche Änderung:

1. alles in Grunddateien exportieren (mit Parameterdatei I-1.APR), 2. F-Wert in der .CFG ändern,

3. Datenbank neu aufbauen (INDEX -f70)

Standard: **`F219`** (in M.CFG : **`F127`**, beim Chinesisch-System ist **`F95`** sinnvoll)

f **Anzahl der Füllzeichen je Datensatz** **Ä**

Hinter f steht eine Zahl n >= 0.

Beim Abspeichern eines neuen Datensatzes im Schnellzugriff (→ 1.5) werden dann n Füllzeichen (siehe F) mitgespeichert. Um diesen Betrag kann der Datensatz bei einer späteren Bearbeitung im Schnellzugriff verlängert werden. Wird diese Länge überschritten, so sucht das Programm automatisch einen neuen Platz für den Datensatz und kennzeichnet den alten Platz als freigegeben.

Standard: **`f0`** (d.h. keine Füllzeichen einfügen) Jederzeit änderbar.

K **Kategorie-Anfangszeichen** **Ä**

Hinter K setzt man den Code des gewünschten Kategoriezeichens, das bei der Anzeige und Eingabe am Bildschirm jeweils vor der Kategorienummer stehen soll. Nachträglich änderbar, denn das Zeichen wird nicht mit gespeichert, es erscheint nur im Editor. Die Deskriptorzeilen müssen jedoch immer mit '#' beginnen.

Standard: **`K35`** (35 ist der Code des Zeichens '#'). *Abweichung vom Standard wird nicht empfohlen.*

M **Mehrfachkennung** **V**

Mehrfachkategorien erhalten das hinter M stehende Zeichen und die darauf folgenden als Fortsetzungszeichen. Es sei denn, man hat per $M in der Kategorieliste festgelegt, welche Zeichen als Wiederholungszeichen hinter einer Kategorienummer auftreten dürfen. Dieser Befehl gilt also nur für diejenigen Kategorien, bei denen keine Angabe $M gemacht wurde.

Standard: **`Ma`**, das bedeutet, man erhält `#xxa`, `#xxb`, `#xxc`, ... als Mehrfachkategorien, wenn für `#xx` keine $M-Angabe gesetzt ist.

In $A.CFG ist `M2` gesetzt, in N.CFG dagegen `M0`.

m **Speicher konfigurieren** (memory configuration) **Ä**

Achtung: Diese Befehle müssen **vor** die erste Zeile, die mit **#** beginnt!

Die Größen mehrerer interner Arbeitsbereiche lassen sich einstellen. Die aktuellen Einstellungen und die Größe der noch freien Bereiche sehen Sie mit **Alt+F7** im Anzeigeschirm (→ 1.5). Als Standards sind eingestellt:

mr16000 maximale Größe der Ergebnismenge (verkleinern Sie bei Problemen zuerst diesen Wert)
Unnötig für die Windows-Programme: sie können Ergebnismengen bis 64.000 bewältigen.

mk2500 Anzahl Kategorien im Aufnahmespeicher Dieser enthält den aktuellen Satz, an dem man arbeitet oder der angezeigt wird (falls es keine extrem großen Sätze gibt, genügt `mk500`)

mK48000 Größe des Aufnahmespeichers in Byte (10000 reicht, wenn keine extrem großen Sätze.
Der größte Satz muß aus internen Gründen doppelt hineingehen; 48000 ist das Maximum.)

md800 Anzahl der Kategoriedeskriptoren (→ A.1.2, für Pica braucht man 1300)

mb200 Anzahl Kategorien im Hintergrundspeicher (auch für Anwendervariable zuständig!)

mB12000 Größe des Hintergrundspeichers in Byte (5000 reichen meistens, Obergrenze 32000)
(Bereich wird für `#uxy`-Variablen gebraucht, nicht zu klein machen, bes. für ***a99***!)

mP4000 Größe des Phrasenspeichers in Byte (wird auch z.B. für die Zwischenteile gebraucht)

mX40000 reservierter Platz für Parameterspeicher (Index, Anzeige, Export) (Maximum 64000)

N **Nichtsortierzeichen** **N**

Hinter N steht der Code des gewünschten Nichtsortierzeichens, also des Steuerzeichens, das eine Übergehung des nachfolgenden Wortes beim Sortieren etc. bewirken soll.

Standard: **N170** : das ist das Zeichen '¬' (in M.CFG : **N95** = Unterstrich)

n **Nichtsortier-Modus** **N**

Wenn man hier n1 vorgibt, wird vor **und** hinter das nicht zu sortierende Wort der bei N angegebene Code gesetzt, sonst (standardmäßig) nur vor das Wort. Außerdem gibt es den Modus n4 für MARC-Formate.

Beim Export läßt sich der Modus ändern, und zwar

von n0 auf n1 : indem man den Manipulationsbefehl **u¬¬** anwendet (→ 10.2.6.3).

von n1 auf n0 : mit dem Ersetzungsbefehl **_¬ _ _**, damit wird das Zeichen hinter dem Wort beseitigt, das vor dem Wort jedoch nicht. Andererseits: wenn Modus n0 eingeschaltet ist, schadet ein zweites Nichtsortierzeichen hinter dem Wort nicht, denn bei Anzeige und Druck wird es ohnehin ausgeblendet (in der Parameterdatei steht dann in der Regel der Befehl **p ¬ 1** .

Vorteil von Modus n1: innerhalb des "Wortes" können Leerzeichen vorkommen, während bei n0 das zu übergehende Wort am ersten Leerzeichen, Bindestrich oder Apostroph automtatisch endet.

Standard: **n0** (in A.CFG und M.CFG : **n1**)

p **Druckername** **Ä**

Standard: **pPRN**

Wenn der zu benutzende Drucker für das Betriebssystem einen anderen Namen hat, z.B. PRT2, setzt man diesen hier ein.

Wenn man hinter p einen beliebigen anderen Namen angibt, der dem System nicht bekannt ist, wird er als Dateiname angesehen und die Druckausgabe wird in diese Datei umgelenkt (sie wird angelegt, wenn sie noch nicht existiert). Sie kann anschließend bearbeitet und mit dem DOS-Befehl PRINT ausgedruckt werden.

S **Nicht-Stopzeichen** **N**

auch "Entstoppungszeichen" genannt. Hinter S steht der Code des gewünschten Nicht-Stopzeichens, also des Steuerzeichens, das die Wirkung der Stopwortliste (→ 10.2.1.3) aufhebt.

Standard: **S64** (= '@' = "Klammeraffe") .

w... **Farbeinstellungen** siehe A.3 **Ä**

Y **YES-NO-Codes Ä**

Für Ja/Nein-Abfragen müssen in fremdsprachigen Versionen evtl. die
Abkürzungsbuchstaben für "JA" und "NEIN" geändert werden. Man setzt die jeweiligen Kleinbuchstaben direkt hinter das 'Y'.

Standard: **Yjn**

x **Ende** der Befehle. Was danach noch kommt, wird ignoriert.

A.2 Abfrageliste

Für ***a99*** wird empfohlen, statt dessen mit Formularen zu arbeiten. Dafür kann man eine Dokumentation "online" abrufen, wenn man im Schreibfeld `h form` eingibt. Mitgeliefert werden Formulardateien CAT.FRM für $A.CFG und BANK.FRM für N.CFG.
Die Abfrageliste wird in PRESTO durchlaufen, wenn ein neuer Datensatz erfaßt werden soll (→ 3.2 und 1.5.4 Befehl 'I'). Sie kann aus mehreren Abschnitten (Abfragefolgen) für unterschiedliche **Satztypen** bestehen.
Mit einem **Typ-Befehl** am Anfang der Liste steuert man die Benutzung der einzelnen Abschnitte: (nehmen Sie das Beispiel $A.CFG hinzu (→ A.4), dort kommt alles vor; auch $U.CFG ist als Beispiel gut geeignet.)

`Typ" 1 = ....  2 = ....  3 = .... x = Irrtum"` **Ä**

Typ-Abfrage. In der anschließenden Liste muß es dann Sprungmarken **`-1`**, **`-2`** usw. für die verschiedenen Satztypen geben. (z.B. 1 = einbändige Monographie)
Diese Typabfrage dient nur dazu, einen bestimmten Teil der Abfrageliste anzuspringen. Gespeichert wird der Typ hierdurch nicht! Wenn man ein Typkennzeichen speichern will, kann man dies auf andere Art machen: siehe dazu unten : "Vergleichs-Sprung".
Ein Abschnitt für einen bestimmten Satztyp endet, wenn eine mit **`-x`** beginnende Zeile kommt.
Eine Abfrageliste kann mehrere Typabfragen verstreut enthalten.

Achtung: eine Typzeile darf nicht umgebrochen sein (d.h. kein Zeilensprung), auch wenn sie sehr lang ist. Sie kann, fortlaufend geschrieben, bis zu 255 Zeichen lang sein.
Mehrfach-Leerzeichen sind nur innerhalb der Anführungszeichen erlaubt, sonst entfällt der Rest der Zeile.
Abfragezeilen haben folgende Formen: (man beachte: kein '#' vorweg, nur die Nummer)

`xxf"Text"+M` **Abfrage mit Sprungbefehl** (Ausführung nur, wenn etwas eingegeben wird)

`xxf"Text"abc<` **Abfrage mit Vorgabe** ('<' bewirkt: `abc` wird hier ohne <Enter> sofort übernommen)

`xxf"Text".` **Abfrage mit Übernahme aus dem vorigen Datensatz**

Dabei ist: xxf = Kategorienummer mit Folgezeichen (evtl. f = Leerzeichen). Wenn f das Zeichen '~' oder 'ú' ist, wird die nächste unbelegte Mehrfachkategorie von `#xx` genommen, falls `#xx` schon belegt ist.
`"Text"` = Hinweistext ("Prompt"), der bei der Abfrage von `#xxf` erscheinen soll, beliebige Länge.

- Wenn `"Text"` mit **!** beginnt, kommt man ohne Eingabe nicht aus der Zeile heraus,
- wenn "Text" mit **?** beginnt, wird man bei Leereingabe gefragt, ob man hier nichts eingeben will. (So programmiert man **Pflicht-** bzw. **Hinweiskategorien**.) Der `Text` zwischen den Anführungszeichen kann übrigens auch Mehrfach-Leerzeichen enthalten. Er darf aber nicht leer sein, sonst muß "" weggelassen werden.

Hinter `"Text"` steht entweder nichts *oder:*

`+M` = Sprungbefehl (M muß ein ASCII-Zeichen sein, kein Wort!) *oder:*

`abc` = Zeichenfolge, die als Vorgabe für die Kategorie xxf auf den Bildschirm geschrieben wird;
Wenn hinter `abc` ein '**<**' steht, wird die Vorgabe ohne <Enter> sofort übernommen; ohne '<' steht der Cursor hinter der Vorgabe `abc` , so daß man diese ergänzen oder ändern kann.
Wenn statt `abc` nur ein **Punkt** steht oder man einen Punkt eingibt, wird `#xxf` aus dem vorigen Datensatz übernommen. Man nutzt das so: Bevor man mit **I** in die Erfassung geht, holt man einen Satz in die Anzeige, der möglichst viel Übernehmbares enthält.
Zwischen `xxf` und `"Text "` können ebenfalls noch Zeichen stehen. Die sind dann dem Bearbeiter zum Zeitpunkt der Abfrage nicht zugänglich und werden fest in die Kategorie übernommen.

Die Zeile mit `+M` bewirkt: wenn bei dieser Abfragezeile etwas eingegeben wird, dann geht es bei der

`-M` **Sprungmarke** `-M` weiter, sonst mit der nächsten Zeile der Abfrageliste.
Die Sprungmarke `-M` muß auf eigener Zeile stehen, ohne Leerzeichen dazwischen.
Wenn alle Zeilen abgelaufen sind oder eine Sprungmarke **`-x`** kommt, gelangt man in den normalen Eingabe-/Befehls-/Bearbeitungszustand des Editors (Kap.3).

`+M` **unbedingter Sprung** : Wenn die Abfrageprozedur an diesen Punkt kommt, wird die Sprungmarke -M gesucht und es geht mit der darauf folgenden Zeile weiter. Die Sprungmarke darf oberhalb des Sprungbefehls stehen. Das bedeutet: wenn eine Sprungmarke doppelt vorkommt, wird immer die erste angesprungen, die zweite ist unerreichbar.

=\"xyz\"+M *Achtung:* In ***a99/alcarta*** funktioniert dieser Befehl nicht.

Vergleichs-Sprung : wenn bei der unmittelbar vorangehenden Abfrage `xyz` eingegeben wurde, geht es mit der auf die Sprungmarke -M folgenden Zeile weiter.

Beispiel:

```
97 "Typ: "          Kat. #97 soll die Typenbezeichnung aufnehmen
="a1"+i             a1 wurde eingegeben : Sprung nach  -i
="a2"+j             a2 wurde eingegeben : Sprung nach  -j
="b"+k              b  wurde eingegeben : Sprung nach  -k
=""+m               es wurde nichts eingegeben : Sprung nach  -m
+n                  in allen anderen Fällen : Sprung nach  -n
```

Wenn also die Abfrage "#97 Typ: " mit "a1" beantwortet wurde, geht es bei Sprungmarke -i weiter, etc.
Hinter '=' ist jeweils der zu erwartende Eingabetext vollständig in "..." anzugeben.
Wenn man eine Typabfrage so gestaltet, wird also der Satztyp auch gespeichert (in diesem Fall in #97).

=F*nr* **Dateinummer für Speicherung** *Achtung:* In ***a99/alcarta*** funktioniert dieser Befehl nicht.

Mit diesem Befehl kann man steuern, daß bestimmte Satztypen in bestimmte Dateien (.cLD-Dateien) zu speichern sind. Die *nr* kann von 1 bis 255 reichen. Am besten setzt man solche Befehle direkt hinter die Sprungmarke des betreffenden Satztyps (Beispiele siehe A.CFG, abgedruckt in A.4).

Vergleichen Sie besonders zu den Standard-Abfragen die beiden mitgelieferten Dateien $A.CFG und $U.CFG sowie die Ausführungen in Kap.3. Beim Ausprobieren werden die Zusammenhänge schnell deutlich.

Identnummern-Vergabe verhindern

Wenn ein bestimmter Satztyp keine Identnummern bekommen soll, baue man in den zugehörigen Abfrage-Abschnitt diese Zeile ein:

```
#nn "nr.: "|9#xxx?3<
```

(wobei #nn die Kategorie für Identnummern ist, die mit Befehl `cg` festgelegt wurde (→ A.1.3), also normalerweise `#00` oder `#000`. `#xxx` ist keine erlaubte Kategorie, deshalb entsteht dann keine Nummer.)

Schleifen in der Abfrageliste

Mit Einführung der Teilfeldtechnik entstand der Wunsch, wiederholbare Felder in einer Schleife abfragen zu können, für jedes einzelne Feld jedoch zwei oder mehr Teilfelder. Diese Möglichkeit wurde eingebaut. Da es auf viele Einzelheiten sehr genau ankommt, stellen wir drei Muster vor, wie solche Schleifen zu konstruieren sind.

Wichtige Voraussetzung: Vor der Schleife muß mindestens ein Feld abgefragt werden, das in der Reihenfolge **vor** dem abzufragenden Mehrfachfeld steht. Aus internen Gründen funktioniert die Schleife nicht, wenn im Arbeitsspeicher noch keine Kategorie ist, die der abzufragenden vorausgeht. Also: als erstes eine Kategorie abfragen, die in der Ordnung weiter vorne steht, dann klappt es.

Die zur Schleife gehörigen Zeilen müssen mit derselben Kategorienummer (evtl. plus Indikator) anfangen, die erste braucht nicht mit einem '$', die weiteren müssen mit zwei solchen Zeichen anfangen, denn daran erkennt das Programm, daß der eingegebene Text an die schon eingegebene Kategorie anzuhängen ist; das zweite '$' wird entfernt.

Wenn ein Fehlerhinweis erscheint, diesen mit <Enter> wegdrücken, dann geht die Schleife weiter.

Beispiel 1: Format ohne Indikator (t2, k4 : Konfiguration A.CFG)

```
-A
85 " Serie ; Band "          erstes Teilfeld abfragen (das erste geht auch ohne Teilfeldkennung!)
=""+D                        wenn nichts eingegeben wurde: Schleife verlassen
     ↓ wenn hier ein $ steht, wird der Inhalt als Teilfeld an die Kategorie angehängt!
85 $$ "  Körperschaft : " 2. Teilfeld      (Leerzeichen als Teilfeldkennung!)
85 $$i"          ISSN : " 3. Teilfeld    Teilfeld $i wird auch noch angehängt
+A                                  zurück zum Schleifenanfang
-D                                  Fortsetzung hier, wenn Schleife durch Leereingabe beendet wurde.
```

Beispiel 2: Ein Indikator (`t3, k6` : $D.CFG = MAB-Format)
(Dieses Beispiel ist nur fingiert, denn MAB kennt keine Teilfelder, außerdem ist die #100 nicht wiederholbar.)

```
-A
100  $a"  Nachname: "                Indikator ist '' (2 Leerzeichen hinter der 100 !)
="$a"+D                          Es wurde nichts eingegeben (Kategorie besteht nur aus "$a")
      ↓ wenn hier ein $ steht, wird der Inhalt als Teilfeld an die Kategorie angehängt. Hier also $b.
100  $$b"  Vorname: "
100  $$c"    Datum: "
+A
-D
```

Beispiel 3: Zwei Indikatoren (`t3, k7` : $U.CFG = MARC-Format, #600 = Personenschlagwort)

```
-A
600 10$a"  Personal name: "              Die Ziffern '10' sind als Indikatoren vorgegeben
="$a"+D                              Es wurde nichts eingegeben (hinter $a steht nichts)
       ↓ In diesem Fall ist hier die Position für das zweite $
600 10$$d"           Dates: "
600 10$$x"    Subdivision: "
+A
-D
```

In der CFG-Zeile für die betreffende Kategorie (#40, #100, #700) muß natürlich sichergestellt sein, daß die Kategorie überhaupt mehrfach besetzbar ist (d.h. wenn eine Angabe $Mxxx vorhanden ist, darf xxx nicht leer sein). Automatisch erhalten dann die eingegebenen Felder die zulässigen Wiederholungszeichen.
Übrigens: in einem Prompttext darf auch das Zeichen '_' (Unterstrich) vorkommen.

A.3 Farben (DOS-System)

Farbbildschirme sind heute der selbstverständliche Standard, was zu den Anfangszeiten von MS-DOS noch nicht so war. Für die konventionellen DOS-Programmen gibt es eine eigene Methode der Farbeinstellung, für UNIX und für Windows gilt das Folgende nicht (d.h. die Befehle wirken nicht).
Es wurden 6 verschiedene Farbkombinationen ausgewählt für die Unterscheidung von 6 Anzeigetypen auf dem Bildschirm. Einstellbar sind diese Kombinationen über den Konfigurationsbefehl **`w`** . Hier eine tabellarische Übersicht:

Befehl	Farbkombination	Verwendung für
`wd160`	hellblau auf schwarz	Eingabe und Anzeige der eigentlichen Daten (Anzeigebildschirm)
`wm020`	grün auf schwarz, normal	informatorische Mitteilungen des Programms
`we110`	rot auf schwarz, hell	Fehlermeldungen (errors), meist ganz unten auf dem Bildschirm außerdem für die Suchbegriffe im "Erweiterten Register" (Shift+F8)
`ws130`	gelb auf schwarz, hell	Menütexte für Selektion durch den Benutzer (z.B. Fußzeilen der Anzeige)
`wb174`	weiß auf blau	Index-Anzeige ("Browse"-Liste)
`wh007`	schwarz auf weiß	Hilfetexte

Mit den Befehlen **`wd`** und **`wb`** kann man bis zu drei Farbkombinationen angeben, wenn man beim Aufruf von PRESTO oder APAC zwei oder drei Datenbanken zugleich aufruft, indem man die Option **`-d`** zwei- oder dreimal angibt (→ 12.1). Damit erreicht man, daß beim Umschalten mit <Alt>+a jede Datenbank in einer anderen Farbe erscheint. Dies hat sich in der Praxis sehr gut bewährt, kann also empfohlen werden.

Hinter dieser Farbwahl stehen relativ simple Überlegungen:
Die angenehmste Kombination, bei der Text am besten lesbar ist, muß für die Anzeige der gefundenen und eingegebenen Daten benutzt werden, weil darauf das Auge die meiste Zeit verweilt. Hier erschien hellblaue Schrift auf schwarzem Grund (subjektiv) am günstigsten. Urteilen Sie selbst.
Rot für Fehler liegt nahe. Grün signalisiert: "alles O.K., dies ist nur eine Information",
Gelb dagegen: "Achtung, jetzt muß reagiert werden".
Und daß schließlich Hilfetexte Schwarz auf Weiß kommen, wie auf dem Papier des Handbuchs, versteht sich von selbst.
Der Hintergrund ist bei allen Kombinationen außer Fehlermeldungen und Hilfe schwarz. Das Bild wird unruhiger, wenn man auch die Hintergrundfarben noch variiert - es sei denn, man arbeitet konsequent mit Fenstertechnik und leistet sich dann auch noch Umrahmungen etc. Solches ist im ***allegro***-System bislang nur in einigen Zusatzprogrammen für besondere Aufgaben und im ***CockPit*** realisiert, denn derartige Techniken sind schwer mit der ganzen Variabilität von ***allegro*** in Einklang zu bringen, wenn zudem das Programm nicht alle Grenzen sprengen soll. Ab Version 12.2 kann es bei DOS ohnehin auch bei Rechnern mit 640K schon Probleme geben, wenn größere Geräte- oder Netzwerktreiber u.a. eingebunden sind. Deshalb wurde die Möglichkeit geschaffen, den Arbeitsspeicher zu konfigurieren (siehe Befehl 'm' in A.2).
Die oben aufgeführten Kombinationen werden vom Programm eingestellt, wenn man nichts anderes in A.CFG verlangt, d.h. diese Befehle dürfen dann wegbleiben. Wer seine eigenen Wünsche einprogrammieren will, füge die entsprechenden Zeilen in seine c.CFG ein, wobei die Ziffern folgendes bedeuten:

1. Ziffer: 0 für Normaldarstellung, 1 für Hervorhebung (highlight)
2. Ziffer: Zeichenfarbe 0 – 7 (Vordergrundfarbe)
3. Ziffer: Hintergrundfarbe 0 - 7.

für die Farben gelten folgende Zifferncodes

0 : schwarz
1 : rot
2 : grün
3 : braun/gelb
4 : (dunkel-)blau
5 : violett (magenta)
6 : hellblau (cyan)
7 : grau/weiß

Sie können den Befehl w auch online im Editor eingeben. Das ermöglicht ein ausführliches Experimentieren bei sofortiger Kontrolle der Ergebnisse und erleichtert wesentlich die Auswahl für den persönlichen Geschmack. Geben Sie probeweise `#wd123` <Enter> im Editor, dann **F5**, um die Anzeige zu erneuern.
Die Hilfsprogramme ASORT und QRIX arbeiten übrigens ohne Farbe.
Hier finden Sie sinnvolle Kombinationen für

SW-Bildschirme:	Weiß-Schwarz:	Grün-Monitore:		
`wd007`	`wd070`	`wd020`	`oder`	`wd007`
`wm107`	`wm170`	`wm120`		`wm007`
`we107`	`we170`	`we120`		`we007`
`ws107`	`ws170`	`ws020`		`ws007`
`wb007`	`wb170`	`wb120`		`wb007`
`wh007`	`wh070`	`wh020`		`wh007`

Damit erhalten Sie auch auf Farbbildschirmen die entsprechende Darstellung.
Ausgesprochene Freaks können natürlich ihre Titelaufnahmen auch kolorieren (Köpfe gelb, Fußnoten violett...). Dazu muß man "nur" die sog. ANSI-Escape-Sequenzen (Befehle der Form **`ESC[1;3x;4ym`** , x und y ist eine der obigen Ziffern für Zeichenfarbe bzw. Hintergrundfarbe, → 10.2.0.) in die Zwischenteile einer Parameterdatei einbauen. Die mitgelieferte Parameterdatei COLOR.APR, die zum Testen erstellt wurde, zeigt diese Methode. (Eine farbige Druckausgabe allerdings kann noch nicht angeboten werden, wäre jedoch ähnlich zu machen.)
Wenn es auf Schwarz-Weiß-Monitoren Probleme gibt, so daß z.B. die Anzeige ganz oder teilweise ausfällt, kann man die MS-DOS-Befehle **`mode bw80`** oder **`mode co80`** versuchen. Außerdem kann man ***CockPit*** mit **`cp -M`** starten, denn ***CockPit*** benutzt nicht die .CFG für seine Farbeinstellungen, sondern reagiert nur auf die Option –M (für Monochrom).

A.4 Beispiel: Standard-Konfiguration `$A.CFG`

(Aus Platzgründen wurde dieser Abdruck der Datei leicht gekürzt - das Original wird ja mitgeliefert!)
Die Datei darf auch A.CFG heißen. Wenn beide existieren, wird $A.CFG vorgezogen.
Ein weiteres Großbeispiel ist ab V26 die **`N.CFG`**, die man vergleichend hinzuziehen kann.

```
Konsolidiertes allegro-Format
**********************
**** Teil 0 : Festlegung der allgemeinen Struktur  ****

t2   es ist ein 2stelliges Kategoriesystem
k4   Kategorietext beginnt auf Position 4
$31  Unterfeld-Code ist das Zeichen ▼ (ASCII 31. Wenn es $ sein soll, dann $36 setzen)

Speicherbelegung: wenn der Arbeitsspeicher nicht reicht, folgende Werte ändern und aktivieren
Diese Zeilen müssen vor der Liste der erlaubten Kategorien stehen!
mr15001          maximale Ergebnismengengröße  (bis 16.000)  (Reduktion auf 1000 bringt 60K !)
mk2500           Arbeitssp. Anzahl Kategorien (1000 reicht)
mb200            Hintergr.Sp. Anzahl Kategorien
mB12000          Größe (in Byte, 1000 reicht meist bei DOS, mehr bei Windows)
mP4000           Phrasenspeichergröße (auch für Zwischenteile nötig!)
mX64000          Parameterspeicher
          PRESTO:  prüfen Sie bei Problemen mit langen Datensätzen mit Alt+F7 die Einstellungen und den freien Platz

**** Teil 1 : Liste der erlaubten Kategorien ********
Hierarchie-Struktur für bis zu 6 Unterstufen
!!! Folgende 9 Zeilen bitte nicht ändern
#u1"Kopfzeile 1"Die Sonderkategorien #u1 und #u2 müssen immer ganz oben stehen
#u2"Kopfzeile 2"
#00"IdNr"                  Empfehlung: IdNr mit einem Typbuchstaben beginnen lassen,
#01"BandNr"                      z.B. 'b' für Buch, 'z' für Zeitschrift ...
#02"TeilNr"
#03
#04
#05
#06
          Hinweise für mögliche Veränderungen:
          So müssen die Zeilen aussehen, wenn KEINE hierarchischen Datensätze gebraucht werden:
   #u1
   #u2
   #00
   #01   wenn nur EINE Stufe gewünscht wird; sonst hier auch #00
   #00   auf die erste Feldnummer, hier die  #00, müssen auf jeden Fall 6 Nummern folgen, die ansonsten
   #00   nicht benutzt werden, wenn man ohne Hierarchie arbeitet!!! Das hat interne Gründe.
   #00
   #00
   #00
!!! Ende der Hierarchiezeilen - ab hier sind Änderungen aller Art erlaubt
ACHTUNG : Statt $ kann nachfiolgend auch das Zeichen ▼ stehen (bis V33 war nur ▼ möglich)
Jetzt kommen die anwendungsspezifischen Kategorie-Definitionen. Die Kategorienummern sind 2stellig (siehe Befehl t2
ganz oben)  die 3. Ziffer ist dann der Wiederholungscode, Indikatoren gibt es wegen k4 nicht.
Wenn $Mabc gesetzt ist, sind nur a, b und c als Wiederholungscodes zulässig
Wenn auf $M keine Buchstaben/Ziffern folgen: keine Wiederholung zulässig
Die Feldbezeichnungen in "..." sind für das Programm nicht zwingend nötig.

(Dies hier ist nicht die Abfrageliste! Die steht unten in Teil 3)

#07"frei07"$M
#08"frei08"$M
#09"VerknNr"$M
```

```
#0a"STypFestFelder"$M
#0c"Codes"$M
#10"frei11"
#11"frei11"$M
#12"frei12"$M
#14"frei11"
#15"frei13"$M
#16"frei11"
#17"frei11"
#18"Verf-Titel-Vorlageform"$M
#19"HST-Vorlage"$M
#20"Titel"$Mgps$C c
#2n"EST-Stammsatz"$Madruz$C c
#8n"ZeitschrStammtitel"$M123akrz$C c    neu: k
#21"SammlVerm"$M$C c
#22"EST"$Mr$C c
#23"Nebentitel"$M2345FPSZabcfskvz$C c     (Teilfeld z ergaenzt)       ~~v
#24"Paralleltitel"$M2345abc$C c
#25"TitelErgz"$M
#26"BeigefWerkTitel"$M2$C c
#27"Kartenkopf"$M2345678abcde
#28"Andere Schreibw"$M
#3n"SachStammsatz"$MacglrstvuxyzPS$Aabfnrs
#30"Notation"
#31"Schlagw"$Macfgikmpstuh
#32"FremdNotation"$MABDLMNZ
#33"FremdSchlagw"$M0123456789BDLMm
#35"Kunstwerk"
#36"Kuenstler"
#37"Sprache"$M
#3s"Thesaur.IdNr"$Mid
#3t"ThesaurusTerm"$Mabdfhnpqrtyz
#3q"Ergz"
#3u"obsolet"
#39"VerfAngabe"$Mgbc
#4n"PersonenStamm"$MadoprtzADOPRTZ
#4N"LokalVar"
#40"Verfasser"$M23456789abc$C d
#41"Hrsg"$M23456789abc$C d
#42"Mitarb"$C d
#43"Bearb"$C d
#44"Begruender"$C d
#45"Sammler"$C d
#46"Texter"$C d
#47"Uebersetzer"$C d
#48"Person"$C d
#49"Person"$C d
#50"Illustr"$C d
#51"Komment"$C d
#52"Komponist"$C d
#53"Praeses"$C d
#54"Respond"$C d
#55"Vorredn"$C d
#56"Nachredner"$C d
#57"anderePerson"$C d
#58"Name-im-Titel"$M23456789$C d
#59"GefeiertePerson"$M2345abc$C d
#60"Urh"$M234ae$C c
#6n"KoerpStammform"$MadfilprszADFILPRSZ$C c
#6N"LokalVar"
#6u"obsolet"
#61"beteilKoerp"$M23456789abcdi$C c
#62"Ausstellungsort"
#63"frei63"
#68"VerfAdresse"
#69"gefKoerp"$M2345abc$C c
#70"Quelle"$M234689caisZ$C c
#71"Ausgabe"$M
#72"ReprVerm"$Mn
#73"ImpressumVorlageform"$M
#74"ErschOrt"$Md
#75"Verleger"$Md
```

```
#76"EJahr"$Mcefrspz$C e
#77"Umfang"
#78"frei78"
#80"dtTitelUebersetz"
#81"Fussnote"$M89BFPSWZabcdefghijklmnopqrstuvwxyz
#82"NE-Verm"
#83"Kongress"$M234abc$C f
#84"UebergeordnWerk"
#85"Serie"$M234abc$A sANiuUv$C c f
#86"ReportNr"
#87"ISBN"$C g
#88"ISSN"$C h
#89"FremdIdNr"
#8e"E-Adresse"
#90"Signatur"$M23456789abcdiszDH$APDLadlnpsuz
#91"ZugangsNr"$M23456789abcd
#92"Bestand"
#93"Inhalt"$M23456789abcd
#95"frei95"
#96"frei96"
#98"Abstract"
#9a"MemoTermin"
#9s"allgVerw"$Maz
#9A"Systemdaten"
#9B"BenutzerDaten"
#9C"Kennungen(z.B.Erw.)"
  Hier NRSU ergaenzt f. Zeitschr.Banddaten: (20070611)
#9D"ExplDaten"$MABCDEFGHIJMNRSUV
#9G"GeschGang Erw"
#9H"Ausl"
#9R"Rechnungs-Teilbetr."$MDHKabcdefghijklmnopqrstuvwxyz
#9X"Ausldaten"
#34"Sonderschlagwort"
#97"LokaleDatei"
#99"Datum"
```

… *und noch einige nichtbibliographische Formalien*

**** Teil 2 : Konfigurationsbefehle **********

Die Befehle folgen hier alphabetisch (s. Anhang A.2):

Erfassungsdatum soll in #99n

```
cn99n
```

Datum der letzten Änderung soll in #99e (aktivieren, wenn gewünscht)

```
ce99e
```

```
cg00
ci9
```

Artikelliste (war bis Version 12.1 automatisch drin)

```
d the a al an as az bir das dat de dei den der det di die dos
d een eene egy ei ein eine eit el en et ett eyn eyne gl gli
d ha hai he heis hen hena henas het hith hin hinar hinir
d hinn ho hoi i il ka ke l la las le les lo los mia n na
d nji o os s t ta to um uma un una une uno y yr des dem eines
d einer einem einen
```

`F219` Füllzeichen-Code

`f0` keine Füllzeichen für neue Datensätze, z.B. f20 schreiben für 20 Füllzeichen je Satz

`K35` Kategoriezeichen bei Anzeige ist #

`k4` Textanfang 4. Position

`lGER` Sprache (language) ist GERMAN

Für Mehrfachkategorien:

Normalerweise wird #40a, #40b etc. gemacht. Für A.CFG soll es anders sein:

