

Folgende Funktionen können bei den Befehlen APPEND, BROWSE, EDIT, INSERT, MODIFY COMMAND zusätzlich genutzt werden:

- 1)  Cursor 1 Zeichen nach links
- 2)  Cursor 1 Zeichen nach rechts
- 3)  Cursor auf 1. Zeichen des nächsten Feldes
- 4)  Cursor auf 1. Zeichen des vorherigen Feldes
- 5)  Kommando beenden und Daten abspeichern
- 6)  Modiwechsel zwischen ÜBERSCHREIBEN und EINFÜGEN
- 7)  Zeichen auf Cursorposition löschen

Folgende Programme können nicht verwendet werden:

- 1) INSTALL.COM: Die von SIEMENS gelieferte dBASE II-Version ist bereits fertig installiert und benötigt daher dieses Installationsprogramm nicht mehr.
- 2) ZIP.COM, ZIPIN.COM: Der Maskengenerator ZIP kann nur auf 8-Bit-Systemen verwendet werden und findet deshalb beim SIEMENS PC-D, der einen 16-Bit Prozessor hat, keine Anwendung.

Folgende Fehler treten bei der Arbeit mit dBASE II auf:

- 1) Die in Kapitel 4 Abschnitt 2 und 3 beschriebenen Beispiele sind fehlerhaft.
- 2) CHANGE: Der gesamte Inhalt eines Datenfeldes läßt sich mit der Tastenkombination CTRL-Y, RETURN nicht löschen.
- 3) CREATE: Die Taste "ESC" wird bei der Abfrage "Daten jetzt eingeben?" als "J" interpretiert.
- 4) INSERT: Wird der Befehl INSERT in Verbindung mit dem Befehl SET FORMAT TO <Formatdatei> verwendet, so führt dieser zur Fehlermeldung "RECORD OUT OF RANGE".
Befindet man sich auf dem letzten Datensatz, so führt INSERT zur Fehlermeldung "NO RECORDS IN DBASE FILE".
- 5) MODIFY COMMAND: Beim Einlesen von Dateien in den Programmeditor werden Umlaute als Schmierzeichen geladen.
- 6) RECALL ALL: Funktioniert nicht, wenn SET-Parameter DELETE auf ON gesetzt ist.
- 7) SET F<n> TO <Belegung>: Die auf Seite 23 beschriebene Änderung der Funktionsbelegung funktioniert nicht.
- 8) QUIT [TO <LISTE VON .COM-DATEIEN >] : Der auf Seite 38 beschriebene Befehl funktioniert nicht.
- 9) FUNKTION "!" : !(äöüß) ergibt "äöüß".
- 10) FUNKTION "#": Ergibt Ø, wenn keine Datenbank eröffnet ist.
- 11) FUNKTION EOF: Ist aktiv, wenn keine Datenbank eröffnet ist.
- 12) § statt @ : Für die deutsche Tastatur wird im Manual die Verwendung von § anstelle von @ erwähnt. Da aber das Zeichen "@" auf der SIEMENS-Tastatur vorhanden ist, darf das Zeichen "§" nicht alternativ verwendet werden.
- 13) @ <Koordinaten> SAY <Ausdruck> GET <Variable> PICTURE <FORMAT>:
Das Formatsymbol "A" läßt keine Umlaute zu.

GESAMTINHALTSVERZEICHNIS

	Seite
1. Kapitel: Anpassung und Installationshinweise	1
2. Kapitel: Die Anwendung von dBASE II	
Abschnitt 1: Grundsätzliches über dBASE	25
Abschnitt 2: Wie man eine Datenbank erzeugt und mit ihr umgeht	47
Abschnitt 3: Arithmetische Funktionen und Datenbankmanipulation	79
Abschnitt 4: Die Programmierung der Datenbank	127
Abschnitt 5: Zusatzfunktionen zu Programmierung und Manipulation der Datenbank	157
Abschnitt 6: ZIP - der Bildschirm- und Drucker-Formular-Generator 1.....	175
3. Kapitel: Alle dBASE-Funktionen auf einem Blick	
Abschnitt 1: Einführung in die wichtigsten Funktionen	199
Abschnitt 2: dBASE-Befehle in alphabetischer Reihenfolge	235
Anhang A : dBASE-Befehle	367
Anhang B : Fehlermeldungen	371
4. Kapitel: Praktische Beispiele mit dBASE II	
Abschnitt 1: Erläuterungen zu den Beispielen und der Testversion	375
Abschnitt 2: Das Beispiel Adreßverwaltung	383
Abschnitt 3: Das Beispiel Rechnungsverwaltung	399
5. Kapitel: Alphabetisches Stichwortverzeichnis	415

