allegro news Nr.54

Ausgabe 99/2, 28. Juni 1999
Diese Ausgabe hat 16 Seiten

Universitätsbibliothek der Technischen Universität Braunschweig, Pockelsstr. 13, D-38106 Braunschweig. Tel. (0531) 391-5011, FAX -5836

www.allegro-c.de/allegro

Summer, this season of splendour and spice,
what riot of colour and scent!
Circumstances
for romances...
No one cares where Winter went,
Yet now and then one longs for ice.

ReFLEXionen über FLEXibilität

Das ständige Wechselspiel der Jahreszeiten in unseren Breitengraden ist nicht jedermanns Sache. Neben allem anderen, was man um die Ohren hat, muß man sich auch noch *darauf* immer wieder einrichten! Besonders Frauen sind bekanntlich die Leidtragenden, wenn sie z.B. feststellen müssen, für den Sommer absolut nichts Tragbares mehr im Schrank zu haben. Und das eine oder andere eigentlich beliebte Teil kann man nur deshalb nicht im Sommer anziehen, weil es farblich oder im Stil nicht mehr paßt. *Muß* das wirklich so sein? Ist nicht ein Komponentensystem denkbar, das zu jeder Saison ein flexibles Kombinieren von Elementen ermöglicht, deren jedes, für sich genommen, nicht auf eine bestimmte Jahreszeit festgelegt und auch nicht starr an eine bestimmte Größe, Altersgruppe oder Geschlecht gebunden ist? Vielleicht müssen einige hergebrachte oder anerzogene Vorstellungen von Ästhetik hinterfragt und letztlich aufgegeben werden, aber der Wegfall der Frustration, der geringere Bedarf an Schrankraum und der enorme Gewinn an Flexibilität wögen das wohl auf. Solchem Fortschritt steht allerdings so manches entgegen - wenn auch fast nur Gründe, die mit der Sache an sich überhaupt nichts zu tun haben...

Der ständige Wechsel zwischen ganz verschiedenartigen Systemen in der Bibliotheksarbeit sagt auch nicht jedem zu. Bei all den Wechselfällen, mit denen einen das Privatleben dauernd konfrontiert, will man doch wenigstens bei der *Arbeit* ein wenig Stabilität! Besonders beim Katalogisieren fällt es bekanntlich oft auf, daß neue Aufgaben auftreten, für die unter dem erworbenen Rüstzeug kein passendes Instrument zu finden ist. Das eine oder andere eigentlich beliebte Werkzeug ist plötzlich unbrauchbar, weil eine bestimmte Wirkung damit nicht erzielbar oder oder eine Nebenwirkung in einem neuen Zusammenhang unerwünscht ist. *Muß* das wirklich so sein? Ist nicht ein Komponentensystem denkbar, das in jeder Situation ein flexibles Kombinieren von Elementen ermöglicht, deren jedes, für sich genommen, nicht auf eine bestimmte Aufgabe festgelegt und auch nicht starr an eine bestimmte Sparte, Größenordnung oder Abteilung gebunden ist? Vielleicht müssen einige hergebrachte oder angelernte Vorstellungen von Angemessenheit hinterfragt und letztlich aufgegeben werden, aber der Wegfall der Frustration, der geringere Bedarf an Speicherplatz und der enorme Gewinn an Flexibilität wögen das wohl auf. Solchem Fortschritt steht in der Tat heute weniger entgegen als je zuvor - allenfalls Gründe, die mit der Sache an sich nichts zu tun haben.

Zwei grundverschiedene Bereiche, ganz ähnliche Phänomene! Im Bekleidungswesen noch Utopie, im Bibliothekswesen aber längst Realität: das flexible Komponentensystem. Eine unflexible Bibliothekssoftware könnte heute gar nicht überleben. Im Falle ***allegro*** war Flexibilisierung von Anfang an immer wieder Prinzip und Herausforderung zugleich. Das zeigt sich in den Möglichkeiten der **Konfigurierung** (= Anpassung für ein beliebiges Kategorienschema) und **Parametrierung** (= Realisierung von individuellen Indexierungen, Ausgabeformen und Importen). Dadurch gibt es kein für alle verbindliches Datenschema, keine Einschränkung auf RAK oder MAB, keine Kompromisse bei den Zugriffsregistern, keine Festlegung auf bestimmte Materialien oder Aufgabenbereiche. Anwender aus allen Sparten und Arbeitsbereichen des Bibliothekswesens wurden nicht müde, neue Anforderungen zu formulieren, aber auch, neu geschaffene Möglichkeiten ausgiebig zu testen und in neuen Projekten wieder bis an die Grenzen auszureizen. Das gilt für die neue Generation der Programme (die C^{++}-Klasse) nicht weniger als für die "alte" DOS-Generation (die C-Klasse). Mittlerweile ist, schon bald nach Verabschiedung der ***V16***, ein weiterer Meilenstein erreicht: die Windows-Programme ***alcarta*** und ***a99*** haben nicht nur eine Reihe von ganz neuen Funktionen, sie besitzen nun auch so gut wie alle Fähigkeiten der alten Programme PRESTO und APAC, so daß einem Umstieg (Win'95/98 oder NT vorausgesetzt) höchstens noch die Scheu vor einer Umgewöhnung im Wege steht. Neue Mode verlangt manchmal auch Mut, doch kann der Umstieg zu jeder Jahreszeit erfolgen, und bis dahin und auch danach noch kann man die DOS-Programme gleichzeitig damit kombinieren. Das tun gerade auch Experten - wie man im Sommer zuweilen nach Eis verlangt, ohne sich gleich den Winter zurückzuwünschen. An den Datenbanken ändert sich nichts! Die neuen Programme geben ihnen ein neues Aussehen und bieten neue Möglichkeiten des Umgangs mit den Daten, das ist der Punkt. Auch Flexibilität hat aber ihren Preis: Das Konfigurieren und Parametrieren ist kein Nebenbei-Job, sondern man muß echte Zeit investieren, wenn man etwas Neues verwirklichen will. Aus den vielen schon durchgeführten Projekten kann man inzwischen jedoch sehr oft Komponenten übernehmen oder braucht sie nur noch leicht zu modifizieren. Kenntnisse in der Parametrierung bleiben wichtig, deshalb werden in dieser Nummer eine Reihe von Tips zur Einarbeitung in dieses weite Feld gegeben. Wer erst die Grundkenntnisse besitzt, kann sich aus einer Vielzahl von fertigen Modellen bedienen, die auf dem FTP-Server bereitstehen. Mindestens 15 unterschiedliche Komplettpakete kann man sich per WWW herunterladen. Darin sind neben den Beispieldaten alle Parameter und Hilfedateien etc. enthalten, die zum Funktionieren des Modells gebraucht werden. Was zum Installieren der neuen Programme nötig ist, stellen die nachfolgenden Seiten knapp zusammen.

Die Flexibilisierung geht nun aber noch einen Schritt weiter: Unter den neuen Funktionen ist jetzt auch so etwas wie eine Makro-Programmierung zu finden. Es gab ja bereits die Flips, die schon einer DOS-Datenbank so etwas wie Hyperlink-Komfort verleihen. In der Windows-Anzeige können Flips so gestaltet werden, daß sie aussehen und funktionieren wie WWW-Links. Nun aber kann sich hinter einem Flip noch viel mehr verbergen als vorher: ein umfangreiches, maßgeschneidertes Programm kann dadurch ausgelöst werden - ähnlich wie ein Makro. Solche flexibel kombinierbaren Komponenten nennen wir **FLEX**e. Da man FLEXe sogar von außen aktivieren kann, wird ***a99*** zu einem neuartigen Server. Diese Ausgabe soll alle FLEX-Befehle dokumentieren und Beispiele für den Einsatz zeigen. Das FLEXen dürfte jetzt sehr schnell in Mode kommen...

Installation von *a99* und *alcarta*

Um noch bestehende Unklarheiten zu beseitigen und um dem eiligen Systemverwalter eine ganz knappe Checkliste an die Hand zu geben, stellen wir hier alle unbedingt zu beachtenden Einzelheiten zur Installation der neuen Programme zusammen.
Eine vollständige Liste aller benötigten Dateien wurde schon in der Nummer 53 veröffentlicht, diese Liste ist unter dem Namen FILELIST.TXT weiter aktualisiert worden und auf dem FTP-Server vorhanden. Diese recht lange Liste ist jedoch ein etwas abstraktes Nachschlagewerk und beschreibt etliche Dateien, um die man sich nur selten kümmern muß.

Wer erst jetzt neu beginnt, entpackt am besten `a99.lzh` und `alcarta.lzh` auf sein vorhandenes ***allegro***-Programmverzeichnis. Diese Dateien findet man auf den FTP-Verzeichnissen `ac15\a99` bzw. `ac15\alcarta`.

Als **Beispiel** nehmen wir folgendes an: (setzen Sie hier Ihre eigenen Werte ein)

Konfiguration **K**
Datenbank **XYZ** (d.h. es gibt eine Datei `xyz.kpi`)
Verzeichnis **D:\DATA**

SEHR WICHTIG

Auf jedem PC, der mit ***alcarta*** oder ***a99*** arbeiten soll, müssen die Schriften (.TTF Dateien) installiert werden. Die *Software* darf zentral auf einem Fileserver liegen und von dort gestartet werden, nur die *Schriften* müssen auf jedem PC liegen und darauf ordnungsgemäß installiert werden, sonst bekommt man keine korrekte Anzeige (Fenster zu groß, falsche Schrift).