`M2` d.h. #402, #403 etc. bei Mehrfachkategorien

```
N170      Nichtsortierzeichen 170 = ¬
n1        Nichtsortiermodus 1, d.h. ¬ vor und hinter dem Wort (das war früher anders!)

pPRN      Name des Druckers (oder : LPT1)
            (setzen Sie für PRN einen Dateinamen ein, dann wird nicht gedruckt,
                    sondern die Ergebnisse in die Datei geschrieben, schreiben Sie
                    pNUL, dann passiert beim Druckbefehl gar nichts)

S64       Entstoppungszeichen soll @ sein
```

Farbkombinationen (s. Anh. A.3; für wd und wb braucht man bis zu 3 Kombinationen)

```
wd160150140             Display (Titelanzeige)
wb170160150             Browse  (REGISTER-Anzeige)
```

für Schwarz-Weiss die folgenden Zeilen aktivieren:

```
 wd107070170            Display
 wm007                  Message (Nachrichten vom Programm)
 ws007                  Select  (Auswahl-Menüs)
 we007                  Error   (Fehlermeldungen)
 wh070                  Help    (Hilfeseiten)
```

Befehl Y : Angabe der Zeichen für "Ja" und "Nein" (s.a. Zeile 301 in UIF0...)

```
Yjn    deutsch (ja/nein)
Yyn    englisch (yes/no)
Yon    frz (oui/non)
Ysn    ital. (si/non)
```

**** Teil 3 : Abfrage-Liste für die Erfassung: **********

Zuerst die Satztyp-Abfrage:

```
Typ" 1 = selbständig (Buch)   2 = unselbst. (Aufsatz, Beitrag)   3 = Stammdaten
     z = Zeitschriften-Stammsatz    r = siehe- u. siehe-auch-Verw.
     7 = Sachdaten   9 = sonstige Kat.   u = Band   v = verknüpfte Bandaufnahme"
```

(Wenn man 1 eingibt, geht es bei Sprungmarke -1 los, usw.)

```
-1        Typ 1: Buch
=F0           d.h. Satz soll in die default-Datei (Option -n vom Programmaufruf)
00 "      Id.Nr.: "
-E
40 "  Verfasser: "+E      (Wiederholungsschleife!)
41 "       Hrsg.: "+T
60 "  Körpersch: "
-T
20 "! Sachtitel: "        Pflichtkategorie: ohne Eingabe geht's nicht weiter
71 "    Auflage: "
74 "?         Ort: "      Wenn man hier nichts eingibt, kommt eine Nachfrage
75 "      Verlag: "
76 "        Jahr: "
77 "      Umfang: "
91 " Zug.Nr.:    "
85 "       Serie: "
87 "        ISBN: "
90 "   Signatur: "
```

```
31 " Schlagwort: "
-x

-2      Unselbständiges Werk
=F0
40 "  Verfasser: "+2
20 "! Aufs.titel: "
70 "  Zeitschr. (oder _Kürzel): "+B
70c"      CODEN: "+B
70i"       ISSN: "+B
84 "   enth. in (Id.Nr.): "+C
-B
704"     BandNr: "
76 "       Jahr: "
706"       Heft: "
-C
708"     Seiten: "
-x
```

Typabfrage für Normsätze: (bei s kommt dann noch eine Typabfrage)

```
-3
Typ" p = Personen    k = Körpersch.  z = Zeitschr./Serie  e = EST
     s = Systematik   a = Signaturgruppe  t = Thesaurus "
```

Besser geeignet ist für solche Sätze das "Referentenprogramm"

```
-s        Klassifikations-Stammsätze verschiedener Art:
Typ" c = Klassifikation   a = Aufstellungsgruppe   l = Sprachschlüssel
     h = Zeitcode         g = geogr. Schlüssel"

  ........
-8
-u    Unteraufnahme bei hierarchischen Aufnahmen
01 "  Band: "
41 " Hrsg.: "
20 " Titel: "
76 "  Jahr: "
77 "Umfang: "
-x

-v     Bandaufnahme , verknüpft gespeichert
   (vorher die Hauptaufnahme in die Anzeige holen, dann bewirkt der folgende
   Befehl, da die IdNr aus #00 übernommen wird und man nur noch mit '+' die
   Bandnummer dranhängen muß
00 "  IdNr+BandNr: ".
41 " Hrsg.: "
20 " Titel: "
76 "  Jahr: "
-9
77 "Umfang: "
-x
.........
-z       Zeitschriften-Stammsatz
=F100          File 249 benutzen
8n " Zeitschrift/Serie (Titel in Zitierform): "
```

```
20 " Titel in RAK-Form: "
8na"            Kürzel: "
8nr"       Verw.Formen: "
23F"    früherer Titel: "
23S"    späterer Titel: "
8n1"    Zitiertitel 1 : "
8n2"    Zitiertitel 2 : "
88 "             ISSN: "
88c"            CODEN: "
76p"     Ersch.Verlauf: "
74 "     Ersch.Ort    : "
75 "     Verlag       : "
92 "     Bestand      : "
8nz"     Kommentar    : "
-x
```

`-p` Personenstammsatz

```
=F252
4n " Personenname Ansetzung: "
4na" siehe auch:             "
4nr" Verweisungsformen:      "
4np" Pseudonyme :            "
4nt" wirklicher Name:        "
4nd" Lebensdaten:            "
4nz" Kommentar: "
-x
```

`-k` Körperschafts-Stammsatz

```
=F253
6n " Körperschaft Ansetzung: "
6na" siehe auch:             "
6nr" Verw.Formen:            "
6nf" frühere Form:           "
6ns" spätere Form:           "
6nd" 'Lebensdaten'  "
6nz" Kommentar:  "
-x
```

`-e` EST-Stammsatz

```
=F254
2n " Einheitssachtitel:  "
40 " Verfasser:          "
2nr" Verweisungsformen (in gleicher Sprache):  "
2nu" Übersetzungen in anderen Sprachen:        "
2nd" Angaben zur Datierung: "
2nz" Kommentar:          "
-x
```

`-r` Siehe- und Siehe-auch-Verweisungen (für alle Register)

Form: `verweistext -> |R ziel` mehrfach, getrennt durch "; "

```
=F248
9s "siehe-Verw.:      "
9sa"siehe-auch-Verw.: "
```

`x` Ende der relevanten Befehle der CFG

A.5 Umstieg von älteren Versionen auf V13 oder neuere

Zur Förderung des Übergangs speziell von `a.cfg` auf das konsolidierte Format `$a.cfg` gibt es das Skript `konsol.bat`. Grundsätzlich ist es so, daß für den Umstieg von einer der älteren Versionen auf V13 oder eine spätere **keine** Umarbeitung der Konfiguration oder der Parameterdateien notwendig ist (siehe A.1.2) . Die Programme erkennen am Fehlen des **`t`**-Befehls, daß eine .CFG alter Art vorliegt. Dann interpretieren sie die Kategorie-Definitionszeilen korrekt und arbeiten ohne erkennbare Unterschiede. Nur dann, wenn Sie die neuen Möglichkeiten zur präzisen Definition der Kategorien nutzen wollen, müssen Sie zur Tat schreiten. Das Hilfsprogramm `cfgconv.exe` nimmt Ihnen dann einen Teil der Arbeit ab, so daß der Aufwand nicht besonders groß wird. Wenn Ihre Konfiguration z.B. **`x.cfg`** ist, starten Sie den Vorgang so:

`cfgconv x`

Dann entsteht eine Konfigurationsdatei neuer Art mit Namen **`$x.cfg`**. Diese können Sie nun Ihren Vorstellungen entsprechend ausgestalten. Wichtig ist dieses: die alten Programme funktionieren solange weiter, wie die alte `x.cfg` noch da ist. Die neuen Programme bevorzugen aber die `$x.cfg`, wenn sie schon existiert, sonst kommen sie auch mit der alten `x.cfg` klar. Wenn Sie schließlich die `$x` in `x` umbenennen, können die alten Programme damit nichts anfangen, sie werden wahrscheinlich abstürzen. Den neuen macht es nichts aus. So ist es möglich, die alten Programme zunächst noch für alle Fälle vorzuhalten.

Die neue Konfiguration beginnt in jedem Fall mit dem Befehl **`t2`**, denn jedes bisherige Kategoriesystem ist genau genommen ein zweistelliges Schema. Für die alten Programme sind alle Kategorienummern zweistellig, die dritte Stelle ist das Wiederholungszeichen. Daher muß man für das weitere Vorgehen drei Situationen unterscheiden:

1. Die `.cfg` enthielt den Befehl **`k4`** (oder keinen **`k`**-Befehl): dann braucht an den Daten nichts verändert zu werden. Das trifft zu auf die Konfiguration **`a.cfg`**. In der neuen `.cfg` ergänzt man zu jeder Kategorie einen Befehl **`▼M`** mit den zulässigen Werten für die Wiederholungszeichen. Wenn man das nicht macht, werden ohne Kommentar alle Zeichen akzeptiert. (Vgl. die offizielle `$a.cfg`)

2. Die `.cfg` enthielt **`k5`** (man hatte also ein Pseudo-dreistelliges Schema), und die vierte Stelle war immer ein Leerzeichen. Dann braucht man nur **`t2`** in **`t3`** zu ändern, die Daten können bleiben, wie sie sind.
   Die vierte Stelle kann künftig für Wiederholungszeichen genutzt werden, d.h. die dreistelligen Nummern sind auf einmal wiederholbar. Die neue `.cfg` muß man nun überarbeiten: Für jede erlaubte dreistellige Nummer muß es eine Deskriptorzeile geben (siehe Abschnitt A.1).

3. Wie Fall 2., aber die vierte Stelle war für Indikatoren genutzt worden (MAB-ähnliche Systeme). In diesem Fall muß man ebenfalls **`t3`** setzen. Dann gibt es jedoch zwei Möglichkeiten:

   a) Man will mit völlig unverändertem Kategoriesystem weiterarbeiten, d.h. die bisher nicht wiederholbaren 3stelligen Nummern sollen auch in Zukunft nicht wiederholbar sein. Dann braucht man keine weiteren Veränderungen vorzunehmen. Die zulässigen Indikatoren müssen aber in der neuen `.cfg` so behandelt werden, als seien es Wiederholungszeichen, denn sie stehen ja auf der Position, wo bei einem dreistelligen Schema neuer Art das Wiederholungszeichen stehen muß. Man würde also z.B. für die MAB-Kategorie 100 zu schreiben haben: `#100▼M*bcef` statt `#100▼I*bcef`. Man muß das nicht sofort machen, denn ohne die **`▼M`**-Befehle akzeptieren die Programme einfach alle Zeichen, die auf Position 4 stehen. Man muß aber, wie im Fall 2., für jede erlaubte 3stellige Nummer eine Zeile in die `.cfg` einsetzen. Die automatische Umwandlung mit `cfgconv` erzeugt z.B. nur die Zeile **`#33`**, man erweitert das also auf **`#331, #333, #335`**...

   b) Man will in Zukunft die 3stelligen Nummern auch wiederholen können. Dann muß man in der neuen `x.cfg` **`k5`** in **`k6`** ändern, denn die Programme benötigen die vierte Stelle für das Wiederholungszeichen, und der Indikator rutscht auf Position 5. Das aber bedeutet: man muß die Daten ändern! Hinter jede Kategorienummer ist ein Leerzeichen einzuschieben. Zunächst erstellt man die neue `$x.cfg`. Als Minimum muß die Liste aller erlaubten dreistelligen Nummern vorhanden sein. Die Umwandlung der Daten macht man am besten so:
      1. Export aller Daten mit `i-1.xpr` (Kopie von `i-1.apr`) in einem SRCH-Durchlauf. Ergebnis: Grunddateien des Typs `.xLG`.
      2. Import dieser Daten mit `import.exe` (ab V13) und einer Parameterdatei `alg3.xim` (= Kopie der mitgelieferten `alg3.aim`).
      3. Jetzt müssen Sie leider alle Export- und Indexparameter durchsehen: aus `b4` bzw. `i5,x` müssen Sie `b5` bzw. `i6,x` machen, da sich der Text um eine Stelle verschoben hat!
      4. Indexierung der dabei entstandenen neuen `.xLG`-Dateien mit `index.exe` ab V13.

      In allen Export-Parameterdateien muß man **`ks=5`** durch **`ks=6`** ersetzen oder **`ks`** beseitigen!

Hinweis: Um die renovierte Uralt-Datenbank dann auch noch mit Windows nutzen zu können, siehe Hinweise im Kap. 2.5.

Anhang B *allegro*-Standard-Datenschema

Vollständige Ausgabe: http://www.allegro-c.de/allegro/doku/form2004/

Das *Konsolidierte Format*

Ab V26 gibt es ein zweites Standardformat: Das Neutralformat:
http://www.allegro-c.de/allegro/doku/neutral/

B.0 Vorbemerkungen

Wer braucht diesen Anhang? Jeder, der die Standardkonfiguration `$a.cfg` einsetzt (→ Anh.A.4). Aber auch, wer sich als nichtbibliothekarischer Anwender ein wenig über die Details von Bibliotheksdaten kundig machen will. In jedem Fall zu empfehlen sind die aktuellen Online-Dokumentationen unter den oben angegebenen Web-Adressen.
allegro setzt kein bestimmtes **Kategorienschema** oder Datenformat voraus (→ Anh.A). Das hier gezeigte, mittlerweile sehr bewährte Schema ist das in der Praxis meistverwendete. Es ist in der Konfigurationsdatei `$a.cfg` enthalten und wird automatisch benutzt, wenn Sie nicht ausdrücklich über die Option `-k` beim Programmaufruf ein anderes Schema verlangen (→ Kap.12). Sie können dieses Schema ändern, erweitern, kürzen oder aber auch ein ganz anderes Schema aufstellen. Solche Dinge werden in der Regel notwendig sein, wenn Sie andere als bibliographische Daten verwalten wollen. Jedoch: Änderungen können in viel Arbeit ausarten, deshalb lieber zweimal überlegen.
Ein Kategorienschema legt grundsätzlich nur fest, unter welchen Feldbezeichnungen (= **Kategorienummern**) die einzelnen Elemente einzugeben und zu speichern sind. Es sagt nichts aus über die innere Form des Datenelements, in welcher Weise also z.B. ein Personen- oder Körperschaftsname oder ein Zeitschriftentitel einzugeben ist, was man mit einem Doppelnamen macht, was mit Abkürzungen in Titeln, mit dem Artikel am Anfang, welche Interpunktion man innerhalb eines Feldes benutzt usw. usf. (→ Kap.3.0). Alles das sind Probleme (z.T. sehr große Problemkomplexe), die nicht per Programm, sondern durch ein **Katalogregelwerk** und seine intellektuelle Umsetzung zu lösen sind.
Die im deutschsprachigen Bibliothekswesen allgemein verbreiteten "Regeln für die alphabetische Katalogisierung (RAK)" (→ Anh.C: Literaturhinweise) sind in vollem Umfang anwendbar, d.h. für alle darin definierten Angaben gibt es geeignete Datenfelder, und es können regelgerechte Ausdrucke produziert werden.
Katalogisierung nach RAK heißt im übrigen noch nicht, daß man auch das Datenformat der DNB (MAB2 bzw. ab 2010 MARC21) verwenden muß. Es wird ein an MAB2 angelehntes PC-Format `$m.cfg` mitgeliefert. (An der Universität Bonn war eine Variante B.CFG entstanden.) Es hat dreistellige Kategorienummern (Übersicht → B.6), wie auch das ebenfalls mitgelieferte MARC-ähnliche `$u.cfg`. Wenn keine zwingenden Gründe z.B. von Seiten eines Verbundes vorliegen, wird empfohlen, das ***allegro***-Standardformat `$a.cfg` zu verwenden, da es übersichtlicher strukturiert ist und die zweistelligen Kategorienummern leichter zu merken sind. Im Bedarfsfall lassen sich beide Formate ineinander umwandeln.
Normierung steht mit Recht hoch im Kurs - ***allegro*** ist jedoch auf kein bestimmtes Katalogregelwerk fixiert. Zwar sind alle wesentlichen, für RAK erforderlichen Datenfelder im Standardformat `$a.cfg` und im `$m.cfg` vorhanden, ob Sie die Felder aber mit regelgerechten Inhalten füllen, ist eine andere Sache - das könnte ohnehin kein Programm überprüfen. Das Format `$a.cfg` enthält darüber hinaus noch weitere Felder, und zwar für Quellenangaben, differenzierte Sacherschließungsdaten, Systematik-Stammsätze, Thesaurus-Datensätze, Namens-Stammsätze für Personen und Körperschaften, Normdatensätze für Einheitssachtitel und Serientitel, Adressendatensätze für Benutzer und Bibliotheken. Damit geht das ***allegro***-Standardschema in seinen Möglichkeiten noch über MAB hinaus. Es soll auch noch den neuen Erfordernissen von RDA und GND angepaßt werden.

Die hier gegebenen Beschreibungen stellen im übrigen keinesfalls auch nur annähernd ein Katalogregelwerk dar. Einige Angaben oder Ausdrücke erscheinen vielleicht Bibliothekaren trivial, unnötig oder ungewohnt, doch soll diese Zusammenstellung auch eine Hilfe für Nichtbibliothekare sein, da ***allegro*** auch außerhalb der Bibliothekswelt Verbreitung gefunden hat. Wenn Sie zur letzeren Gruppe gehören, sollten Sie etwas mehr Zeit mit der Demodatenbank verbringen, Sie sparen hinterher Zeit beim routinemäßigen Erfassen.
Sie sehen auf den nächsten zwei Seiten zunächst eine Kurzübersicht. Dann folgen noch einige Erläuterungen zur Terminologie und schließlich eine Übersicht mit Abschnitten für Normdaten und Geschäftsgangsdaten.

B.1 Kurzübersicht zum "konsolidierten" Format

Einige weniger wichtige Kategorien sind in dieser Übersicht weggelassen (Vollständige Übersicht: Format 2004 bei der UB Braunschweig: http://www.allegro-c.de/doku/form2004/).
Die Buchstaben vor den Kategorienummern deuten auf bestimmte automatische Prüfungen hin, die für die betreffenden Kategorien ausgeführt werden. (c = Artikelprüfung, e = Jahreszahl, d = Kommaprüfung, f = Semikolon-Prüfung, g/h = Prüfziffer-Berechnung ISBN/ISSN, → Anh.A.1). Mit Register a ist das ALL-Wortregister gemeint.
Angaben in **eckigen Klammern** sind nur bei Bedarf einzugeben. Die Klammern sind dann nicht mitzuschreiben.
Rechts ist als Hilfe angegeben, in welchen Registern die Kategorien indexiert werden (gilt nur für `cat.api`).

Gruppe #0 : Identifikations-Kategorien

indexiert im Register

	#00	**Identifikationsnummer[+*BandNr*[+*TeilNr*[+...]]][=*Bandbezeichnung*]** (notwendig nur bei selbständig gespeicherten Untersätzen)	**9**
	#01	**Band-, Abteilungs-, Teil- etc. -Bezeichnung:** (bei unselbständig gespeicherten Untersätzen)	
	... #06	(Ziffer 1...6 kennzeichnet die Hierarchiestufe des Untersatzes; empfohlen wird "flache" Hierarchie, nur mit #01)	
	#09	**IdNr des übergeordneten Hauptsatzes+*BandNr*[=*Bandbezeichnung*]**	**9**
	#0c	**Formschlüssel (= Dokumenttyp, Liste s. "Format '2004") bzw. GND Normsatztyp**	

Gruppe #1 : frei verwendbare Kategorien

	#07, #08	**frei verwendbar**	
	#18	**Sachtitel + Verfasserangabe in Vorlageform** (faßt die Gruppen #2, #4, #5 und #6 zusammen)	
	#19	**Sachtitelangaben in Vorlageform** (faßt Gruppe #2 zusammen) auch: **Bandtitel** (in Untersätzen statt #20), falls nicht für Index geeignet	

Gruppe #2 : Titelangaben

c	**#20**	**Hauptsachtitel. Körperschaftliche Ergänzung : Zusatz** (siehe auch #25)	**3,4,a**
	#21	**Sammlungsvermerk (Werke, Sammlung u.ä.)**	**1**
c	**#22**	**Einheitssachtitel <Sprache> (z.B. Originaltitel bei Übersetzungen)**	**3,4,a**
c	**#23**	**Nebensachtitel (Rücken-, Umschlag- etc. -Titel)**	**3,4,a**
c	**#23F**	**früherer Titel** (bei Hauptaufnahmen von Zeitschriften	
c	**#23S**	**späterer Titel** und Serien)	
c	**#23k**	**"Key Title"** (gehört zur ISSN, siehe #88)	
c	**#24**	**Parallelsachtitel (in anderer Sprache)** (Ansetzung von #23 und #24 wie #20)	**3,4,a**
	#25	**Sachtitelergänzungen** (nicht für Index bestimmt, sonst in #20 hinter " : " angeben)	
	#26	**Titel enthaltener Werke (z.B. Beigaben bei Zeitschriften)**	**3,4**
	#27	**zusätzliche Köpfe für Karten**	
	#28	**zusätzliche Stichwörter für Index** (Hilfsfeld für Wörter, die nicht im Titel stehen)	**3**

Gruppe #3 : Sachliche Erschließungsdaten

	#30	**Systematik-Notationen, Klassifikation** (Trennung: Leerzeichen)	**7**
	#30a,b	**Sachgruppen (Grobsystematik)**	**7**
	#30i	**Lokale Systematik (Institut, Teilbibliothek)**	**7**
	#30x	**diverse Spezialnotationen (x=g,h,l,k,m** s. Systematik-Stammsätze, #3n)	**7..**
	#31	**Schlagwörter, Thesaurusbegriffe** (Trennung: Semikolon)	**3,a**
	#31x	**diverse Spezialschlagwörter (nach RSWK: x=c,g,k,p,s,t,u,** → #3n)	**1-4,a**
	#32x	**Notationen aus Fremddaten** (z.B. #32L für LoC)	**7..**
	#33x	**Schlagwörter aus Fremddaten** (z.B. #33L für LoC, #330,#331.. RSWK-Ketten)	**3..,a**
	#35/#36	**Angaben zu einem Werk / zu einem Künstler** (beides mit Unterfeldern)	**1/4**
	#37	**Sprache(n) des Textes** (Trennung: Leerzeichen)	
	#39	**Verfasserangabe (Personen u. Körperschaften)** (faßt die Gruppen #4 bis #6 zusammen) **in "Vorlageform" (vgl. #18, #19) (optional)**	

Gruppe #4 : Personen (immer in der "Ansetzungsform" : Name, Vorname) **1,a**

d	**#40**	**Verfasser**	**#50/#50f**	**Illustrator / Photograph**	
d	**#402,3**	**2. und 3. Verfasser**	**#51**	**Kommentator**	
d	**#41**	**Herausgeber**	**#52**	**Komponist**	
d	**#42**	**Mitarbeiter**	**#53**	**Präses** (Bei alten	
d	**#43**	**Bearbeiter**	**#54**	**Respondent** Dissertationen)	
d	**#44**	**Begründer**	**#55/#56**	**Vorredner / Nachredner**	
d	**#45**	**Sammler**	**#57x**	**sonstige Personen (z.B. #57i Interpret)**	
d	**#46**	**Texter**	**#58**	**Personenname im Sachtitel**	
d	**#47**	**Übersetzer**	**#59**	**"gefeierte" Person**	

indexiert im Register

Gruppe #6 : Körperschaften

			2,a
c	**#60**	**Urheber (verantwortliche Körperschaft im Sinne von RAK)**	
c	**#61**	**Beteiligte Körperschaft (Sekundärkörperschaft)** (Trennung " = ") (Veranstaltungen → #83)	
c	**#61x**	**Besondere Typen beteiligter Körperschaften** (z.B. #61i Interpreten, z.B. Orchester)	
	#62	**Ausstellungslokal** (*Form:* Museum : Anfangsdatum-Endedatum)	
	#68	**Körperschaft, der der erste Verf. angehört,** (evtl. Adresse, Zeilentrennung '¶')	
c	**#69**	**sonstige Körperschaft (z.B. "gefeierte" K. bei Festschr.)**	

Gruppe #7 : Erscheinungsdaten, physische Daten

	#70	**Quelle (Zeitschriftentitel[; Band(Jahrgang)Heft, Seiten])** (für Zeitschriftenaufsätze)	**5**
oder		**_kürzel ...** (wobei das "kürzel" in der #8na eines Stammsatzes vorkommen muß)	**9**
	#70x...	**Detaillierte Quellenangaben** (wahlweise, wenn nicht in #70)	
		(x=4 Band **6** Heft **8** Seiten **9** Beigaben **a** ISO-Abk. **c** Coden **i** ISSN **s** Signatur **z** ZDB-Nr**)**	
	#71	**Auflage-/Ausgabebezeichnung**	
	#72	**Reprintvermerk** (z.B. "Nachdr. d. Ausg. ...")	
	#73	**Orte und Verlage bzw. Drucker in Vorlageform** (optional, ersetzt im Druck #74 + #75)	
	#74/#74d	**Erscheinungsort(e) / Druckort** (in Ansetzungsform , Trennung "; ")	**6,a**
	#75/#75d	**Verlag(e) / Drucker** (Ansetzungsform , Trennung "; ")	**6,a**
e	**#76**	**Erscheinungsjahr** (stets mit 4 Ziffern, evtl. "ca. ..." oder "vor ...", Klammern erlaubt)	**6,a**
	#76x	**Erscheinungsangaben** (x = Erscheinungsform, z.B. bei Zeitschriften: #76p Ersch.Verlauf)	
	#77	**Umfangsangabe : Illustr. + Begleitmaterial ; Format**	
	#77x	**Medienspezifische Angaben** (x = Mediencode = physische Form, z.B. #77v Video)	

Gruppe #8 : Fußnoten, Gesamttitel, Identnummern

	#80	**deutsche Titelfassung (vom Katalogisierer übersetzt) (optional)**	
	#81x	**Fußnoten** (#81 Allg.Fußnote, spezielle z.B. #81F/#81S Fußnote zu früheren/späteren Titeln)	
	#818	**Hochschulschriftenvermerk nach RAK**	
	#819	**Ort:Jahr** (normierter Hochschulschriftenvermerk)	**1D**
	#82	**NE-Vermerke** (nur für Zettelkatalog relevant)	
f	**#83**	**Veranstaltung ; Zählung (Ort) : Datum** (vor allem Kongresse; deutscher Ortsname)	**3,4**
	#84	**Sachtitel des übergeordn. Werkes [/ Verf.Name] [; Zählung]**	
	oder	**_IdNr des übergeordneten Werkes** (bei Aufsätzen aus Sammelbänden)	
f	**#85**	**Gesamttitel (Serientitel) ; Zählung**	**5,a**
	oder	**_Ident ; Zählung** (Ident = Kürzel oder andere Identifikation des Serien-Stammsatzes)	
	#86	**Reportnummer**	
g	**#87**	**ISBN = 2.ISBN = 3.ISBN** [die ISBN darf auch 13stellig sein]	**9i,a**
h	**#88**	**ISSN, #88x = andere offiz. Id.-Nrn. für fortlfd. Sammelwerke** (z.B. #88c CODEN)	**9I,a**
	#89x	**Andere IdNummern (z.B. #89D** DBNr, **#89L** LCNr, **#89Z** ZDB**, #89o** OpusNr...**)**	**9X**
	#8e	**Elektronische Adresse (URL etc.),** ▼t oder ▼y Nutzerhinweise, ▼z interne Angaben	**9**

Gruppe #9 : Bestands- und Verwaltungsangaben

#90	**[Standort']Signatur[= Magazinsignatur]**	**8**
#91	**Zugangs- oder Inventarnummer**	**9Z**
#92x	**Bestands- und Lückenangaben** (Zeitschr.u.Reihen; #92 = Hauptbestand, #92i = Institute)	
#93	**Inhaltsangabe (Kapitel, Bestandteile)** (Zeilentrennung mit ¶)	
#94	**Verknüpfung zu externen Ressourcen** (Graphikdateien, Textdateien etc.)	
#95,#96	**frei verwendbar (z.B. bibliotheksspezifische Daten)**	
#97x	**Geschäftsgangsdaten** (z.B. #97a Zeitschr.kennung: 1=laufend, 3=abgeschlossen)	
#98x	**Abstract** (zusätzlich #980, #981, ..., wenn 3000 Zeichen nicht genug sind), oder z.B: **#98E engl.**	
#99e	**Änderungsdatum** (→ A.CFG : Befehl ce99e)	
#99n	**Zugangsdatum (Erfassungsdatum)** (→ A.CFG : Befehl cn99n)	**9D**
#9A - #9E	**System-, Erwerbungs- Ausleih-, Exemplardaten** (bilden eigene Datensätze)	**10,11**
#9s	**Allgemeine Verweisungen für die Register (z.B. #9s |5centralblatt -> zentralblatt)**	

Jede Kategorie ist bei Bedarf beliebig oft belegbar: man wiederholt die Eintragungen entweder innerhalb der Kategorie und setzt ein Trennzeichen dazwischen, meistens ";" (bei #6x und #87: " = ") oder '¶' (Eingabe mit <Strg>+t). Oder man wiederholt die Kategorie als solche, indem man hinter der Kategorienummer ein Wiederholungszeichen (empfohlen: 2,3,4,...) eingibt. Zum Beispiel wird meistens `#402` für den zweiten und `#403` für den dritten Verfasser verwendet.

Katalogisieren - Was ist das überhaupt? Zur Terminologie

Mehr zum Thema: http://www.allegro-c.de/regeln

In knapper Form finden Sie hier die Grundbegriffe der **Formalkatalogisierung** in Anlehnung an die "Regeln für die alphabetische Katalogisierung (RAK)". In Fettdruck sind die festgelegten Fachbegriffe hervorgehoben.

Man hat bei der Erfassung von Buch- oder Literaturdaten immer ein konkretes Exemplar (Buch, Aufsatz, etc.) vorliegen. Dieses Exemplar, die sogenannte **Vorlage**, gehört zu einer bestimmten **Ausgabe** (oder Auflage) einer Veröffentlichung, denn jedes **Werk**, als **geistige Schöpfung** definiert, kann in verschiedenen, mitunter sehr zahlreichen Ausgaben veröffentlicht worden sein. Man könnte auch sagen: das "Werk" ist eine abstrakte Vorstellung, man sieht nie das Werk selbst, sondern immer nur eine konkrete, in bestimmter Form erschienene Verkörperung, und *diese* ist es, die man katalogisiert. Selten wird dies bewußt, doch hat es wichtige Konsequenzen.

Die **Vorlageform** eines Namens oder Titels ist jeweils diejenige Schreibweise, die in der Vorlage tatsächlich zu finden ist. Die **Ansetzungsform** ist dagegen eine vereinheitlichte, nach festen Regeln "normierte" Schreibung des Namens oder Titels. Konsistente Ansetzungsformen sind von großer Bedeutung für das **Zusammenführen** aller Werke und Ausgaben desselben Verfassers und für die **alphabetische Ordnung** in Zettelkatalogen, sortierten Listen und natürlich auch in Indexdateien! Denn wird ein Name einmal so und das nächste Mal anders eingegeben, ist das für jede Art von Katalog bei der Suche ein Problem. Wenn die normierte Form von der Vorlageform deutlich abweicht, ist die letztere in der Regel zusätzlich zu erfassen – wenn der Name so zitiert wird, muß man ihn auch so finden können! Ansetzungs- und Verweisungsformen können auch als eigene Datensätze (sog. **Stammsätze**) in **Normdateien** festgehalten werden (→ B.3). Version 14 ermöglichte erstmals ein komfortables Verknüpfen von Titelsätzen mit Stammsätzen (→ 10.2.6.8). Ein Stammsatz muß dabei eine eindeutige Nummer oder ein "Kürzel" haben (Beispiel: "tucho" in der Demodatenbank). Dieses wird statt des Namens, Schlagworts, Serientitels etc. in die betreffende Kategorie eingetragen, d.h. man schreibt z.B. #40 _tucho. Die Programme können, wo notwendig, die Nummer bzw. das Kürzel durch den Klartext ersetzen, also "_tucho" durch "Tucholsky, Kurt".

Am Zustandekommen von Werken sind **Personen** und/oder **Körperschaften** (das sind Personenvereinigungen) beteiligt. Wenn nur eine einzelne Person oder bis zu drei Personen ein Werk erarbeitet haben, gelten diese als **Verfasser**. Wenn es mehr als drei sind, oder wenn sie in irgendwelchen anderen Funktionen an der Erarbeitung eines Werkes beteiligt gewesen sind, spricht man von **beteiligten Personen** statt von Verfassern. Welche von diesen und wieviele man erfaßt, ist Sache des Regelwerks, nach dem man arbeitet. (Für Nichtbibliothekare sei angemerkt, daß man im deutschen Katalogisierungswesen den Terminus "Autor" nicht benutzt, sondern "Verfasser".)

Wenn keine Personen als Verfasser genannt oder zu ermitteln sind, spricht man von einem **anonymen Werk**. Ein solches kann immerhin noch von einer Körperschaft **erarbeitet** oder wenigstens **veranlaßt und herausgegeben** worden sein. In diesem Fall gilt die Körperschaft als **Urheber** des Werks. Sonst, wenn sie nicht so maßgeblich beteiligt war, gilt sie nur als **beteiligte Körperschaft**. Ein Verlag gilt nicht als Körperschaft!

Nun erscheint nicht jedes Werk immer zusammenhängend und abgeschlossen in einem Band. Auch ein in sich abgeschlossenes **Einzelwerk** kann in zwei oder mehr Teilen, als **mehrteilige Ausgabe** erscheinen. (Die gängige Bezeichnung "mehrbändiges Werk" ist eigentlich unsinnig, denn ein Werk als *geistige Schöpfung* kann nicht aus mehreren Bänden bestehen! Das kann nur eine konkrete Ausgabe des Werkes – und es kann daneben ja auch eine einbändige geben.)