1. Kapitel: Anpassung und Installationshinweise

Inhalt

Einführung.....	1
Im Handbuch verwendete typographische Regeln.....	1
Anforderungen an das Hard- und Software-System.....	2
Allgemeine Daten von dBASE II.....	2
Herstellen einer Sicherheitskopie.....	3
Anpassung von dBASE II auf Ihrem Rechner.....	4
Individuelle Terminalanpassung.....	7
Modifizieren einer früheren Anpassung.....	17
Hinweise für die Nutzer von 16-Bit-Mikrocomputern.....	21

Einführung

dBASE II ist ein Datenbank-Verwaltungssystem, das die leichte Handhabung kleiner und mittlerer Datenbanken mit Hilfe von Befehlen erlaubt, die der englischen Sprache ähneln. Mit dBASE II können Sie:

- Vollständige Datenbanken erzeugen.
- Auf einfache Weise Daten zu Ihrer Datenbank hinzufügen, löschen, verändern, darstellen und ausdrucken, wobei die Dateien ein Minimum an Mehrfacheintragungen aufweisen.
- Ein hohes Maß an Programm/Daten-Unabhängigkeit erzielen, so daß Sie, wenn Sie Ihre Daten verändern, nicht Ihre Programme umzuschreiben brauchen (und umgekehrt).
- Übersichten und Auswertungen (Reports) aus einer oder mehreren Datenbanken anfertigen, wobei jederzeit automatisch Multiplikation, Division, Teil- und Totalauswertungen und andere Datenmanipulationen ausgeführt werden können.
- Freie bildschirmorientierte Editierung benutzen, um eine Bildschirm-Maske zu erstellen, so daß Sie einen genauen Überblick darüber haben, woran Sie arbeiten, und Daten einfach eingeben, indem Sie "die leeren Zwischenräume ausfüllen".

dBASE II ist ein außerordentlich leistungsfähiges System. Nehmen Sie sich bitte die Zeit, die Gebrauchsanleitung zu lesen, bevor Sie es benutzen, damit Sie seine Fähigkeiten vollständig ausnutzen können. Der Zeitaufwand wird sich lohnen.

In diesem Handbuch werden folgende typographische Regeln benutzt:

Kleingeschriebene Texte in den Beispielen, die einen Bildschirm-Dialog darstellen, zeigen an, was Sie eingeben.

Großgeschriebenes in den Bildschirm-Beispielen zeigt, was das dBASE II-System zu Ihnen sagt oder Ihnen antwortet. Im erläuternden Text des Handbuches wird Großschreibung verwendet, um dBASE II-Befehle hervorzuheben.

'...' wird im Text dieses Handbuches verwendet, um dBASE II-Befehle und -Mitteilungen, die Sie eingeben, hervorzuheben. Gelegentlich wird diese Hervorhebung auch in den Bildschirm-Beispielen verwendet, wenn es zur Klarstellung erforderlich ist. DIE KLAMMER-SYMBOLS (' ', d.h. das einfache Anführungszeichen) SOLLEN SIE NICHT EINGEBEN!

[...] eckige Klammern werden benutzt, um Teile eines dBASE II-Befehls anzuzeigen, die auf Wunsch angegeben oder fortgelassen werden können.

< ... > spitze Klammern zeigen Teile eines dBASE II-Befehls an, der mit echter Information ausgefüllt werden muß. Beispielsweise bedeutet "<Filename>", daß Sie statt "Filename" den Namen eines Files (einer Datei) einfügen sollen, die Sie tatsächlich benutzen wollen.

<RETURN> bedeutet, daß Sie die Wagen-Rücklauf-Taste betätigen sollen (üblicherweise als "RETURN" oder "ENTER" bezeichnet). GEBEN SIE BITTE NICHT DAS WORT "ENTER" ODER DIE KLAMMERN EIN!