0. INI-Datei

Eine INI-Datei ist anzulegen. Man kopiert dazu `a99.ini` auf `xyz.ini` und ändert darin mindestens folgende Befehle:

```
Konfiguration=k      (default: a)
DbName=xyz
DbDir=d:\data
access=3       (volle Schreibberechtigung in a99)
```

Bei Mehrplatzbetrieb muß jeder User auf einem eigenen Verzeichnis starten, die INI-Datei kann aber, wie alle Dateien, zentral liegen. Auf dem Userverzeichnis werden einige temporäre Dateien und die Phrasendatei PHRASE.A99 untergebracht.

1. IndexParameter

1.1 Symbolische Registernamen (Handbuch 10.2.1.3, S. 176 und ***avanti***-Dokumentation)

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 die Suchmaske, die man mit dem [Find]-Button erscheinen läßt. (*Achtung:* für Register 10 `:` statt 10)

1.2 Überschriften der Register, der Titelanzeige und der Kurzanzeige (Handbuch 10.2.1.3, S. 177)

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 Umcodierung ASCII←→ANSI

An das Ende der Indexparameter: (wird gebraucht für korrekte Anzeige der Register und Kurztitel etc.)

`to` zum Laden der O.APT (wird sonst aber defaultmäßig gemacht)

Man muß aber eine eigene `o.kpt` schaffen, wenn man nicht OSTWEST.FON verwendet oder nicht den PC-Satz 437 (siehe Kommentar in `o.apt`). Sonst wird automatisch `o.apt` geladen. Um das zu *verhindern* (z.B. wenn die Datenbank selbst ANSI ist, oder bei den Sinologischen Anwendungen) muß man eine leere `o.apt` auf's Datenverzeichnis legen.

2. Hilfsparameter (werden mitgeliefert)

`d.kpt` ASCII → ANSI Umcodierung für Titelanzeige (normalerweise Kopie von d.apt)
`d-rtf.kpt` RTF-Attribute für die Anzeige (incl. Farben) (Kopie von d-rtf.apt)
`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 `d.apt` etc. stehen Kommentare). Die ersten zwei muß man modifizieren, wenn man nicht mit dem OSTWEST-Zeichensatz arbeitet, sondern z.B. mit Pica oder ANSI (dem Windows-Originalzeichensatz).

3. Anzeigeparameter Gebraucht wird 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.APR. Die wichtigen Stellen sind dann folgende:

3.1 Grundparameter

```
zm=0
zl=70   Zeilenlänge (mehr oder weniger möglich, je nach Schriftart)
fl=0
dx=1
    ke=""  vermeiden (unter DOS sowieso ohne Wirkung)
```

3.2 Einbindung vonTabellen

Am Ende der Datei sollten diese 2 Zeilen stehen (siehe 2.):

`td-rtf` enthält Farbbefehle (Zwischenteile 71-78) und Befehle für Schriftattribute
`td` Zeichencodes ASCII → ANSI für die Anzeige

3.3 Abschnitt für Alternativanzeige: (mit Alt+z oder über Menü "Ansicht")

Folgenden Abschnitt baut man in die `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 }
#+#
```

3.4 Zwischenteile

Wenn darin oder in indirekten Prä-/Postfixen (`p{...}` oder `P{...}`) Umlaute vorkommen, erscheinen diese falsch.
Abhilfe: die Umlaute in der D-WRTF.KPR durch ANSI-Codes ersetzen (z.B. mit NOTEPAD).

3.5 Schriftart

Wenn eine spaltenrichtige Anzeige wichtig ist, baue man diese Zeile ein:

```
as=" { \f2 "
```

dann wird Courier als Schriftart genommen. Die zu verwendenden Schriftarten lassen sich in DISPHEAD.RTF aber individuell vorgeben und im Text jeweils mit `#t{ "\fn " }` einstellen und mit `#t{ "}" }` abstellen, wobei n die Nummer einer der in DISPHEAD.RTF definierten Schriften 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-drtf.kpt`.

5. Hilfeseiten

Dazu gibt es eine eigene Anleitung: HELP.TXT. Für die eigene Datenbank empfiehlt sich, mindestens folgende anzulegen:
`DbNameGER.RTF` mit WinWord anzulegen (Vorbild: CATGER.RTF, Demodatenbank)
`DbNameENG.RTF`
`HA_InGER` ASCII-Dateien für die einzelnen Register, n=1..9

Die Dateien HELP.TXT und INPUT.TXT beschreiben auch, wie man Flips in Hilfedateien einbaut. 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***. Wo das Programm a99.exe bzw. alcarta.exe selbst liegt, ist dabei nicht wichtig. Sinnvoll ist, es im normalen Programmverzeichnis unterzubringen.

`a99`	`a99.ini` wird genommen
`a99 name`	`name.ini` wird geladen (Einen solchen Befehl kann man in eine Desktop-Verknüpfung einbauen)
`a99 0`	Man kann nach dem Start eine INI-Datei suchen und auswaehlen
`a99 1`	`a99.ini` wird gelesen, dann Auswahl einer Grunddatei
`a99 2`	Suche und Auswahl einer Datenbank `(.?DX)` ohne INI

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. Jetzt geht es noch weiter: die Hilfetexte können wiederum Flips und jetzt 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. Mitgeliefert wird HELP.TXT mit ausführlicher Beschreibung der Möglichkeiten.
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`, also in sinnvoller Weise anders als bei Parametern: zuerst Lokal.
Noch eine Möglichkeit: eigene Datensätze mit Hilfetexten als Teil der Datenbank. Auch solche können dann wieder Flips zu anderen Hilfethemen enthalten! In beispielhafter Weise wurde dieses Konzept umfassend realisiert für den Theologischen Zeitschriften-Inhaltsdienst der UB Tübingen, der soeben auf CD-ROM mit ***alcarta*** ausgeliefert wurde. Eine Demo-Version davon liegt auf dem Download-Verzeichnis unter ZID (`www.allegro-c.de/allegro/alcarta/download.htm`).

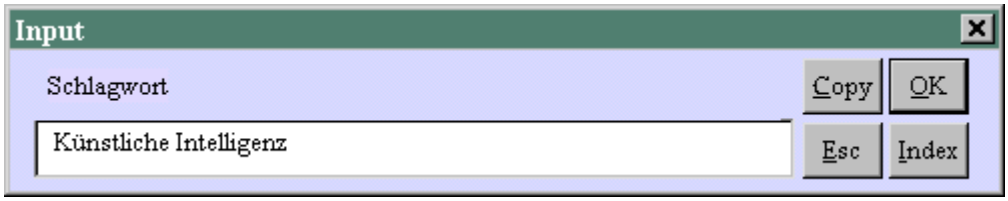
Was ist ein FLEX? (Es heißt *der* FLEX, nicht *das* FLEX)

Knapp gesagt: eine automatisierte Folge von Aktionen, die man auch hintereinander mit der Hand auslösen könnte. Bei anderen Softwaresystemen wird so etwas meistens **Makro** genannt. Buttons und Menüpunkte aktivieren, Suchbefehle eintippen, Datenfelder eingeben - das sind alles Einzelschritte, die man von Hand durchführen kann, die nun aber auch automatisch hintereinander ablaufen können. Es muß dafür natürlich Befehle geben, die irgendwie und irgendwo aufgezeichnet und dann jederzeit bei Bedarf ausgelöst werden. Wie das geht, wird nun ausführlich dargestellt.
Achtung: Was nachfolgend beschrieben wird, funktioniert noch nicht oder nicht alles mit der Version auf der CD! Man muß sich mit FTP die aktuelle Version besorgen, mit Datum ab 28.6.1999.
Beginnen wir mit einem Beispiel, das schon mehrere Möglichkeiten zeigt, insbesondere die Bildung einer Schleife. Man kann es als Muster für ähnliche Aufgaben verwenden.
Aufgabe: In vielen Datensätzen soll jeweils die Kategorie #31 bearbeitet werden, jedoch geht es nicht mit einer globalen Ersetzung, sondern es muß von Fall zu Fall etwas anderes gemacht werden. Ziel ist eine Prozedur, die mit einem Minimum von Tastendrücken auskommt. Und so wird's gemacht:
Im **Schreibfeld** (unter dem Anzeigefenster) gibt man folgendes ein (schreiben Sie es genauso ab, wie es hier steht):

```
#uX6x ask |3Schlagwort=#31\insert #31\next\repeat
```

Wenn man nun eine Ergebnismenge hat und läßt sich den ersten Satz anzeigen, dann **Alt+6**. Damit wird der FLEX #uX6 aktiviert. Er besteht aus 4 Befehlen: (Der Anfangsteil "#uX6x " bedeutet nur: speichere das, was jetzt kommt, als FLEX unter dem Namen `#uX6`. Statt 6 kann jede andere Ziffer genommen werden.) (*Tip:* \ zweimal drücken, wenn es nicht kommt)

`ask |3Schlagwort=#31`
Inhalt von Kategorie #31 in einem Eingabefenster zum Bearbeiten bereitstellen.
Das sieht so aus, wenn in #31 z.B. gerade "Künstliche Intelligenz" steht:



Bei Druck auf [Index] wird zum Register 3 gesprungen! Das bewirkt die Angabe |3 im ask-Befehl.
Bei [Enter] oder [OK] kommt der Inhalt des Eingabefeldes in einen Zwischenspeicher (die "interne Variable")

`insert #31`
Die bearbeitete Form wieder in #31 speichern (d.h. aus dem Zwischenspeicher in #31 kopieren).
`next`
Zum nächsten Satz der Ergebnismenge übergehen
`repeat`
Wiederholung der Befehlskette. Das läuft bis zum Ende der Ergebnismenge; durch eine leere Eingabe kann man die Schleife unterbrechen, danach mit Alt+6 weiterlaufen lassen.

Die geänderten Datensätze werden nicht sofort gespeichert, sondern zuerst im Offline-Speicher aufbewahrt. Daher kann man auch die Änderungen leicht wieder rückgängig machen. (Über Menü "Extras" oder einzeln mit [Wechseln].)
Soll jeweils mehr als eine Kategorie bearbeitet werden, kann man mehrere `ask...\insert...` Kombinationen einbauen.
Bis zu 10 FLEXe kann man anlegen (#uX0 ... #uX9). Diese werden in der Phrasendatei PHRASE.A99 gespeichert und bleiben deshalb für nachfolgende Sitzungen erhalten, bis man sie ändert. (Die Phrasendatei ist auch manuell änderbar.)
So **ändert** man einen vorhandenen FLEX: Mit Alt+r umschalten auf den Reservespeicher (Hintergrundspeicher), dann sieht man die #uX-Kategorien. Man wählt diejenige an, die man ändern will, drückt [Enter] und hat den FLEX im Schreibfenster. Nach Änderung: mit [Enter] wieder speichern, mit Alt+r zurück zur Titelanzeige.
Hinweis: Bei PRESTO schaltet Alt+1 immer um auf Register 1 usw. Das ist bei ***a99*** nicht so, sondern funktioniert nur, wenn ein Indexfenster aktiv ist. (Den Index holt man mit Alt+i wieder hervor, oder Druck auf [Index].)
Ein FLEX kann auch mit Parameterbefehlen eingerichtet werden. Um das obige Beispiel einzurichten, müßte in den Anzeigeparametern irgendwo stehen:

```
#dt p" x ask |3Schlagwort=#31\insert #31\next\repeat!" e"!" =X6 e0
```

Wer will, kann auch noch einen (im Anzeigefenster sichtbaren) Flip zum Anklicken erzeugen. In D-WRTF.APR baut man ein:

`#t{ C t69 "Schlagwort bearbeiten" t68 }`	Flip anzeigen, blau und unterstrichen
`#dt p"Schlagwort bearbeiten!" e"!" =Y6`	Denselben Text in #uY6 speichern

Man sieht dann in der Anzeige in blau und unterstrichen Schlagwort bearbeiten, und mit Doppelklick auf diese Angabe wird der FLEX ausgelöst: Das Programm stellt fest, daß #uX6 der zu #uY6 gehörige Befehl ist. Wenn es #uX6 nicht gibt, wird #uZ6 genommen (d.h. #uX6 hat Vorrang).
Alternative: Längere FLEXe speichert man besser in einer Datei von Typ .FLX. Dies ist eine ASCII-Datei, die man unter DOS bearbeiten sollte, jedenfalls müssen die Sonderzeichen (Umlaute etc.) in DOS-codiert sein.
Wenn man also eine Datei **SW.FLX** anlegt, in der steht (derselbe Text wie oben, aber jeweils neue Zeile statt '\')

```
ask |3Schlagwort=#31
insert #31
next
repeat
```

dann muß man eine #u-Kategorie haben mit dem Inhalt `#uX6X sw.flx`. Das *große* 'X' sagt dann dem Programm: suche die Datei mit Namen `sw.flx` , lade sie und führe sie als FLEX aus. Die Wirkung bei Alt+6 ist dieselbe.

Es folgt die Liste aller Befehle, die man beim FLEXen einsetzen kann.

FLEX-Befehle

Wichtig: es gibt eine **interne Variable** (kurz **iV**). Das ist ein temporärer Zwischenspeicher, der nur solange lebt, bis er durch einen weiteren Befehl überschrieben wird. Das geschieht durch die Befehle "ask", "date" und "variable".
Die iV hat und braucht keinen Namen, das vereinfacht einiges. Man verwendet die interne Variable in der Regel sofort in einem "insert"-Befehl. Will man die iV länger aufbewahren: mit `insert #uxy` in die Uservariable #uxy kopieren. (Und mit `variable #uxy` von dort wieder zurück in die iV.)

`#xyz`*Text*

Die Kategorie `#xyz`*Text* wird in den aktuellen Satz eingefügt. Wenn *Text* leer ist, wird `#xyz` gelöscht.
#xyz kann auch eine Nutzervariable #uxy sein.
Wenn der Text auf kompliziertere Weise zusammengesetzt werden soll, muß man zuerst mit Hilfe des Befehls `variable ...` den Text in der internen Variablen erstellen, dann mit `insert #xyz` abspeichern.

`$a+#xyz`*Text*

Das Teilfeld ▼a mit Inhalt *Text* wird an die Kategorie `#xyz` angehängt. Wenn #xyz noch nicht da ist, entsteht sie.
Statt '$' kann das Teilfeld-Dreieck verwendet werden. z.B.
`▼u+#90 Meier` ▼uMeier an #90 anhängen

`$a-#xyz`*Text*

Wenn Teilfeld ▼a existiert, wird es ersetzt, sonst wird ▼a*Text* an `#xyz` angehängt.
Wenn *Text* leer ist, wird das Teilfeld ▼a aus `#xyz` beseitigt.

`*#nnn_ABC_XYZ_` Ersetze in Kategorie #nnn die Zeichenkette "ABC" durch "XYZ"

`_ABC_XYZ_` Ersetze die Zeichenkette "ABC" im gesamten Satz durch "XYZ"

`? |i`*abc*

Register i aufblättern an der Stelle *abc,* z.B. `? |1goethe`
Der Flex wird dann erst fortgesetzt, wenn man den Index verläßt. Daher kann man mehrere solche Befehle aufeinander folgen lassen und andere Befehle dazwischenschalten.

Von den nachfolgenden Befehlen genügen jeweils die ersten 3 Buchstaben, also z.B. "dow" statt "download", "var" statt "variable", "ins" statt "insert", aber auch "down" etc. würde funktionieren.
(Bei ***avanti*** muß man die Befehlswörter vollständig angeben. Wenn man FLEXe schreibt, die auch als ***avanti***-Aufträge funktionieren sollen, muß man das beachten.)

ask |**iprompt=**Vorgabe
ask +|**iprompt=**Vorgabe
Aufforderung zu einer Eingabe. Die eingegebene Zeichenkette wird in der **internen Variablen** gespeichert.
Wenn ein + gesetzt ist, wird die Eingabe an die iV hinten angehängt.
Für die Eingabe erscheint ein kleines Dialogfenster mit einer Eingabezeile. (Beispiel siehe S. 4)
i = Nummer des Index, der aufgeblättert wird, wenn man [Index] drückt
prompt = Aufforderungstext
(Wenn man |**i** wegläßt, kommt ein rein zufälliger Indexabschnitt)
Leereingabe führt zum Abbruch der Befehlskette.
Der Teil "=*Vorgabe*" kann entfallen; der Text *Vorgabe* erscheint im Eingabefenster.
Vorgabe kann auch eine Kategorie oder eine #u-Variable sein:
`ask |1Verf.?=#40` legt den Inhalt von #40 als Vorgabe vor. Die Eingabe geht aber nicht direkt zurück in die Kategorie, sondern in die iV und muß mit `insert #40` zurückübertragen werden, falls gewünscht.

call *Programmaufruf*
Hinter "call " kann man einen kompletten Startbefehl für ein externes Programm hinschreiben, wie beim Flipbefehl ~.
Es kann sich auch um einen DOS-Befehl handeln, z.B. `call del e.adt`
Wenn das externe Prog. eine Datei E.ADT produziert, kann diese anschließend mit "`read`" (s.u.) eingelesen werden: sie wird dann in den aktuellen Satz eingemischt. So kann man von außen Daten einschleusen.
Die Abarbeitung der Befehlskette geht erst weiter, wenn man das externe Programm verläßt.

copy
Der aktuelle Datensatz, so wie er gerade ist, wird als neue Kopie behandelt.. Nachfolgend evtl. `#xyz-` oder `insert-`Befehle und/oder Ersetzungen, dann evtl. **put** zum Speichern des neuen Satzes.

date *b* (*b*=Breite des Datums, 8-17 Byte; 8 ist nur das Datum, 17 ist Datum/vollständige Uhrzeit)
Datum (und Uhrzeit) werden in die interne Variable kopiert.
Diese speichert man mit `insert ...` in eine Kategorie oder ein Teilfeld

display *xABC*
deposit *xABC*
Abschnitt `#-x` in den Anzeigeparametern ausführen. Die Zeichenkette *ABC* ist dann in der Variablen #u1 enthalten. An den betr. Sprungmarken können z.B. Manipulationen am Datensatz stattfinden. Die gesamte Mächtigkeit der Exportsprache kann somit in die Tätigkeit eines FLEX einbezogen werden.
Der Befehl `deposit` bewirkt, daß die Anzeige sich nicht ändert, d.h. die Verarbeitung nur intern durchgeführt wird. Das ist nützlich, wenn man komplizierte Manipulationen machen will, die mit der FLEX-Sprache allein nicht gehen. Man verwendet dann #u1, um in den Anzeigeparametern zu entscheiden, was zu tun ist. Das Ergebnis der Verarbeitung ist dann in der Internen Variablen deponiert - daher der Name "deposit". Also: mit den zwei Befehlen
```
deposit
write
```
kann man die Anzeigeform in die Ausgabedatei überführen! Das wird man selten machen. Das Normale wird sein, daß man einen bestimmten Abschnitt ausführen läßt, um z.B. #u-Variablen zu besetzen oder eine Zusammensetzung aus mehreren Kategorien oder -bestandteilen in die interne Variable zu bringen. Man gibt also normalerweise `deposit yTEXT.` Damit wird der Abschnitt `#-y` ausgeführt, und es liegt dort `#u1 TEXT` vor.
Wenn eine Ausgabe entsteht, hat man sie anschließend in der iV und kann sie mit "write" ausgeben lassen oder mit "insert" in eine Kategorie kopieren, oder aber ignorieren, wenn es nur auf die internen Manipulationen ankommt.

download
Aktuellen Satz exportieren (wie über Menü Export). Soll nur ein bestimmter Abschnitt in den Exportparametern ausgeführt werden: vorher eine Variable besetzen, z.B. #uFL (mit Befehl `var xxx\ins #uFL`), und in den Parametern am Anfang einen Sprung einbauen: `#uFL +X e0`

download set
Aktuelle Ergebnismenge exportieren (wie über Menü Export)

erase
Der aktuelle Satz wird gelöscht. Dazu braucht nicht in der .INI-Datei `access=3` gesetzt zu sein, d.h. man kann das Löschen in erwünschten Fällen per Parametrierung ermöglichen. Sogar unter ***alcarta***! Eine Gefahr besteht dort nicht, weil es kein Schreibfeld zum Eingeben eigener FLEXe gibt.

extern (wie Alt+t)
Der aktuelle Satz wird in E.ADT ausgegeben und der externe Editor aufgerufen.
Nach Rückkehr kann der Satz wieder eingelesen werden.

find
find `suchbefehl`
Bilde eine Ergebnismenge. Diese wird nicht angezeigt, sondern sie wird anschließend als aktuelle Ergebnismenge benutzt, wenn man mit den Befehlen `next, prev`, oder `download set` arbeitet. Wenn *suchbefehl* fehlt, wird der Inhalt der iV benutzt. Wenn man diese vorher geeignet besetzt, kann man den Suchbefehl in Abhängigkeit vom aktuellen Satz gestalten. Der erste Satz der Ergebismenge wird sofort geladen, wird also zum aktuellen Satz.
Der FLEX bricht nach diesem Befehl ab, wenn die Suche fehlschlägt.

form *i*
Formular i wird aufgeblättert (i = 1...) (in der Reihenfolge der .FRM-Datei)
Man kann mehrere solcher Befehle hintereinanderschalten.

help *name*
Hier ist *name* der Name einer RTF-Datei (ohne .rtf), welche dann als Hilfetext in das Anzeigefenster geholt wird. Entspricht dem Flip h. Statt RTF darf es auch eine gewöhnliche ASCII-Datei sein. Das Programm setzt als Kopf HELPHEAD.RTF davor, damit sie angezeigt werden kann - sie erscheint dann natürlich in ganz schlichter Form.

input *n*
Damit kann die Nummer der Datei verändert werden, in welche die neuen Datensätze zu speichern sind.
Allerdings muß die Berechtigung (`access=` in der INI-Datei) mindestens 2 sein.

insert #xyz
Inhalt der internen Variablen in die Kat. #xyz kopieren.
Wenn man `insert #uxyABC` schreibt, wird noch ABC vor die interne Variable gesetzt.
Der Befehl nimmt also den nachfolgenden Text, hängt die iV hinten an, und interpretiert das ganze als Kategorie.

insert $a+#xyz
Inhalt der internen Variablen als Teilfeld $a an #xyz anhängen

insert $a-#xyz
Teilfeld a ersetzen, wenn es vorh. ist, sonst anhängen. Wenn die interne Variable leer ist: Teilfeld $a löschen

Jump xTEXT
wie "display", aber Ausgabe per Exportparameter statt Anzeigeparameter.
Sicherer funktioniert die Ausgabe per "download", siehe dort.

message *Text*
Text wird in einer Messagebox angezeigt und muß mit OK bestätigt werden.
Wenn der Nutzer etwas entscheiden soll: statt dessen Befehl `yesno`
Wenn er etwas eingeben soll: Befehl `ask`

new
Es wird ein neuer, leerer Datensatz angelegt. Der aktuelle Satz wird vorher in den Hintergrundspeicher kopiert. Nachfolgend manuelle Eingabe oder #- und `insert`-Befehle, bzw. `transfer`, um Kategorien aus dem Hintergrundspeicher zu übernehmen!

next
Der nächste Satz der Ergebnismenge, in der vorher eingestellten Sortierfolge, wird geladen und angezeigt.
Abbruch der Befehlskette erfolgt, wenn es keinen nächsten gibt, also nach dem letzten Satz.

order *M P*
Ergebnismenge ordnen (sortieren)
M = Modus: a=aufsteigend, d=absteigend, n=Nach Satznummern
P = Position: Das erste Zeichen der Kurzliste ist Position 0

prev
Der vorige Satz der Erg.Menge wird geladen und angezeigt
Abbruch der Befehlskette erfolgt, wenn es keinen vorigen gibt (sondern der erste geladen ist).

print
Anzeigefenster ausdrucken (wie Print-Button)

Put Speichern mit Rückfrage
put Speichern ohne Rückfrage
Der aktuelle Satz wird gespeichert (wie bei [Speichern]-Button)

Put new

put new

Der aktuelle Satz wird als neuer Satz gespeichert. (Dasselbe würde `copy\put` bewirken.)
War es ein Online-Satz und wurden keine Veränderungen gemacht, entsteht damit eine Dublette.

read

Datei E.ADT wird eingelesen. Diese kann eine Reihe von Kategorien enthalten, die dann alle in den aktuellen Satz eingefügt werden. (Erfassungshilfe)

repeat

Die gesamte Befehlskette wird so lange wiederholt, bis ein Befehl nicht ausführbar ist.
Sinnvoll in Verbindung mit `next / prev`, um automatische Schleifen zu bilden. Kann nur als letzter Befehl in einer Kette stehen (wenn noch was folgt, wird es ignoriert)
Während des Ablaufs einer Schleife kann man mit 'x' unterbrechen und dann wahlweise weiterlaufen lassen oder abbrechen.

sleep *n*

n Millisekunden untätig verharren, bevor weitergemacht wird. (Für 3 Sekunden muß man also schreiben: sleep 3000)
Nutzbar z.B., wenn man eine Folge von Hilfeseiten abrollen lassen will. Das kann durch eingestreute yesno-Befehle noch flexibilisiert werden:

```
help name1\sleep 4000\yesno Weiter?\help name2\sleep 4000...
```

transfer #nnn

Kategorie #nnn aus dem Hintergrundspeicher in den aktuellen Satz kopieren.
Wenn eine Variable #uxy in eine Kategorie #nnn kopiert werden soll: `var #uxy\ins #nnn`

undo

Entspricht dem Button [Wechseln] : Umschalten zwischen Bearbeitungs- und Originalzustand.

`update` Diese Funktionen sind noch in Vorbereitung. Sie sollen den gleichnamigen ***avanti***-Funktionen entsprechen.
`upload` Das DOS-Programm UPDATE wird dadurch ersetzt werden. Der Ablauf wird allerdings interaktiv überwacht und kontrolliert werden können. Und nicht nur Grunddateien (Typ .ALG) wird man einmischen können, sondern auch (wie bei ***avanti***) die einfacheren Externdateien (Typ .ADT). (S.a. Beispiel 3 unten)

variable *mixed string*

Der *mixed string* wird in die interne Variable kopiert. Der *string* kann genauso strukturiert werden wie bei dem Befehl **write** (siehe unten)

variable +*mixed string*

Inhalt an die iV hinten anhängen. Hiermit kann man beliebige Inhalte zu einer Zeichenkette verknüpfen, z.B.

```
var #40 ": " #20        Inhalt von #40 und #20 in iV kopieren mit ": " dazwischen
var +" (" #76 ")"       (Inhalt von #76) an iV hinten anhängen
ins #upt                iV in #upt speichern (Die iV als solche bleibt erhalten!)
```

write
Inhalt der internen Variablen in die Ausgabedatei schreiben

write *mixed string* [Das ist genau der ***avanti***-Befehl]
So kann man am schnellsten Satzinhalte und beliebigen Text ausgeben. Der *mixed string* kommt zusätzlich in die iV!
Beispiel:
```
write "Titel: " #20 n "Verfasser: " #40 n "Ort: " #74 " (" #76 ")"
```
produziert eine Ausgabe in dieser Form:

Titel: Hamlet
Verfasser: Shakespeare, William
Ort: London (1982)

Das 'n' ist der Befehl für eine "neue Zeile".
Schlichte Exporte kann man hiermit, genau wie bei ***avanti***, ohne Exportsprache machen.

xport p *name*
Parameterdatei *name.c*PR für den Export laden

xport f *filename*
xport f +*filename*
Nachfolgende Exporte sollen in die Datei *filename* gehen.
Das '+' bewirkt, daß an die existierende Datei angehängt wird, sonst wird sie überschrieben.

yesno *Frage*
noyes *Frage*
Die *Frage* wird in einer Ja/Nein-Box gezeigt. Ohne Antwort geht es nicht weiter.
Bei "Nein" wird der Job abgebrochen, d.h. nachfolgende Befehle nicht mehr ausgeführt.
Bei "noyes" ist der Button [Nein] statt [Ja] der default-Button! Dann muß der User bewußt auf [Ja] gehen - [Enter] bedeutet sonst Nein und Abbruch. Also ganz klar: weiter geht es in beiden Fällen nur bei OK (= Yes)

zzz Kein Befehl. Jeder ungültige Befehl wird ignoriert, kann also als Kommentar benutzt werden.

Was ist ein ***mixed string***? (siehe Befehle **variable** und **write**)
Eine Kette von Elementen, beliebig zusammengesetzt, wobei die Elemente aus vier Typen bestehen, getrennt durch Leerzeichen:

`"xyz"` Zeichenketten, in "..." oder '...' eingeschlossen

d d d ASCII-Codes als Dezimalzahlen (z.B. 27 69 für Esc E)

#nnn Kategorietexte (nnn kann auch eine #u- oder Sondervariable sein)
#nnn$a Teilfelder (nur der Inhalt des Teilfeldes wird ausgegeben)

i Interne Nummer des Datensatzes
l Größe der Ergebnismenge
n Neue Zeile
p Primärschlüssel des aktuellen Satzes
r relative Nummer des Satzes in der Ergebnismenge
s Kurzzeile des Satzes (aus der .STL-Datei)

FLEX-Beispiele

Beispiel 1 **: Teil-Automatisierung einer Bearbeitung**
Die Aufgabe ist folgende: Man hat eine Menge Datensätze zu bearbeiten, die man jeweils über die Inventarnummer aufruft, dann eine bestimmte Kategorie hinzufügt oder ändert und den Satz dann wieder abspeichert - manchmal aber auch nicht.
Annahme: es gibt ein Register INV im Index 9 mit der Inventarnummer.
Dazu eignet sich ein FLEX nach diesem Muster:

```
  Die Variable #uia mit dem Text "inv" belegen:
#uia inv
  Inventarnummer abfragen (Vorgabe: Inhalt von #uib) Index 9 bei Druck auf [Index]
ask  |9Inventarnummer=#uib
  Eingegebene Nummer wieder in #uib kopieren:
ins #uib
  Text der internen Variablen zusammensetzen aus #uia und #uib mit " " dazwischen
var #uia " " #uib
  und als find-Befehl ausführen (d.h. "find inv ...")
```

```
find
  Kategorie #xyz mit Inhalt Text einfügen in den geladenen Satz
#xyz Text
  Fragen, ob gespeichert werden soll
yesno Speichern?
  und ausführen, wenn Antwort "OK"
put
```

Statt der beiden letzten Befehle kann man auch **Put** setzen, dann kommt ebenfalls die Aufforderung zur Bestätigung

Beispiel 2 **: Eingabe-Unterstützung**
Neue Datensätze sollen immer schon mit einer Anzahl fester Kategorien vorbesetzt werden. Statt über die Formulartechnik kann man auch einen FLEX anlegen, indem man im Schreibfeld eingibt:

```
#uX3x new\#nn1 Text1\#nn2 Text2\#nn3 Text3\trans #nn4\form 2
```

Dadurch wird zuerst ein neuer (leerer) Satz angelegt, dieser mit drei vorbereiteten Kategorien belegt, Kategorie #nn4 wird aus dem Hintergrundspeicher übernommen, dann Formular 2 aktiviert für die weitere Eingabe. Mit Alt+3 aktiviert man diesen FLEX. Man sieht leicht, wie man dieses Beispiel ausbauen kann.
Man kann mehrere solche FLEXe für unterschiedliche Satztypen anlegen.

FLEX von außen starten : *a99* als neuartiger Server

Mit der Version 2.2 von ***a99*** und ***alcarta*** (ab 28.6.99) gibt es ein neues Werkzeug der gehobenen Mächtigkeitsklasse: den Externen FLEX, kurz ExFLEX. Ein ExFLEX ist ein FLEX wie jeder andere, aber gespeichert in einer Datei des Typs .FLX.
Das Neue ist: Ein solcher FLEX kann von außen gestartet werden, d.h. von einem DOS-Fenster aus. Während ***a99*** läuft, kann man ihm von DOS aus Botschaften zuschicken, welche FLEXe es ausführen soll. Hierdurch werden ***a99*** und ***alcarta*** zu neuartigen Serverprogrammen. Sie können (noch) nicht alles, was ***avanti*** kann, aber schon eine Menge, die Nutzung ist jedoch viel einfacher. Die FLEX-Sprache wurde mit der ***avanti***-Sprache harmonisiert. Es fehlen aber noch die update/upload-Funktionen sowie "list" und "qrix", ferner "if" und "jump". ExFLEX bedeutet also: man kann sich auf einfache Art neue Schnittstellen zu einer ***allegro***-Datenbank einrichten. Ergebnisse eines Auftrags kommen als Ausgabedatei heraus.
Die FLEX-Dateien müssen einen Namen mit bis zu 8 Zeichen und den Dateityp .FLX haben, um sie von anderen zu unterscheiden. Für diese Art der FLEXibilisierung muß man nichts parametrieren! Es wurde ein ganz kleines Hilfsprogramm geschaffen, mit dem man die Befehlsbotschaft an ***a99*** oder ***alcarta*** sendet. Dieses Programm FLEX.EXE (in a99.lzh enthalten) wird so gestartet, um `abc.flx` ausführen zu lassen: (nur der reine Dateiname `abc`, ohne Pfadname!)

```
flex abc
```

Das kann von Hand geschehen, es kann auch in einem Batch stehen. "flex" kehrt erst dann zurück (d.h. der Batch geht erst dann weiter), wenn `abc.flx` komplett abgearbeitet ist. Auf diese Weise kann man sich Befehle und Prozeduren von großer Mächtigkeit schaffen. (Klar, das setzt einen verantwortungsbewußten Datenbankverwalter voraus, der weiß, was er tut.)
Die Flexdateien werden in dieser Reihenfolge gesucht:
`DbDir / ProgDir\FLEX / ProgDir / Lokal`
("`Lokal`" bezieht sich auf das Arbeitsverzeichnis von a99, nicht dasjenige, wo der "`flex`"-Befehl gegeben wird.)
Man hat also hier, wie bei den Hilfedateien, die Möglichkeit, ExFLEXe auf ein separates Verzeichnis (ProgDir\FLEX, meistens also C:\ALLEGRO\FLEX) zu legen. (Einen anderen Namen kann man dafür nicht wählen.)
Wenn `abc.flx` nicht gefunden wird, erscheint "`FLEX abc.flx not found`" im Anzeigefenster, man sieht also wenigstens, daß die Botschaft angekommen war. Wenn gerade kein *a99* oder *alcarta* läuft, passiert einfach nichts.
Wenn mehr als ein *a99* oder *alcarta* läuft, verarbeitet jedes davon die Botschaft, daher sollte dann der FLEX auf seinem eigenen Verzeichnis liegen. Es ist nicht möglich, die Nachricht gezielt nur an ein Programm zu schicken, jedoch kann man die ExFLEX-Bearbeitung unterbinden: in die betreffende INI-Datei muß man dazu `exflex=0` schreiben (Default ist 1).

ExFLEX eignet sich besonders gut für die folgende Situation:
Man braucht regelmäßig eine bestimmte Auswertung, die sich durch Export einer bestimmten Ergebnismenge darstellen läßt. Das kann eine Berechnung sein, es kann auch eine Liste sein. Die Prozedur packt man in eine Datei `abc.flx`, dann gibt man nur bei Bedarf den Befehl `flex abc` , evtl. eingebettet in ein Batch, das die Ergebnisse noch weiterverarbeitet.
In einer FLEX-Datei dürfen Kommentare vorkommen, aber nur separate Zeilen, die mit blank beginnen.
Die nachfolgenden Beispiele sind Muster für ähnliche Situationen, die in der Praxis häufiger auftreten.

Beispiel 1 : **Summierung einer Ergebnismenge**

SUMME.FLX für eine Summenbildung sieht so aus, und man startet es mit dem Befehl **`flex summe`** :

```
  SUMME.APR laden
xport p summe
  Ergebnisse sollen in ERGEBNIS
xport f ergebnis
```

```
  Gesamte Erg.Menge durcharbeiten
down set
  Variable #uSU mit "xxx" besetzen
#uSU xxx
  aktuellen Satz nochmals verarbeiten (Endabschnitt!)
down
```

Als Exportparameter wird SUMME.APR genommen, geschrieben wird in die Datei ERGEBNIS.
Zuletzt wird die Hilfsvariable #uSU mit "xxx" belegt, dann noch der aktuelle Satz exportiert. Das ist ein Trick: wenn #uSU belegt ist, erfolgt in SUMME.APR ein Sprung zu einem Abschnitt, wo die Ergebnisvariablen ausgegeben werden. (Damit ist auch das Problem des Export-**Endabschnitt**s zu lösen, der von ***a99*** und ***alcarta*** bekanntlich nicht ausgegeben wird.)
Da nach jedem Export die Datei abgeschlossen wird, kann man nach Beendigung von `flex` sofort, ohne ***a99*** beenden zu müssen, von außen auf ERGEBNIS zugreifen.
Berechnet wird hier immer die gerade aktuelle Ergebnismenge. Um auch diese noch automatisch zu bilden, kann man eine kleine Batchdatei SUM.BAT machen, in der nur drei Zeilen stehen:

```
echo find sys %1? >zzz
copy zzz +summe.flx sum.flx
flex sum
```

Mit `sum abc` läßt man die Summe für die Ergebnismenge aus `find sys abc` errechnen. (Angenommen ist dabei, man hat ein Register SYS, mit dem sich die Ergebnismengen für den Zweck der Sache bilden lassen.) Eingefleischte Allegrologen dürften das Potential dieser Methode nun erkennen.

Beispiel 2 **: Liste der Ergebnismenge geordnet ausgeben**

Dafür wurde KURZLIST.FLX mit KURZ.APR vorbereitet (in a99.lzh). Diese sollten für alle Kategoriesysteme funktionieren:

```
  kurz.apr laden
xport p kurz
  Ausgabe soll in KLIST:
xport f klist
  nach titel ordnen
order a 0
  aktuelle Erg.Menge Ausgeben
download set
```

Hiermit kann man jederzeit, wenn man eine beliebige Ergebnismenge hat, schnell in ein DOS-Fenster gehen, einmal `flex kurzlist` eingeben, und schon hat man eine Datei KLIST mit der sortierten Liste. Auch dieses Modell kann natürlich beliebig abgewandelt werden, indem man die Parameter in kurz.apr ändert.

Beispiel 3 **: Einzelne Sätze einspeisen**

Situation: Es fallen außerhalb des Programms in unregelmäßigen Abständen Datensätze an, die jeweils sofort in die Datenbank eingemischt werden sollen. Jedesmal UPDATE aufzurufen ist möglich, aber unelegant.
Jetzt kann man ad hoc jeden Datensatz in einen kleinen FLEX einpacken: (den man mit einem geeigneten Prográmmchen extern erzeugt)

```
new
#00 ...
#20 ...
...
#90 ...
put
```

Die einzelnen Kategorien müssen hier ohne Zeilenbruch fortlaufend geschrieben werden! Wenn man diese Datei "neusatz.flx" nennt, kann man sie jeweils mit dem Befehl `flex neusatz` an ***a99*** schicken zum sofortigen Einmischen.
Es können mehrere Datensätze hintereinander mit einem FLEX eingemischt werden, jeweils getrennt durch `put \ new`.
Wenn "`put`" fehlt, werden die Sätze zunächst im Offline-Speicher gehalten und noch nicht gespeichert.

***Beispiel 4* : Neue Kategorien in vorhandene Sätze einschleusen**

Hat man die Primärschlüssel oder andere eindeutige Schlüssel von Datensätzen vorliegen, und sollen in bestimmte Sätze bestimmte neue Kategorien eingebracht werden, kann das nach diesem Muster geschehen:

```
find xxx schlüssel
    z.B.  find isb 3-519-12343-5
#nnn text
put
```

In solchen Fällen kann also ***a99*** statt UPDATE zum Einsatz kommen, denn der FLEX kann aus einer langen Folge solcher Dreizeiler bestehen. (Max. FLEX-Länge: 30.000 Byte)

***Beispiel 5* : Ausdruck von Barcode-Etiketten, Signaturen o.a.**
Viele Bibliotheken haben sehr individuelle Signaturen oder Barcodes und können diese nicht von einer vorgefertigten Rolle abziehen, sondern jedes Buch braucht sein individuelles Label oder Rückenschild. Dazu muß dann freilich einmal eine individuelle Parameterdatei geschrieben werden, sagen wir `LABEL.APR`. Es gibt dann zwei Möglichkeiten:

1. Druck aus ***a99*** oder ***alcarta*** heraus
Man setzt in der INI-Datei
`ExportParameter=LABEL`
`OutputFile=PRN`
und gibt im Schreibfeld ein:
`#uX7x download`
Dann muß man nur jeweils den Titelsatz aufrufen und dann Alt+7 betätigen. #uX7 bleibt für nachfolgende Sitzungen erhalten!

2. Druck mit einem **FLEX-Auftrag** von außen
Das ist machbar, wenn jeder Satz leicht mit einem eindeutigen Schlüssel aufgerufen werden kann. Sagen wir, dies sei die Inventarnummer, und sie sei in einem Register INV indexiert. Dann müßte die Eingabe der Inventarnummer in einem externen Programm geschehen, und jedesmal der folgende FLEX aktuell erzeugt und an das Programm gegeben werden:

`find inv` *inventarnummer*
`download`

Fremddatennutzung per *a99*

Es folgt eine detaillierte Anleitung, wie man auch per ***a99*** mit hohem Komfort Fremddaten aus anderen ***allegro***-Datenbanken übernehmen kann. Von seiten des Programms sind dafür alle Vorkehrungen getroffen, die Sache ist nur leider nicht total selbsterklärend. Die nachfolgende Beschreibung ermöglicht ein schnelles Einrichten der gewünschten Funktionen.
Mit dem "alten" Programm PRESTO hat man die Möglichkeit, zwei oder drei Datenbanken parallel zu schalten, d.h. beim Start zwei- oder dreimal die Option **`-d`** zu setzen (Handbuch Kap.12). Mit Alt+a schaltet man zyklisch von einer Datenbank zur anderen. Mit 'C' kopiert man einen Satz der zweiten oder dritten Datenbank in den Editor, mit F10 speichert man ihn ab in die erste Datenbank. Diese Arbeitsweise kommt mit einem Minimum an Tastendrücken aus und ist somit die wohl schnellste denkbare Fremddaten-Übernahmemethode.
Ähnliches wird selbstverständlich von ***a99*** erwartet. Noch schneller, d.h. mit noch weniger Schritten als bei PRESTO, kann es nicht gehen, soviel wird klar sein. Mehr sollen es aber auch nicht sein, sonst werden viele eingefuchste PRESTO-Anwender das neue Programm als Rückschritt empfinden, trotz aller sonstigen Annehmlichkeiten. Im Alltagsgeschäft zählt Schnelligkeit mehr als Oberflächenästhetik, man mache sich da nichts vor.
Folgendermaßen kann man mit ***a99*** verfahren: Für jede Datenbank macht man eine geeignete INI-Datei. Sagen wir, AB ist die Arbeitsdatenbank, FB die Fremddatenbank, dann machen wir uns eine AB.INI und eine FB.INI, beide auf dem Programmverzeichnis. Wo man startet, ist allerdings egal. Folgende Dinge müssen für die Fremdbank drinstehen, also in FB.INI:

```
Extern=no              # Mit Alt+t wird nicht der externe Editor gestartet
ExportParameter=e-w    # Parameter für Externformat (= Austauschformat)
ExEdFile=e.adt         # Mit Alt+t wird der aktelle Satz in E.ADT exportiert
OutputFile=extern.dat  # Ergebnismengenesxport geht in EXTERN.DAT
access=0               # Wenn Bearbeitung in der Fremdbank nicht gewünscht
```

Die fettgedruckte Zeile muß identisch auch in AB.INI stehen.
Das ist schon die gesamte Vorbereitung. Die Arbeit besteht dann aus folgenden Schritten:

1. Fremddatenbank starten mit **`a99 FB`**

2. Arbeitsdatenbank starten mit **`a99 AB`**

3. Wenn ein Fremdsatz gebraucht wird: Alt+TAB zum Umschalten (statt Alt+a)

4. Fremdsatz suchen, in die Anzeige bringen, Alt+t zum Exportieren in die Externdatei E.ADT.
Oder Menü "Bearbeiten | Extern" (Man sieht keine Bestätigung)

5. Alt+TAB zurück in die Arbeitsbank.

6. [Neusatz] / Nein betätigen, um einen leeren Neusatz zu erstellen.

7. Alt+ä Einlesen des Fremdsatzes aus E.ADT (oder Menü Bearbeiten | Get)
(Alt+g ist ja leider durch das Menü "Global" belegt) (Wenn man [Neusatz] vergessen hat, wird der Fremdsatz in den aktuellen Satz eingemischt! Mit dem Button [Wechseln] kann man das rückgängig machen.)

8. Datensatz bearbeiten, dann Alt+c oder Button [Speichern]. Bei Bedarf wieder Schritt 3.

Schritte 6 und 7 kann man durch einen FLEX zusammenfassen, in dem steht `new\read .`

Das ist, zugegeben, ein ganz klein wenig weniger direkt als bei PRESTO. Und beim Umschalten landet man nicht in der Fremdbank an derselben Registerstelle. Aber es gibt zum Ausgleich neue Möglichkeiten:

Mehr als zwei Fremddatenbanken
Es können auch mehr als zwei Fremdbanken zugleich gestartet sein. In jeder INI müssen die o.g. Befehle stehen.

Fremdes Fremdformat
Eine Fremddatenbank muß nicht mehr dasselbe Format (CFG) haben! Dann allerdings muß als ExportParameter eine Datei genommen werden, die im Format der Arbeitsdatenbank exportiert. Die Übergabe geschieht in jedem Fall über die Datei E.ADT, in welcher das Externformat der ARBEITSbank erwartet wird.

Sieben auf einen Streich (oder mehr)
Statt bei jedem einzelnen Satz umzuschalten, kann man auch:

- in der Fremddatenbank eine beliebige Anzahl Sätze zuerst exportieren, und zwar über Menü Export statt mit Alt+t (Einzeltitel oder Erg.Mengen) Die Daten gehen dann in die Datei EXTERN.DAT
- in der Arbeitsdatenbank alle zugleich einlesen mit dem Menüpunkt Datei | Externe Ergebnismenge (= EXTERN.DAT laden)

Die Datei EXTERN.DAT wird jedoch immer verlängert, solange die Fremddatenbank läuft. Daher muß man sie zwischendurch löschen (kann z.B. auch über einen FLEX geschehen, d.h. einen geeigneten Batch-Aufruf), oder die Fremddatenbank kurz verlassen, del extern.dat geben, und wieder starten. Der einfachste FLEX für diesen Zweck sieht so aus:

`#uX9~del extern.dat` *oder gleichwertig:* `#uX9x call del extern.dat`

Das gibt man im Schreibfeld ein, und mit Alt+9 wird es aktiviert. (Das zweite kann Teil eines längeren FLEX sein.)

Wie lernt man Parametrieren? Ein Leitfaden

Muß man es überhaupt noch lernen? Oder wird die Windows-Version es über kurz oder lang überflüssig machen? Nun, über kurz mit Sicherheit nicht, und über lang wollen wir nicht spekulieren, das würde wenig nützen, wenn jemand vor der akuten Frage steht, ob man sich einschlägige Kenntnisse aneignen solle. Wir wollen deshalb an dieser Stelle Tips geben, wie man sich in die Materie einarbeiten kann, insbes. welche Hilfsmittel dazu vorhanden sind. Als Einsteiger halten Sie sich am besten genau an diese Vorschläge, wenn Sie wenig Zeit aufwenden können. Ganz ohne Zeitaufwand geht aber leider in der EDV gar nichts. In wenigen Stunden können Sie sich die Grundkenntnisse aneignen und einfache Probleme lösen. Um aber versierter Allegrologe zu werden, dazu bedarf es einiger Monate intensiver Beschäftigung, neben der Neigung zu abstraktem Denken.

0. DOS-Kenntnisse

Wer seine Computerlaufbahn direkt mit Windows begonnen hat, muß erst noch diese Hürde überwinden: die wichtigsten Befehle zum Agieren in einem MS-DOS-Fenster muß man einfach kennen. Das sind eigentlich nur `cd, ren, copy, del, dir`. Wissen muß man auch, was **Verzeichnisse** sind, wie man sie anlegt und beseitigt. Wenn man eine Datenbank benutzt, gibt es immer drei wichtige Verzeichnisse: 1. das **Daten**verzeichnis (z.B. C:\ALLEGRO\DEMO), 2. das **Start-** oder **Arbeits**verzeichnis (oft C:\ALLEGRO, im Netz z.B. so etwas wie F:\USER\user*name*) und 3. das **Programm**verzeichnis (bei Einzelplatzbetrieb meistens ebenfalls C:\ALLEGRO). Die Programme suchen eine Parameterdatei immer zuerst auf dem Datenverzeichnis, steht sie dort nicht, dann auf dem Startverzeichnis, zuletzt auf dem Programmverzeichnis. Hat man also z.B. eine Parameterdatei namens xyz.apr auf dem Programmverzeichnis und eine gleichnamige, aber mit ganz anderem Inhalt, auf

dem Datenverzeichnis, so wird die letztere genommen. Das kann für Verblüffung sorgen, kann aber auch sinnvoll ausgenutzt werden.
Unter DOS ist ferner das Konzept der Stapeldatei sehr wichtig, auch **Batchdatei** genannt (Dateityp .BAT).
Die meisten Arbeiten macht man mit dem ***CockPit***, das man auf C:\ALLEGRO mit dem Befehl `cp` startet. Hiermit kommt man leicht an die Parameterdateien heran: Menü **`Dateien`**. Wählt man eine Export-Parameterdatei aus, und sodann den Menüpunkt "`edit`", wird automatisch der X-Editor gestartet und die Datei zum Bearbeiten präsentiert. Wenn es z.B. die Datei E-1.APR sein soll, kann man statt vom ***CockPit*** auch unter DOS im Verzeichnis `c:\allegro` den Befehl **`x e-1.apr`** geben, also den X-Editor mit der Hand aufrufen, das ist genau dasselbe.Der X-Editor ist ein Text-Editor:
Ansonsten legt das ***CockPit*** jedesmal, wenn ein Vorgang gestartet wird, eine Stapeldatei namens CCC.BAT an. Darin stehen die Befehle und Programmaufrufe, aus denen sich der Vorgang zusammensetzt. *Tip:* Starten Sie das CockPit mit **`acp`** statt `cp`, und lösen Sie dann den gewünschten Vorgang aus. Es passiert scheinbar nichts, aber die Datei CCC.BAT wird angelegt! Diese können Sie studieren, um zu sehen, welche Programme für die betreffende Aufgabe aufgerufen werden und auf welche Weise.
Die ***allegro***-Programme werden immer mit geeigneten Optionen aufgerufen, das sind Werte, die hinter den Programmnamen geschrieben werden, z.B. `srch -dkatalog -ka -f4 -es-vj/uuu.alg -m1 -v0` . Hier ist "srch" der Programmname (des Volltext-Suchprogramms), was dahinter folgt, sind die **Optionen**, wobei `-e...` für den Export zuständig ist. Das Kapitel 12 im Handbuch beschreibt alle Optionen, die man beim Aufruf der Programme angeben kann. (Die Windows-Programme werden nicht mit Optionen aufgerufen, sondern die entsprechenden Angaben stehen in einer INI-Datei.)

1. Text-Editor

Unverzichtbar sind einige wenige, sehr einfache Kenntnisse eines Text-Editors. Nicht Word oder dergl., auch nicht Notepad, sondern ein ganz schlichter DOS-Texteditor muß es sein, denn Parameterdateien sind ASCII-Dateien. EDIT von Microsoft tut es, viel besser ist aber der bei ***allegro*** mitgelieferte X-Editor. Was Sie darüber wissen müssen, ist in wenigen Minuten erlernbar und übersichtlich dargestellt im **Anhang D** (S. 327). Wenn Sie keine *ganz* triftigen Gründe haben, einen anderen Editor zu benutzen, dann wenden Sie die ***zwei Minuten*** für X unbedingt auf! Diese Kenntnisse sind unbezahlbar, ohne Übertreibung, und nicht nur für *allegro*. Andere Instrumente sind in der Regel klobiger und umständlicher und kosten deshalb per Saldo mehr Zeit. (Die besten Allegrologen verwenden fast alle X und sonst nix für die Parameter.)
Im Kapitel 10.1 wird ganz genau beschrieben, wie man eine neue Parameterdatei mit DOS-Methodik anlegt. Gehört der geplante Export zu einem der 10 Standardtypen (10.0), kann man die dafür vorhandene **Prototyp**-Datei als Vorlage benutzen.

2. Handbuch Kapitel 10

Dringend zu empfehlen ist, vor dem Experimentieren doch die Einführung zum Kapitel 10 einmal durchzulesen. Diese Einführung ist keine wissenschaftliche Abhandlung, sondern auf allgemeine Verständlichkeit hin mehrfach überarbeitet worden. Konzepte und Grundbegriffe *müssen* ja einfach irgendwo mal erklärt und sie müssen auch halbwegs verstanden werden, und ohne das kommt höchstens ein völlig unflexibles System aus - oder wie sehen Sie das? (Wenn Sie das Kapitel 0 auch noch nie so richtig gelesen haben, kann das in diesem Zusammenhang ebenfalls nicht schaden. Es erklärt das Datenbankkonzept, also den "Gesamtzusammenhang", Kap. 10 nur die Export-Konzepte.)

3. "Die erste selbstgemachte Datenbank"

In der "news"-Ausgabe 47 (1997/3) gab es eine noch immer zu empfehlende Lektion "**Die erste selbstgemachte Datenbank**". Darin lernt man, welche Dateien zu einer Datenbank gehören, und um welche man sich wirklich kümmern muß. Dazu gehört die Anzeige-Parameterdatei, beim Standardsystem D-1.APR genannt. Wenn Sie die Lektion nachvollziehen, erhalten Sie u.a. die einfachste mögliche Anzeige-Parameterdatei: sie zeigt auf dem Bildschirm nur den Inhalt von Kategorie #20 an. Von diesem Punkt aus können Sie mit eigenen Versuchen weitermachen: Wenn Sie die Konfigurationsdatei B.CFG erweitern (über ***CockPit*** "Optionen / Konfiguration" aufrufen und bearbeiten), indem Sie neue Kategorien hinzufügen, können Sie diese Kategorien auch in die Anzeigeparameter einbauen: dabei wird auch die Methode der **Merseburger Testschleife** vermittelt, die viel Zeit spart. So geht's:
In der PRESTO-Titelanzeige F2 F2, die Datei D-1.BPR anwählen, F10. Jetzt sind Sie im X-Editor und haben D-1.BPR vor sich. Nun 'x' drücken (Schreibmodus einschalten), gewünschte Änderung durchführen, [Esc] q s zum Speichern, '+' vor den Namen setzen, [Enter]. Jetzt ist die geänderte D-1 geladen und wird für die Anzeige benutzt - d.h. Sie sehen das Ergebnis Ihrer Änderung sofort vor sich. Mit erneutem F2 F2 ... können Sie das Spielchen wiederholen, beliebig oft. Schneller kann ein Testverfahren für diese Zwecke wohl kaum sein, was eingearbeitete Anwender immer wieder schätzen. Man kann die meisten Parameter auf diese Weise testen, auch solche, die nicht für die Bildschirmanzeige gedacht sind. (Mit ***a99/alcarta*** kann man eine veränderte Parameterdatei über das Menü "Ansicht | Anzeigeparameter wechseln" ebenfalls sofort testen.)

4. Allers-Lernsystem : Großes Menü mit vielen kleinen Lektionen

Der "Allers", das bekannte Lehrbuch für die Parametrierung, ist leider vergriffen und kann nicht so bald neu herauskommen. Wer eine der ***allegro***-CD-ROMs hat, verfügt jedoch über das Lernsystem; per FTP kann man es sich ebenfalls besorgen.
Installation: auf der CD-ROM schalte man auf das Verzeichnis **\param\allers**. Dort liegt **allers14.exe**.
Man legt eine leere Diskette ein, öffnet eine DOS-Box und schaltet auf das Diskettenlaufwerk, also A:. Dort gibt man den Befehl **e:allers14** (wenn E: das CD-Laufwerk ist). Dann, immer noch auf A:, den Befehl **install**. Es kommt ein Menü, in dem man höchstens an einigen Stellen C: in D: ändern muß, wenn man nicht auf C: sondern auf D: installieren will. Mit F10 wird dann die Installation vollendet. (Die Diskette kann anschließend auch noch zum Installieren auf anderen Rechnern genutzt werden.)
"Allers" verrät dann noch dieses:

```
Beginnen Sie die Übungen mit einem der folgenden CockPit-Aufrufe:

     cp        - für 'normalen' Datenbankaufruf
     start1    - für Exportparametrierung I  (incl. Satzverkn., Rechenbefehle)
     start2    - für Exportparametrierung II und Sonstiges (Index-Parameter!)
     start3    - für Importparametrierung
```

Und das alles spielt sich ab auf dem Lernverzeichnis ACLERN, das bei der Installation entstanden ist.
Geben Sie dort einfach **cp** ein - die anderen drei Aufrufe stellt dann das ***CockPit*** auf dem Hauptmenü bereit.
Für das Starten einer neuen Sitzung immer in einer DOS-Box auf ACLERN gehen und **cp** geben.
Fast alles beruht auf dem Prinzip der "Allers'schen Testschleife", die jeweils automatisch vom ***CockPit*** in Gang gesetzt wird. Eine solche Testschleife (ein .BAT-Programm) besteht immer aus zwei bis drei Vorgängen: Einblick in eine Parameterdatei (mit Hinweisen für Versuche, die man dort machen kann), Start der Datenbank und Betrachtung der Exportergebnisse, evtl. noch Anzeige der Exportergebnisse, wenn der Export nicht auf den Bildschirm gegangen ist, sondern in eine Datei.
Die Beispieldateien enthalten zwar viele Kommentare, leider aber nicht immer den Hinweis auf den zuständigen Handbuchabschnitt. Doch dieser ist leicht zu finden: Die Lektionen sind in der Reihenfolge der Abschnitte des Kapitels 10 angeordnet. Außerdem kann man im Register des Handbuchs jeden Befehl unter seinem Namen finden.

5. Kompliziertere Fälle: Sortierte Listen und Auswertungen

Wenn es um die Erstellung von sortierten Listen geht, oder um statistische Auswertungen, müssen oftmals zwei oder mehr Durchläufe mit unterschiedlichen Parametern stattfinden. Außerdem kommt dabei nicht nur das Programm PRESTO zum Einsatz, sondern auch SRCH, das Volltext-Suchprogramm. Dieses kann nicht nur suchen, es kann auch exportieren, und zwar auch Dateien, die nicht zu einer Datenbank gehören, sog. Grunddateien (Typ .ALG), die beim Exportieren aus einer Datenbank und beim Importieren aus Fremddaten (Kap. 5) oft entstehen und dann weiterverarbeitet werden müssen.
Hier ist Kapitel 6 sehr wichtig: es erklärt die Verfahrensweise bei der Herstellung von sortierten Listen und Statistikauswertungen, wofür es vorbereitete Hilfspakete gibt:

- QUEX und QUANT. Diese automatisieren menügesteuert das Erstellen der Parameterdateien für einfache Listen bzw. statistische Auswertungen. Man muß dabei nichts vom Parametrieren wissen. Es besteht dann aber immer noch die Möglichkeit, die dabei automatisch entstandenen Parameter weiter auszubauen und zu modifzieren. (Kap. 6.3)
- Wenn die Sortierung oder die Druckaufbereitung höhere Ansprüche erfüllen soll: das ***CockPit*** bietet unter "Volltextsuche / Listen" schon einige fertig vorbereitete Parameterdateien, die sich vielfältig kombinieren lassen. Kapitel 6.1 und 6.2 beschreiben das Konzept dieser Methode. Die Stapeldateien PR-LIST.BAT und SR-LIST.BAT kann man auch als Beispiele für zusammengesetzte Abläufe benutzen.
- EXPEX : Übungen mit USMARC. Mit dem Kernsystem ebenfalls ausgeliefert wird ein Paket, das einerseits auf dem MARC-Format der Library of Congress beruht, andererseits insgesamt 10 Parameterdateien für die Bildschirmanzeige von MARC-Daten. Diese sind besonders deshalb zum Lernen geeignet, weil sie mit dem einfachsten möglichen Modell anfangen und dann fortschreitend immer mehr Elemente hinzufügen, bis am Ende eine komplette katalogkartenähnliche Anzeigeform entstanden ist. Starten Sie dieses Lernpaket auf C:\ALLEGRO mit dem Befehl EXPEX. Etwas mehr dazu erfahren Sie in Kap. 6.3.3 (fälschlich hinter 6.4 abgedruckt).
- Listen mit Hauptteil und alphabetischen Registern gehören zu den schwierigsten Aufgaben, weil mehrere Vorgänge ineinandergreifen müssen. Die Methodik und Beispiele dazu stellt Kap. 6.5 dar.

Reicht die Anwendung der vorgefertigten Methoden nicht aus, dann muß man tiefer in die Parameterdateien eingreifen. Um das Abarbeiten einer Parameterdatei und der einzelnen Befehlszeilen darin richtig zu interpretieren, liest man die Abschnitte auf S. 167 oben und auf S. 186-188 (Kap. 10.2.6). Hat man diese gut verstanden, ist man dem Expertentum schon recht nahe.

6. Schnellschuß-Projekte mit PRONTO

Will man eine völlig neue Datenbank mit eigener Struktur aufbauen, kann PRONTO eine Menge Starthilfe geben. PRONTO.BAT liegt auf Ihrem Programmverzeichnis, Sie starten es also dort mit dem Befehl PRONTO. Aus Fertigteilen setzt Ihnen dieses Programm alle zur Datenbank gehörigen Dateien zusammen: die CFG, die Indexparameter und eine Parameterdatei für die Anzeige. Dieses Grundgerüst können Sie anschließend ausbauen. Es kann eine Menge Zeit sparen, wenn auf diese Weise die grundlegend wichtigen Dateien schnell und einfach zusammengestellt werden. Eine genauere Beschreibung finden Sie in der Datei PRONTO.TXT (gehört ebenfalls zum Kernsystem) und in ***news*** Nr. 42 (1996/2).

7. Neue Möglichkeiten mit *a99*

- **Sortierte Listen mit einfacher Struktur**
  Ausgabedateien mit einfacher Struktur können sogar ohne Parametrierung mit der FLEX-Methodik erstellt werden.
  Wenn man im Schreibfeld eingibt:
  `#uX4x write #20 " / " #40 ". [" #90 "]" n\next\repeat`
  hat man auf Alt+4 eine Prozedur gelegt, die folgendes macht: Die Inhalte der Kategorien #20, #40 und #90 werden mit der in "..." angegebenen Interpunktion ausgegeben, am Ende eine neue Zeile, dann wird der nächste Satz der Ergebnismenge genommen und der Vorgang wiederholt - bis zum Ende der Ergebnismenge.
  Man benutzt diese Prozedur so: Ergebnismenge bilden, im Kurzanzeigefenster sortieren lassen, dann den ersten Satz anzeigen lassen, dann Alt+4. Die Ausgabe beginnt mit dem aktuellen Satz – ist dies nicht der erste, kommt nur der Rest ab diesem Satz heraus.
  Dieses Muster kann man leicht abwandeln und ausbauen. Wird der Vorgang häufiger gebraucht, kann man statt dessen eine FLEX-Datei machen, Name z.B. LISTE.FLX, siehe S. 5, und dann nur eingeben: `#uX4X kurzlist`. Von außen kann man nun sogar den Befehl `flex liste` geben, und ***a99*** führt den Vorgang aus.
- **Summen und Durchschnitte**
  Auswertungen, bei denen nur gerechnet werden muß (meistens Summen und Durchschnitte), lassen sich jetzt mit ***a99*** sehr leicht und mit wenig Parametrierung automatisieren, siehe dazu Beispiel 1 auf S. 9 in dieser Ausgabe. Die dabei verwenderte Parameterdatei SUMME.APR kann man als Vorlage für ähnliche Aufgaben verwenden. Ansonsten gehört zum Kernsystem die Datei R-0.APR als kommentierte Vorlage für Rechenauswertungen. Darin findet man einige Beispiele für die Anwendung der **Rechenbefehle** in den Exportparametern. Auch Allers widmet diesen ein paar Lektionen (im Kapitel D unter "Start1").

Exportfragen

Anwendervariablen

Es wurden neue Möglichkeiten zum Anlegen und Löschen von #u-Kategorien geschaffen.
Manchmal muß man eine unbestimmte (vom Datensatzinhalt abhängige) Anzahl von #u-Variablen anlegen. Das ist nun sehr einfach: Innerhalb einer Schleife kann ein Befehl nach diesem Muster vorkommen:
`#kkk ... =Y~`
Dadurch wird bei jedem Durchlauf der Schleife die nächste #uYx angelegt, beginnend mit #uY0. (Beispiele: D-WRTF.APR)
Weil #u-Variablen aber bei Beginn eines neuen Satzes nicht verschwinden, sondern im Gegenteil erhalten bleiben, muß man auch eine Möglichkeit haben, z.B. alle vorhandenen #uYx en bloc zu löschen. Dazu wurde der Manipulationsbefehl **`d`** erweitert. So kann man z.B. schreiben:
`#nr dY~ e0`
und alle #uYx verschwinden. Es ist darüberhinaus möglich, mit einem Befehl wie
`#nr d~~ e0`
sämtliche #u-Variablen auf einen Schlag zu beseitigen.

Kurzzeile stark verlängert

Der Befehl **`i0=`***k* in den Indexparametern regelt die Länge der Kurzanzeige-Zeilen. Bisher war das Maximum **`i0=72`**. Ein größerer Wert wurde ignoriert. Jetzt ist **250** das Maximum. Sichtbar werden weiterhin nur die ersten 72 Zeichen, aber der Rest kann von ***avanti*** für Sortierungen genutzt werden. PRESTO, INDEX und UPDATE der V16 können die verlängerten Kurzanzeigen korrekt erzeugen, auch wenn PRESTO sie nicht anzeigt.

Expertentreffen '99

Das diesjährige soll stattfinden am 16./17. September in der UB Braunschweig. Anmeldungen werden schon angenommen. Das Programm geht dann rechtzeitig den registrierten Anmeldern zu. Zu diesem Treffen können wieder bis zu 50 Personen kommen. Die Kostenbeteiligung an dieser Veranstaltung beträgt wie immer DM 50 pro Person.