Andererseits können auch mehrere Einzelwerke (Aufsätze, Vorträge o.a.) vereinigt als **Sammelwerk** herauskommen (**Sammlung** genannt, wenn alle vom selben Verfasser sind). Die Bestandteile gelten dann als **unselbständige Werke**. Eine Aufsatzsammlung oder ein Tagungsbericht wäre ein **abgeschlossenes Sammelwerk**, eine **Schriftenreihe** oder erst recht eine **Zeitschrift** bezeichnet man in ihrer Gesamtheit als **fortlaufendes Sammelwerk**. Die einzelnen Stücke einer Schriftenreihe (auch "Serie" genannt) sind meist Einzelwerke, in Bezug auf die Reihe sind es **Stücktitel**.

Die regelgerechte, beschreibende Darstellung aller dieser Arten von Werken in einer Datenbank, die sogenannte **bibliographische Beschreibung**, erfordert die Erfassung einer Anzahl unterschiedlicher Datenelemente, wie Sie aus der Kurzübersicht bereits entnehmen. In der separat erhältlichen ausführlichen Darstellung wird auf die Einzelheiten genauer eingegangen. Um die **Interpunktion** zwischen den Elementen braucht man sich wenig zu kümmern, die kann ***allegro*** mit seiner Exportsprache selbst einsetzen, auch z.B. Klammern vor und hinter dem Serientitel.

Von der **Sachkatalogisierung** oder **Sacherschließung** war bis hierher noch gar nicht die Rede. Die Zuordnung des Werkes zu einer **Sachgruppe,** die Charakterisierung seines Inhalts mit **Schlagwörtern**, Schlagwortketten oder genormten Fachbegriffen (**Thesaurus**begriffe) ist ein ganz eigener Problemkomplex. Die notwendigen Kategorien finden sich in der Gruppe #3 = Sachliche Erschließungsdaten. Für Nichtbibliothekare: die Wörter, die für den Index automatisch aus den Titeln entnommen werden, sind keine Schlagwörter, sondern bloß sog. **Stichwörter**.

Mit ***allegro*** können Sie neue Konzepte der Sacherschließung realisieren. Mehrere Ansatzpunkte enthält die Index-Parameterdatei CAT.API, die Sie anhand der mitgelieferten Beispieldatenbank studieren können.

Die vollständige Dokumentation des "konsolidierten Formats" (online veröffentlicht als **Format 2004**) enthält auch Listen von Standardcodes für Formschlüssel und Sprachen und eine (erweiterungsfähige) Grobklassifikation, mit denen Sie arbeiten oder experimentieren können. Für die Sachkategorien kann man sich in a99 eigene Eingabeformulare schaffen. (In a99 den Befehl `h form` geben, darin wird das Konzept erklärt.)

Beispiel: Erfassung eines 2bändigen Werkes, Variante A: Verknüpfte Speicherung

```
#00 IdNummer        für die Verknüpfung braucht man eine Identifikationsnummer,
                    z.B. könnte es eine Zugangsnummer sein oder auch die ISBN
#20 Was können wir wissen?
#30aNA PH
#40 Vollmer, Gerhard
#74 Stuttgart
#75 Hirzel
#76 1985
#77 Bd.1.2.
```

Nachdem diese Hauptaufnahme eingegeben ist, diese **speichern** (DOS: **F10** , a99: Alt+s).
Dann wieder mit **I** bzw. F9 die Eingabe aufrufen, jetzt aber Typ "Verknüpfte Bandaufnahme" wählen.

```
#00 IdNummer+1      die IdNummer der Hauptaufnahme wird vorgegeben, "+1" gibt man ein
#20 Die Natur der Erkenntnis : Beiträge zur Evolutionären Erkenntnistheorie
#25 Mit einem Geleitw. v. Konrad Lorenz
#77 337 S.
#87 3-7776-0403-8
#90 2647-3611
```

und nun wieder **F10** bzw. Alt+s, dann in derselben Weise die Daten für Band 2:

```
#00 IdNummer+2=Bd. 2      Hinter '=' kann man eine Bandnummer in Textform eingeben
#20 Die Erkenntnis der Natur : Beiträge zur modernen Naturphilosophie
#77 350 S.
#87 3-7776-0404-6
#90 2650-9705
```

Band 2 ebenfalls speichern.
DOS: Bei verknüpfter Erfassung kann man **nicht** mit **Bild↓** und **Bild↑** zwischen den vorhandenen verknüpften Aufnahmen während der Eingabe blättern! (Das geht nur bei Variante B, siehe nächste Seite.)
Wenn zwischen zwei Bandaufnahmen eine neue einzufügen ist, hat man 2 Möglichkeiten:

a) man holt die Hauptaufnahme in die Anzeige, gibt **I** bzw. F9 und erfaßt dann genau wie oben bei Band 1 und 2 gezeigt. (bei Typ v entnimmt das Programm die *IdNummer* aus der angezeigten Aufnahme!)

b) man holt einen der Bände in die Anzeige, drückt **C** (bzw. F9 und "Kopie") und bearbeitet die Kopie der Bandaufnahme, insbesondere in #00 setzt man die Bandnummer ein.

Da die Bände getrennt gespeichert werden, ist die in #00 angegebene *IdNummer*+*BandNummer* sehr wichtig, um dem Programm die Anzeige in der korrekten Reihenfolge zu ermöglichen. Geben Sie hinter dem '+' evtl. eine 2- oder 3-stellige *BandNummer* mit führenden Nullen ein, dann dahinter mit '=' eine Textform der Bandnummer. Die letztere wird für die Anzeige benutzt, die erstere für die Ordnung. Die Reihenfolge wird automatisch über die *BandNummer* hinter dem '+' hergestellt, sie kann also leicht verändert werden, indem man diese Nummer ändert.
Blättern: Steht bei der Suche einer der Bände oder die Hauptaufnahme in der Anzeige, kann man mit **B** zum nächsten Band vorblättern, mit **b** zum vorigen zurück. (Dieser Vorgang stoppt bei der letzten bzw. ersten Einheit.)
Jeder Untersatz kann mit Druck auf <Entf> gelöscht werden, der Hauptsatz jedoch nicht, solange noch ein verknüpfter Untersatz vorhanden ist.
In der Demodatenbank finden Sie ein umfangreicheres Beispiel unter **037277219** im Register 9. Machen Sie damit eigene Versuche in der oben beschriebenen Weise. Sehen Sie sich auch mit Shift+F7 an, welche Registereinträge aus den verschiedenen Teilen entstehen.

Vorteile dieser Speichermethode: bei umfangreichen mehrbändigen Werken spürbar schneller als die hierarchische Methode. Keine Grenze für die Anzahl der Bände. Besser brauchbar für das Ausleihsystem.

Nachteile: Die Parametrierung ist schwieriger. Eine UND-Verknüpfung mit Begriffen aus zwei verschiedenen Teilen ist nicht möglich (also z.B. "vollmer + erkenntnis" liefert nichts). Dieses Problem kann aber seit V15 mit besonderen Maßnahmen umgangen werden (sog. "Satzübergreifende Suche", S. 220, Kap. 10.2.6.9).

Beispiel: Erfassung eines 2bändigen Werkes, Variante B: Hierarchische Aufnahme

Als Typ für die Erfassung wählt man zunächst `1 = selbständig (Buch)` und beantwortet dann die Fragen der Abfrageliste so:

```
#20 Was können wir wissen?
#30aNA PH
#40 Vollmer, Gerhard
#74 Stuttgart
#75 Hirzel
#76 1985
#77 Bd.1.2.
```

Nachdem diese Hauptaufnahme eingegeben ist: noch **nicht** abspeichern, also nicht F10 sondern
DOS: **F9** drücken , bei der Typabfrage **`u = Band`** wählen, um eine Unteraufnahme für den Band 1 einzugeben.
a99: Alt+# und Formular "Hierarchischer Untersatz", dann folgendes eingeben:

```
#01 Bd. 1
#20 Die Natur der Erkenntnis : Beiträge zur Evolutionären Erkenntnistheorie
#25 Mit einem Geleitw. v. Konrad Lorenz
#77 337 S.
#87 3-7776-0403-8
#90 2647-3611
```

und nun wieder **F9**, **Typ u** wählen, dann die Unteraufnahme für Band 2 eingeben:

```
#01 Bd. 2
#20 Die Erkenntnis der Natur : Beiträge zur modernen Naturphilosophie
#77 350 S.
#87 3-7776-0404-6
#90 2650-9705
```

Während der Bearbeitung eines mehrbändigen Werkes kann man mit **Bild↓** und **Bild↑** zwischen den vorhandenen Untersätzen **vor- und zurückblättern** (→ Kap.3.5.3) und den jeweils aufgeblätterten Untersatz bearbeiten. a99: man sieht links im Auswahlfeld immer den gesamten hierarchischen Satz. Wenn man ein mehrbändiges Werk in der Anzeige hat und mit 'E' in den Editor geht, erscheint zunächst nur die Hauptaufnahme; die Bände sind aber auch alle im Arbeitsspeicher! Mit **`#+ Enter`** erhalten Sie den ersten Band usw. Geben Sie den Befehl **`#s`**, dann sehen Sie die Gesamtzahl der Einheiten.
Wenn zwischen zwei Bandaufnahmen eine neue einzufügen ist, geht man so vor: man blättert zu dem Band, **vor** welchem der neue Band stehen soll, und gibt dann den Befehl `#I .` Auf der dann erscheinenden Typabfrage wählt man wieder u = Band und gibt die zugehörigen Daten ein. Jede weitere Kategorie, die in der Abfrage nicht berücksichtigt ist, kann natürlich mittels ihrer Nummer zusätzlich eingegeben werden.
Weitere Hierarchiestufen, `#02, #03, ..., #06,` kann man eingeben, indem man zuerst so tut, als würde man eine neue #01 erfassen wollen. Wenn die Abfrage zu Ende ist, fährt man zur #01 zurück und macht daraus eine #02 oder eine der anderen Stufen, je nachdem, was man braucht.
Einen Untersatz kann man **löschen**, wenn man ihn aufblättert und dann den Befehl `#V` gibt. (→ Kap.3.5.3)
Wenn ein Untersatz **verschoben** werden muß: man blättert ihn auf, gibt den Befehl `#M (markieren),` blättert dann zu dem Untersatz, **hinter** welchen er verschoben werden soll, und gibt dann den Befehl #m (move).
Erst wenn **alle Teile** korrekt erfaßt sind: zum **Abspeichern** der gesamten mehrteiligen Aufnahme dann F10 und zur Bestätigung 'j' geben.
Vorteile dieser Speichertechnik: die Erfassung ist einfach und schnell; bei der Suche ist die Verknüpfung "vollmer + erkenntnis" möglich (allgemeiner: bei Verknüpfungen dürfen die Suchbegriffe in unterschiedlichen Teilen der Gesamtaufnahme stehen).
Nachteil: bei umfangreichen Werken wird das Speichern langsam und das Auffinden und Bearbeiten einzelner Bände etwas mühsam. Die gesamte Länge ist zudem begrenzt auf ca. 20.000 Zeichen.
Empfehlung: Nur Werke bis etwa 8 oder 10 Bände auf diese Weise katalogisieren, umfangreichere entweder nach Art einer Serie (wenn es ein "Sachtitelwerk" ist) oder in verknüpfter Form; für dasselbe Beispiel finden Sie auf der vorigen Seite die verknüpfte Eingabetechnik.

Anzeige beim Schnellzugriff (gesteuert durch D-1.APR (PRESTO) bzw. d-wrtf.apr (a99)) :

```
Vollmer, Gerhard:
Was können wir wissen? / Vollmer, Gerhard. - Stuttgart : Hirzel,
1985. - Bd.1.2.

Bd. 1. Die Natur der Erkenntnis : Beiträge zur Evolutionären
       Erkenntnistheorie. - 337 S.                          2647-3611
       ISBN 3-7776-0403-8
Bd. 2. Die Erkenntnis der Natur : Beiträge zur modernen
       Naturphilosophie. - 350 S.                           2650-9705
       ISBN 3-7776-0404-6
```

und Ausgabe in Kartenform: (bei Druckbefehl d40+H, auch unter D-1.APR)

```
Vollmer, Gerhard:
Was können wir wissen? / Vollmer, Gerhard. - Stuttgart :
Hirzel, 1985. - Bd.1.2.


      Vollmer, Gerhard:                                        2)
      Was können wir wissen?...
                                                    2647-3611
      Bd 1. Die Natur der Erkenntnis : Beiträge zur Evolutionären
      Erkenntnistheorie. - 337 S.
         ISBN 3-7776-0403-8

          Vollmer, Gerhard:                                    3)
          Was können wir wissen?...
                                                2650-9705
          Bd. 2. Die Erkenntnis der Natur : Beiträge zur modernen
          Naturphilosophie. - 350 S.
                 ISBN 3-7776-0404-6
```

Unter CAT.API entstehen u.a. diese Zugriffsschlüssel: (mit **F7** können Sie das nach dem Speichern sehen)

```
|1vollmer, gerhard
|3beitraege
|3erkenntnis
|3erkenntnistheorie          ( |3xyz bedeutet: xyz kommt in Index 3 )
|3evolutionaeren
|3koennen
|3modernen
|3natur
|3naturphilosophie
|3wir
|3wissen
|4erkenntnis der natur       (auch die Bandtitel werden indexiert!)
|4natur der erkenntnis
|4was koennen wir wissen
|61985
|6hirzel,1985
|6stuttgart,1985
|7na,1985
|7ph,1985
|82647-361
|82650-970                   (Signaturen spezifisch für UB Braunschweig)
|9i3-7776-0403
|9i3-7776-0404
```

B.2 Normdatensätze = Stammsätze

http://www.allegro-c.de/doku/form2004/for3.htm

Die `cat.api`-Datenbankstruktur auf Basis `$a.cfg` kennt einige Zusatzkategorien für Normdateien (engl. "authority files"). Außer den Ansetzungsformen sind alle Kategorien, soweit sinnvoll, in sich wiederholbar, indem man Mehrfacheinträge durch `"; "` trennt. Für den Index wird dann jede Eintragung gleichwertig berücksichtigt.
Will man V14-Verknüpfungen realisieren, muß jeder Stammsatz eine eindeutige **`#00`** mit IdNummer oder Kürzel haben.

B.2.1 Personennamen

Normdatensätze für Personennamen können folgende Kategorien enthalten: (vgl. oben die Ausführungen zur Gruppe **#4**)

`#00` IdNummer oder Kürzel
`#4n` Personenname in Ansetzungsform
`#4na` Verweisungsformen (andere Schreibweisen etc.)
`#4nd` Lebensdaten (Geburtsjahr-Sterbejahr)
`#4np` Pseudonyme (falls Wirklicher Name in #4n)
`#4nr` siehe-auch-Hinweise
`#4nt` Wirklicher Name (falls Pseudonym in #4n)
`#4nz` Anmerkungen

Beispiel: (rechts darunter sehen Sie die automatisch entstehenden Indexeinträge im Register 1 (bei `cat.api`)

```
#00 tucho                    Kürzel für V14-Verknüpfung
#4n Tucholsky, Kurt
#4nd 1889 - 1938
#4np Hauser, Kaspar; Tiger, Theobald
                                    tucholsky, kurt _tucho
                                    hauser, kaspar [pseud.] -> tucholsky,
                                    kurt
                                    tiger, theobald [pseud.] -> tucholsky,
                                    kurt
```

Der erste dient als "Übernahmeschlüssel": Bei der Erfassung kann man damit das Kürzel finden und übernehmen.

B.2.2 Körperschaftsnamen

Stammsätze für Körperschaftsnamen können folgende Kategorien enthalten:

`#6n` Körperschaftsname in Ansetzungsform
`#6na` Verweisungsformen
`#6ni` Identnummer (GKD) (wichtig für Zugriff auf Stammsatz)
`#6nr` siehe-auch-Hinweise
`#6nf` frühere Namen
`#6ns` spätere Namen
`#6nd` "Lebensdaten" (dates)
`#6nz` Anmerkungen

Beispiel: (rechts darunter sehen Sie die automatisch entstehenden Indexeintragungen, bei `cat.api` im Index 2)

```
#6n SED
#6nr Sozialistische Einheitspartei Deutschlands
#6ns SED-PDS
#6nd 1946 - 1989                    sed **
                                    sozialistische einheitspartei
                                    deutschlands -> sed
```

Für die 1989 aus dieser Partei hervorgegangene SED-PDS müßte ein entsprechender Datensatz mit `#6n SED-PDS und #6nf SED und #6ns PDS` angelegt werden, und für letztere noch einer mit `#6n PDS und #6nf SED-PDS` (Allerdings nur, wenn entsprechende Veröffentlichungen vorliegen.)

B.2.3 Zeitschriften- / Serienstammsatz

```
#00   IdNummer
#20   RAK-Ansetzungstitel      (falls abweichend von #8n )
#8n   Zitiertitel Langform
#8na  Kürzel (für Verknüpfung mit Aufsatzdaten, → #70 )
#8ni  Zitiertitel, Kurz- und andere Formen  (i=1,2,3...)
#8nr  Verweisungsformen
#23k  "Key Title" (zur ISSN gehörig)
#23F  früherer Titel                  oder #81F
#23S  späterer Titel                  oder #81S  falls
                                 Registereintrag unerwünscht
#74   Erscheinungsort▼gCode für Ersch.Land
#75   Verlag
#76s  (bzw. #76p bzw. #76r) Erscheinungsverlauf (s.o. Gruppe #7)
#88   ISSN
#88c  CODEN
#89Z  ZDB-Nummer
```

Die Kategorien #88 und #88c sind wichtig für den Fall, daß man Verknüpfungen realisieren will. In den Parameterdateien P-*.APR und CAT.API sind Vorkehrungen dafür getroffen. Siehe auch **#70...**
Die #8na übernimmt bei diesen Stammsätzen die Rolle einer IdNummer für die V14-Verknüpfungen. Es ist für die Aufsatzkatalogisierung praktischer, ein sprechendes Kürzel zum Verknüpfen zu verwenden, als eine nicht-mnemonische IdNummer. Die #00 wird oft für die IdNummer eines Verbundes gebraucht.

Beispiel: (rechts darunter sehen Sie die entstehenden Indexeinträge, bei `cat.api` in Register 5)

```
#8n J.-G.-Herder-Jahrbuch
#23s Johann-Gottfried-Herder-Jahrbuch
#76r 1.1920 - 4.1923
                                    j-g-herder-jahrbuch ;
                                    johann-gottfried-herder-jahrbuch ->
                                    j-g-herder-jahrbuch
```

Ein entsprechender Datensatz muß für den späteren Titel angelegt werden; dort ist dann #23f zu benutzen.

B.2.4 Einheitssachtitel

```
#2n   Einheitssachtitel, Ansetzungsform
#2nr  Verweisungsformen in derselben Sprache
#2nd  Angaben zur Datierung
#2nz  Anmerkungen
#2nu  Übersetzungen mit <Sprachbezeichnung> als Ordnungshilfe
```

Ansonsten können die anderen Kategorien benutzt werden. Z.B. wird man #40 eingeben, wenn es sich um ein Verfasserwerk handelt.

Beispiel: (rechts darunter sehen Sie die entstehenden Indexeinträge, bei CAT in Register 4)

```
#2n Chanson de Roland
#2nr Rencevals <franz.>
#2nu Rolandslied <deutsch> ; Song of Roland <engl.>

                          chanson de roland *
                          rencevals -> chanson de roland
                          rolandslied -> chanson de roland <deutsch>
                          song of roland -> chanson de roland <engl.>
```

B.2.5 Thesaurus (→ 1.7)

In Klammern die geläufigen Abkürzungen für Thesaurusbestandteile.

```
#3s   Sortierbegriff (optional, kann zur Herstellung einer
      logischen Reihenfolge dienen)
#3t   Deskriptor = Thesaurus-Normbegriff
#3ta  Synonyme                  (UF : "used for"-Begriffe)
#3tr  verwandte Begriffe        (RT : related terms)
#3tb  Oberbegriffe              (BT : broader term)
#3tt  Sachgruppe                (TT : top term)
#3tp  frühere Begriffe          (PT : prior term)
#3ty  Definition, Anwendung     (SN : scope note)
#3tz  Anmerkungen
#3td  Datum der Erfassung/Bearbeitung
#3u   Übersetzungen in andere Sprachen (z.B. #3ue engl.)
```

Beispiel: wenn man den INSPEC-Thesaurus verwendet, könnte ein Datensatz so aussehen:

```
#3t microcomputers
#3ta personal computers; desk-top computers
#3tb digital computers
#3tt computers
#3tr workstations; minicomputers
#3td 1975.07
#3tp general purpose computers
```

Daraus lassen sich diese Einträge für den Index erstellen: (bei CAT in Register 3)

```
computers  NT> microcomputers
desk-top computers  USE> microcomputers
digital computers  NT> microcomputers
general purpose computers  USE> microcomputers
microcomputers **
microprocessor chips  RT> microcomputers
minicomputers  RT> microcomputers
personal computers  USE> microcomputers
workstations RT> microcomputers
```

Hinweis: Unterbegriffe werden nicht eigens erfaßt! Es reicht völlig aus und vereinfacht die Verwaltung eines Thesaurus, wenn man nur die Oberbegriffe mit eingibt. Denn daraus entstehen dann im Register die Hinweise vom Oberbegriff zu allen Unterbegriffen (vgl. oben "digital computers" und "computers").

Will man die Technik der Satzverknüpfungen einsetzen, wäre folgendermaßen vorzugehen:
In #00 wird eine eindeutige Identifikationsnummer untergebracht. Diese Nummer gibt man statt des Deskriptors im Dokumentsatz ein, und zwar in #31, #311, #312, ... (getrennte Kategorien hier günstiger).
Vor die Nummer setzt man ein '_' , also z.B. `#31 _222222` . Mit der Methode der Stammsatzverknüpfung (→ 10.2.6.8) kann man den Deskriptor dann aus der Stammdatei holen. Im Index findet man damit alles unter den Deskriptoren, während in den Datensätzen nur die Identnummern stehen. Die Verfahrensweise ist in den Parametern der Beispieldatenbank von Version 14 allerdings noch nicht enthalten.
`cat.api` und `d-1.apr` bieten die Thesaurus-Stammsätze als Alternative an - es ist auch möglich, Schlagwort- und Systematikstammsätze im selben Format unterzubringen, → B.2.6.

B.2.6 Sacherschließungs-Stammsätze

Normsätze für alle Arten von Sacherschließungen können mit diesen Kategorien einheitlich verwaltet werden:

```
#3n   TypNotation Hauptbegriff
#3na  Synonyme: Begriffe, von denen auf die Notation verwiesen wird
#3nr  siehe-auch-Verweisungen ("related terms")
#3nt  Textform der Begriffe (evtl. für gedruckte Ausgabe der Systematik)
#3nz  Kommentar zur Notation
#3q   Fremdnotationen, z.B.
      #3ql  Library of Congress Classification
      #3qd  Dewey-Notation
```

Der **Typ** hat einen oder zwei Buchstaben zur Kennzeichnung der **Notation**. Empfohlen werden hier folgende Typen, die in `cat.api` berücksichtigt sind:

c = Grobsystematik
d = Dezimalklassifikation (UDK)
l = Sprachschlüssel
g = Geographischer Schlüssel
h = historischer Code (Jahreszahlen einer Epoche)
a = Aufstellungssystematik (dahinter eine Ziffer für den Aufstellungsbereich), **k** = Basisklassifikation (Pica-Verbund)
s = Schlagwort-Stammsatz; hinter dem **s** folgt dann ein zweiter Buchstabe für den Schlagworttyp:

- **s** = Sachschlagwort
- **p** = Personen-Schlagwort
- **k** = Körperschafts-Schlagwort
- **c** = Ortsgebundenes Körperschafts-Schlagwort
- **g** = Geographikum
- **t** = Titel-Schlagwort (*Form:* Person: Titel)

#3na, #3nr und #3q sind in sich wiederholbar, indem man ein '; ' zwischen die Eintragungen setzt.

Beispiel 1: wenn man die UDK verwendet, könnte ein Datensatz so aussehen:

```
#3n d58 Botanik   (Der  Typ  ist 'd', die Notation ist 58)
#3na Pflanzenkunde
#3nr Ackerbau
#3ql QK Botany
#3qd 581 Botany
```

Und daraus würden bei `cat.api` diese Index-Einträge entstehen:

```
|3botanik !
|3pflanzenkunde !
|3ackerbau SIEHE AUCH -> botanik
|7DK58              Botanik
|7LC:QK Botany -> DK58
|7DY:581 Botany -> DK58
```

Fährt man beim Index-Zugriff auf eine der Zeilen, die "->" enthalten und drückt dort Enter, so wird durch die Referenzierungstechnik sofort die hinter dem "->" stehende Stelle des Index aufgeschlagen, im Beispiel wird so von "ackerbau" umgeblättert zu "botanik".
In den Buchdatensätzen braucht man also nur unter #30 die Notation 58 einzutragen, dann sind alle so klassifizierten Dokumente über alle oben genannten Indexeinträge zu finden.

Vom Konzept her etwas anderes als die herkömmliche Systematik ist die sogenannte **Grobsystematik** (→ #30a) , auch **Sachgruppenschlüssel** genannt. Dieses Konzept ist auch oder gerade dann anwendbar, wenn man keine Klassifikation im eigentlichen Sinne benutzt. Die Grobsystematik erlaubt es, größere Bestände bei Bedarf nach einem fachlichen Kriterium zu differenzieren und Fachbestände zu selektieren, was über die verbale Sacherschließung nicht möglich ist. Im Index läßt sich der Fachgruppenschlüssel zusammen mit dem Formschlüssel (→ #0c) und dem Erscheinungsjahr zu einem recht nützlichen Zugriffskriterium verarbeiten. Dies leistet die

Indexparameterdatei `cat.api`. Mehr zu dem Thema finden Sie in der ausführlichen Dokumentation des konsolidierten Formats, einschließlich einer Vorschlagsliste für eine Grobsystematik. Diese Liste ist als Normdatei in den Online-Katalog der UB Braunschweig integriert und auf Anfrage erhältlich. Wenn Sie die Struktur `cat.api` verwenden, können Sie diese kleine Normdatei mitverwenden: Sie erzeugen zunächst die Datenbank (wenn Sie noch keine haben) und geben einige Titel ein. Dann kopieren Sie `cat_255.ald` auf Ihr Datenverzeichnis und lassen z.B. mit der Funktion `INDEX -f71 -n255` diese Datei in Ihre Datenbank einarbeiten, wobei letztere die Nummer 255 behält, also von den Titeldaten getrennt bleibt. (Unter a99: Datei als "Weitere Offline-Datei" laden und dann mit "Datei / Offline-Datei -> Datenbank" speichern.)

Beispiel 2 SWD-Stammsätze. Hier drei Datensätze, wobei der dritte einen Oberbegriff zu den beiden anderen darstellt. In `#3nr` stehen die SIEHE AUCH-Begriffe, wobei die Oberbegriffe noch zusätzlich mit ▼b gekennzeichnet sind.

```
#00 s4048829-9
#3n sspl Rechtsradikalismus ("ss" = Typ Sachschlagwort,  "pl" = Sachgruppe)
#3naNeonazismus; Rechtsextremismus
#3nrFaschismus; Neue Rechte; Rechtsradikaler;▼bRadikalismus
#3nz▼aW Pol▼S8.1▼P292
```

Satzanzeige: (mit D-1.APR)

Sach-SW: Rechtsradikalismus
Synonym: Neonazismus; Rechtsextremismus
Sachgebiet: Politik
siehe auch: Faschismus; Neue Rechte; Rechtsradikaler
Oberbegriff: Radikalismus
Quelle: W Pol
DB-Sachgrp.: 8.1 (*Erfassung:* 292)

```
#00 s4136650-5
#3n sspl Linksradikalismus
#3naLinksextremismus
#3nr;▼bRadikalismus
#3nz▼aAu=DB▼S8.1▼P292
#00 s4048171-2
#3n sspl Radikalismus
#3naExtremismus; Radikale <Politik>
#3nz▼aM▼S8.1▼P292
```

Insgesamt entstehen folgende Schlüssel in den Registern 3, 7 und 9:

```
|3faschismus SIEHE AUCH -> rechtsradikalismus
|3neonazismus -> rechtsradikalismus
|3neue rechte SIEHE AUCH -> rechtsradikalismus
|3radikalismus SIEHE AUCH -> rechtsradikalismus
|3rechtsextremismus -> rechtsradikalismus
|3rechtsextremismus SIEHE AUCH -> rechtsradikalismus
|3rechtsradikaler SIEHE AUCH -> rechtsradikalismus
|3rechtsradikalismus
|7pl    rechtsradikalismus
|9s4048829
|3linksextremismus -> linksradikalismus
|3linksradikalismus
|3radikalismus SIEHE AUCH -> linksradikalismus
|7pl    linksradikalismus
|9s4136650
|3extremismus -> radikalismus
|3politik SIEHE AUCH -> radikalismus
|3radikale <politik> -> radikalismus
|3radikale SIEHE AUCH -> radikalismus
|3radikalismus
|7pl    radikalismus
|9s4048171
```

B.2.7 Praxistips für Normdaten

http://www.allegro-c.de/doku/form2004/

Allgemeine Hinweise zu Normdateien werden in Kap. 1.6 gegeben. Hier geht es nun um die konkrete Realisierung verschiedener Möglichkeiten im konsolidierten Format.
Die beste Methode, sich mit der Technik der Normdatensätze vertraut zu machen ist die, daß man alle Möglichkeiten praktisch ausprobiert. Legen Sie eine `cat`-Datenbank an (→ 1.1) oder, einfacher, benutzen Sie zum Testen die Beispieldatenbank, und geben Sie von jedem Typ einen oder mehrere Datensätze ein. Dazu brauchen Sie nur in der Erfassungsfunktion (→ `1.5, Funktion` 'I') zuerst den Typ '3' = Normdaten zu wählen und dann den entsprechenden Untertyp für den Normdatensatz (z.B. 'p' für Person). Dann fragt der Editor die nötigen Elemente ab.
Nach dem Abspeichern mit F10 geben Sie sofort **F7** und sehen die Liste der Indexeinträge, die aus dem gerade eingegebenen Datensatz entstanden sind. Blättern Sie mit **F6** oder "Cursor rechts" um zur Indexanzeige und sehen Sie sich die Eintragungen im Kontext an.

Methoden für die Übernahme von Normdaten in bibliographische Sätze (Arbeitsweise in PRESTO)

1. Vor der Erfassung einer Aufnahme suchen Sie z.B. im Index die Eintragung eines benötigten Personennamens, setzen den Pfeil drauf und geben ***Esc*** und einen Großbuchstaben, z.B. ***Esc*** `G`. Wenn Sie den Namen dann bei der Eingabe benötigen, tippen Sie nur ***Esc*** `g`, und er erscheint. Evtl. ist es nötig, Umlaute wiederherzustellen, die ja im Index aufgelöst erscheinen.

2. Eine andere Möglichkeit: zuerst den Normdatensatz aufschlagen, z.B. einen Personennamenssatz. Mit **Alt+F5** eine Kopie anfertigen, dann mit 'I' in die Erfassung gehen. Man gelangt in die Abfrageroutine und gibt alles ein bis auf die Namensangabe. Die holt man sich, indem man nach der Abfrage den Befehl **#a4n** gibt (bzw. **#a6n**, wenn es eine Körperschaft ist) : dann erscheint der vorher in den Hintergrund kopierte Namenssatz. Mit dem Cursor fährt man auf die Kategorie #4n und macht aus dem 'n' eine '0'. Schon hat man die Namensform, und diesmal völlig korrekt auch mit Umlauten. (Vgl. → Kap.3.5.3, Befehl #a)
   Oder: Man gibt den Befehl `#t4n ,40` , dann wird #4n in #40 umgewandelt und in die aktuelle Aufnahme übernommen.
   Bis zu 3 Normsätze kann mit mit der Alt+F5-Methode kopieren. Im Hintergrundspeicher stehen dann die Kategorien unter #4n, #4na und #4nb. Mit **`#t4na,41`** kopiert man z. B. den unter #4na im Hintergrund stehenden Namen in die Kategorie #41 der aktuellen Aufnahme.
   Mit **Shift+F5** wird der Hintergrundspeicher wieder aufgeräumt (nur der aktuelle Satz verbleibt darin). Nötig ist das zwar nicht, aber bisweilen kann es dort unübersichtlich werden.

3. **V14-Methode** unter ***a99***. In der Demo-Datenbank sind ab Version 14 einige Beispiel-Stammsätze enthalten.
   Im Register 1 finden Sie einen für Kurt Tucholsky, im Register 5 einen für die Zeitschrift "Bibliotheksdienst". Hinter der Eintragung sieht man jeweils das "Kürzel" (es könnte genausogut eine Identnummer sein) mit einem Unterstrich davor. Nun arbeitet man bei der Erfassung so:
   In dem Moment, wo der Name bzw. der Zeitschriftentitel für die Eingabe gebraucht wird, macht man folgende Schritte: F6 um in den Index zu springen, im Übernahmeregister (10 unter ü) den Namen bzw. den Titel suchen (also in diesem Fall z.B. `ütuc` eingeben), Zeiger auf die Zeile setzen, in der das Kürzel (die Identnummer) hinter dem Namen zu sehen ist, <Alt+k> um in die Erfassung zurückzuspringen : `_tucho` erscheint im Eingabefeld.
   (In PRESTO: <Strg><Enter> um das Kürzel (die Identnummer) zu übernehmen.)
   Diese Methode ist mit Abstand die rationellste. Sie setzt eine Parametrierung voraus, die die neuen Möglichkeiten von V14 ausnutzt (→ 10.2.6.8).