Anforderungen an das Hard- und Software-System Ihres Computers

dBASE II benötigt folgende Hard- und Software-Umgebung:

- 8080, 8085 oder Z80 Mikrocomputer. (wie z.B. TRS-80 Model II, Northstar, Apple mit Z80-Karte).
- 8086 oder 8088 Mikrocomputer (wie IBM-PC, NEC, APC etc) mit dem CP/M 86 oder MSDOS Betriebssystem.
- mindestens 48 KBytes Schreiblesespeicher
- CP/M 1.4 o. 2.x, TurboDOS, CDOS- oder CROMIX-Betriebssystem.
- Ein oder mehrere Massenspeicher-Geräte (üblicherweise Floppy-Disk-Laufwerke oder Festplattenspeicher).
- Einen Bildschirm oder ein Terminal mit frei adressierbarem Cursor, wenn die bildschirmorientierte Editierung verwendet werden soll.
- Die Möglichkeit, einen Text-Drucker zu benutzen (bei einigen der Befehle).

Allgemeine Daten des dBASE II-Systems

Anzahl Einträge (Records) per Datei:	65535 max.
Zeichen pro Record:	1000 max.
Felder pro Record:	32 max.
Zeichen pro Feld:	254 max.
Größte darstellbare Zahl:	etwa $+1.8 \times 10$ hoch 63
Kleinste darstellbare Zahl:	etwa $+1 \times 10$ hoch -63
Rechengenauigkeit:	10 Stellen
Länge einer Zeichenkette (String):	max. 254 Zeichen
Länge einer Befehls-Zeile:	max. 254 Zeichen
Kopf eines Reports:	max. 254 Zeichen
Länge eines Index-Schlüssels:	max. 100 Zeichen
Ausdrücke in SUM-Befehl:	max. 5

Anfertigen einer Sicherheitskopie

Falls Sie mit dem Rechner noch nicht umgehen können, dann lassen Sie die Anpassung von Ihrem Fachhändler oder sonst einer kundigen Person vornehmen. Lassen Sie sich dann auch gleich zeigen, wie die nötigen Handgriffe aussehen und welche Grundbegriffe Sie über das Befehlssystem Ihres Computers wissen müssen.

Bevor Sie irgendetwas anderes tun, machen Sie eine Sicherheitskopie der Originaldiskette von dBASE III! Verwahren Sie das Original an einem sicheren Platz und benutzen Sie die Kopie! Besonders wenn verschiedene Leute mit dem System arbeiten, machen Sie von dieser "Master"-Kopie weitere Arbeitskopien. Diese Grundregel sollten Sie und Ihre Mitarbeiter nie außer Acht lassen!

Setzen Sie ins Laufwerk A eine Systemdiskette Ihres Betriebssystems (CP/M, CDOS oder welches Sie sonst benutzen) und ins Laufwerk B eine Sicherheitskopie von dBASE II. Nun geben Sie folgendes ein:

'PIP A: = B:*. * [OV]'

Der Buchstabe "O" ist erforderlich, um sicherzustellen, daß Ihr Betriebssystem alle Daten von der Vertriebsdiskette überträgt.

(Anmerkung des Übersetzers: "O" und "V" sind sogenannte Optionen. "V" bewirkt, daß das Programm PIP die übertragenen Daten zur Kontrolle nochmals liest, um Übertragungsfehler zu erkennen. "O" soll verhindern, daß bestimmte Codes (File-Ende-Zeichen), die in Textfiles nicht, wohl aber im Maschinencode [COM-Files etc.] vorkommen, den Abbruch der Übertragung bewirken. Der Übersetzer konnte auf seinem CP/M 2.2 bislang nicht beobachten, daß diese Option bei der Übertragung zwischen Massenspeicher-Dateien (Disketten) nötig wäre.)

Wenn Sie mit einem System arbeiten, das nur ein Laufwerk besitzt, verwenden Sie die COPY oder BACKUP-Befehle und befolgen Sie die über den Bildschirm ausgegebenen Anweisungen.