Achtung: Wenn die Stammsätze in eigenen Dateien stehen sollen: vor der Erfassung eines Normsatzes mit Funktion 'F' die Dateinummer umschalten! Sonst werden die Normsätze in die Dokumentdaten mit eingemischt. Das macht zwar nichts, und mit SRCH können Sie alles später wieder entwirren (Suchbegriff z.B. "#3n", um alle Sacherschließungs-Stammsätze herauszufischen). Allerdings muß man danach neu indexieren. Wenn also alles säuberlich getrennt sein soll, muß man darauf achten.
In a99: `x set n251` eingeben, um etwa die Dateinummer 251 einzustellen.

B.3 Geschäftsgangsdaten (Erwerbung und Ausleihe)

Im Erwerbungssystem hat man es mit **Bestellsätzen**, **Lieferantensätzen** und **Exemplarsätzen** zu tun. Letztere werden auch im Ausleihsystem gebraucht. Dort gibt es dann natürlich auch noch **Benutzersätze**. Neben diesen 4 Satztypen kommt noch eine Anzahl von Systemsätzen hinzu. Deren Beschreibung ist der Dokumentation zum Erwerbungs- bzw. Ausleihprogramm zu entnehmen. Wir dokumentieren hier die genannten vier Satztypen deshalb, damit interessierte Anwender, die einen späteren Einsatz der anderen Programme vorhaben, schon jetzt die entsprechenden Daten aufbauen können, auch ohne die Programme schon zu besitzen.
Wichtig: Die hier beschriebenen Satzstrukturen können auch dann ohne Änderungen benutzt werden, wenn nicht das Kategorienschema $A.CFG angewendet wird, sondern irgendein anderes. Da es zur Zeit keine im Bibliothekswesen gebräuchliche Normierung dieser Datenelemente gibt (auch nicht innerhalb der MAB-Formate), besteht kein Grund, andere als die hier beschriebenen und erprobten Strukturen erfinden zu wollen. Im Gegenteil: wenn man sich an diese Vorschläge hält, kann man sehr viel Parametrierungsarbeit sparen, denn die unter dem A-Schema entwickelten Parameter können dann übernommen werden. Aus diesen Gründen werden hier diese Felder in aller Ausführlichkeit dargestellt.

1. Bestellsätze

Man benötigt nur **eine** zusätzliche Kategorienummer für alle Erwerbungsdaten. Die Programme ORDER und aLF sehen standardmäßig #9D dafür vor, im Bedarfsfall kann man eine andere nehmen.
Ein Bestellsatz besteht aus nur zwei Kategorien:

`#9DA`*bestellnummer*`▼T`*titelidentnummer* ... (Identor)

und

`#9DB`*status*`▼` ... (Bestelldaten)

Die "Identor"-Kategorie identifiziert die Bestellung, die "Bestelldaten"-Kategorie enthält die eigentlichen Bestellangaben. In der Identorkategorie können anstelle der Punkte Datumsangaben über den Geschäftsgangsablauf stehen:

▼O*datum*(*Kürzel des Vormerkenden*)
▼V*datum*(*Kürzel des Referenten*)
▼E*datum*(*Kürzel des Bearbeiters in der Erwerbung*)
▼B*datum*(*Kürzel des Bestellers*)
▼S*datum*(*Kürzel des Stornierenden*)
▼R*datum*(*Kürzel des Reklamierenden)*
▼I*datum*(*Kürzel des Inventarisierenden)*
▼A*datum*(*Kürzel des Abschließenden)*
▼F*datum*(*Kürzel des Freigebenden)*

Diese Eintragungen können u.U. jeweils mehrfach auftreten. Das Erwerbungsprogramm legt sie automatisch an.

Beispiel:

```
#9DA0010282▼T0-7100-0632-2▼V911002(kra)▼E911010(mue)▼B911012(sch)▼I911210(mue)
```

Interpretation:

Am 10.10.91 wurde durch die Erwerbung (Mitarbeiter mue) der Titel mit der Identifikation 0-7100-0632-2 vorakzessioniert und durch den Verantwortlichen sch ma 12.12.91 bestellt. Die Bestellung basiert auf einem Vorschlag des Referenten kra vom 02.10.91. Mitarbeiter mue hat am 10.12.91 die eingegangene Lieferung inventarisiert. Die Rechnung ist noch nicht beglichen worden (es fehlt der Zusatz ▼A...), eine Freigabe ist auch noch nicht erfolgt (es fehlt der Zusatz ▼F...).

Der status in der Bestelldatenkategorie kennzeichnet den Zustand der betreffenden Bestellung:

1 durch den Referenten vorgeschlagen,
2 durch die Erwerbung vorakzessioniert,
3 bestellt,
4 Bestellung reklamiert,
5 Bestellung storniert,
6 Bestellung inventarisiert,
7 Bestellung abgeschlossen,
8 Desiderat,
9 vorgemerkt.

Dem Status folgen die eigentlichen Bestelldaten: es sind Unterfelder, die mit dem Steuerzeichen ▼ und einem Kennbuchstaben identifiziert werden:

#9DB*status* Geschäftsgangs-Status

▼A...	Auftraggebercode,
▼a...	Kontingentcode,
▼c...	Währung, in der die Bestellung erfolgt,
▼C...	Anmerkung, Quelle,
▼d...	Bestelldatum in der Form `jjmmtt`,
▼D...	Fälligkeitsdatum in der Form `jjmmtt`,
▼e...	Eingangsdatum in der Form `jjmmtt`,
▼E...	Rechnungseingangsdatum,
▼f...	1. Mahnfrist,
▼g...	2. Mahnfrist,
▼h...	3. Mahnfrist,
▼i...	Rechnungsbetrag,
▼I...	Kommentar zur Bestellung,
▼j...	Lieferantencode,
▼J...	Lieferanten-Anschrift (falls kein Stammsatz),
▼K...	Rechnungskommentar,
▼l...	Nummer des Lieferscheins,
▼m...	Mehrwertsteuersatz in %,
▼M...	Mahnungszähler,
▼n...	Anzahl der bestellten Exemplare,
▼N...	Rechnungsnummer,
▼o...	Datum der letzten Mahnung in der Form `jjmmtt`,
▼p...	Preis je Exemplar in der Währung, in der bestellt wird,
▼P...	Bestellpreis (Basiswährung),
▼q...	Preis bei Lieferung,
▼Q...	Kommentar zur Lieferung,
▼R...	Rechnungsdatum in der Form `jjmmtt`,
▼r...	Rabatt in %,
▼s...	Code mit 6 Zeichen, der Publikationsform, Erwerbungsart, Fachgebiet und Bestellart in verschlüsselter Form enthält,
▼t...	Sonstige Nebenkosten,
▼T...	Titelzusatz (nur für Bestellung gedacht),
▼u...	Reklamationsgrund,
▼U...	Reklamationsdatum in der Form `jjmmtt`,
▼V...	Rechnungsvermerke,
▼v...	Anzahl der Bände je Exemplar (bei mehrbd. Werken),
▼w...	Bestellweg,
▼z...	Zeitraum (bei Abonnements).

Es gibt keine festgelegte Reihenfolge der Unterfelder. Es können auch Unterfelder fehlen, wenn sie nicht gebraucht werden.

2. Exemplarsätze

Exemplarsätze werden bei Inventarisierungen erzeugt und zwar je ein Satz für jede buchbinderische Einheit, d.h. jedes separat ausleihbare Objekt. Exemplarsätze enthalten immer die Kategorien

#9DF*titelidentnummer***▼B***bestellnummer* (Inhalt von Kategorie #00 des Titelsatzes bzw. verknüpften Bandsatzes, zu dem das Ex. gehört)

#9DG*status exemplarangaben...*

#9DH*ausleihangaben...* (temporär, nur solange ausgeliehen oder vorgemerkt)

#9DI*ausleihangaben...* (temporär, nach Rückgabe steht hier der vorherige Inhalt von #9DH)

Der *status* in der Kategorie #9DG spezifiziert die Exemplarverfügbarkeit. Bei der Inventarisierung wird er auf **'g'** (im Geschäftsgang) gesetzt. Andere Indikatorwerte werden bei den Vorgängen des Ausleihprogramms gesetzt, und zwar 'l' für "ausge**l**iehen", 'W' für "**W**arteregal für Vormerker", 'v' für "**v**erloren", 'x' für "Fahndung", 'f' für "**F**ernleihe", 'g' für "**G**eschäftsgang" und '0' für "verfügbar", 'A' für "ausgesondert".

Die Kategorie #9DH dient zur Aufnahme von Entleihungsangaben. Sie besteht nur während der Dauer der Entleihung und hat dann die Unterfelder ▼u (Benutzernummer), ▼D (Ausleihdatum), ▼R (berechnetes Rückgabedatum), ▼E Ende der Abholfrist (bei Vormerkung) und evtl. ▼M (Mahnungsdaten).

Unterfelder der Kategorie #9DG: (Die Angaben der rechten Spalte sind für alte Bücher gedacht!)

#9DG*status* Verfügbarkeits-Zustand (s.o.)

▼L...	Sigel der besitzenden Bibliothek (falls mehrere beteiligt sind)		
▼l...	Standort'Aufstellungssignatur		
▼s...	Signatur (Magazinsignatur)	▼P...	Provenienz
▼e...	Exemplarnummer	▼G...	Gaskell-Code
▼a...	Verbuchungsnummer (Barcode)	▼D...	Schäden des Exemplars
▼z...	Zugangsnummer	▼I...	Unregelmäßigkeiten
▼d...	Zugangsdatum	▼M...	fehlende Seiten
▼m...	Medientypkürzel		
▼B...	Sortierfähige Bandnummer, wenn Teil eines mehrbd. Werks oder fortlfd. Sammelwerks, falls dieses nicht in verknüpfter, sondern in hierarchischer Form erfaßt ist.		
▼b...	Bandbezeichnung in Textform, wenn abweichend von ▼B		
▼E...	Ausgabe/Auflage (bei Lehrbuch-Exemplaren)		
▼Y...	Erscheinungsjahr (bei Lehrbuch-Exemplaren)		
▼n...	Fußnote/Kommentar zum Exemplar/Bestand		
▼T...	Titelanmerkung		
▼v...	Ausleihbarkeit/Benutzungsmodus (z.B. k=Kurzausl., p=Präsenz, c=Kopieren nicht erlaubt)		
▼V...	Vormerkfrequenz (wie oft war das Ex. schon vorgemerkt)		
▼f...	Exemplarleihfrist		
▼w...	Wartezeit o.a. Angaben zur Bereitstellung		
▼F...	Ausleihfrequenz-Zähler (für Statistik)		
▼H...	Zusammenfassende Bestandsangabe (Zeitschriftenbesitz; nicht für Ausleihsystem!)		

Die Struktur des Exemplarsatzes kann auch in allgemeinerem Sinne für Besitzangaben (Holdings, s. Unterfeld ▼H) verwendet werden, wenn man einen Gesamtkatalog führen will. Dann stellt der "Exemplarsatz" eigentlich einen Bestandssatz dar.

Unterfelder der Kategorie #9DH:

#9DH▼u...	Identnummer des Ausleihers		
▼R...	Rückgabedatum	▼W...	Wartedatum (Abholtermin)
▼D...	Ausleihdatum		

Das Erfassen von Exemplarsätzen ("Inventarisierung") geht schnell und einfach mit dem ***aLF***-System - auch dann, wenn man es (noch) gar nicht zur Ausleihe einsetzt, oder mit dem Programm **INVENT**. Man sucht die Titelaufnahme, und mit `X a-exemp` (a99) bzw. mit 'I' (PRESTO) erhält man ein Eingabeformular, auf der alle Unterfelder bequem einzugeben und zu bearbeiten sind. Diejenigen Unterfelder, die sich aus der Titelaufnahme bereits ergeben, sind dann schon ausgefüllt. Zum Eintragen der Barcodenummer ist ein Lesestift oder Handscanner einsetzbar. Mit der Tastenkombination TAB ü bekommt man die Bestandsübersicht, aus der man ein Exemplar zur Bearbeitung auswählen kann.

3. Systemsätze : Bibliotheksadressen, Lieferantensätze, etc.

Für eine Anzahl von Aufgaben im Zusammenhang mit der Bestandsverwaltung und mit den Ausleih- und Erwerbungsfunktionen braucht man Daten, die nicht bibliographischer Natur sind, aber gleichwohl in flexibler Weise gespeichert werden müssen. Diese **Systemdaten** werden allesamt mit nur einer einzigen Kategorie, der **#9A**, abgespeist. Das erste Zeichen in dieser Kategorie zeigt dann an, um welchen Typ von Systemsatz es sich handelt:

- **A** Auftraggeber, Bibliotheksadresse
- **B** Bestellnummerngenerator
- **C** Klassensatz für Benutzerdaten (beschreibt die Eigenschaften einer Benutzergruppe)
- **I** Identnummerngenerator
- **K** Kontingentdaten
- **L** Lieferantendaten
- **P** Paßwortdaten
- **R** Kalenderdatensatz (für Datumsberechnungen in der Ausleihe)
- **S** Signaturgenerator
- **W** Währungstabelle
- **X** Textbaustein

Die Struktur dieser Datensätze ist in den Dokumentationen der Programme ***aLF*** und ***ORDER*** dokumentiert. Hier werden nur die zwei Typen näher beschrieben, die eine gewisse Relevanz zu den Katalogdaten haben:
Für die Bestandsverwaltung in Zentralkatalogen und Zeitschriften-Gesamtverzeichnissen braucht man Datensätze für Bibliotheksadressen.
Ein Adressensatz beginnt mit **#9A A...** und enthält die folgenden Teilfelder:

#9A	**A***Sigel*	Das Sigel der Bibliothek steht direkt hinter dem A
	▼N...	Eintrag für die Registerzeile in Reg. 11 (für Nachladung)
	▼n...	Namensform für die Anschrift
	▼a...	Anschrift
	▼t...	Telefonnummer
	▼O..	Öffnungszeiten
	▼B...	Benutzungsbedingungen
	▼Z...	Ansprechperson

Im Erwerbungssystem gibt es für eine Reihe von Systemdatensätzen die Verwaltungskategorie #9A (falls gewünscht eine andere). Ein besonders wichtiger Systemsatztyp ist hier der Lieferantensatz. Er beginnt mit **#9A L...** und enthält hinter dem **L** eine Anzahl von Datenfeldern für den Lieferanten:

#9A	**L***code*	Kurzbezeichnung des Lieferanten, bis zu 4 Zeichen
	▼n...	Lieferantenname (wie für Adresse erforderlich)
	▼a...	Lieferantenanschrift (mit '¶' als Zeilentrenner)
	▼k...	Kontaktperson (inkl. Anrede)
	▼l...	Land
	▼p..	Sprachkürzel (wird für Textbausteine benötigt)
	▼s...	Standardlieferfrist (wie mit Lieferant vereinbart)
	▼f/g/h...	Mahnfrist1/2/3
	▼t...	Telefon
	▼b...	Bank (Name, Ort und evtl. Anschrift)
	▼K...	Kontonummer
	▼z...	Bankleitzahl
	▼w...	Standardlieferweg
	▼F...	Faxnummer
	▼B...	Bemerkungen

4. Benutzerdaten

Ein **Benutzerdatensatz** besteht nur aus der Kategorie **#9B** und enthält maximal folgende Unterfelder: (welche tatsächlich genutzt werden, ist abhängig von der Benutzerklasse und bibliotheksseitig im jeweiligen "Klassensatz", einem Verwaltungssatz des Ausleihsystems, einstellbar) (Mehr: http://www.allegro-c.de/doku/form2004/)

#9B	i	Indikator ('0' oder '1')
	BenNr	Benutzernummer (Barcodenummer des Ausweises)
	▼k...	Klasse (Ziffer 1..9 oder Buchstabe a..z)
	▼n...	Name, Vorname
	▼u...	m oder w
	▼d...	Geburtsdatum
	▼a...	Alter
	▼f...	Fachgebiet
	▼b...	Beruf
	▼D...	Sperrdatum
	▼R...	Sperrgrund (Kommentar)
	▼G...	Summe der aufgelaufenen Gebühren
	▼s...	Straße und Hausnummer oder Postfach
	▼o...	Wohnort
	▼p...	Postleitzahl
	▼t...	Telefon
	▼N...	Name der Eltern (bei Studenten!); dann auch deren Anschrift:
	▼S...	Straße und Hausnummer
	▼O...	Ort
	▼P...	Postleitzahl
	▼T...	Telefon
	▼Z...	Ansprechperson (Wenn der "Benutzer" keine Person ist)

Das erste Zeichen der Kategorie #9B, vor den Teilfeldern, ist ein Indikator, und zwar '0', wenn der Benutzer ausleihberechtigt ist, und '1', wenn er gesperrt ist. Dann folgt die Benutzernummer, dann erst die Teilfelder. Die tatsächliche Reihenfolge der Teilfelder innerhalb des Datensatzes ist ohne Bedeutung.

Diese Satzstruktur kann auch für das Einfach-Ausleihprogramm ALFA benutzt werden, obwohl ALFA keine Benutzerdatensätze unbedingt benötigt. Es existiert für diesen Zweck eine leicht erweiterte Version der Standard-Parameterdateien (CAT.API und D-1.APR).

Anhang C

Diagramme - Fehler - Literatur

C.1 Funktionsdiagramme

Ein Gesamtüberblick über ***allegro*** läßt sich graphisch besser als verbal vermitteln. In diesem Anhang wird deshalb mit drei Diagrammen der Versuch gemacht, die auf den vorangegangenen ca. 300 Seiten wortreich erklärten Bestandteile als Gesamtkomplex überschaubar zu machen.

Diagramm 1 bildet extrem abstrakt ***allegro*** gewissermaßen als Kachel in einem nach allen Seiten offenen, das heißt ausgestaltbaren Verbund ab. Hier sehen Sie, nebenbei, was es mit dem Signet auf sich hat, das zum ersten Mal auf dem ***allegro***-Newsletter Nr. 11 verwendet wurde.

Diagramm 2 zeigt Abhängigkeiten und Interaktionen der wichtigsten Programme und Dateitypen. Man zieht es am besten bei der Lektüre der Abschnitte 0.3 und 0.4 als Illustration heran.

Diagramm 3 schließlich mag dem Einsteiger die beste Hilfestellung sein: es zeigt die "vier Gesichter" einer ***allegro***-Datenbank, d.h. die vier wesentlichen Bildschirmtypen, auf denen sich die ganze Arbeit mit einer Datenbank abspielt, und es zeigt, von welchem der Bildschirme man mit welchen Tasten zu welchen anderen Bildschirmen kommt. Die Erfahrungen auf Schulungen zeigen, daß so das Vorstellungsbild vom Funktionieren der Datenbank leicht aufzubauen ist. Benutzen Sie als dieses Diagramm als Orientierungshilfe bei den ersten Versuchen an der Demodatenbank.

Diagramm 1

Das Diagramm auf der folgenden Seite zeigt, daß ***allegro*** zwar ein in sich abgeschlossenes System, aber dennoch sozusagen nach vier Seiten offen ist. Dieses Handbuch stellt nur zwei dieser Seiten dar (3 und 4), die anderen zwei müßten einem Programmierhandbuch vorbehalten bleiben, welches noch nicht existiert.

1 Betriebssystem

In vollem Umfang läuft ***allegro*** auf MS-DOS-Rechnern und auf Windows- und OS/2-Rechnern in der sog. "DOS Box", für die Windows-Programme braucht man mindestens Win'95.. Eine UNIX-Implementierung war zunächst an der Herzog August Bibliothek in Wolfenbüttel im Einsatz; diese wurde von zwei weiteren Bibliotheken für die Erfassung von alten Büchern übernommen. Ab Sommer 1993 gingen weitere Implementierungen in Betrieb, zunächst auf SUN-Solaris, dann auch AIX und Linux, 1996 auch auf DEC-Alpha. Während des Jahres 1994 konnte eine Vereinheitlichung der Quellcodes durchgeführt werden. Das hatte zur Folge, daß die Version 15 sehr bald auch für die UNIX-Plattformen verfügbar wurde, und daß die Weiterentwicklungen praktisch synchron durchgeführt werden können. Ein relativ kleines C-Modul enthält die betriebssystemspezifischen Routinen. Nur dieses muß angepaßt werden, wenn eine weitere UNIX-Plattform bedient werden soll. Der vorher beträchtlich hohe Portierungsaufwand ist damit auf ein Minimum reduziert worden.

2 C-Programmierung / Klassenbibliothek C^{++}

Auf der C-Ebene gibt es sogenannte "user exits", die es einem C-Programmierer ermöglichen, neue Funktionen an den ***allegro***-Kern anzuhängen. Ohne diesen Kern überhaupt kennen und verstehen zu müssen, kann der Programmierer Datensätze abrufen, anzeigen lassen, exportieren, verändern und zurückspeichern. Der C-Programmierer benötigt nur das C-Modul mit den "user exits" und dazu die fertig übersetzten anderen Teile (Objektmodule), um arbeiten zu können. Auf diesem Wege sind bereits die Programme APAC (OPAC-Modul), REF (das "Referentenprogramm" für die Sacherschließung incl. SWD-Anbindung), ORDER (Monographien-erwerbung), ***aLF*** (Ausleihprogramm) und ***aLFA*** (Einfach-Ausleihe) erstellt worden.
Im Jahre 1995 entstand eine "Klassenbibliothek" in der Sprache C^{++}. Hiermit ist es für Programmierer noch sehr viel einfacher, neue Programme zu schreiben, die auf ***allegro***-Datenbanken zugreifen können. Auf dieser Basis entstand der ***avanti***-Server, der zum Kern der WWW-Kataloge wurde, und dieser ist wiederum die Vorstufe zur Realisierung einer Z39.50-Schnittstelle, an der von 1996 bis 1999 mit Unterstützung der DFG gearbeitet wurde. Auch diese Software ist nun im Einsatz. Die Klassenbibliothek ist aber auch der Kern der Windows-Programme ***a99*** und ***alcarta***, die ab 1999 allmählich die DOS-Programme ablösen.

3 Konfiguration, Parametrierung

Freie Definition des Kategorienschemas (→ Anh.A) in Verbindung mit der Exportsprache für die Gestaltung von Bildschirmanzeigen, Druckausgaben, Indexgestaltung und Dateiproduktion (→ Kap.10) sorgen für eine hohe Flexibilität in der Gestaltung von Datenbanken für sehr unterschiedliche Anwendungen, nicht nur für Bibliotheksdaten. Die Importsprache (→ Kap.11) ermöglicht in weiten Grenzen die Umwandlung von Alt- und Fremddaten. Da auch die Parameterdateien plattformunabhängig sind, kann man mit einer Anwendung leicht zwischen den Betriebssystemen "migrieren".

4 Einbindung als Prozeß

Alle Teilprogramme von ***allegro*** können von der Betriebssystemebene, also auch aus Batchfiles (MS-DOS) oder Shell-Programmen (UNIX) aufgerufen und mit den jeweils erforderlichen Einstellungen (Optionen) gestartet werden (→ Kap.12). Die Programme 4 (Selektion und Export) und 5 (Import) ergeben zusammen mit den Datenmanipulationssprachen für Export und Import eine Art "Datenkonverter", den man in jeder Umgebung einsetzen kann, wo routinemäßig Datenkonvertierungen durchgeführt werden müssen. Die anderen Teile von ***allegro*** würden dann überhaupt nicht benutzt. Von "außen" wäre nicht erkennbar, daß gewisse Arbeiten von einem ***allegro***-Programm ausgeführt werden.
Die Windows-Programme ***a99*** und ***alcarta*** bieten mit der FLEX-Technik (→ Kap.2) weitreichende Möglichkeiten der Einbindung als Server in andere Prozesse.

Diagramm 1

Wie anders wirkt dies Zeichen auf mich ein!
Welch Schauspiel! Aber ach! Ein Schauspiel nur!
Wo faß ich dich, unendliche Natur?
(J.W.v. Goethe, Faust I)

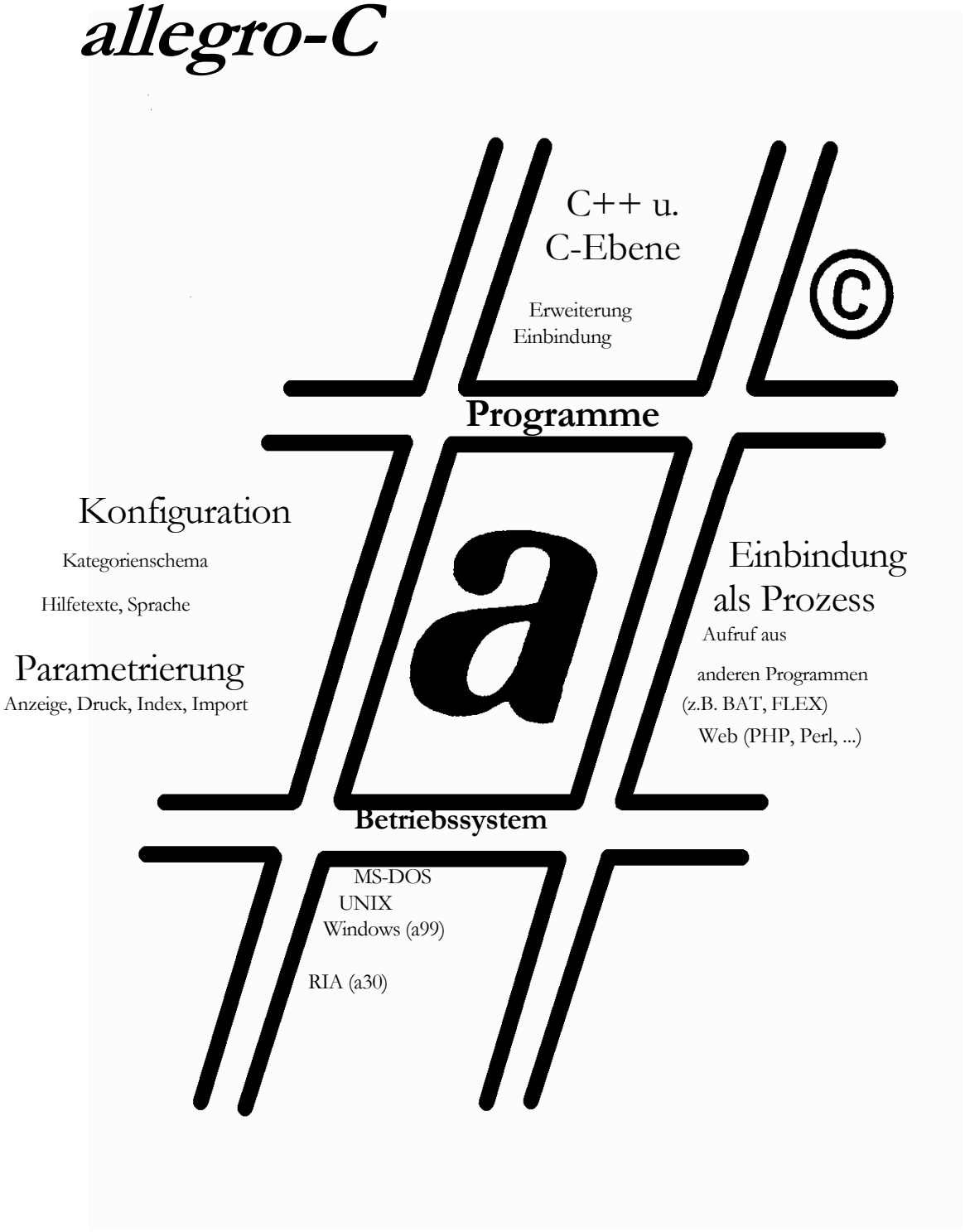


Diagramm 2

Dieses Diagramm zeigt Ihnen die wichtigsten funktionalen Zusammenhänge zwischen den Programmen und Dateitypen, die zu ***allegro*** gehören (→ 0.3, 0.4).

Im Zentrum steht Ihre Datenbank **dbn** mit den bekannten drei Bestandteilen (Dateitypen .cLD, .cDX und .TBL).

Alle Programme, die auf die Datenbank zugreifen, benötigen die Konfigurationsdatei c.CFG, in der das Kategorienschema steht.

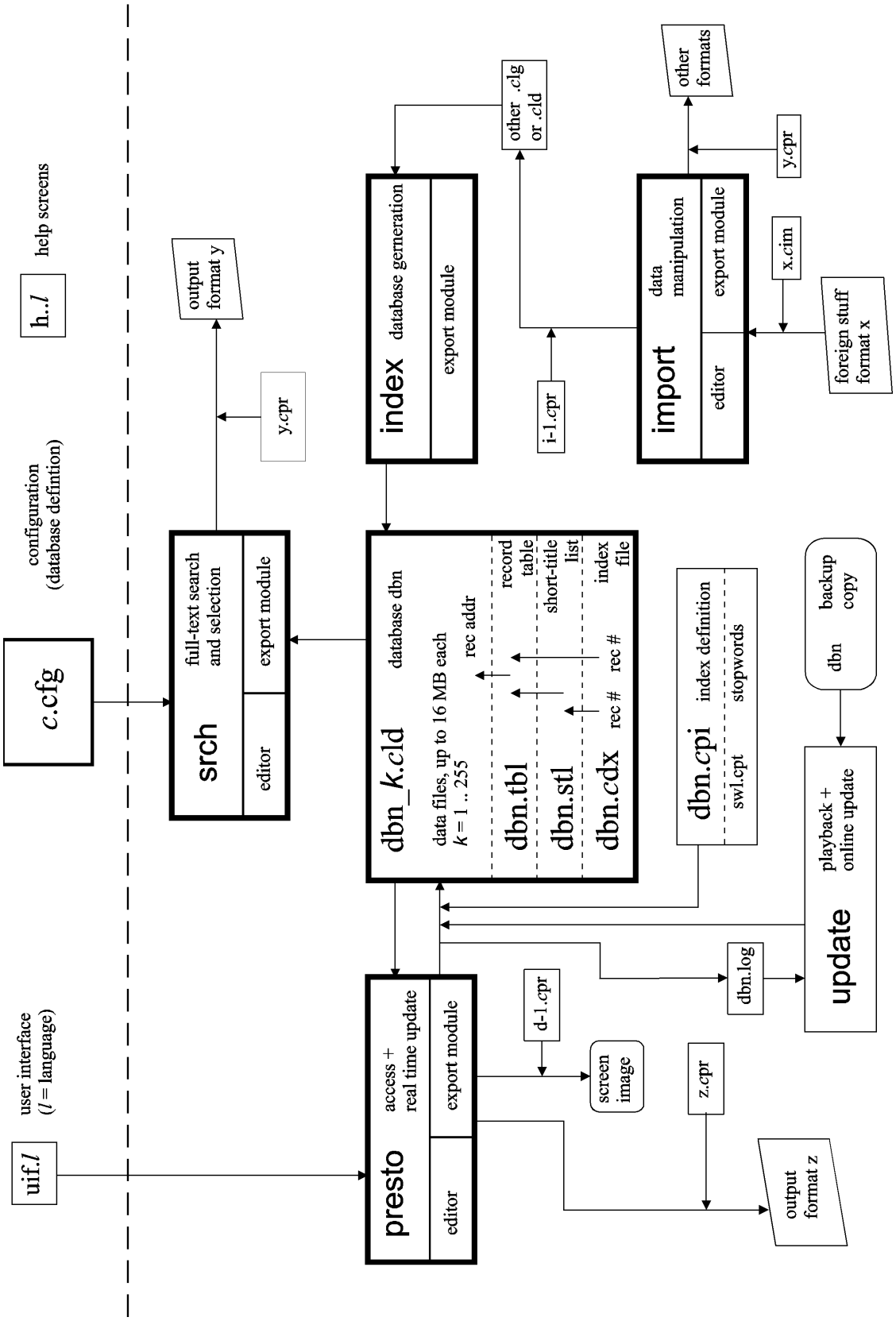
Menütexte (UIF...) und Hilfetexte (H...) werden ebenfalls von allen Programmen gebraucht. Diese Dateien können in verschiedenen Sprachen vorliegen.

Diejenigen Programme, die verändernd an der Datenbank tätig werden können, müssen auch die Index-Parameterdatei (Typ .cPI) lesen, in der die Art und Struktur der Indexeintragungen genau beschrieben sind. Veränderungen an der Datenbank (Eintragung neuer Sätze, Korrektur oder Löschung vorhandener Sätze) wirken sich oft auch auf den Index aus. Die Programme "wissen" aus der Datei dbn.cPI, was da zu tun ist und führen es vollautomatisch und sofort durch.

Genauso gilt: wenn ein Programm Daten auf Bildschirm oder Drucker oder in eine Datei ausgibt, entnimmt es die Anweisung, in welcher Form das zu geschehen hat, aus einer Export-Parameterdatei (Typ .cPR).

Im Windows-System tritt nur das Programm ***a99*** an die Stelle von PRESTO. Die Datenbank ist ansonsten dieselbe! Es gibt keine DOS-und Windows-*Datenbanken* sondern nur DOS- und Windows-*Programme*, die beide auf dieselben Daten zugreifen, was sie sogar gleichzeitig tun können.

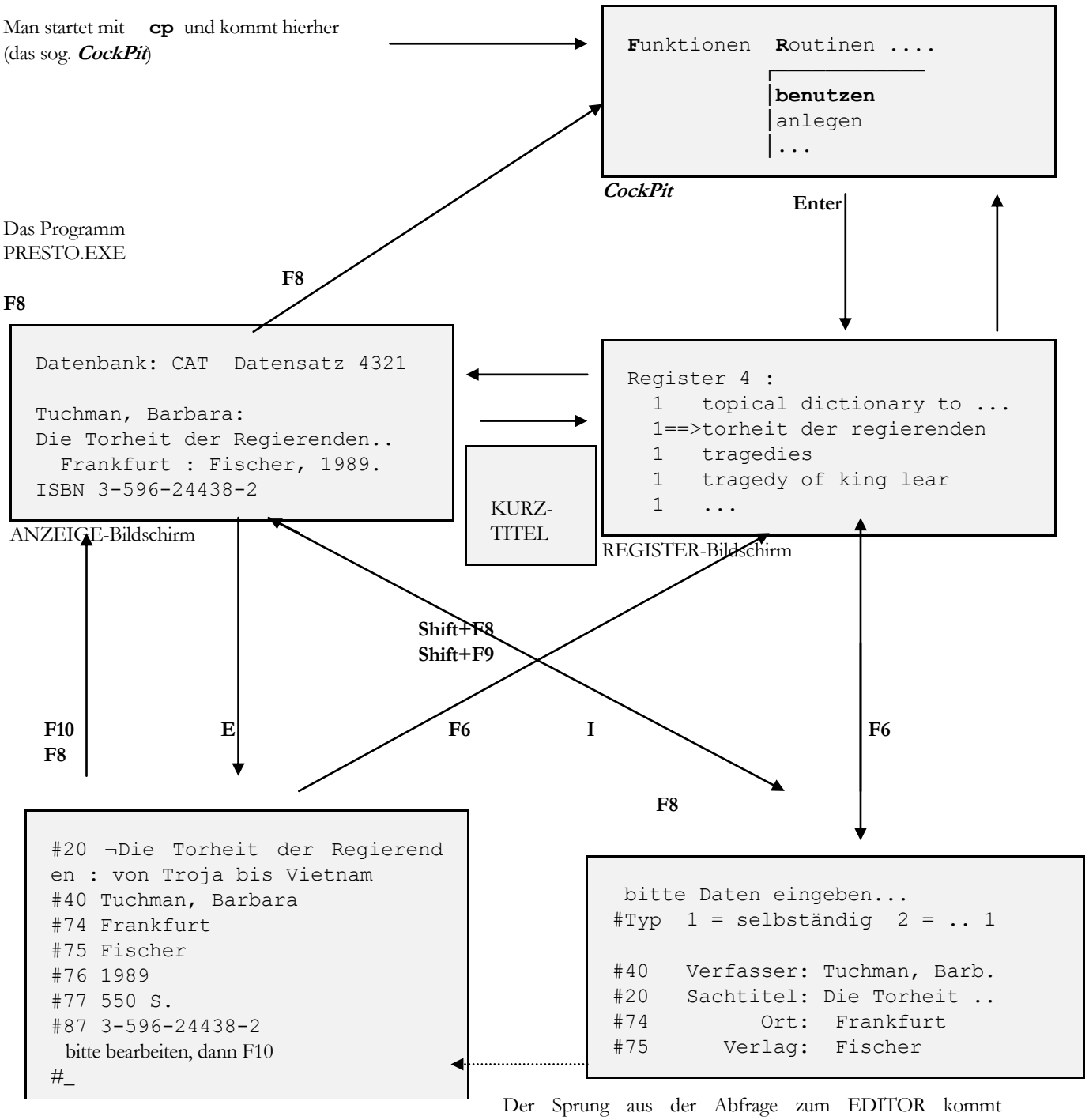
Diagramm 2



Statt "presto" kann man sich hier auch a99 und alcarta vorstellen. Diese umfassen mehr Funktionen, z.B. auch die von "upd", aber die Zusammenhänge und Datenstrukturen sind dieselben.

Diagramm 3

Zur Veranschaulichung der Zusammenhänge werden hier die 4 "Gesichter", also die 4 verschiedenen Bildschirmtypen, des DOS-Hauptprogramms PRESTO noch einmal dargestellt, und zwar so, daß für eine bestimmte Titelaufnahme jeweils nur der wichtigste Bildschirmausschnitt gezeigt wird. Pfeile zwischen den 4 Bildschirmen deuten an, welche Zusammenhänge bestehen und wie man vom einen Schirm zum anderen wechselt. Dieselben Elemente finden sich auch im Windows-System (es ist ja dieselbe Datenbank!), dort jedoch kann man alles zugleich sehen.



Der Sprung aus der Abfrage zum EDITOR kommt automatisch, wenn die ABFRAGE beendet ist. Oder "per Hand": x eingeben

C.2 Fehlermeldungen

Viele Fehlermeldungen erklären sich selbst und sind unmittelbar verständlich, z.B. wenn die Ursache eine falsche Eingabe ist. Die nicht so leicht verständlichen sind hier aufgelistet. Die meisten Meldungstexte stehen in UIF-Dateien und können deshalb auch anders lauten, besonders wenn in einer anderen Sprache gearbeitet wird. Das Indexsystem bringt gelegentlich Meldungen hervor, die nur aus einem knappen englischen Hinweis und einer Fehlernummer bestehen. Im Teil 2 finden Sie diejenigen Fehlernummern, die nach bisheriger Erfahrung auftreten können. Wenn als Abhilfe "Neu-Indexieren" angegeben ist, wählen Sie die Funktion **`INDEX -fi0`** (→ Kap.7 oder per ***CockPit*** **`µr o i`**).
Eine Angabe wie diese bedeutet: ***CockPit***-Menü "**R**outinen", Punkt "**O**rganisieren" und dann "**I**ndex erneuern".
In ***a99*** gibt es entsprechende Funktionen auf dem Menü "Reorganisieren" (**`h org`** eingeben).

Teil 1: Meldungen der Programme (einige treten nur im DOS-System auf)

`[1;32;47m` ... oder ähnliche störende Zeichen in der Bildschirmanzeige (DOS)
Dieses Phänomen tritt oft beim ersten Start nach der Installation auf. Ursache: der Bildschirmtreiber ANSI.SYS ist nicht vorhanden oder fehlerhaft. *Abhilfe:* → Kap.0.10 "Installation".

Arbeitsspeicher reicht nicht. Zuviele residente Programme? (DOS)
Abhilfe: die Speichereinstellungen in der .CFG (→ Anh. A.1.3) auf ein vertretbares Minimum herabsetzen, und/oder prüfen, ob durch Maßnahmen in der CONFIG.SYS und/oder AUTOEXEC.BAT der verfügbare DOS-Speicher vergrößert werden kann.

bitte warten (.TBL-Datei gesperrt)
es wird im Mehrplatzbetrieb gerade ein anderer Datensatz gespeichert. Wenn dies länger als 1 Sekunde dauert, läuft entweder gerade ein UPD, oder die .TBL-Datei ist blockiert. Im Einzelplatzbetrieb kann das nur höchst selten, wenn überhaupt, vorkommen. Freigabe dann mit ***CockPit*** **`µr o e`**. (**a99**: h org "Datenbank freigeben")

CFG enthält keine Artikelliste
in der .CFG fehlt die Liste der Artikel, die ab Version 12.2 vorhanden sein muß. Aus der mitgelieferten A.CFG kann man sie in andere .CFG-Dateien übernehmen. Prog. arbeitet trotzdem - ohne Artikelprüfung.

CREIDX error ...
Dahinter kann eine der Nummern auftreten, die weiter unten aufgelistet sind. Ursache: die Indexdatei konnte nicht angelegt werden. Evtl. wurde beim Start von PRESTO oder UPD eine falsche Option angegeben, oder es ist zuwenig Platz auf der Platte.

ctree fatal error nnn siehe Teil 2 unter Fehler nnn

D-1.cPR fehlt, Anzeige deshalb nur im Internformat (DOS)
PRESTO braucht für die Titelanzeige eine Datei namens D-1.APR (oder D-1.MPR etc.). Diese wird zuerst auf dem Datenverzeichnis, dann auf dem Programmverzeichnis gesucht. Wird sie nicht gefunden oder ist sie fehlerhaft, erhält man nur eine Anzeige im Kategorieformat.

Datei nicht zugänglich
Eine Datendatei (Typ .ALD) wurde nicht gefunden. Evtl. wurde sie gelöscht oder umbenannt (das darf man nicht machen) oder die .TBL enthält falsche Angaben. Wenn die .ALD-Dateien alle vorhanden sind: .TBL erneuern (***CockPit*** **`µr o t`**). Sonst Datenbank wiederherstellen (**`µr w`**).

Dateien vom Typ .A8D zuerst prüfen (DOS) (auf dem Menü "Routinen / Organisieren")
CockPit hat festgestellt, daß es auf Ihrem Datenverzeichnis noch Dateien mit dem Typ .c8D gibt. Diese müssen Sie zuerst löschen, bevor der Index erneuert werden kann. Die Ursache kann aber auch ein Versagen oder Absturz sein, der sich vorher ereignet hat. Dann müssen diese Dateien in .cLD zurückbenannt werden.
Also: nachsehen, ob es .cLD-Dateien gibt;
wenn ja, aber 0 Byte: Dateien vom Typ .cLD löschen, Dateien vom Typ .c8D auf .cLD umbenennen;
wenn ja: die Dateien vom Typ .c8D löschen; (***CockPit***: **`µr o a`**)
wenn es keine .cLD-Dateien gibt : .c8D wieder auf .cLD umbenennen;

Dateifehler
Eine Datei konnte nicht geöffnet werden. Aus dem Zusammenhang sollte meistens zu entnehmen sein, um welche Datei es sich handelt. Mögliche Ursache: keine Schreibberechtigung oder Platte voll.

default.opt not found → Vorgabendatei nicht gefunden

Eingabe/Änderung unzulässig
Es wurde versucht, eine Kategorie einzugeben oder zu ändern, die unter Änderungsschutz steht. D.h. die "Eigenschaftszahl" dieser Kategorie (Anhang A.1.2) ist vermutlich auf 4 oder 5 eingestellt.

Eingabefehler
Im Bearbeitungsmodus wurde eine Zeile eingegeben, die weder mit einer erlaubten Kategorienummer noch mit einem Befehlszeichen (Kap. 3) anfängt.

Fehler ... bei Indexdatei ...
Siehe Teil 2 unter der Fehlernummer

fehlerhafte Parameter nicht benutzbar
zuvor wird eine Parameterzeile angezeigt (Index-, Anzeige-, Druck- oder Exportparameter), die einen schwerwiegenden Fehler enthält, z.B. eine ungültige Kategorie. Dieser muß behoben werden, damit das Programm richtig läuft. Meistens genügt es, die ungültige Kategorienummer in die .CFG einzutragen (→ Anh. A.1.2), z.B. eine Zeile mit **`#62`** einzubauen, wenn dies die Nummer wäre. Wenn die Fehlermeldung auf eine nicht vorhandene Tabelle hindeutet, lautet die Meldung z.B. "Fehlerhafte Zeile: tprinter". Dann fehlt die Datei PRINTER.cPT.

kein Eintrag unter dieser Satznummer oder **fehlerhafte Satznummer** siehe → **wrong recn**

kein Index, Zugriff nur über Satznummer
es ist zwar eine .TBL-Datei vorhanden, aber keine .ADX-Datei. Siehe unten Fehler 12. Im Gegensatz zu dieser Meldung kann man i.d.R. NICHT zugreifen, sondern das DOS-Programm stürzt sogar ab.

Kein Platz mehr für Phrasen
Mögliche Ursache: zuviele Zwischenteile in den Exportparametern und/oder zu große Phrasenliste.
Abhilfe: Wert `mP` in der .CFG heraufsetzen, z.B. `mP5000`, oder Datei PHRASE.APH verkleinern.

Kein Speicherplatz mehr im Umgebungsbereich [kommt von DOS, nicht von *allegro*]
Ursache: in einer Batchdatei wurden zu viele SET-Befehle gegeben.
Abhilfe: DOS: in AUTOEXEC.BAT einen höheren Wert im `shell`-Befehl einsetzen, z.B. `/E:2048`.
Windows: in SYSTEM.INI unter [NonWindowsAppl] die Zeile `CommandEnvSize=2048` einfügen.

keine Datei ausgewählt
Es ist keine geeignete Datei auf dem gewünschten Verzeichnis gefunden worden (Typ .cLG oder .cLD), oder man hat auf der vorher angebotenen Auswahlliste keine Datei mit '+' markiert. Ursache kann auch sein, daß man die Option `-k` beim Programmaufruf vergessen oder eine falsche angegeben hatte.

Keine Datenbank auf Verzeichnis ...
Auf dem gewählten Verzeichnis müssen mindestens eine .TBL und eine .cDX existieren! Vielleicht wurde das falsche Verzeichnis angegeben oder ein fehlerhafter Name. Wenn das Verzeichnis z.B. `c:\katalog` ist, wäre es falsch, z.B. `\katalog` oder `c:/katalog` zu geben. Vielleicht hat man auch mit der Option `-k` eine falsche CFG angegeben.

Keine Formulardatei gefunden [Nur in a99]
Es gibt keine `.frm`-Datei. Das macht nichts, nur funktioniert der Button [Form] dann nicht..

Keine Index-Parameterdatei ...
Im Programmaufruf wurde wahrscheinlich der Datenbankname falsch geschrieben, oder es wurde eine falsche Konfiguration angegeben.

Keine Löschung, da verknüpft
Es wurde versucht, einen Satz zu löschen, von dem andere Sätze abhängen, z.B. den Hauptsatz eines verknüpft gespeicherten mehrteiligen Werkes. Voraussetzung, damit das funktioniert, ist das Vorhandensein eines Löschkontrollschlüssels (→ 10.2.7). Die Löschung kann erst erfolgen, wenn alle abhängigen Sätze beseitigt sind.

Konfiguration ... fehlt
Die angegebene Konfigurationsdatei wurde nicht gefunden. Entweder wurde die Option **`-k`** vergessen oder falsch gesetzt, oder die verlangte Datei ist wirklich nicht vorhanden.

Ladefehler 58 … oder …LOAD ERROR (INDEX oder QRIX)
doppelter Schlüssel oder falsche Reihenfolge der Schlüssel beim Einordnen. Kommt vor, wenn ein Schlüssel mit einem Zeichen oberhalb '|' beginnt, z.B. '}' oder '~'.
Wenn das Programm danach stehen bleibt, ist die Platte voll. *Abhilfe:* vor dem Indexlauf Platz schaffen.

max exceeded Ergebnismenge kann nicht gebildet werden, sie würde zu groß. Bis V14c auch dann, wenn schon die Summe der Zwischenergebnisse zu groß ist. *Abhilfe:* Wert **`mr`** in der CFG heraufsetzen, falls noch möglich.

Mehrfachcode ... unzulässig, erlaubt: ...
Es wurde eine Kategorie mit falschem Wiederholungszeichen eingegeben. Die laut CFG erlaubten Codes werden gezeigt. Die Angabe ▼M bei der betr. Kategoriezeile in der CFG ändern, wenn der eingegebene Code zulässig sein soll. (→ Anhang A.1.2)

Momentan kein Zugang wegen Wartung
Auf dem Datenverzeichnis liegt eine Datei namens *.sgf, z.B. cat.sgf. Darin steht nur eine Zeile, und die beginnt mit 1. Diese Datei wird entweder automatisch gesetzt, wenn man eine Neuindexierung vornimmt, oder manuell vom Systemverwalter (Menü ORG, Punkt "Totalsperre").
Abhilfe: Auf dem Menü ORG den Punkt "Totalsperre" anklicken und dann die Frage mit ja beantworten. Oder einfach die SGF-Datei löschen.

Nicht genug Arbeitsspeicher, Fehler nnn bei Indexdatei
Das Problem ist meistens ein zu geringer Arbeitsspeicher. Prüfen Sie aber auch die angegebene Nummer nnn im zweiten Abschnitt dieser Fehlerliste, bevor Sie die Werte der m-Befehle in der CFG verändert (→ Anh. A.1.3). Der Fehler kann aber auch eine fehlende Indexdatei sein. Schauen Sie nach, ob die Datei vom Typ *.?DX im Datenverzeichnis vorhanden ist. Wenn nicht, dann Index erneuern.

nichts gefunden, Satznummer nicht besetzt

Unbedenklich. Eine nicht belegte Satznummer wurde verlangt. Solche Nummern gibt es, wenn man entlüftet hat oder eine "nummerntreue Reorganisation" durchgeführt hat (→ Kap.7). Es entstehen dann Eintragungen unter dem Sonderschlüssel **`/[0]`** im Index 1. Sie verschwinden automatisch, wenn neue Sätze erfaßt werden.

no config file : .CFG missing, keine Konfigurationsdatei

Die verlangte .CFG-Datei ist nicht zu finden. Achten Sie vor dem Start über ***CockPit*** darauf, ob in der Statuszeile bei "Konfiguration" eine Angabe steht, zu der es auch eine .CFG-Datei gibt. Wenn Sie ein Programm direkt starten, also z.B. SRCH, und Sie geben Option `-k xyz`, dann muß XYZ.CFG existieren. Gesucht wird die Datei auf dem Datenverz., dann auf dem Startverzeichnis (von dem aus man startet), und wenn sie dort auch nicht ist, auf dem Programmverzeichnis.

no write access to this database

Es besteht keine Schreibberechtigung für die gewählte Datenbank. Versucht wird, wenigstens mit Leseberechtigung zuzugreifen, wenn das auch nicht geht, kommt "Fehler ... bei Indexdatei", s. dort.

Parameter xyz nicht gefunden

Die angegebene Parameterdatei xyz wurde nicht gefunden. Man prüfe, was damit los ist, oder ob man beim Programmaufruf einen Fehler bei einer der Optionen `-p, -q, -e` gemacht hat.

Satz *nnn* : falsche Nnummer

Programm INDEX hat in einer .ALD-Datei einen Satz mit offensichtlich falscher (zu hoher) Nummer gefunden. Die ersten 2 Kategorien werden mit angezeigt, damit man ihn identifizieren kann. Er wird nicht mit indexiert. Das Problem kann von einer älteren Version verursacht worden sein: der Anfang des betr. Satzes wurde überschrieben und damit auch die Satznummer, an deren Stelle nun etwas falsches steht. Nach dem Indexieren prüfen, ob der angezeigte Satz vorhanden ist; wenn nicht: neu erfassen.

Satz gesperrt

im Mehrplatzbetrieb: es arbeitet gerade jemand anders an dem Datensatz. Versuchen Sie es später noch einmal. Wenn er wieder frei ist, bekommen Sie natürlich die bearbeitete Fassung. Wenn der Satz fälschlich gesperrt ist, weil jemand während der Bearbeitung sein Gerät ausgeschaltet hat oder der Strom ausfiel: Freigabe mit Strg+Z.

Wenn UPD diese Meldung bringt, sehen Sie nur die interne Satznummer. Diese müssen Sie dann in PRESTO mit Expreßzugriff aufrufen (→ 1.5.2) und dann mit Strg+Z freigeben. (a99: Menü "Bearbeiten: Satz freigeben")

Satztabelle gesperrt

siehe **bitte warten (.TBL-Datei gesperrt)**

`t` muss der erste Befehl sein

In der .CFG muß der Befehl **`t`**, der die Breite der Kategorienummern angibt, ganz oben stehen.
(Das Programm hat festgestellt, daß vor dem Befehl noch andere stehen, was nicht sein soll.)
Setzen Sie in Ihrer .CFG den Befehl, z.B. **`t3`**, an die erste Stelle, oberhalb aller anderen Befehle.

text too long, check your 're' command

IMPORT gibt diese Meldung aus, wenn kein Satzende in den Fremddaten gefunden wird. Parameter `re` ist mit Sicherheit falsch; dieser Parameter muß die Zeichenkombination angeben, die das Ende eines Fremdsatzes markiert (dazu → Kap. 11.2, Typ C).

user interface missing UIF...

Es fehlen UIF-Dateien (→ 0.3). Die Programme erwarten ihre Menütext-Dateien auf dem Datenverzeichnis oder auf C:\ALLEGRO. Kopieren Sie die Dateien also entweder auf's Datenverzeichnis oder geben Sie vor dem Start, wenn Sie z.B. auf Laufwerk D: arbeiten, den DOS-Befehl `set -P=D:\ALLEGRO` . Ursache kann auch eine falsche Sprachangabe sein (Befehl **`l`** in der Vorgabendatei cp.opt oder Option **`-l`** beim Programmaufruf).

Vorgabendatei nicht gefunden (DEFAULT.OPT not found)

CockPit wurde gestartet, fand aber keine Vorgabendatei. Prüfen Sie, wie der Aufruf des Programms ACP aussieht. Wenn darin keine Option -o vorkommt, muß es im Startverzeichnis eine CP.OPT geben.

wrong recn (ab 11.2: "kein Eintrag unter dieser Satznummer")

Das Programm findet im gewünschten Datensatz (oder an der Stelle, wo er laut Index stehen sollte) keine oder eine falsche Satznummer ("wrong record number"). Es stimmt etwas nicht mit Ihren Dateien. Die Satztabelle (Typ .TBL) und die Datenbankdatei (Typ .ALD) passen nicht zusammen. Wenn bei einer Datensicherung (RESTORE) z.B. nur eine von beiden zurückkopiert wurde, kann dieser Fehler auftreten. Untersuchen Sie dies, sonst: .TBL-Datei erneuern (**`index -ft`** oder ***CockPit*** **`µr o t`**). Evtl. Hilfsprogramm SNIFFER einsetzen.

Zugriff auf Index ... nicht möglich! (Wahrscheinlich unproblematisch)

Kommt nach dem Start von SRCH, wenn keine Indexdatei gefunden wurde. Problematisch ist das NUR, wenn bei dem durchzuführenden Export auf den Index zugegriffen werden muß, und zwar wegen V14-Ersetzungen oder wegen Nachladungen, sonst kann man die Meldung ignorieren. *Wenn* der Index also für den Export gebraucht wird, muß man die Option `-b` beim SRCH-Aufruf richtig angeben. Wenn die Datenbank CAT heißt und auf C:\KATALOG liegt, muß man schreiben `srch ... -bc:\katalog\cat` .

Teil 2: Fehlermeldungen des Indexsystems : (statt "Fehler" kann auch "error" oder "fatal error" kommen)

Fehler 10

Nicht genug Platz im Arbeitsspeicher. Prüfen Sie, ob "residente" Programme geladen sind oder größere Gerätetreiber, die den verfügbaren Arbeitsspeicher einschränken. Wenn weniger als 540K im DOS-Bereich frei sind, ist so etwas zu erwarten. *Abhilfe:* Beseitigen Sie nötigenfalls in der CONFIG.SYS oder AUTOEXEC.BAT die Installation dieser störenden Programme. Zur Not verkleinern Sie noch die Werte `mr, mB, mK, mk, mX` (in dieser Reihenfolge) in Ihrer CFG (→ Anh. A.1.3), was jedoch wieder andere Fehler zur Folge haben kann, z.B. 222. Die Windows-Programme sind frei von solchen Problemen.

Fehler 12

Indexdatei nicht auffindbar oder nicht benutzbar. Prüfen Sie, ob das richtige Datenverzeichnis angesprochen wurde und dort die nötigen Dateien, insbesondere die .ADX-Datei und die .CFG-Datei, vorhanden sind, und ob beim Start des Programms die Option **`-k`** für die Konfiguration korrekt gesetzt wurde..

Fehler 13 und 14

Indexdatei ist nicht vorhanden oder korrumpiert. Dies passiert z.B. bei einem Stromausfall, wenn dieser zufällig genau zu dem Zeitpunkt einer Indexspeicherung eintritt. Die sicherste *Abhilfe* ist Neu-Indexieren. PC-Kenner können in der .ADX-Datei mit einem geeigneten Programm (z.B. DEBUG) das Byte 16 von FFh auf 00h ändern. Vielleicht funktioniert dann der Index wieder, aber für seine Zuverlässigkeit gibt es dann keine Garantie.

Fehler 16

Indexdatei kann nicht angelegt werden. Beim Anlegen einer neuen Datenbank wird festgestellt, daß nicht genügend Platz für die Indexdatei vorhanden ist. *Abhilfe:* Platz schaffen auf der Platte.

Fehler 18 (ebenfalls beim Neu-Anlegen einer Datenbank)

Indexdatei kann nicht angelegt werden, weil sie schon existiert. Sie muß von einem Fehlversuch oder einer unvollständigen Datenbanklöschung übrig geblieben sein. Evtl. steht eine auf dem Programmverzeichnis, wo sie nicht hingehört. Löschen und neu starten.

Fehler 22

Falsches Index-Präfix, erlaubt sind nur |1, … |9, |: und |;
Ursache: Fehler in den Indexparametern; evtl. dadurch, daß das Zeichen | durch Umcodierung verschwindet

Fehler 34

Inkonsistenz in der Indexdatei. Kann nur bei Zusammenbrüchen auf Mehrplatzanwendungen eintreten.
Abhilfe: nur durch Neu-Indexierung (**`ur o i`**).

Fehler 35 oder 37

Ursache: kein Platz mehr auf der Platte, Indexdatei kann nicht geschrieben werden oder ist beschädigt.
Abhilfe: Index erneuern (**`ur o i`**), aber vorher, wenn nötig, Platz schaffen.

Fehler 40

Einstellung der maximalen Schlüssellänge zu klein. Der Wert des Parameters `il` in der Index-Parameterdatei ist wahrscheinlich vergrößert worden. *Abhilfe:* Wert von `il` korrigieren oder neu indexieren.

Fehler 46

Zu wenig Systembereich für Dateien. *Abhilfe:* In der CONFIG.SYS muß der Files-Befehl auf einen höheren Wert gesetzt werden. Setzen Sie FILES=20 ein, das reicht (andere Software braucht meistens mehr).
Oder: selbe Ursache und Abhilfe wie 222.

Fehler 49

Platz auf der Platte erschöpft. Während eines Index-Schreibzugriffs war auf einmal kein Platz mehr auf der Platte. Der zuletzt eingegebene Datensatz ist dann zumindest nicht vollständig indexiert.
Abhilfe: Platz schaffen und neu indexieren (**`ur o i`**).

Fehler 54

Es wurde versucht, in eine Datei zu schreiben, für die keine Schreibberechtigung besteht, oder die schreibgeschützt ist oder die gar nicht beschrieben werden kann (z.B. auf CD-ROM!). *Abhilfe:* Schreibrechte kontrollieren, evtl. die Option **`-a`** ändern. Wenn der Index auf einer CD liegt, muß man beim Start von PRESTO die Angabe `-a0` machen. Wenn die CD-Datenbank als zweite parallel benutzt wird, z.B. `presto -a30` angeben.

Fehler 208

Fehler bei einem Indexzugriff. Kann bei einem QRIX-Lauf auftreten. Wenn die Indexdatei anschließend nicht benutzbar ist, muß man neu indexieren, oder den QRIX-Lauf nochmals starten (→ Kap. 7.5 "Wiederanlauf").

Fehler 214

Fehler beim Indexzugriff. Arbeitsspeicher-Knappheit kann auch hier die Ursache sein; → Fehler 10.

Fehler 222 : ctree fatal error

Parameterspeicher nicht ausreichend. *Abhilfe:* In der .CFG den Wert mX hochsetzen, z.B. `mX45000`.

Fehler 231 / 233 : ctree fatal error

Inkonsistenzen im Index. Diese Probleme können nach einem fehlerhaften QRIX-Lauf auftreten (→ Kap.7) oder durch Eingriffe mit einem Hex-Editor in die Indexdatei. *Abhilfe:* zuverlässig nur durch Neu-Indexieren.

C.3 Literaturangaben

Aktuelleres Material online → http://www.allegro-c.de

Hier wird keine auch nur annähernd vollständige Bibliographie geboten, sondern nur eine Liste von Dokumenten, die unmittelbar für die Arbeit mit ***allegro*** interessant sein könnten. Neuere Dokumente sind fast nur noch im Web zu finden!

Ahlers, Torsten: Das Projekt *allegro* : Entwicklung, Anwendungen, Einsatz in Netzen. - Braunschweig: UB, 1994. - 102 S. - ISBN 3-927115-24-X
[mit Abriß der Entwicklungsgeschichte und umfangreicher Bibliographie]

aLF : ***allegro*-Leih-Funktionen**. Technische Referenz. / Universitätsbibliothek Braunschweig. - Braunschweig: UB, 1995. 86 S.
[beschreibt Konzept und Benutzung des Ausleihprogramms; wird als Textdatei mitgeliefert]

Das *allegro*-Erwerbungssystem : Datenstrukturen, Parametrierung, Funktionen, Version 1995.2 (2 Teile) / UB Braunschweig. - Braunschweig: UB, 1995. 70+48 S.
[beschreibt Konzept und Benutzung des Erwerbungsprogramms *ORDER*]

***allegro*-Format 2004** : Konsolidiertes Format für Bibliotheksdaten. - http://www.allegro-c.de/doku/form2004/

***allegro* im Netz** : Installationshinweise. - Braunschweig: UB, 1994. - 28 S.

***allegro*-Sacherschließung** : Konzept, Anleitung, Normdatei-System, Hilfstabellen / UB Braunschweig. - Überarb. Ausg. - Braunschweig: UB, 1993. 43 S.
[beschreibt Konzept und Benutzung des Referentenprogramms]

Allers, Heinrich: *allegro-C* Lehrbuch : Version 14. - Braunschweig: UB, 1995, - ca. 260 S. + Diskette mit Beisp. u. Demodaten. [vergriffen, teilw. Neuaufl. s. Tews.]
[Überarb. Ausg. des zuerst 1993 zu Version 12.2 herausgegebenen Lehrwerks; Beisp. u. Demodaten auch auf der CD-ROM]

Brandes, Dietmar u. Bernhard Eversberg: Arbeitsplan für die *allegro*-Entwicklung 1991/92. - In: Bibliotheksdienst 25(1991), Heft 6, S.861-871.
[Projektplan für die stufenweise Entwicklung von Ausleih- und Erwerbungsmodulen]

Dudeck, Jochen u. Gaby Maidorn: allegro - allegro vivace : ein Arbeitsbericht aus Niedersachsen. - In: Buch und Bibliothek 44(1992)9, S.809-815.

Eversberg, Bernhard: "allegro-C" : a new database concept for micros - MS-DOS and UNIX. In: 12th Essen Symposium 1989. ISBN 3-922602-13-4. - S.127-144
[Aus den ganz frühen Jahren. Zum Datenbankkonzept und zu den Sprachen für Import und Export.]

*** 14 Jahre ***allegro*** : Fragen, Anforderungen, Nutzerkreise. - In: "mb" (1995)Heft 97/98, S.69-82.
[auch auf Englisch erschienen in: Program 29(1995)2, S.147-158.]

*** ***allegro-C***: Eine Software im Epochenwandel. - Braunschweig : Univ.-Bibl., 2013. – 32 S.
http://www.digibib.tu-bs.de/?docid=00050229
[Zusammenfassung der Entwicklung bis 2013]

*** Was sind und was sollen bibliothekarische Datenformate? - Überarb. u. erw. Neuausg. - Braunschweig: UB, 1994. - 182 S. + 1 Diskette. - (Veröffentlichungen der Universitätsbibliothek Braunschweig ; 9) - ISBN 3-927115-21-5
[Enthält Übersichten und echtes Beispielmaterial zu einigen wichtigen Datenformaten, darunter MAB1 und verschiedene MARC-Versionen, und versucht einen theoretischen Überbau. Entstand während der Arbeiten an der allegro-Importsprache. Diskette enthält eine *allegro*-Datenbank der Formate, 1996 aktualisiert, 1999 als Online-Dokument: http://www.allegro-c.de/formate/ .]

Gradmann, Stefan: Katalogisierung mit dem PC : Datenbankgestützte Systeme für die Verarbeitung bibliothekarischer Daten. Allegro-C, BIS-LOK, IBAS IV/BIBLIO. - Wolfenbüttel: Tandem, 1992. - 149 S. - (TANDEM Informationen ; 3) ISBN 3-927651-04-4.
[Neufassung eines schon 1989 erschienen Werkes, in dem neben allegro-C die Systeme LIDOS und TINman untersucht wurden. Bewertung der Programme nach einem an der Praxis orientierten Kriterienkatalog, mit Testergebnissen eigener Versuche.]

Homann, Benno: Kriterien und Konzepte einer PC-gestützten Medienerschließung an Medien/Sprachenzentren (MSZ) : Die Entwicklung von KATGALAV am IKM/SL der Universität Mannheim.
In: Zeitschrift für Bibliothekswesen und Bibliographie 36(1989)3, S.191-215.
[Mit ausführlicher Darstellung eines eigenen, sehr detaillierten Kategoriesystems für a.v. Medien.]

Münnich, Monika: PC-Katalogisierung mit RAK : nach dem Format des DBI-Pflichtenheftes. - München u.a. : Saur, 1992. - 362 S.
In: Bibliotheksdienst 22(1988), Heft 9, S.841-856.
ISBN 3-598-11068-5
[Voraussetzung sind RAK-Kenntnisse. Kein eigentliches Kategoriesystem liegt zugrunde, sondern das in einem DBI-Projekt 1988/79 entwickelte "Pflichtenheft für Formalkatalogisierung auf dem PC". Insofern ist die Darstellung unabhängig von konkreter Software, wenngleich dadurch nicht eben leichter verständlich. In der Praxis muß dann oftmals zweigleisig gedacht werden, denn mindestens bei der Parametrierung muß man mit den "echten" Kategorienummern arbeiten]

Regeln für die alphabetische Katalogisierung : (RAK) / [hrsg. vom Bibliotheksverband der Deutschen Demokratischen Republik, Kommission für Katalogfragen ; red. Bearb. u. Reg.: Elisabeth Lotte von Oppen]. - 1. Aufl. - Leipzig : Bibliographisches Institut, 1989. - 571 S. ISBN 3-323-00183-4
[Der umfassendste und konsequenteste Regelcodex in deutscher Sprache. Diese Ausgabe spiegelt die Praxis auf dem Gebiet der ehemaligen DDR und bietet den breitesten Überblick über alle Probleme der alphabetischen Katalogisierung, reiches Beispielmaterial und eine unschätzbare Zusammenstellung von Tabellen aller Art im Anhang; das Register läßt keine Frage offen.]

Regeln für die alphabetische Katalogisierung - RAK / Hrsg. von der Kommission des Deutschen Bibliotheksinstituts für Alphabetische Katalogisierung. - Wiesbaden: Reichert, 1983 -
1. Regeln für Wissenschaftliche Bibliotheken - RAK-WB / Hrsg. von Irmgard Bouvier. - 1983. ISBN 3-88226-166-8
Online als Datenbank: http://www.allegro-c.de/regeln. Dort auch einiges zu RDA.
[Der Standard-Regelkodex der westdeutschen wissenschaftlichen Bibliotheken bis 2015.]

Regeln für den Schlagwortkatalog : RSWK / bearb. von d. Komm. d. Dt. Bibliotheksinst. für Sacherschließung. - 2. Aufl. - Berlin: Dbi, 1986. - 281 S. - ISBN 3-87068-359-7

Rösch, Hermann & Ruth Großgart: Lernbuch für Einführung von *allegro* in der Bibliothek der Friedrich-Ebert-Stiftung. - Bonn: Bibl. d. Friedrich-Ebert-Stiftung, 1994. - 105 S. - ISBN 3-86077-279-1
[Die sehr durchdachte Verfahrensweise der Bibliothek der Ebert-Stiftung wird hier ausführlich dargestellt. Es wird auf der Grundlage des NRW-Schemas gearbeitet!]

Tews, Annemarie: *allegro*-Ouvertüre : Partitur für Einsteiger. - Braunschweig: UB, Mai 1997. - 139 S. - ISBN 3-927115-32-0, DM 25.
[Enthält wichtige Teile des Lehrbuchs in einer etwas anderen Aufmachung. Hauptsächlich für Einsteiger: Konfiguration und Parametrierung werden nicht behandelt. Umfangreiches Glossar aller *allegro*-Fachbegriffe.
Auch als PDF-Datei verfügbar: http://www..allegro-c.de/doku/, das Glossar auch als Datenbank.]

UNIMARC manual : bibliographic format / International Federation of Library Associations. - 2. ed. München: Saur, 1994. - Losebl.-Ausg. - (UBCIM publications ; N.S., Vol. 14)
ISBN 3-598-11211-4
[Dieses Handbuch enthält Codelisten für Geographika, Formschlüssel und Sprachen, die in der Parametrierung der UB Braunschweig verwendet wurden. In der mitgelieferten Demodatenbank sind solche Codes zu finden.]

Weigel, Harald: Autographendatei und Nachlaßrepertorium durch Katalogisierung mit dem Datenbanksystem Allegro-C. - In: Der Einsatz der Datenverarbeitung bei der Erschließung von Nachlässen und Autographen, S.149-162. - Berlin: DBI, 1991.
[Beschreibung des *allegro*-basierten Systems **HANS**, das an der SUB Hamburg entwickelt wurde und sich inzwischen auch im Museumsbereich verbreitet hat. Das sehr umfangreiche Format H.CFG wird in einer eigenen Veröffentlichung von 1996, beschrieben.]

Wie katalogisiert man ein Buch? Ein Leitfaden nicht nur für Einsteiger. / B. Eversberg. 2011-.
http://www.allegro-c.de/regeln/rak-einf.htm
[Einiges Material zum Thema Katalogisierung, darunter auch eine Einleitung zu RDA in deutscher Sprache]

Anhang D Texteditor [historisch]

Empfehlung:

Benutzen Sie im DOS-Fenster X.EXE, wenn Sie an keinen anderen Texteditor gewöhnt sind. Sonst nehmen Sie den, mit dem Sie vorzugsweise arbeiten. (Einbau in die "Vorgabendatei" CP.OPT: siehe Befehl E, → 0.11.6). Es muß ein ASCII-Editor sein, also z.B. nicht WordPad oder NotePad, denn diese arbeiten mit ANSI-Zeichen. Die Umlaute etc. erscheinen dann hinterher falsch.
Ausnahme: die Datei UIFEGER. Hierfür empfiehlt sich NotePad. Für .RTF-Dateien: WordPad.

Anwendung:

Bearbeitung von "normalen" Textdateien, auch "ASCII-Dateien" genannt
= Dateien, die aus schlichtem Text bestehen (nur druckbare Zeichen), wobei die Zeilen durch die Steuerzeichen "Carriage Return / Line Feed" (Codes 13 10) getrennt sind
Darunter fallen die Parameterdateien (.APR, .APT, .API und .AIM), ferner die .CFG-Dateien, auch UIF- und H-Dateien sowie die ***CockPit***-Vorgaben CP.OPT,
ABER NICHT Datenbank- und Indexdateien (Typen .ALG, .ALD, .ADX, .TBL, .STL)!
Editor X wird vom ***CockPit*** automatisch aufgerufen, wenn eine Datei zu bearbeiten ist. Sie können aber mit dem E-Befehl in der Datei CP.OPT auch einen anderen Editor verlangen (→ Kap.0.11.6).

Installation:

X wird automatisch auf Ihr Programmverzeichnis kopiert. Die Dateien X.EXE und X.MAC (= Funktionstasten-Belegung, Farbeinstellungen) gehören zusammen.

Aufruf: **`x`** *dateiname*
z.B. **`x cat.api`**

Dann erscheint der Anfang der Datei (22 Zeilen). Wenn sie nicht existiert, wird sie am Ende beim Speicherbefehl (siehe unten: Befehl **Q**uit mit **`s`**) neu angelegt.

Zustand und Hilfe:

Der Editor befindet sich nach dem Start im **Befehlszustand**.
Hilfe: Oben steht eine Liste von Befehlen, die man gerade geben kann. Wenn man die **Leertaste** ("space") drückt, kommt die Liste der restlichen Befehle.
Von jedem Befehl gibt man nur den Anfangsbuchstabe ein (siehe unten). Im Befehlszustand kann man auch den Cursor bewegen, aber nicht schreiben. Druck auf 'x' schaltet den Schreibzustand ein.

Die wichtigsten Handgriffe : Zwei Minuten, die sich lohnen

Die folgenden drei Sätze erklären schon alles, was Sie für normale Bearbeitungen an den ASCII-Dateien (Parameter, Konfiguration, Stapel-, Hilfedateien etc.) brauchen.

1. Wenn die Datei erscheint: mit dem Cursor an die zu bearbeitende Stelle fahren. (Bei großen Dateien ist der **Find**-Befehl dabei hilfreich (siehe unter **f**).

2. **<Einfg>** : **Schreibzustand einschalten**. Oben links steht "Insert", man ist also im Einfügemodus. Mit Taste **<Einfg>** kann man ständig zwischen Einfüge- und Überschreibmodus ("Exchange") umschalten, wie bei Textprogrammen üblich.

3. **<Esc>** : Schreibzustand verlassen, **Befehlszustand** wird wieder eingeschaltet,
q : Quit-Befehl zum Beenden der Bearbeitung, dann gibt es 2 Möglichkeiten:
s : Save-Befehl zum Speichern
e : Ende ohne Speichern, d.h. die ausgeführten Änderungen werden ignoriert. (Bestätigen mit **y**)

a

Die nachfolgenden Seiten enthalten trotzdem eine vollständige Beschreibung aller Funktionen des Editors.

Schreibzustand:

Im Befehlszustand **'x'** drücken, dann ist man im **Überschreib**modus,
mit **<Einfg>** im **Einfüge**modus.

Die Funktionstasten haben in jedem Zustand folgende Wirkungen:

<Cursortasten> und <Bild↑>/<Bild↓>: normale Wirkung, wie Sie es erwarten.

<Pos1>	Sprung an den Zeilenanfang. Zweimal <Pos1> : Sprung an den Bildschirmanfang.
<Strg>+<Pos1>	Sprung an den Textanfang.
<Ende>	Sprung ans Zeilenende. Zweimal <Ende> : Sprung an den Anfang der letzten Bildschirmzeile.
<Strg>+<Ende>	Sprung an das Textende.
<Entf> <Rücktaste> <Einfg>	normale Wirkung wie in anderen Programmen
<Esc>	Sprung in den Befehlszustand, siehe oben, mögliche Befehle siehe unten.
<Enter>	oder <RETURN> : Zeile wird an der Stelle, wo sich der Cursor befindet, geteilt. (Passiert nicht im Befehlszustand.) Der hintere Teil wird eine Zeile tiefer gesetzt. Wenn man das nicht bewußt machen will, sondern einfach nur an den Anfang der nächsten Zeile gehen will: <Home> und <Cursor down> statt <Enter> geben.

Befehle

Hier die Liste der Befehle und ihrer Wirkungen:
Auslösung jeweils durch den **A**nfangsbuchstaben - klein oder groß.
Wenn Befehlszustand nicht eingeschaltet: vorher <Esc> drücken!

Hilfe: Oben steht eine Liste von Befehlen, die man gerade geben kann. Wenn man die **Leertaste** ("space") drückt, kommt die Liste der restlichen Befehle.

Again ein vorher gegebener Befehl, z.B. ein Suchbefehl (**f** oder **-**) oder ein Ersetzungsbefehl (**r**) wird wiederholt. Gibt man vorher eine Zahl, wird entsprechend oft wiederholt: z.B. **100 a** um 100 mal zu wiederholen (ohne <Enter> dazwischen).

Buffer Die Cursorposition wird markiert. Man fährt dann mit dem Cursor (dabei sind auch Bild↑ und Bild↓ sowie Find-Befehl erlaubt) an eine beliebige andere Stelle weiter unten in der Datei und drückt abermals **b** . Von der vorher markierten Stelle bis zu dieser Endposition wird alles in einen "Pufferspeicher" kopiert. Sodann:

Copy Der Inhalt des Pufferspeichers (siehe **B**uffer, **D**elete und **F3**) wird an der Stelle eingefügt, wo der Cursor gerade steht. Wiederholbar!

Delete Wie Buffer, nur wird beim zweiten Druck auf **d** der markierte Bereich gelöscht. Wenn man gleich danach **c** drückt, wird das rückgängig gemacht. Sonst kann man den gelöschten Teil auch woanders wieder einfügen. Auch mehrmals an verschiedenen Stellen. Beim nächsten **b-** oder **d**-Befehl oder F3 wird der Pufferspeicher durch einen neuen Inhalt ersetzt, d.h. sein Inhalt ist dann verloren.

Find man gibt danach eine beliebige Zeichenfolge und dann <Enter>. Falls die Zeichenfolge im Text (ab Cursorposition nach unten) vorkommt, landet der Cursor sofort genau dahinter.
Wiederholung mit **a** .

- (Minus) dasselbe, nur rückwärts, d.h. von der Cursorposition nach oben.

Get Einen anderen Text, dessen Name dann abgefragt wird, laden und an der Cursorposition in den aktuellen Text einkopieren. Der andere Text kann z.B. einer sein, den man vorher mit dem **P**ut-Befehl in eine Datei geschrieben hat.

Insert Übergang in den Schreibzustand (siehe unten), wobei die eingegebenen Zeichen an der Cursorposition eingesetzt werden (= Einfügemodus). Wechsel in den normalen Überschreibmodus und zurück: jeweils mit der <Einfg>-Taste (= <Ins>-Taste).

Jump Zu einer vorher gesetzten Marke (a,b,c,d) springen: dazu gibt man nach dem **j** einen dieser Buchstaben (siehe **T**ag).

Macro Einschalten der Makro-Prozedur: sie dient zur Belegung der Funktionstasten mit beliebigen Tastenfolgen. Es erscheint oben ein Untermenü
Delete **L**oad **R**ecord **S**ave
Man gibt einen der Anfangsbuchstaben; die Wirkung ist:

d: ein vorher mit **r** gespeichertes Makro wird gelöscht

l: eine vorher gespeicherte Funktionstastenbelegung wird geladen

r: Makro aufzeichnen: danach gibt man eine der Tasten F1 - F8. Auf dieser Taste soll die nachfolgende Tastensequenz gespeichert werden. Man gibt nun beliebige Befehle oder auch Text ein, geht dann wieder in den Befehlszustand (Esc) und drückt abermals **m** . Damit ist die gewählte Funktionstaste mit der eingegebenen Tastenfolge belegt. Beispiel siehe unten.

s: die zuvor gespeicherten Makros werden als Datei `x.mac` gespeichert und bei erneutem Aufruf von x automatisch geladen.

**** (Backslash) Eingabe von **Sonderzeichen**, die sonst per Tastatur nicht eingegeben werden können. Es erscheint oben die Frage `Literal:` , die man mit dem ASCII-Zahlenwert des gewünschten Zeichens beantwortet. An der Cursorposition erscheint dann dieses Zeichens. (Die Eingabe mittels <Alt>-Taste und Ziffern funktioniert nicht mit allen Zeichen.)
Will man z.B. das Teilfeld-Trennzeichen ▼ auf die Taste **F2** legen, kann man folgendes Makro dafür definieren (Tasten genau in dieser Reihenfolge drücken):
<Esc> m r F2 \ 31 <Enter> m m s
(Durch **m s** am Ende dieser Folge werden die aktuellen Makros für später gesichert.)

Other Ein zweiter Text kann zusätzlich geladen werden.
Zwischen beiden schaltet man anschließend nach Belieben durch Druck auf **o** hin und her und bearbeitet sie unabhängig voneinander. Jedoch:
Teile, die man in dem einen Text mit **b** markiert oder mit **d** gelöscht hat, kann man mit **c** in den anderen einkopieren!
Weil das Hin- und Her-Schalten sehr schnell geht, kann man diese Funktion auch benutzen, um zwei Dateien visuell zu vergleichen.
Hat man an beiden Texten Veränderungen gemacht, muß beim Quit-Befehl für jeden einzeln entschieden werden, ob er zu speichern ist oder nicht.

Put **Ausschnitt abspeichern** : Textbereich wird zunächst markiert wie bei **b**. Nach dem zweiten **p** wird aber nach einem Namen gefragt. Unter diesem Namen wird dann der markierte Bereich als Datei gespeichert. So kann man leicht Teile aus Dateien kopieren, die man später separat verwenden will. Mit dem Befehl **G** kann man einen solchen Abschnitt jederzeit an anderer Stelle wieder einfügen.
Gibt man PRN als Dateiname, wo wird der Textausschnitt sofort gedruckt.

Quit **Ausstieg aus dem Editor**. Es erscheint ein Untermenü:

Backup **E**xit **I**nitialize **S**ave-Exit **U**pdate **W**rite

Die wichtigsten Unterfunktionen sind hier **s** und **e** :

s : der Text wird in der bearbeiteten Form neu abgespeichert, die alte Form ist verloren. Das ist der Normalfall.
Text speichern:
Man gibt also normalerweise, um eine Bearbeitung abzuschließen, hintereinander **<Esc> q s**.

b : dasselbe, aber die alte Form wird umbenannt (Typ .BAK) und bleibt somit erhalten.

e : die Bearbeitung wird ergebnislos abgebrochen, d.h. die Datei bleibt unverändert erhalten. Wenn man Änderungen am Text vorgenommen hat, wird man noch gefragt, ob man wirklich die Änderungen ignorieren will. Man bestätigt mit **y** oder geht mit **n <Esc>** zurück in die Bearbeitung.

w : der Text kann in eine andere Datei geschrieben werden. Der Name wird dann abgefragt. Die andere Datei wird kommentarlos überschrieben, wenn sie existiert!

Replace Wie bei Find wird nach einer zu suchenden Zeichenfolge gefragt. Dann wird nach einer dafür einzusetzenden Ersatzzeichenfolge gefragt. Nach <Enter> wird die Ersetzung, falls möglich, ausgeführt.
Wiederholung mit **a** .
Mehrfachausführung: vor **r** eine Zahl eingeben! Wenn man z.B. eingibt: **999 r xyz <Enter> abc <Enter>**
(hinter der Zahl kein <Enter>, sondern sofort **r** !)
werden die nächsten 999 vorkommenden **xyz** jeweils durch **abc** ersetzt.

Set **Settings ändern**. Verschiedene Optionen können gesetzt werden.
Am wichtigsten: Case
Wenn man nach **s** hintereinander **c** und **y** gibt, wird die Groß-Klein-Unterscheidung eingeschaltet. Sonst wirken die Befehle **f s** und **r** nämlich unterschiedslos auf groß und klein geschriebene Buchstaben.
Farbsetzungen: s p f <Ziffer> für Zeichenfarbe, **s p b** <Ziffer> für Hintergrundfarbe.
(0=schwarz, 1=blau, 2=grün, 3=cyan, 4=rot, 5=magenta, 6=orange, 7=weiß. Auf die Vordergrundfarbe kann man 8 addieren, dann wird sie heller.) Mit **m s** macht man die Farbeinstellungen permanent, also auch für nachfolgende Sitzungen.

Tag **Sprungmarke setzen**. Danach drückt man a , b , c oder d.
Damit hat man die Cursorposition (unsichtbar) markiert. Also kann man bis zu vier Stellen in der Datei gleichzeitig markieren.
Mit **j** (siehe dort) kann man diese Stellen jederzeit wieder anspringen.
Die Marken bleiben bei Abspeicherung nicht erhalten!

Xchange Vom **Befehls**zustand geht man mit **x** in den **Überschreibmodus**,
d.h. in den "normalen" Schreibzustand. Aus diesem geht man mit <Esc> wieder in den Befehlszustand.

Die Nummer der aktuellen Zeile wird angezeigt.
Sprung auf eine bestimmte Zeile:
im Befehlsmodus die Nummer der Zeile eingeben (sie erscheint dann oben links), dann '#' (ohne <Enter> vorher oder hinterher!)

Funktionstasten

Jeder kann sie selbst belegen: siehe oben Befehl **m** und Beispiel unter **** (Backslash). Die Datei x.mac, die automatisch jedesmal geladen wird, enthält bei Lieferung diese Belegungen:

F3 : **Zeile löschen**, in der sich der Cursor befindet.
Anschließend kann man die Zeile woanders einkopieren, indem man den Cursor dorthin bewegt und Befehl **c** gibt. Die gelöschte Zeile wandert also zunächst in den Pufferspeicher (siehe Befehle **b** und **d**).

F4 : Vor die Zeile, in der sich der Cursor befindet, eine **Leerzeile einfügen**. Wenn man vorher (im Befehlszustand!) eine Zahl eingibt, werden entsprechend viele Leerzeilen eingesetzt. Also z.B. **7 F4** (kein <Enter> hinter der 7 !) um 7 Zeilen einzufügen.

F5 : **Zeilenende** (ab Cursor) **löschen**. Der gelöschte Teil geht in den Pufferspeicher und kann mit **c** woanders wieder einkopiert werden.

F8 : Wenn der Cursor gerade auf einem der Klammerzeichen ([{ }]) steht, dann springt er vor bzw. zurück zur dazugehörigen schließenden bzw. öffnenden Klammer. (Unschätzbar mindestens für den C-Programmierer)

F9 : **Exkursion:** Bearbeitung kurzfristig unterbrechen. Anschließend kann man jeden MS-DOS-Befehl ausführen und dann mit dem Befehl EXIT in den Editor zurückkehren - an die vorher verlassene Stelle!

D.h. F1, F2, F6 und F7 sind frei für eigene Belegung (siehe Befehl **M**). F10 ist nicht belegbar.

Anhang E Zeichensatz

Zum Thema Unicode: in ***a99*** `h unicode` eingeben

Das Konzept für DOS

Zum Lieferumfang ab V15 gehören einige Dateien, mit denen Sie den Zeichensatz der DIN 31628 (Stufe 2) verwenden können, der für Bibliothekszwecke geschaffen wurde und allgemein anerkannt ist.
Die Norm definiert einen Satz von 128 Zeichen, aber sie schreibt **nicht** vor, welche **Codes** den Zeichen zuzuordnen sind. Das wäre auch jederzeit, zumal bei ***allegro***, mit einer Umwandlungstabelle wieder zu ändern, wenn es einmal sein muß. Die Zuordnung, die seitens der ***allegro***-Entwicklung getroffen wurde, ist jedoch keineswegs willkürlich, sondern mit einiger Überlegung zustandegekommen. So wurde als Grundlage die Standard-Zeichensatztabelle 437 genommen, die jeder PC "von Natur aus" hat. Darin wurden solche Zeichen durch die noch fehlenden Zeichen der Norm ersetzt, die für ***allegro*** keine Bedeutung haben **und** in anderer, gängiger Software, soweit überschaubar, kaum oder nicht verwendet werden. Das sind im Wesentlichen diejenigen Rahmensymbole, bei denen Doppel- und Einzellinien zusammenstoßen. Die ***allegro***-Programme benutzen diese Zeichen nicht. So konnte die vollständige Norm verwirklicht werden, wobei die im Standardzeichensatz vorhandenen Akzentbuchstaben aber uneingeschränkt erhalten bleiben. (Auf der Grundlage der Tabelle 850 hätte man das nicht so machen können: es hätte nicht genügend viele freie Zeichen gegeben.)

Anmerkung. Mit Version 11.2 bis 13 wurde eine andere Lösung unter dem Namen VGAFONT mitgeliefert. Diese Programme und Dateien sind überholt. Wer sie bisher nicht eingesetzt hat, kann das Verzeichnis VGAFONT auf seiner Platte löschen.

Installation und Nutzung (DOS)

Wenn Sie eine Version ab V15 installiert haben, stehen auf Ihrem Programmverzeichnis (im Normalfall C:\ALLEGRO) alle zugehörigen Dateien. Das ***CockPit*** hat unter "eigene Routinen / Aktionen" einen Menüpunkt "Zeichensatz". Wenn Sie diesen auslösen, wird
1. der erweiterte Zeichensatz OSTWEST.FON geladen (mit `fontload <ostwest.fon`)
2. das Hilfsprogramm AW.EXE aktiviert, ein sog. "residentes" Programm.

Finden Sie den Menüpunkt "Zeichensatz" auf Ihrem ***CockPit*** nicht, wählen Sie über das Menü "Makros / Produktion starten" die Datei OSTWEST.BAT an.
Betätigen Sie danach die Tastenkombination <Alt>+w. Sie sehen eine Tabelle mit allen 256 Zeichen. Mit <Esc> verschwindet sie, mit <Enter> kopiert man das angewählte Zeichen auf die Cursorposition. Diese Hilfe können Sie nun in fast allen Programmen nutzen, mit denen Sie arbeiten, jedenfalls in den ***allegro***-Programmen.

Wenn Sie ständig damit arbeiten wollen, bauen Sie die Aufrufe am besten in CP.BAT oder sogar in AUTOEXEC.BAT ein. Die notwendigen Dateien FONTLOAD.COM, OSTWEST.FON, AW.EXE und OSTWEST.BAT können Sie natürlich auch woanders lagern.
Die Datei OSTWEST.FON ist eine ASCII-Datei, die mit dem X-Editor bearbeitet werden kann. Sie werden schnell erkennen, wie sie aufgebaut ist und wie sie verändert werden kann. Ändern Sie nicht die Zahlen und die Anzahl der Zeilen in dieser Datei, sonst wird die Funktion gefährdet. Es gibt daneben übrigens eine entsprechende Datei PICA.FON, die den im Pica-Verbund verwendeten Zeichensatz definiert. Mit dem Programm FONTLOAD kann man jederzeit eine andere Font-Datei laden. Der Befehl lautet:
`fontload <`*`dateiname`*. (Das Zeichen '<' vor dem Dateinamen ist notwendig!)

Drucken

Dies ist unter DOS ein viel schwierigeres Problem, als die Zeichen am Bildschirm sichtbar und benutzbar zu machen. Die allgemeine Lösung ist, einen "Druckertreiber" zu erstellen. Das ist eine Datei vom Typ .APT, die praktisch eine Tabelle von Zeichenzuordnungen enthält. Als Beispiel für eine solche Tabelle können Sie P-DSKJET.APT für den HP-Deskjet zu Hilfe nehmen. Wenn Ihr Drucker andere Befehle braucht, haben Sie mit P-DSKJET wenigstens eine Vorlage, die alle Zeichen enthält, und die Sie anpassen können. Das geht erheblich schneller, als selber eine neue Tabelle zu machen. Wenn Ihr Drucker bestimmte Zeichen nicht kennt, müssen diese per Tabelle auf ein anderes Zeichen, z.B. Buchstabe ohne Akzent, abgebildet werden.
Unter Windows ist das Drucken dagegen überhaupt kein Problem: Der Drucker-Button druckt alles, was im Anzeigefeld steht, genauso ab, wie man es dort sieht.

Tabelle der DOS-Zeichen und Tasten

Alle vom PC her gewohnten Textzeichen werden unverändert dargestellt und eingegeben. Zusätzlich gibt es dann diese: (die Tastenbelegung entspricht weitgehend der DIN und wird durch die **echo**-Befehle in OSTWEST.BAT erreicht) Stand: 6.3.2000

Taste	Code	Zeichen	
	213	ʿ	Ain
ALT-B	123	{	geschweifte Klammer auf
ALT-C	211	¸	Cédille
ALT-D	222	–	Querstrich durchgezogen
ALT-E	189	¨	Trema
ALT-F	64	@	sog. Klammeraffe
ALT-G	91	[	eckige Klammer auf
ALT-H	93	]	eckige Klammer zu
ALT-I	208	‾	Querstrich drüber
ALT-J	170	¬	Nichtsortierzeichen
ALT-K	221	€	Euro-Währungssymbol
ALT-L	214	©	Copyright
ALT-M	220	→	rechtsweisender Pfeil
ALT-N	125	}	geschweifte Klammer zu
ALT-O	176	ı	kleines türk. dumpfes i
ALT-P	178	þ	kleines isländ. Thorn
ALT-Q	198	″	Doppelakut

Taste	Code	Zeichen	
ALT-R	190	ˍ	Halbkreis übergeseetzt
ALT-S	219	/	Schrägstrich durchgezogen
ALT-T	126	~	Tilde
ALT-U	207	°	Ringel übergesetzt
ALT-V	212	.	Punkt untergesetzt
	184	·	Punkt übergesetzt
ALT-X	210	«	Ogonek (Haken rechts unten)
ALT-Y	209	×	Halbkreis untergesetzt
ALT-Z	199	ˇ	Haček
ALT-F2	177	Þ	großes isländ. Thorn
ALT-F3	215	Ø	gr. dänisches Ø
ALT-F4	216	¤	allg. Währungszeichen
ALT-F6	181	´	Akut (accent aigu)
ALT-F7	92	\	inverser Schrägstrich
ALT-F8	182	`	Gravis (accent grave)
ALT-F9	179	|	vertikaler Strich

Die folgenden Akzentbuchstaben gehören nicht zur DIN 31628/2: (in der ASCII-Reihenfolge)

018	ő	o mit Doppelakut	171	Š	S mit Haček	246	Ó	O mit Akut
019	ű	u mit Doppelakut	172	š	s mit Haček	242	Ł	L mit Schrägstrich
022	ų	u mit Ogonek	173	ý	y mit Akut	243	ł	l mit Schrägstrich
127	ě	e mit Haček	174	Ž	Z mit Haček	244	Ń	N mit Akut
155	ť	t mit Haček	175	ž	z mit Haček	245	ń	n mit Akut
157	ş	s mit Cédille	188	ă	a mit Breve	247	Ś	S mit Akut
158	ů	u mit Ringel	233	İ	gr. I mit Punkt	249	ś	s mit Akut
159	Ů	U mit Ringel	234	ţ	t mit Cédille	251	Ź	Z mit Akut
166	Č	C mit Haček	238	ą	a mit Ogonek	252	ź	z mit Akut
167	č	c mit Haček	239	ę	e mit Ogonek	253	Ż	Z mit Punkt
168	ř	r mit Haček	240	Ć	C mit Akut	254	ż	z mit Punkt
169	Ř	R mit Haček	241	ć	c mit Akut			

Für das Zeichen 223 (diakritischer Unterstrich) wurde keine Kombination voreingestellt, da ALT-_ nicht funktioniert. Man könnte z.B. den Befehl `echo ←[127;223` in OSTWEST.BAT ergänzen, dann wäre das Zeichen auf Strg-Rücktaste.
Alle Zeichen können über <Alt>+w, [Auswahl in der Tabelle], <Enter> eingegeben werden. Diejenigen, die oben mit einer ALT-Kombination angegeben sind, können auch mit dieser Kombination, also schneller, erzeugt werden.
Man unterscheide die hier ersichtlichen Werte der Akzente (akut=181 und gravis=182) von den normalen Zeichen, die auf der Tastatur vorhanden sind. Wichtig ist das im Falle des akut (' = ASCII 39), denn dieses dient zugleich als Apostroph. Wenn man Akzentbuchstaben wie é und ó eingibt, benutzt man die normale Akzenttaste und tippt gleich dahinter den Grundbuchstaben, dann wird beides automatisch kombiniert, wie man es gewohnt ist. Wenn jedoch der Haček z.B. mit dem n kombiniert werden soll, muß man ALT-Z n geben und erhält auf dem Bildschirm ˇn. Nur die von Hause aus im PC eingebauten Kombinationen (d.h. die des Zeichensatzes 437) sind als kombinierte Zeichen, wie sonst auch, darstellbar, die anderen muß man hintereinander schreiben. Diese Lösung wird beiden Extremen gerecht: diejenigen, die sich mit Sonderzeichen nicht abgeben wollen, können OSTWEST ignorieren), aber ihre Daten sind trotzdem mit denen der anspruchsvollen Anwender kompatibel, die die vollständige Norm einsetzen. Und bei einem späteren Umstieg auf die Norm kommt man ohne Umcodierung davon.
Achtung: Wenn Sie mit PRESTO zwei oder mehr Datenbanken gleichzeitig benutzen wollen, muß die Kombination <Alt>+a

funktionieren. In der Stapeldatei OSTWEST.BAT darf dann kein Befehl `echo ←[0;30;...p` stehen, denn der Wert 30 würde die Kombination <Alt>+a umdefinieren.

Das Konzept für Windows *(Online-Dokumentation:* **`h char`** *eingeben)*

Um die Diskrepanz zwischen den DOS- und Windows-Zeichensätzen zu überwinden und insbesondere den *allegro*-Anwendern ein kompromißloses Arbeiten unter Windows zu ermöglichen, wurde ein TrueType-Zeichensatz zusammengestellt, der alle Zeichen des verbreiteten OSTWEST-Font in sich vereinigt. Die Programme ***a99*** und ***alcarta*** sind dann in der Lage, zwischen den DOS- und Windows-Zeichencodes unter der Oberfläche hin und her zu schalten, ohne daß der Anwender dies merkt - es erscheinen einfach alle "von früher her" gewohnten Zeichen.

An den Datenbanken braucht nichts verändert zu werden (!), an den Parameterdateien sind höchstens sehr geringfügige, leicht durchzuführende Veränderungen vorzunehmen.

Das bedeutet: der Umstieg auf Windows verläuft schmerzlos, man kann sogar die alten Programme noch parallel und gleichzeitig auf denselben Datenbanken benutzen. (Deshalb wude die interne Codierung nicht auf Windows umgestellt.)

Diese Tabelle der Windows-ANSI-Codes kann in ***a99*** abgerufen werden: Man gibt `h char` im Schreibfeld ein, dann erscheint sie. Die mit □markierten Plätze sind unbesetzt (sie haben für Windows eine Sonderfunktion).

Windows-Standardzeichensatz

		1	2	3	4	5	6	7	8	9
030				!	"	#	$	%	&	'
040	(	)	*	+	,	-	.	/	0	1
050	2	3	4	5	6	7	8	9	:	;
060	<	=	>	?	@	A	B	C	D	E
070	F	G	H	I	J	K	L	M	N	O
080	P	Q	R	S	T	U	V	W	X	Y
090	Z	[	\	]	^	_	`	a	b	c
100	d	e	f	g	h	I	j	k	l	m
110	n	o	p	q	r	S	t	u	v	w
120	x	y	z	{	|	}	~		€	
130	,	*f*	"	…	†	‡	ˆ	‰	Š	‹
140	Œ	•	•	•	•	'	'	"	"	•
150	–	—	˜	™	š	›	œ	•	ž	Ÿ
160		¡	¢	£	¤	¥	¦	§	¨	©
170	ª	«	¬	-	®	¯	°	±	²	³
180	Î	µ	¶	·	¸	Î	º	Î	Î	½
190	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç
200	È	É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ
210	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û
220	Ü	Ý	Þ	ß	à	á	â	ã	ä	å
230	æ	ç	è	é	ê	ë	ì	í	î	ï
240	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù
250	ú	û	ü	ý	þ					

Umdefinierter Bereich von 128 bis 255

	0	1	2	3	4	5	6	7	8	9
120									€	→
130	♫	☼	◄	ğ	ű	ų	ˆ	ő	Š	↓
140	Ω	□	□	□	ţ	`	˚	δ	∞	˙
150	~	α	Γ	π	š	σ	œ	İ	□	L
160		⁄	t’	£	¤	▲	˘	§	¨	©
170	ą	¸	¬	–	Ř	¯	°	γ	▼	ˇ
180	´	µ	¶	·	¸	ı	º	►	ă	↑
190	ž	·	Ś	ś	ř	ů	Ä	Å	Æ	Ç
200	ę	É	ě	Ż	ş	ć	Ć	ż	Č	Ñ
210	ź	Ó	Ź	Ń	Ö	˛	Ø	Ł	ł	Ž
220	Ü	Ů	Þ	ß	à	á	â	ä	˝	å
230	æ	ç	è	é	ê	ë	ì	í	î	ï
240	č	ñ	ò	ó	ô	ń	ö	_	ø	ù
250	ú	û	ü	ý	þ	ÿ				

Mit einer einzigen TTF-Schrift kommt man allerdings nicht aus, man braucht mindestens zwei: eine proportionale und eine nichtproportionale (konstante Zeichenbreite):

`a-times.ttf`	Auf der Basis der TimesNewRoman – die z.B. in WinWord meistbenutzte Schrift.
`a-arial.ttf`	Als "serifenlose" Alternative
`a-letter.ttf`	Abgeleitet von der LetterGothic, eine Schrift mit konstanter Zeichenbreite. Man braucht sie, wenn es auf spaltengenaue Darstellung ankommt, z.B. in Index- und Kurztitellisten.
`a-lucida.ttf`	Abgeleitet von der Lucida Console, Alternative zu `a-letter`
`a-dos.fon`	Schrift für das DOS-Fenster (dort wirksam nur bei Schriftgrößen 8x13 und 10x19)

Es folgt noch eine zum Nachschlagen gedachte Tabelle aller Sonderzeichen-Codes, alphabetisch geordnet:

DOS	WIN	Zeichen		DOS	WIN	Zeichen		DOS	WIN	Zeichen	
160	225	á	´a	162	243	ó	´o	178	254	þ	kleines Thorn
133	224	à	`a	149	242	ò	`o	177	222	Þ	großes Thorn
131	226	â	^a	147	244	ô	^o	214	169	©	Copyright
132	228	ä	Umlaut ä	148	246	ö	Umlaut ö	221	128	€	Euro
134	229	å	a mit Ringel	018	137	ő	o Doppelakut	156	163	£	Brit. Pfund
228	188	ă	a mit Breve	237	248	ø	/o dänisch	216	164	¤	Währ.Symbol
238	170	ą	a mit Ogonek	---	156	œ	oe Ligatur	170	172	¬	Nichtsortierz.
142	196	Ä	Umlaut Ä	246	211	Ó	´O	236	148	∞	"[unendlich]"
143	197	Å	A mit Ringel	153	214	Ö	Umlaut Ö	248	176	°	Gradzeichen
145	230	æ	ae Ligatur	215	216	Ø	/O dänisch	220	129	·	Pfeil nach rechts
146	198	Æ	AE Ligatur	168	194	ř	r mit Haček	181	180	´	Akut
135	231	ç	c mit Cedille	169	174	Ř	R mit Haček	182	145	`	Grave
128	199	Ç	C mit Cedille	249	193	ś	´s	183	136	^	Circonflexe
241	205	ć	´c	172	154	š	s mit Haček	232	150	~	diakrit.Tilde
240	206	Ć	´C	157	204	ş	s mit Cedille	189	168	¨	Trema/diaeresis
167	240	č	c mit Haček	247	192	Ś	´S	184	149	·	Punkt oben
166	208	Č	C mit Haček	171	138	Š	S mit Haček	190	166	˘	Breve
130	233	é	´e	225	223	ß	scharfes s	198	227	″	Doppelakut
138	232	è	`e	155	162	ť	t mit Haček	199	179	ˇ	Haček/caron
136	234	ê	^e	234	144	ţ	t mit Cedille	213	146	ʿ	Ain
137	235	ë	e mit Trema	163	250	ú	´u	207	186	°	Ringel
127	202	ě	e mit Haček	151	249	ù	`u	208	175	¯	Macron
239	200	ę	e mit Ogonek	150	251	û	^u	209	215	ˬ	Halbkr. unten
144	201	É	´E	129	252	ü	Umlaut ü	250	183	·	Pkt Mitte
023	133	ğ	g mit Breve	019	134	ű	u Doppelakut	219	161	/	diakritischer /
176	185	ı	dumpfes i (türk)	022	135	ų	u mit Ogonek	222	173	-	diakr. Mitt.strich
161	237	í	´i	158	195	ů	u mit Ringel	210	171	˛	Ogonek
141	236	ì	`i	154	220	Ü	Umlaut Ü	211	184	¸	Cedille
140	238	î	^i	159	221	Ů	U mit Ringel	212	191	̣	unterg. Punkt
139	239	ï	i mit Trema	173	253	ý	´y	223	247	_	diakr. Unt.strich
233	157	˙	I mit Punkt	152	255	ÿ	y mit Trema	224	151	α	"[alpha]"
242	217	Ł	/L	251	212	Ź	´Z	231	177	γ	"[gamma]"
243	218	ł	/l	252	210	ź	´z	226	152	Γ	"[Gamma]"
245	245	ń	´n	253	203	Ż	Z mit Punkt	235	147	δ	"[delta]"
244	213	Ń	´N	254	207	ż	z mit Punkt	230	181	µ	"[mue]"
164	241	ñ	~n	174	219	Ž	Z mit Haček	227	153	π	"pi"
165	209	Ñ	~N	175	190	ž	z mit Haček	229	155	σ	"[sigma]"

Tip: Braucht man bestimmte Zeichen sehr häufig, legt man sie am besten auf Phrasenbuchstaben. In ***a99*** macht man das so: Sagen wir, das ę wird oft gebraucht und soll auf Strg+e. Man gibt im Schreibfeld ein: p ^e ę <Enter>. Von da an wird das ę erscheinen, wenn man Strg+e drückt. Oder: mit Alt+m die Tabelle holen, Schreibmarke vor das Zeichen, nochmal Alt+m. (Mehr zu Phrasen: `h phras` eingeben.)

Zum Schluß und zur Abwechslung eine dichterische Äußerung zum Thema Kataloge.
Die Echtheit des folgenden Fragments ist nicht verbürgt. Vielleicht stellt es aber das dar, was Goethe in seiner Schülerszene im Faust I geschrieben hätte, wäre ihm dieser Gegenstand genügend wichtig gewesen ...

....

Schüler: Denn was man schwarz auf weiß besitzt
Kann man getrost nach Hause tragen. soweit der bekannte Text
und hier beginnt das Fragment

Da Ihr nun von den Büchern sprecht,
Ach, könnt Ihr mich in diesem Punkt beraten?
Sie sind nicht wohlfeil, sind auch eben knapp,
Man läuft darnach sich manche Sohlen ab.
In meinem Beutel steht's mit Kreuzern schlecht
Und nicht zu reden von Dukaten!

Mephisto: Das hat nicht Not, wenn Ihr nur wißt,
Wo hiererorts die Bibliotheca ist.
Sie birgt Libelli und Folianten,
Faszikeln, Codices, Atlanten,
Wohl achtzigtausend an der Zahl...

Schüler: Wird dann das Suchen nicht zur Qual?
Durch solche Stapel, diese Mengen,
Zum Rechten sich hindurchzuzwängen?
Bedeckt von Staub kömmt man heraus!

Mephisto: Nein, damit sieht es anders aus.
Ihr geht nur hin und fragt also
Nach Auctor und nach Titulo.
Drauf wird Euch der Curator weisen
Wie es anders müßte heißen:
Der Nam' sei so und dies der Titel:
Der Catalog, das ist sein Mittel,
Das führt ihn zu des Buches Ort
In jenem Schrank, auf diesem Bord.
Bald habt Ihr, ohne daß Ihr hüstet,
Das Buch, nach dem es Euch gelüstet.

Schüler: Der Catalog? Ein großes Wort!
Verzeichnet jedes Buches Ort?
So kann in jenem Bücherhaufen
Ich alles finden, muß nichts kaufen?
Noch nie ist mir so wohl gewesen,
Ich eile, um darin zu lesen!

Mephisto: Halt, Sterblicher! wo denkt Ihr gleich hinaus!
Im Catalog der Bibliothek
Zu lesen, ist des Curators einzig Privileg!
Und er allein kennt drin sich aus.
Die Seiten starren von Symbolen,
Die keiner noch zu deuten weiß,
Und hätt' er sich dem Teufel anbefohlen ...

Schüler: Was! Steht nur durch Curators Gunst
Das Tor mir offen zu den Quellen
Der Wissenschaft und auch der Kunst?
Wer kann die Sinne mir erhellen
In dem unendlichen Geschwafel?
Muß hungern angesichts gedeckter Tafel!

Mephisto: Es wird ein neues Saeculum erst dämmern müssen
Bevor wir mehr von Catalogi wissen.
So grämt Euch nicht, den wirklich weisen Mann
Ficht alles das nicht ernstlich an.

Doch wählt mir eine Fakultät! ...usw., der bekannte Text ...

INDEX

= Editor-Promptzeichen
  (Kategorie- und Befehls-Eingabezeichen) 100
#u1, #u2 : Sonderkategorien 192
@ = Nicht-Stoppzeichen 97
_ : CockPit-Menü 2
¬ = Nichtsortierzeichen 97
¶ = Absatz-Endezeichen 97

A
A.CFG = a-Schema 300
a30 92
a35 92
a99 als Server 92
a99/alcarta: 89
ab = Aufnahme-Beginn (Exp.Param.) 174
Abbruch
  Volltextsuche bzw. Export 122
Abbruch des Programmlaufs
  Export, Volltextsuche 122
  Import 126
Abbruch in Kategorieliste
  Import 272
Abbruchbefehl
  Export 189
Abbruchbefehl (Export) 206
Abfrageliste
  abbrechen/fortsetzen (F8) 96
  Beispiel 96
  CFG-Datei 9
  Definition 296
Ablaufsteuerung
  Export 204
  Import 271
Absatz-Endezeichen
  Codierung für Export 185
  Zeilenvorschub-Steuerung (Editor) 97
Absturz 24
acon
  Programm 17
ACREPLY : Hilfsprogramm 20, 288
ad = Ausgabesatz ohne Kopf (Exp.Param.) 174
Adressen-Länge (Imp.Param. al) 256
Adressen-Modus (Imp.Param. am) 256
ADT (Dateityp) 8, 12
ADX (Dateityp) 10
ae = Aufnahme-Ende (Exp.Param.) 174
Aenderung, globale
  definieren 116
Aenderung, globale, ausführen 63
ag = Aufnahme-Gliederungsmodus (Exp.Param.) 174
AIM (Dateityp) 13
Akronyme als Zugriffsschlüssel 203
aktuelle Aufnahme 103
aktuelle Aufnahme auflisten (Editor:#l) 110
aktuelle Kategorie: Export-Sonderkategorie #cc 191
aktuelle Kopfkategorie: Export-Sonderkateg. #ch 191
al = Adressenlänge (Imp.Param.) 256
alcarta für Database Publishing 92
ALD (Dateityp) 10
aLF : Ausleihprogramm 19
ALFA
  Einfach-Ausleihprogramm 17
ALFA: Einfach-Ausleihe 49
ALG (Dateityp) 12
ALLEGRO (Programm) 17
allegro-Datenstruktur 8, 250
Allers'sche Testschleife 236
Alternativtabelle (Umcodierung bei Export) 182
am = Adressen-Modus (Imp.Param.) 256
am = max. Anzahl Exportsätze (Exp.Param.) 175
an = Nummer der ersten Aufnahme (Exp.Param.) 175
Anführungszeichen in Zeichenfolgen (Export) 170
Ansetzungsform (Begriff) 310
ANSI<->ASCII Codetabelle 185
ANSI-Escape-Sequenzen
  Ausnutzung für Bildschirm (Exp.Param dx) 182
  für Bildschirm-Anzeige 173
  für Farben 299
Anweisung (Export)
  Syntax 189
Anwendervariable
  ansehen 104
  bei Export 199
  bei Import 267
  beim Programmaufruf besetzen 286
  Export, Definition und Benutzung 199
  verändern, löschen 199
Anwendervariablen u1, u2 192
Anzahl Sätze in der Datenbank 61
Anzeige gefundener Titel bei Volltextsuche 122
Anzeige-Bildschirm 59
Anzeigefeld (a99) 67
Anzeigeparameter
  Auswahl per Option -p 284
  Beispiel 239
  Grundmuster 239
  Standard D-1.APR 13
APAC
  OPAC-Programm 17
  Start, Besonderheiten 49
APH (Dateityp) 14
API (Dateityp) 13
APR (Dateityp) 13
APT (Dateityp) 13
Arbeitsspeicher
  Belegung anzeigen 64
  konfigurieren 295
Arbeitstext (Export) 196
  Länge 202
Arbeitstext (Import) 264
  zwischenspeichern 267
aresqa
  Austausch mit SQL 92
Artikel
  am Titelanfang, Liste 179, 294
Artikelprüfung bei Eingabe 292
as = Aufnahme-Start (Exp.Param.) 174
ASCII<->ANSI Codetabelle 185
ASCII-Datei ausgeben 131
ASCII-Datei bearbeiten 337
ASORT (Programm) 18
ASP (Dateityp) 14
Aufbohrung (Dateigröße erhöhen) 178
Aufnahme = Titelaufnahme = Datensatz 5

Aufnahme entsperren (zum Bearb. freigeben) 333
Aufnahme gezielt mit Satznummer aufrufen 60
Aufnahme löschen 63
Aufnahme löschen mit UPD 160
Aufnahme retten (nach Löschung) 63
Aufnahme-Beginn (Exp.Param. ab) 174
Aufnahme-Ende (Exp.Param. ae) 174
Aufnahme-Gliederungsmodus (Exp.Param. ag) 174
Aufnahme-Nummer der ersten Aufnahme (Exp.Param. an) 175
Aufnahme-Parameter (Export) 174
Aufnahmespeicher: Größe 295
Aufnahme-Start (Exp.Param. as) 174
Ausgabedaten/Eingabedaten 1
Ausgabefunktionen
  im Schnellzugriff 64
Ausleihdaten 320
Ausleihprogramm
  aLF: mit allen Funktionen 19
Ausleihprogramm
  ALFA (einfach) 19
Auswahlfeld (a99) 68
Auswahl-Listen (Dateiauswahl) 22
Auswertungen, zahlenmäßige 245
authority files -> Normdaten 314
automatische Nummernvergabe 293
avanti
  update 159
avanti für Client/Server-Zugriff 43
AW : Zeichensatz-Hilfsprogramm 20

B
Backspace-Liste (Exp.Param. pb) 185
BACKSPACE-Taste
  im Editor 98
  Indexfunktion (Ergebnismenge rücksetzen) 56
Backup-Kopie der Datenbank 24
Bandaufführung 174
  auf eigener Karte 180
  Begriff 5
  blättern 311
  kopieren (Editor: #k) 109
  löschen 312
  Vertauschen (Editor: #m) 111
  zwischen vorh. Bände einfügen 108
Basis-Parameter
  Import 253
Basisparameter ändern (Export) 204
Basis-Parameter für Export 168
BAT (Dateityp) 15
Batchdatei
  bearbeiten 37
  starten 37
Batchdatei 15
Bearbeiter : Export-Sonderkategorie #op 192
Bearbeiterkennung
  Vorgabe mit Option - O 284
Bearbeitung einer Kategorie (Editor-Befehl #b) 104
Bearbeitungsbefehle
  Export 194
  Import 267
Bearbeitungsfunktionen im Schnellzugriff 62
Bearbeitungsmodus
  Volltextsuche 122
Bedingter Sprung (Export) 189
Bedingter Sprung (Import) 271
bedingtes Postfix 190
Befehle: allgemeine Form 102
Befehls- und Eingabemodus (Editor) 96
Befehlszeile (a99) 68
Beginnposition im Fremddatenfeld (Import) 266
Beginnposition setzen (Exp. Manipulationsbef. b) 196
Benutzerdaten, Struktur 324
Benutzereingabe
  umcodieren (Index-Par.) 206
  Umcodierung gewünscht 178
Benutzeroberfläche
  CockPit 32
  Konzept 23
  Portabilität 42
Benutzeroberfläche
  UIF-Dateien 15
Benutzervariable -> Anwendervariable 199
Berechtigungsstufen 68
Berechtigungsstufen : Option -a 281
Beschleunigung des Updates 155
Bestellsätze (Datenelemente) 320
Betriebssystem : Schnittstelle 326
Betriebssystem, temporärer Ausstieg 64
Bibliofile
  Datenstruktur 250
  Spezialbefehl bei Import 265
Bibliotheksadressen 323
Bildschirmanzeige
  Farbwahl 298
  Parameter 13
  Zeichenumwandlung (Exp.Param. dx) 182
Blättern
  im Anzeigebildschirm 61
  im Arbeitsspeicher (Editorbefehle) 103
  im Editor (Befehle #z, #+, #-) 117
  im Index 53
  in der Ergebnismenge 61
  in der Menge der vorher bearbeiteten Aufnahmen 61
  in langen Aufnahmen 59
  in Satznummernfolge 61
  in verknüpft gespeicherten Sätzen 61
Boolesche Operatoren
  im Schnellzugriff 56
  in der Volltext-Suche 122
Breite der Kategorienummern
  Konfig.-Befehl t 290
Buttons (a99) 75

C
CAT.API : Indexstruktur 43
cc = aktuelle Kategorie (Exp.-Sonderkat.) 191
cca = aktuelle Kategorie incl. Nummer (Exp.-Sonderkat.) 191
cDT (Dateityp) 8, 12
cDX (Dateityp) 10
CFG (Dateityp) 9
cfgconv (Hilfsprogramm) 19
CFL (Dateityp) 9
ch = aktuelle Kopfkategorie (Exp.-Sonderkat.) 191
Character String (Steuerzeichenfolge)
  Export 170
  Import 253
Check
  Funktion von Programm UPD 155
cIM (Dateityp) 13
cLD (Dateityp) 10
cLG (Dateityp) 12
CockPit
  allgemeines 32
  Englisch 33

Optionen 39
Parameterübergabe 37
Programm 17
Unerwünschte Menüpunkte 40
Codierbefehle p und q (Exp.Param.) 183
Codiertabelle
Export 182
Import 258
CP.BAT
CockPit-Start 33
CP.OPT : Vorgaben für CockPit 15, 40
cPI (Dateityp) 13
cPR (Dateityp) 13
C-Programmierung : Zusatzfunktionen 326
cPT (Dateityp) 13
CS (Character String) siehe: Steuerzeichenfolge 170
Cursortasten 98
Blättern im Index 51
Blättern in der Datenbank 61

D
Database Publishing 26
Database publishing mit alcarta 92
Datei wechseln (Eingabedatei bei Schnellzugriff) 62
Dateiauswahl (DOS-Programme) 22
Dateiende beim Export 208
Dateien-Hierarchie 7
Dateigröße, Maximalwert 286
Dateikopf erzeugen
Exp.Param. di 182
Kopfabschnitt der Exportparameter 208
Dateikopf-Länge (Imp.Param.) 253
Dateiname: Export-Sonderkategorie #fn 191
Dateitypen 5, 16
Datenausgabe -> Export 161
Datenbank
aktualisieren mit a99 159
aktualisieren mit avanti 159
als Ganzes erneuern 148
entlüften 147
entsperren 36, 331
Generierung aus vorhandenen Daten 47, 144
Import von Fremddaten in eine D. 127
Indexdefinition -> Index / Parameter 13
Kernbestandteile 26
maximale Dateigröße 178
neu anlegen 35, 44
neu aufbauen mit UPD 159
neu entwerfen 45
nummerntreue Erneuerung 148
organisieren 36
Schema : siehe Konfiguration 9
Sicherung 24
Sperre aufheben 331
sperren/freigeben 36
Umfang (Anzahl Datensätze) 61
veröffentlichen (Database Publishing) 26
wechseln 52
Datenbank-Dateien
durchsuchen (Volltext-Suche) 119
Typ 5
Datenbankdefinition 13, 289
Datenbank-Generierung 47, 143
Dateneingabe (DOS), allgemeine Form 100
Datenerfassung
Abfrageliste 96
Erleichterungen 55, 62
in a99 71
online (Schnellzugriff) 17, 62
Phrasen als Eingabeerleichterung 95, 112
Übernahme von Elementen zur nächsten Aufnahme 114
Vorüberlegungen 93
Datenfeld -> Kategorie 6
Datenmanipulation (Export) 167
Datenmanipulation (Import) 267
Datenquelle, Auswahl mit Option -d 281
Datensatz -> Aufnahme 5
Datensicherung 24
Datenumwandlung siehe Import 249
Datum
als Phrase "Leertaste" 294
als Phrase 0 95, 112
Exp.-Soderkategorien #dt, #dts 191
Länge festlegen 294
Datumsprüfung bei Eingabe 292
dBase-Daten umwandeln (Import) 252
DEFAULT.OPT -> CP.OPT 15
Del-Taste = Entf-Taste 98
di = Drucker-Initialisierung (Exp.Param.) 182
DIN 1506 (Import) 256
DIN 31628 Zeichensatz 342
DIN 31628/2 (Zeichensatz) 29
Directory Anfangsposition (Imp.Param. fd) 256
Directory: ISO-Struktur 249
Directory-Eintragung suchen (Import) 265
Direktes Postfix (Exp. Manipultionsbefehl PX) 196
Direktes Präfix (Exp. Manipulationsbef. pX) 196
Direktzugriff ohne Umweg über Index 60
dn = Dateiname der Parameterdatei (Exp.Param.) 182
Dokumentdaten: Dateitypen .cLG und .cLD 5
DOS-Ausstieg 64
Download -> Export 161
Druckbefehl
Index-Abschnitt drucken : F2 53
Schnellzugriff : F2 64
Drucker-Ersatzdarstellung (Export) 184
Drucker-Initialisierung
Export-Parameter di 182
vom Editor aus (Befehl #p/i) 113
Druckername 295
Druckerparameter (Export) 182
Druckersteuerung (Export) 171
Druckersteuerung mit Steuerzeichen 171
Druckertreiber
Dateityp 13
Tabellendatei 168
Druckmenü
im Schnellzugriff 64
Druckparameter, Beispiel 243
dt, dts = Datum (Exp.-Sonderkategorie) 191
Dublettenprüfung 18
Dublettenprüfung 155
dx = Zeichenumwandlung für Bildschirm (Exp.Param.) 182

E
EBCDIC-Daten : Zeichenumwandlung 258
Editor 99
  Befehls- und Eingabemodus 96
  Begriff, Konzept 3, 93
  Form der Befehlseingabe 102
  Form der Dateneingabe 100
  Korrektur von Eingabefehlern 101
  während Import benutzen 126
  während Volltextsuche benutzen 122
Editor-Befehl
  #, #_ : Suchen und Ersetzen 118
  #+ #- : im Arb.Speicher vor-/zurückblättern 117
  #a : Hintergrundspeicher anzeigen 104
  #b : Bearbeite ein Feld/einen Datensatz 104
  #e : Ende der Sitzung / des Programmlaufs 122, 126
  #h : Hilfe 107
  #i : Einfügen eines neuen ((Unter-)Satzes 108
  #k : Kopiere einen Satz 109
  #l : aktuellen Satz auflisten 110
  #M #m : markieren und verschieben 111
  #p : Phrasen und parameter laden/speichern/ändern 112
  #r : Rücksprung ins Menü / Forts. d. Progr. 114
  #r : Rücksprung ins Menü / Fortsetzung des Programms 122, 126
  #t : Transfer (Übernahme) aus Hintergr.Speicher 114
  #v #V : Feld bzw. Satz vergessen 115
  #X : Ersetzung (eXchange) definieren 116
  #z : bestimmten Satz im Arb.Speicher aufblättern 117
Editor-Befehl: 106
Editor-Befehle
  Form 102
  Kurzübersicht 102
Einfg-Taste 98
Einfügemodus (Insert-Modus) bei Dateneingabe 98
Einfügen Satz/Bandaufführung (Editor #i) 108
Eingabedaten/Ausgabedaten 1
Eingabemaske : siehe Abfrageliste 96
Eingabeprüfung, eingebaute 292
Eingabeprüfung, programmierbare 228
Einheitssachtitel
  Normdaten 315
Einmischen von Daten 155
Einrückung am Zeilenanfang
  Änderung (Strukturbefehl #) 171
  Exp.Param. zi 181
Einschränkung von Suchergebnissen Siehe Restriktionen
Einschränkungs-Operator (Volltext-Suche) 122
Einzelplatzbetrieb 285
Einzelplatzbetrieb : Option -S 50
Endabschnitt für Listenproduktion 208
Ende der Sitzung : F8 64
Endebefehl
  Export 189
Endebefehl in der Kategorieliste
  Export 206
  Import 272
Endemarke (variable Trunkierung im Index) 54
Endemarke abschalten 54
Endezeichen, globales, bei Export 183
End-of-file Routine bei Export 208
Endposition setzen (Exp. Manipulationsbef. e) 197
Enter-Taste 98
Entf-Taste = Del-Taste 98
Entlüften von Datenbankdateien 36, 147
Entsperren
  Datenbank 36
  Datensatz 333
Entstoppungszeichen 44, 97, 179, 295
environment variables -> Umgebungsvariable 280
Erfassung : siehe Datenerfassung 62
Erfassungsdatum 293
Erfassungsmaske -> Abfrageliste 9
Ergebnismenge
  anzeigen und durchblättern 61, 70
  Begriff 56
  der bearbeiteten Sätze 61
  Einzeltitel herausnehmen/ergänzen 60
  exportieren 64
  gesamte Datenbank als E. 60
  global korrigieren 63
  Globale Manipulation 63
  Kurzanzeige 57
  löschen 63
  maximale Größe 295
  Satnummernbereich als E. 60
  sortieren (a99) 70
  um einz.Titel ergänzen/vermindern 60
Ergebnismenge rücksetzen 56
Ersatzdarstellungen für Einzelzeichen (Export) 184
Ersetzung
  ein Zeichen durch mehrere 184
Ersetzungsfunktion
  bei Export, global 186
  bei Export, lokal 197
  bei Export, Zeichen 182
  bei Import, global 258
  bei Import, lokal 267
  im Schnellzugriff 63
Ersetzungsmodus 216
Ersetzungsschlüssel 217
erweitertes Register 58
Erwerbungsdaten 320
Erwerbungsprogramm 19
Esc-Menü (APAC) 49, 65
Esc-Taste
  als Phrasen-Funktionstaste 95, 112
Exemplarsatz, Struktur 322
Expandieren = Trunkierung aufheben 54
EXPEX : Exportieren lernen 141
Export
  aus APAC 282
  Ergebnismenge (Schnellzugriff) 64
  Komfort-Funktionen 81
  Konzept 161
  mit Komfort (a99) 80
  Programmablauf 120
  Start im Batch 123
  und Import (Verknüpfung) 125
Exportparameter
  Auswahl per Option -e 282
  im Editor online ändern (zum Testen) 114
  vom Editor aus laden (Befehl #p) 113
Exportparameterdatei
  Dateityp 13
  Inhalt 164
Exportsätze
  maximale Anzahl je Aufnahme (Exp.Param. am) 175
  Steuerung der Auswahl und Anzahl durch Kopfbefehle 175
Exportsprache 167
Exportvorgänge: mehrere gleichzeitig 121
Expresszugriff über Satznummer 60
Externdatei (Typ .ADT) 8

externe Funktion 60
Externformat-Ausgabe 244

F
FAQ (Frequently Asked Questions) 27
Farben der Menütexte
  Änderung 23
  Festlegung 298
Farbumschaltung in der Bildschirmanzeige 171
fb = Formularbruch (Exp.Param.) 180
fc = Feldlängen-Korrektur (Imp.Param.) 257
fd = field directory position (Imp.Param.) 256
fe = Feldende (Imp.Param.) 254
Fehlerkorrektur bei der Eingabe 101
Fehlermeldungen : UIF-Dateien 15, 23
Feld -> Kategorie 6
Feld im Fremdsatz ungültig machen (Import) 268
Feldanzahl je Datensatz (Imp.Param fx) 254
Feldbelegungs-Statistik 80
Felddeskriptor 291
Feldende (Imp.Param. fe) 254
Feldlängen-Korrektur (Imp.Param. fc) 257
Feldrepetierfaktor (Imp.Param. fr) 257
Feldstart (Imp.Param. fs) 257
Feldwiederholung -> Mehrfachfeld 294
Fenster-Bearbeitung (Befehl #b) 104
Feste Satzlänge: Import 253
ff = Fußzeilen-Anzahl je Formular (Exp.Param.) 180
fh = File Header Länge (Import-Parameter) 253
Field Directory Position (Directory Anfangsposition)
  (Imp.Param. fd) 256
FILELIST : Liste der Dateien 15
Findebefehl (Editor: #f) 106
Find-Menü (a99) 70
fl = Formularlänge (Exp.Param.) 180
FLEX-Befehle 84
FLEX-Konzept (a99) 82
Flip 225
fm (UPD-Funktion "mischen") 157
fm = Formular-Modus (Exp.Param) 180
fm = Mehrfachfeld-Kennzeichnung (Imp.Param.) 257
fn = Dateiname (Export-Sonderkategorie) 191
fn = Formular-Nutzen je Seite (Exp.Param.) 180
Folgezeichen bei Mehrfachbesetzung von Kategorien 100
Formalkatalogisierung 310
Formalprüfung eingegebener Kategorien 101
Format für Bibliotheksdaten
  -> Kategorienschema 9
Formularbruch (Exp.Param. fb) 180
Formulardatei für Erfassung/Bearbeitung 72
Formulare zur Dateneingabe (a99) 72
Formularlänge (Exp.Param. fl) 180
Formularmodus (Exp.Param. fm) 180
Formularnutzen je Seite (Exp.Param. fn) 180
Formularparameter (Export) 180
fortlaufendes Sammelwerk (Begriff) 310
Fortsetzungskarte 172, 202
  Numerierung 172
Fortsetzungskarten-Anfang
  Exp.Param fa 180
fr = Feld-Repetierfaktor (Import) 257
Frage an den Benutzer 205
Fragen, häufig gestellte 27
FREE : Hilfsprogramm 20
Freigabe = Entsperrung
  einer Datenbank 36
  eines Datensatzes 333
freigeben
  Datenbank 36
Fremddaten
  Bedeutung für rationelle Erfassung 93
  direkt kopieren 62
  einmischen 127
  konvertieren 125
  Konvertierung: siehe Import 18
  Typen 249
Fremddatenübernahme mit a99 89
FRM (Dateityp)
  Formulardatei 72
fs = Feld-Start (Imp.Param.) 257
führende Nullen beseitigen (Export) 197
führende Nullen ignorieren (Import) 267
Füllhorn (a99) 75
Füllzeichen
  Anzahl je Aufnahme 294
  Code 97
  Code definieren 294
Funktionstasten
  eigene Belegung 65
  im Editor (Übersicht) 98
  Schnellzugriff (Übersicht) 65
Fußabschnitt bei Karten- und Seitendruck 208
Fußzeilen-Anzahl (Exp.Param. ff) 180
fx = Feldanzahl je Datensatz (Imp.Param.) 254

G
gelöschte Aufnahmen
  beseitigen (Datenbank "entlüften") 148
  durchblättern 53, 63
  reaktivieren 63
Generierung einer Datenbank 143
Geschäftsgangsdaten 320
GK : Hilfsprogramm 20, 288
Gliederungsmodus (Exp.Param. ag) 174
Globale Ersetzungen
  Export 186
  Import 258
Globale Manipulation 63, 202, 206
  bei UPD 160
Globaler Änderungsbefehl 63
  Berechtigung (Option -a3) 281
  Definition 116
globales Endezeichen bei Export 183
go to - Befehl (Import) 264
Grafikanbindung 225
Groß-/Kleinschreibung bei Volltextsuche ignorieren 14
Großbuchstabe/Kleinbuchstabe, Prüfung 201
Grunddateien
  Bearbeitung 6
  Dateitypen ALG und ADT 5
  durchsuchen (Volltext-Suche) 119
  Struktur 8, 250
gt = Gesamttitel, übergeordneter (mehrbd. Werke)
  Sonderkategorie bei Export 193

H
häufig gestellte Fragen 27
Hauptsatz einer Datensatzgruppe 5
HELP :Taste F1 im Editor 107
Hierarchie-Code (Export) 171
Hierarchie-Ebene erhöhen (Import) 268
Hierarchie-Gliederung (Exp. Sonderkat. #hi) 193

Hierarchie-Separator (Exp.Param. kh) 187
hierarchisch gegliederte Datensätze 5
  Export 174, 188
  Import 255, 268
  Update 160
Hilfe :Taste F1 107
HILFE-Funktion
  im Schnellzugriff 64
HILFE-Funktion im Editor 107
Hilfetexte 91
Hilfe-Texte 14, 23
  für Datenerfassung 107
  für Restriktionen 234
  zu einz. Datenfeldern 107
Hilfsabschnitt (PV) 228
Hilfsvariable (Import)
  prüfen 271
  setzen 268
hinteres Ende abschneiden (Export) 197
Hintergrundspeicher
  aktuelle Aufnahme hineinkopieren (Shift+F5) 62
  anzeigen (Editor-Befehl #a) 104
  Datenfelder übernehmen (Editor #t) 114
  maximale Größe 295
Hintergrundspeicherbefehle
  Import 267
Hinweiskategorien 296

I
i0 = Länge der Kurzanzeige 177
i1 = Trennzeichen für zweiteilige Schlüssel (Exp.Param 178
i2 = Verbotszeichen für Indexeintragungen (Exp.Param.) 178
i3 = ungültige Endzeichen (Exp.Param.) 178
i4 = Ersetzungsmodus (V14) 216
i5 = Ersetzungs-Steuerzeichen (V14) 216
i6 = Ersetzungs-Register 216
i7 = Register für SR-Schlüssel 223
i8/i9 = Verweisungszeichen im Index 178
ia,ib = Steuerzeichen für Sequenzen-Ersetzung 178
ic = Umcodierung einschalten 178
Identnummern, automatische Vergabe
  siehe Nummernvergabe 293
ii = Multiplikator f. max. Dateigröße 178
II-Dateien 15
il = max. Länge der Schlüssel 178
Import 125
  Konzept 249
  Programmablauf 128
  Programmaufruf 130
  Programmunterbrechung 126
  und Export (Verknüpfung) 125
IMPORT
  Programm 18
Importparameterdatei
  Auswahl per Option -i 283
  Dateityp .cIM 12
  Erstellung 252
  Inhalt 251
include-Dateien (Export, Typ .APT) 13
Index
  Abschnitt drucken 53, 152
  Dateityp .cDX 10
  durchblättern 51
  Einträge als Phrasen kopieren 55
  Eintragserzeugung als Exportvorgang 177
  Eintragung löschen 55
  erneuern 36, 147
  Erstellung (Datenbank-Generierung) 18
  für Schnellzugriff 43
  kompaktieren 36
  manuell ändern 55
  neue Eintragung erzeugen 55
  Parameterdatei (Beispiel) 248
  Parameterdatei (Konzept) 43
  Parameterdatei (Typ) 13
  Parameterdatei anlegen 46
  Register-Bildschirm 51
  Spezialparameter für Indexproduktion 177
  symbolische Registernamen 178
  teilweise erneuern 151
  Überschriften 179
  umblättern 53
  umschalten zwischen mehreren 52
  Zuordnung eines Schlüssels 203
INDEX (Programm) 18
INDEX oder UPD zum Einmischen? 127
Indexcodes (Sortierwerte) 185
Indexieren
  nach Unterbrechung fortsetzen 150
Indexparameter
  Auswahl per Option -i 283
  Beispiel 248
Index-Präfix 196, 203
Indikator-Aktion (Exp.-Steuerbefehl #i) 205
Indikatoren
  erlaubte Werte 291
Indikatorprüfung (Exp.Manipulationsbefehl i/I) 201
Indirektes Postfix (Exp.Manipulationsbefehl Pz) 201
Indirektes Präfix (Exp.Manipulationsbefehl pz) 200
INI-Datei (a99) 10, 89
Initialisierung des Druckers (Exp.Param. di) 182
Initialisierungsdatei (a99 und alcarta) 10
Installation 30
  Windows-Programme 89
Ins-Taste-> Einfg-Taste 98
integrierter Editor --> Editor 3
Internformat als Exportformat (I-1.APR) 126
Interpunktion bei Druckformat 162, 169
Interpunktionszeichen am Kategorie-Ende
  automatische Beseitigung (Import) 267
INVENT
  Exemplar-Erfassung 322
  Programm 19
Inventarisierung von Exemplaren 322
ir = Länge des Restriktionsschlüssels 231
ISBN-Prüfziffertest 292
ISO-Norm 2709 (Import) 256
ISSN-Prüfziffertest 292

J
Ja/Nein-Abfragen : Codes für "ja"/"nein" 295
Jahreszahl-Prüfung bei Eingabe 292
JanaS
  allegro-Web-Browser 92
JUMPS : Hilfsprogramm 20

K
k1 : Konfigurationsbuchstabe 280
Kartendruck 64
Kartendruck, Formularmodus 180
Katalogisierung 310
Katalogkartenköpfe 175
Katalogregelwerk 307

Kategorie
= Datenfeld 6
kopieren (Editor: #t) 114
löschen (bei Export) 202
löschen (Editor: #v) 115
löschen per UPD 160
neu definieren 290
teilen/zusammenlegen (Import) 262
Kategorie-Anfangszeichen definieren 294
Kategoriedeskriptor 290
Kategorieliste
Export 188
Kategorienschema 9
A.CFG (allegro-Standard) 307
Auswahl per Option -k 283
Definition in der .CFG-Datei 290
Kategorienummer (direkte Eingabe) 100
Kategorienummer, Breite 290
Kategorienummern ändern, bei Import 262
Kategorienummern ändern, im Editor 101, 118
Kategorieparameter (Export) 187
Kategoriespezifische Ersetzung (Export) 186
Kategoriespezifische Hilfe 107
Kategorie-Startposition (Exp.Param. ks) 187
Kategorietextanfang: Position festlegen 290
Kategorietextlänge begrenzen (Exp.Param. kt) 187
kb = Kategorie-Folgezeichen blank setzen (Exp.Param.) 187
ke = Kategorie-Endecode (Exp.Param.) 187
Kein Eintrag unter dieser Satznummer 333
kh = Hierarchie-Separator (Exp.Param.) 187
Klammeraffe ("Entstoppungszeichen") 44
Klammern in Suchbegriffen (Volltext-Suche) 124
Kleinbuchstabe/Großbuchstabe, Prüfung 201
Koerperschaftsnamen
Normdatensätze 314
Komma bei Volltextsuche 124
Kommentare
in .CFG-Datei 289
in Exportparametern 168
in Importparametern 251
Komplementärmenge (logisches NICHT) 56
Konfiguration
Dateityp 9
Kennbuchstabe 5, 9
Konzept 6
Option -k 283
Standard-Datei $A.CFG 300
Konfigurationsbefehle 290
Konfigurationsbuchstabe 9
Konfigurationsbuchstabe: Env.var. -k1 280
Konkordanzliste (Import) 268
Anwendung bei Pauschalimport 263
Konsolidiertes Format 307
Konvertierung von Fremddaten
-> Import 125
Konvertierungsbefehle, globale (Import) 258
Kopfabschnitt bei Karten- und Seitendruck 208
Kopfbefehl (Exp.Param. ak) 175
Kopfkategorie
Zerlegung (Exp.Param. ak) 176
Kopftext-Wiederholung (Export) 205
Kopieren einer Aufnahme
im Schnellzugriff (Eingabeerleichterung) 62
in a99 71
in den Hintergrundspeicher (Shift+F5) 62
Kopieren eines Satzes im Editor (Befehl #k) 109
Korrektur von Eingabefehlern
durch Neueingabe 101
global: siehe Globaler Änderungsbefehl 63
im Schnellzugriff 62
mit Befehl #b 104
mit Befehl #b (horizontale Bearbeitung) 101
mit Cursor 101
mit Such- und Ersetzungsbefehl 101, 118
Korrekturdatum 293
kt = Kategorie-Textlänge begrenzen (Exp.Param.) 187
Kurzanzeige
Länge 177
maximale Länge (i0) 177
Kurzliste der Ergebnismenge 57
Kurztitelanzeigen 57
Kurztitelliste drucken (F2) 57
Kurztitelregister
Dateityp 11
erneuern 36, 143
Kurztitelzeile, Aufbereitung 206

L
Länge des Arbeitstextes 202
Längenbegrenzung bei Export (Exp.Param. kt) 187
LANGUAGE CODE : Sprache wechseln 64
Leersätze (gelöschte Sätze) beseitigen 147
Lieferantendaten 323
Lieferantensätze, Struktur 323
Linkstrunkierung 69
LISTE : Standard-Ausgabedatei 15
Listendruck, Formularmodus 180
Listenproduktion 81, 132
Endabschnitt 208
mit QUEX ganz einfach 136
Loeschen
einer Aufnahme aus der Datenbank 63
einer Aufnahme bei offline-Bearbeitung (#V) 115
einer Aufnahme rückgängig machen 63
einer Kategorie (Editor #v) 115
einer Kategorie beim Import 262
eines Satzes mit UPD 158
Loeschkontrolle 218
Loeschkontrolle, Sonderschlüssel 217
LOG (Dateityp) 14
LOG2ALG 158
LOG2ALG (Hilfsprogramm) 19
LOG-Datei auswerten 158
logische Funktionstasten 56
logische Kombination
in der Volltext-Suche 122
Logische Kombination
im Schnellzugriff 56
Lokale Ersetzung
Export 197
Import 267
Löschen bei UPD 160
Löschen eines Datenfelds per UPD 160

M
MAB.DPI : Indexparam. für MAB 44
MAB2-Daten umwandeln 252
MAB-Diskette 273
Makro -> Batchdatei 37
Makros: CockPit-Menü 37
Makrosprache FLEX 82

Manipulationsbefehle
Export 189, 194
Import 267
manueller Eingriff
Volltextsuche bzw. Export 121
MARC21 exportieren 80
MARC-Daten umwandeln 252
MARC-Format 141
Markieren eines Satzes (Editor): #m) 111
Markieren von Treffern
-> Ergebnismenge 56
Maskierung
im Suchbegriff bei Volltextsuche 122
von Zeichen bei Ersetzungen (Import) 259
Maskierungszeichen (Export-Manipulationsbefehle) 196
Maskierungszeichen in Manipulationsbefehlen 202
Massenänderung -> globale Änderung 63
Matchcodes als Zugriffsschlüssel 203
Maximale Anzahl Exportsätze (Exp.Param. am) 175
Maximale Dateigröße 178
Maximale Dateigröße >16MB 178
Maximale Länge der Registereinträge 178
Maximalwert für Dateigröße 286
mehrbändiges Werk
Eingabe eines neuen Bandes 106
Erfassungsbeispiel 312
neuen Band zwischenfügen 108
s.a. hierarchisch gegliederte Datensätze 174
mehrbändiges Werk (Begriff) 310
mehrere Schlüssel im selben Abschnitt aufbereiten 203
Mehrfachbesetzung
Exportbehandlung (Exp.Param. ak) 177
von Importkategorien 272
Mehrfachfeld einfügen mit UPD 160
Mehrfachfeld-Kennzeichnung (Imp.Param. fm) 257
Mehrfachkennung für Kategorien 294
Mehrfachkennungen, erlaubte 291
Mehrfach-Leerzeichen
erlaubt / nicht erlaubt 291
Mehrfach-Präfix (Exp.Manip.Bef. m) 201
Mehrplatzbetrieb 50
Datenbank sperren/freigeben 36
MENUED
Start 49
MENUED (Programm) 17
Menüs (a99) 73
Menütexte
Datei 15
Konzept 23
Merge
Funktion von Programm UPD 155
Merseburger Testschleife 236
Mischen
Funktion von Programm UPD 155
Multix (mehrere Indexdateien) 10, 178
MultiX, mehrere Indexdateien 203
mv : Exp.-Sonderkategorie 192

N
nachladen eines Satzes (Export) 210
Neuerwerbungsliste 80, 81, 135
NEWINX: neue Indexdatei 152
Nichtsortiermodus festlegen 295
Nichtsortierwort kennzeichnen (Import) 267
Nichtsortierwörter am Titelanfang prüfen 292
Nichtsortierzeichen 44, 97
bei Ausgabe weglassen 198
definieren 295
Nichtsortierzeichen verdoppeln 198
Nicht-Stoppzeichen 97, 295
NICHT-Verknüpfung
im Schnellzugriff 56
Volltextsuche 122
NMN.API : Indexstruktur für Standardschema 43
Normaltabelle (Umcodierung bei Export) 182
Normdaten
Allgemeines 66
Einheitssachtitel 315
Körperschaftsnamen 314
Personennamen 314
Praxistips 319
Serientitel 315
Systematik 317
Thesaurus 316
Übernahme von Ansetzungsformen 115
Zusatzkategorien 314
nra = Export-Sonderkategorie 192
nummerntreue Erneuerung einer Datenbank 148
Nummernvergabe 293
Nummernvergabe verhindern 297
Nutzervariable: siehe Anwendervariable 199

O
O.APT (Tabelle für ASCII-ANSI-Zuordnung) 90, 185
OAI-Schnittstelle 92
ODER-Verknüpfung
im Schnellzugriff 56
Volltextsuche 122
Offline-Daten (a99) 78
Offline-Speicher (a99) 72
OLDIX: alte Indexdatei 150
online-Erfassung
als Funktion im Schnellzugriff 62
Eingabe eines Datensatzes (Beispiel) 96
op : Export-Sonderkategorie 192
OPAC-Programm 49
Optimieren (Exportparameter) 237
Optionen für Programmaufrufe
-@ : Primär-/sonstige Schlüssel erzeugen 281
-+: Hauptschalter f. Satzübergr. Suche 281
-a : Berechtigungsstufe 281
-b : Datenbankname 281
-c : Sortierposition 154
-d : Datenquelle 281
-e : Export-Parameter und -Datei 282
-f : Funktion 155
-f : Funktionswahl 150, 282
-F : Schnelles/Langsames Updating 283
-I : andere Index-Parameter 283
-i : Import-Parameter 283
-i : Index-Parameter 283
-K : maximale Schlüssellänge 283
-k : Wahl der Konfiguration 283
-L : .LOG abschalten/umlenken 284
-l : Wahl der Sprache 283
-m : manuelles Eingreifen 284
-M : Monochrom (nur CockPit) 284
-n : Dateinummer für neue Daten 284
-N : Speichermodus für Neusätze 284
-O : Bearbeiterkennung 284
-o : CockPit-Vorgabendatei 284

-p : Anzeigeparameter 284
-P : Programmpfad 284
-Q : QRIX-Protokolldatei 285
-q : Wahl der Druckparameter 285
-r : Programmwiederholung 285
-R : UPD-Aktionen aufzeichnen 285
-S : Einzelplatz-Modus 285
-s : Suchbegriff 285
-t : Phrasendatei laden 285
-T : Testmodus 285
-u : Update-Datei 155, 286
-U : Uservariable vorbesetzen 286
-v : Anzeige der Datensätze beim Durchlauf 286
-V : Anzeigezeilen für Register 286
-W : Register weglassen 151
-w,-W : Index-Auswahl (QRIX) 286
-x : Schwelle für Indexanzeige 151, 286
-X : Umfang der Datenbank 286
-x : Update-Protokoll 286
-y : Pfad der Indexdatei 286
-z : Maximalwert für Dateigröße 286
ORDER : Erwerbungsprogramm 19
Ordner --> Verzeichnisse 7
Ordnung : siehe Sortierbare Dateien 132

P
p = Codierbefehl für Normaltabelle (Exp.Param.) 183
p C abcde vwxyz (Protyp-Umwandlung bei Import) 259
p0, p1, p2, pa = Seitenzahlen (Sonderkategorien) 192
Paragraph = Importprogramm-Abschnitt 260
Parameter
nachladen (Export) 168
online ändern 114
online ändern (Export) 235
Parameterdatei
Name (Exp.Param. dn) 182
Suchreihenfolge 13
Typ (.cPR, .cPT, .cPI, .cIM) 13
Parameterspeicher: Größe 295
Parametrierte Vorbehandlung 160
Paßwörter 68
Pauschalexport-Befehl 191
Pauschalimport-Befehl 261
für MARC-Formate 265
pb = Backspace-Liste (Exp.Param.) 185
Personennamen
Invertierung umkehren 208
Normdaten 314
Pflichtkategorien 296
Phrasen
definieren, verwenden, löschen 95, 112
in a99 68
Phrasen in a99 345
Phrasen laden, speichern, drucken (Editor:#p 112
Phrasendatei (Typ .APH) 14
Phrasenspeicher: Größe 295
Phrasenspeicherung
Editorbefehl #p/s 113
von Index-Einträgen 55
Phrasen-Steuertaste
Esc-Taste (DOS) 98
PICA.PPI : Indexstruktur 44
Pica3-Daten umwandeln 252
Plausibilitätsprüfungen bei Eingabe 101, 292
Playback
CockPit-Routine 35
Funktion von Programm UPD 155
Makro 24
Positionierung im Druckbild (Export) 171
Positionierung im Fremdsatz (Import) 264
Postfix
= Export-Textbaustein 169
Abhängigkeit von vorhandenen Kategorien 190
bedingtes 190
direktes (Exp. Manipulationsbefehl PX) 196
indirektes (Exp.Manipulationsbefehl Pz) 201
Konzept 165
Standard (Exp.Param. ke) 187
Postfix bei Import 267
Präfix
= Export-Textbaustein 169
direktes (Exp. Manipulationsbefehl pX) 196
Indirektes (Manipulationsbef. pz) 200
Konzept 165
mehrfach (Exp.Manipulationsbefehl m) 201
Präfix bei Import 267
PRESTO (Programm) 17
Aufruf als Modul 50
Primärschlüssel 127, 158, 159, 177
und Ersetzungsschlüssel 217
Produktions-Parameterdatei 13
Programmaufrufe aus der Datenbank 225
Programmierte Validierung 228
Programmpfad
Option -P 284
Prompt (Hinweistext bei Abfrage) 296
PRONTO 45
PROTOQ (Meldungen von QRIX) 149
Prototypen für Export 13, 161
Beispiele 239
Protyp-Ersetzungen (Import) 259
Prüfroutinen, kategoriespezifische 292
Pseudoschlüssel 217
Punkt am Kategorie-Ende nicht beseitigen (Import) 267
q = Codierbefehl für Alternativtabelle (Export) 183

Q
QRIX 18, 149
nach Unterbrechung fortsetzen 150
QUANT : Statistik-Auswertungen 139
QUEX : einfache Listenproduktion 136
RDWR : Hilfsprogramm 20

R
re = record end (Imp.Param.) 255
Rechenauswertung, Beispiel 245
Rechenbefehle 209
Recherche-Ergebnis -> Ergebnismenge 56
rechtsbündig ausgeben (Exp.Manipuationsbef. r,R) 198
Record End (Satzende) (Imp.Param. re) 255
Record Länge (Imp.Param. rl) 253
Record Start (Satzanfang) (Imp.Param. rs) 255
Redigieren
nächste Aufnahme holen (Editor: #r) 114
REF : Referentenprogramm 19
Referenzierung (Verweisung im Index) 52

Register
  erneuern 151
  erweitertes 58
  gedruckte 142
  löschen 152
  scheinbares (a99) 208
  Überschriften 179
Register für Schnellzugriff : siehe Index 43
Registerabschnitt exportieren 81
Registeranzeige beschleunigen (%) 53
Registerauszüge drucken 152
Register-Bildschirm 51
Registereinträge
  anzeigen (F7) 64
  löschen 55
  maximale Länge 178
Registermanipulation (a99) 208
Register-Präfix
  (Export-Parameter) 203
Registerproduktion (Export) 200
Reihenfolge
  der Kategorien im Datensatz 290
Rekonstruktion einer Datenbank 24
Report (Statistik-Liste) 139
Report, mit Berechnungen 245
Reportgenerator 131
RES (Dateityp) 12
Reservespeicher --> Hintergrundspeicher 62
Reservierte Sprungmarken (Export) 206
Restore (Sicherungskopie einspielen) 24
Restriktionen 231
  benutzen 58
  Benutzung im OPAC 233
  Hilfeseiten 14
Restriktionsdatei
  Dateityp 12
  erneuern 143
Restriktionsdatei erneuern 36
Restriktionsschlüssel 206, 231
Retrieval.über Volltext 119
Return-Taste = Enter-Taste 98
rl = Record Länge (Import-Parameter) 253
Routinen : CockPit-Menü 35
rs = Record Start (Satzanfang, Imp.Param.) 255
RÜCKTASTE
  im Editor: 1 Zeichen löschen 98
  Indexfunktion (Ergebnismenge rücksetzen) 56
RuckZuck (a99 als Server für WWW-Katalog) 92

S
S1.ASP (Zeichentabelle für Volltextsuche) 14, 119
Sacherschließung 310
  Hilfsprogramm REF 19
Sacherschließungs-Stammsätze 317
Sammelwerk (Begriff) 310
Satz = Datensatz siehe Aufnahme 5
Satz gesperrt (Fehlermeldung) 333
Satzanfang (Imp.Param. rs) 255
Satzart -> Satztyp 296
Satzende (Imp.Param. re) 255
Satzlänge (Imp.Param. rl) 253
Satznummern
  Export-Sonderkategorie #nr 192
  Export-Sonderkategorie #nra 192
  Zugriff über die Satznummer 60
Satztabelle
  Dateityp 11
  erneuern 36
Satztyp (Erfassung per Abfrageliste) 296
Satzübergreifende Suche 202, 210, 220
Satzübergreifende Suche (Index-Param.) 223
Satzverknüpfungen 210
Schaltbefehl 213
Schemakennung siehe Konfiguration/Kennbuchstabe 5
Schiller-Räuber-Problem 220
Schlagwort (Begriff) 310
Schlagwortregister, Parameterdateien für 200
Schleifenprogrammierung
  Export 206
  Import 272
Schlüssel -> Registereintrag 51
Schlüssellänge, Maximum 178
Schnellkopplung (a99) 89
Schnellzugriff
  allgemeine Beschreibung 43
  Anzeige-Bildschirm 59
  Programm 17
  Register-Bildschirm 51
Schnittmenge (logisches UND) 56
Schranken (Speicherbegrenzungen) 295
Schreibfeld (a99) 68
Schreibzugriff : Berechtigung (Option -a) 281
Schriftart einstellen (a99) 91
Schriften f. Windows 89
Schwellenwert
  für Indexanzeige 53, 286
Seitenparameter (Export) 180
Seitenumbruch 205
  bei Export 180
  Export ohne S. 181
Seitenvorschub
  Manipulationsbefehl N 202
  Strukturbefehl in Zwischenteilen 172
Seitenvorschub-Steuerung (Exp.Param. sv) 180
Seitenzahl der 1. Seite (Exp.Param. sz) 180
Seitenzahl: Export-Sonderkategorien #p0,#p1,#p2,#pa 192
Sekundäraspekt -> Restriktionen 231
Selektion
  im Schnellzugriff (Ergebnisse exportieren) 64
  Makrobefehl 287
  mittels Volltextsuche 119
  mittels Volltextsuche, Aufruf 123
Sequenzen-Ersetzung 178, 183
Serientitel : Normdaten 315
SET-Variable -> Umgebungsvariable 280
SGF (Dateityp) Signaldatei 15
Sichern einer Datenbank
  CockPit-Funktion 35
Sicherungsdatei (Typ .LOG) 14
Sicherungskonzept 24
Sicherungskopie einer Datenbank 24
Sicherungsprotokoll
  abschalten/umlenken 284
Signaldatei 15
Simultanzugriff auf 2 Datenbanken 52
Sitzungsablauf (a99) 80
Skriptsprache FLEX 82
SNIFFER
  Datenbank-Diagnose 20
Sonderkategorie
    #uxq
      Antwort auf letzten q-Befehl 193
  #cc : aktuelle Kategorie 191
  #cca : aktuelle Kat. Incl. Nummer 191

#ch : aktuelle Kopfkategorie 191
#dt : Datum in Kalenderform 191
#dts : Datum/Uhrzeit in Sortierform 191
#fn : Dateiname 191
#gt : Gesamttitel 193
#hi : Hierarchie-Gliederung 193
#mv : multi-volume indicator 192
#nr : Satznummer 192
#nra : Satznummer nachgel.Satz 192
#op : Bearbeiter 192
#p0,#p1,#p2,#pa : Seitenzahl 192
#pz0,#pz1,#pz2 Zeilenzähler 192
#pz0/1/2 Zeilenzähler 192
#pzp Pos. in d. aktuellen Zeile 192
#u00...
Anzahl Kategorien des aktuellen Satzes 193
#u01, #u02, #u03...
Erste, nächste, letzte Kat. d. aktuellen Satzes 193
#u1/#u2 : Kopfkategorien 192
#ui0
Letzte Nutzereingabe im Register 193
#uxa
Indexzeile, von der zugegriffen wurde 193
#uxb
Nummer des aktuellen Index 193
#uxc
Kurzzeile, über die zugegriffen wurde 193
#uxd
Nr. der aktuellen Datenbank 193
#uxi
Indexzeile des letzten Nachladens 193
#uxy : Anwendervariable 192
Sonderzeichen 343
a99, Eingabe) 77
als Phrase (a99) 345
DOS-System, Eingabe 97
eliminieren (Export) 183
im Suchprogramm ignorieren 14
im X-Editor eingeben 339
in Suchbegriffen bei Volltextsuche 124
Sortierbare Dateien, Konzept und Erstellung 132
Sortieren von Zahlen 153
Sortieren, absteigend 154
Sortierparameter, Prototyp 241
Sortierprogramm 18
Aufruf 154
Sortierte Listen 81, 132
Sortierwert eines Zeichens ändern 185
Speicher konfigurieren 295
Speicherbedarf 238
Speicherbefehle (Editor) 103
Speicherzustand anzeigen (Alt+F7) 64
Sperre aufheben
Datenbank 36
Datensatz 333
Sperrung/Entsperrung
einer Datenbank 36
Sprache der Menütexte 23, 283
Sprache wechseln 64
Sprungbefehl
Abfrageliste 296
Export, bedingter S. 189
Export, unbedingter S. 205
Import, bedingter 271
Import, unbedingter 272
Sprungmarke
Abfrageliste 296
Export 205
im Kopfbefehl 176
Import 272
Sprungmarken
dokumentieren 20
reservierte, für Index 206
Sprungverteiler (Export) 238
SQL 92
SRCH 119
Aufruf im Batch 123
Volltext-Suchprogramm 18
Stammsätze 314
Stammsätze -> Normdaten 314
Stammsatz-Verknüpfungen 215
Standard-Abfragen (Dateneingabe) 296
Standard-Postfix (Exp.Param. ke) 187
Stapeldatei = Batchdatei 15
Startposition im Kategorietext (Exp.Param. ks) 187
Statistik-Listen 139
Steuerbefehle
Export 204
Import 271
Steuerzeichenfolge
Export 170
Import 253
Stichwort (Begriff) 310
STL (Dateityp) 11
Stoppbefehl (Export) 206
Stoppbefehl (Import) 272
Stoppbefehl in der Kategorieliste
Import 272
Stoppwort-Tabelle 44, 179
Strukturbefehle (Export) 170, 171
Stücktitel (Begriff) 310
Stücktitelaufnahme 175
Gesamttitel-Angabe (Sonderkategorie #gt) 193
Subfields bei Export -> Teilfeld ausgeben 198
Such- und Ersetzungsbefehl
bei Export 186
bei Import, global 258
bei Import, lokal 267
Editor-Befehl #_ 118
global (F10) 63
Suchbefehl (Editor:#f) 106
Suchbefehl (Import) 266
Suchbefehlsfenster (a99) 70
Suchbegriff
bei Volltextsuche 122
Suchbegriff des Nutzers umcodieren 178
Suchbegriffsliste
für Schnellzugriff -> Index 43
Suchen und Ersetzen (Editor: #_) 118
Suchprogramm (Volltextsuche) 18, 119
Suchprogramm-Sonderzeichentabelle (S1.ASP) 14
Suchreihenfolge
Parameterdatei 13
Suchreihenfolge Parameterdateien 13
Suchvorgang unterbrechen (Volltextsuche) 122
Summierung von Feldinhalten 81
sv = Seitenvorschub-Steuerung (Exp.Param.) 180
Symbolische Registernamen 178
Systematik : Normdaten 317
Systemdaten
für Ausleihe u. Erwerbung 323
sz = Seitenzahl der 1. Seite (Exp.Param.) 180

T
t : Export-Parameter nachladen 168
Tabelle zum Nachladen (Dateityp .cPT) 13
Tabellen erstellen (a99) 81
TAB-Taste : externe Funktion aufrufen 60
Tabulatorsprung (Export) 171
tail : Endstück einer Kategorie 197
Tastenfunktionen (a99) 77
TBL (Dateityp) 11
Teilfeld
aktualisieren per UPD 157
ersetzen/löschen per UPD 160
löschen per UPD 157
Teilfeld ausgeben (Export) 198
Teilfeld beseitigen (Export) 198
Teilfeld-Abfragen 297
Teilfeld-Eingabe/-Abfrage 297
Teilfeld-Trennzeichen 97
Code festlegen 293
eingeben 339
Testhilfen
Export 235
F7 für Indexparameter 64
Texel-Ersetzung 183
Textbausteine (Export) siehe Zwischenteile 169
Texteditor 3
Texteinfügung (Exp.-Steuerbefehl #tZ) 204
Textersatzdarstellung (Export) 184
Textzeiger (Import) 264
Thesaurus
Normdatensätze 316
Titel des Katalogs 179
Titelanzeige-Funktionen 59
Titelaufnahme
= Katalogisierung 310
Titelaufnahme = Datensatz 5
Titelstichwörter (Zerlegung des Titels) (Exp.Param. ak) 176
Transaktionssicherung : .LOG-Datei 14
Transferbereich (Import) 267
Transferprogramm siehe Import 127
Treffer -> Ergebnismenge 56
Trefferzählung 51
Trennzeichen für Mehrfacheinträge 97
Trennzeichen für zweiteilige Schlüssel (Exp.Param. i1) 178
trim : Hinterende abschneiden 197
truncator -> Endemarke 54
Trunkierung
aufheben (expandieren) : F7 54
mit fester Position 54
variable (mit Endemarke) 54
Trunkierung, Windows-Programme 69
Typ A-E (Fremddaten) 249
Typ einer Datei 5
Typ-Abfrage bei Dateneingabe 296
u1 und u2: Anwendervariable 176
Übergehwörter herausnehmen (Exp.Manipulationsbefehl u) 198
Übernahmeschlüssel 217
Überschrift der Titelanzeige 179
Überschriften für die Register 179
Übersetzung der Menütexte : siehe Sprache 23
Uhrzeit siehe: Datum 95, 112
UIF-Dateien 23
UIF-Dateien (Menütexte) 15
Umbruchcode (bei Export) 205
Umcodierung
Export 182
Import 258
Umcodierung abschalten/erzwingen 198
Umcodierung der Benutzereingabe 178, 206
Umgebungsvariable 280
Unbedingte Texteinfügung (Exp.Steuerbef. #tZ) 204
unbedingter Sprung (Export) 205
UND-Verknüpfung
im Schnellzugriff 56
Volltextsuche 122
ungültige Endzeichen (Indexeintragungen, Exp.Param. i3) 178
UNICODE 92, 186
Unicode Entitätscodes 178
UNIX : Portabilität 42
UNIX<->DOS Zeichensatz 185
Unselbständiges Werk (Begriff) 310
Unterbrechung
Import 126
Volltextsuche 122
Unterdrückung führender Nullen etc. 197
Unterfeld Siehe Teilfeld
Unterfeld-Trennzeichen
Code festlegen 293
Unterprogramm-Aufruf
im Exportbefehl (bedingter UP-Aufruf) 189
Unterprogramme
Export 207
Import 270
Untersatz -> Bandaufführung bei mehrbd. Werken 5
Untersatz löschen 312
Untersätze (Import) 255, 268
Untersätze aktualisieren per UPD 160
UPD (Programm) 18
UPD : Aufruf 155
UPD oder INDEX zum Einmischen? 127
Update
Beschleunigung/Verlangsamung 155
Datenbank-Rekonstruktion 24
Update-Datei : Option -u 286
Urheber (Körperschaft) 310
User Interface siehe Benutzeroberfläche 15
USMARC : Indexstruktur 44
UTF-8 (Unicode) 186

V
Validierung -> Eingabeprüfung 228
Variabel lange Sätze (Import) 255
Variable (Export) -> Anwendervariable 199
Verbotszeichen für Indexeintragungen (Exp.Param. i2) 178
Vereinigungsmenge (logisches ODER) 56
Vergessen siehe Löschen 115
Vergleichsbefehle (Import) 271
Vergleichsoperatoren (Volltext-Suche) 122
Verknüpfungen zwischen Sätzen 210
Verknüpfungs-Steuerzeichen 215
Verlangsamung des Updates 155
Verschachtelung von Zwischenteilen 171
Verschiebung eines Satzes im Arb.Speicher
Editor-Befehl #m 111
Vertauschen von Bandaufführungen (Editor: #m) 111
Verweisungen
aus Stammsätzen 217
Verzeichnisse 7
ViewListen 15
View-Technik (a99) 81

Volltextsuche
  Aufruf im Batch 123
  aus a99 starten 81
  in Ergebnismenge (a99/alcarta) 71
  Konzept 119
  Programm SRCH 18
  Programmablauf 120
Vorbearbeitungsbefehle
  Export 189, 194
  Import 267
Vorgabendatei (CP.OPT) 40
Vorgespiegeltes Register 208
Vorlage (Begriff) 310
.vw
  Dateityp 15

Web-Anbindung 92
Weglassung von Kategorien
  bei Pauschalexport 191
  bei Pauschalimport 261
Werk (Begriff) 310
wh : Export-Steuerbefehl 180, 205
Wiederherstellen einer Datenbank 24
  CockPit-Funktion 35
Wiederholungsbefehl für Mehrfachkategorien (Export) 189
Wiederholungsfaktor in Steuerzeichenfolgen (Export) 170
Wiederholungsfeld -> Mehrfachfeld 294
Wildcard (Import) 259
Windows Datenanzeige (RTF) 173
Windows<->DOS Zeichensatz 185
Windows-Programme 67
Write-Befehle für direktes Schreiben (Import) 267
wrong recn (Fehlermeldung) 333
WWW-Schnittstelle 92
  Ausgabeparameter 173

X
XML 92
XML-Ausgabe 80
XML-Export 81

Y
y x k (Import-Zeichenumwandlungsbefehl) 258

Z
Zahlen korrekt sortieren 153
Zählungen von Feldinhalten 81
zb = Zeilenbruch (Exp.Param.) 181
ZDB-Daten Import 273
ze = Zeilenende-Steuerzeichen (Exp.Param.) 182
Zeichen mit Sonderbedeutung, DOS-Eingabe 97
Zeichenfolgen-Ersetzungsbefehl
  Export 186
  global (Schnellzugriff) : F10 63
  Import, global 258
  Import, lokal 267
Zeichenfolgensuche siehe Volltextsuche 43
Zeichensatz
  DIN 31628 342
  Hilfsprogramm AW 20
Zeichenumwandlung (Umcodierung)
  Export 182
  Import 258
Zeiger im Index 51
Zeilenbruch-Zeichen (Exp.Param zb) 181
Zeileneinrückung (Exp.Param. zi) 181
Zeilenende-Steuerzeichen (Exp.Param. ze) 182
Zeilengruppe (Import) 270, 271
Zeilenlänge (Exp.Param. zl) 181
Zeilenlänge, Änderung (Export) 171
Zeilenmaximum je Seite (Exp.Param. zm) 181
Zeilen-Parameter (Export) 181
Zeilentrennzeichen (Export) 185
Zeilenumbruch
  erzwingen 185
  Export ohne Z. 181
  in Fremddaten beseitigen (Import) 258
Zeilenvorschub
  Manipulationsbefehl C 201
  Strukturbefehl in Zwischenteilen 172
Zeilenvorschub-Auslösung : Absatzendezeichen 97
Zeilenwechsel erzwingen (Export) 185
Zeitbedarf 237
Zeitstempel s. Erfassungsdatum 293
zi = Zeilen-Einrückung (Exp.Param.) 181
zl = Zeilenlänge (Exp.Param.) 181
zm = Zeilenmaximum je Seite (Exp.Param.) 181
Zugriffsberechtigung 281
Zugriffsfunktionen im Schnellzugriff 60
Zugriffsmethoden 43
Zusammensetzung von Zeichenfolgen (Export) 200
zweiteilige Schlüssel, Trennzeichen 178
Zwischenteile (Export) 169