Backups (Sicherheitskopien) sind von grundlegender Bedeutung, sie sollten regelmäßig angefertigt werden. Wenn Sie jeweils nur kurze Zeit mit Ihrem Computer arbeiten, wird eine Backup-Kopie pro Sitzung ausreichen, andernfalls sollten Sie mehrmals eine Backup-Kopie anfertigen. Sie können besser als wir die Kosten einer Backup-Kopie mit den Kosten für die Erstellung Ihrer Daten vergleichen, aber da Sie Disketten wieder überschreiben können, sind die Backup-Kosten gering. Wieviel aber ist Ihre Datenbank wert?

Dieser Gesichtspunkt kann nicht genug betont werden!

Das Anpassen von dBASE II auf Ihrem Rechner

Setzen Sie die Kopie (Sie haben doch eine Kopie angefertigt, nicht wahr?) des dBASE II-Systems in Ihr eingerastetes Laufwerk ein (eingerastet oder logged-on bedeutet das Laufwerk, dessen Prompt unter CP/M auf dem Bildschirm gerade ausgegeben wird. Nach 'B:' z.B. meldet sich das System mit "B>"). Führen Sie irgendwelche sonst nötigen Handgriffe zur Initialisierung aus (Eingabe von Control-C in CP/M, RESET oder Sonstiges).

Nun geben Sie 'install' ein, um das System mit den Besonderheiten Ihres Rechners vertraut zu machen (GEBEN SIE DIE (!) NICHT EIN!).

Vergleichen Sie die folgenden Ausführungen mit Abb. 1.1.

Um Ihnen das Verständnis des Dialoges zu erleichtern, bestehen viele Abbildungen aus zwei Teilen (a und b). Abb. 1.1a zeigt den Dialog mit den englischen Original-Äußerungen des Systems, Abb. 1.1b zeigt eine "Übersetzung" dieses Dialoges, die 1:1 dem Original entspricht.

Wenn Ihr Terminal keine X-Y-Adressierung des Cursors besitzt (d.h. Befehle, mit denen der Cursor unmittelbar an jede gewünschte Stelle des Bildschirms geschickt werden kann, schlagen Sie in Ihren Unterlagen nach), beantworten Sie die erste Frage mit 'N'. Andernfalls antworten Sie mit 'y'.

("N" = No (nein), "Y" = Yes (ja)).

Die letztgenannte Alternative ('y') macht Ihnen die bildschirmorientierte Editierung verfügbar, eine bequeme Arbeitsweise bei der Eingabe von Daten und dem Umgang mit Ihren Datenbanken. Statt daß Ihre Eingaben am unteren Ende des Bildschirms anstossen, wobei der ganze Inhalt des Bildschirms nach oben geschoben wird, können Sie mittels gewisser dBASE II-Befehle den Cursor frei auf dem Bildschirm bewegen wie den Bleistift auf einem Formular und immer gerade dorthin schicken, wo Sie etwas eingeben oder korrigieren wollen.

Anschließend schreibt dBASE II eine Liste der vorbereiteten Terminal-Typen aus. Wenn Ihr Bildschirm dabei ist, tippen Sie den entsprechenden Buchstaben ein. Wenn Ihr Terminal oder Computer/Betriebssystem-Typ nicht aufgeführt ist, so geben Sie 'Z' ein.

Abb. 1.1a:

```
A> install

dBASE II INSTALLATION PROGRAM VER 2.7
ARE FULL SCREEN OPERATIONS WANTED (Y/N) y
SELECT TERMINAL TYPE
A - HAZELTINE 1500          B - SOROC, TELEVIDEO
C - HEATH 89                D - PERKIN ELMER 1100
E - ADM-3A                  F - ADM-31
G - VDP-80                  H - INTERCOLOR
I - GNAT SYSTEM 10          J - TRS-80 II (P&T)
K - APPLE                   L - VEKTOR GRAPHICS
M - SUPERBRAIN              N - VISUAL 100
O - OSBORNE                  P - H 2621, HP-125
Q - CROMEMCO 3102           R - TRS-80 (FMG)
S - ADDS VIEWPOINT          T - XEROX 820
U - NEC
Y - CHANGE/MODIFY PREVIOUSLY INSTALLED TERMINAL
Z - USER SUPPLIED TERMINAL CHARACTERISTICS.
```

k

Abb. 1.1b:

A>install

dBASE II EINRICHTUNGS PROGRAM VER 2.7

WIRD BILDSCHIRMORIENTIERTE EDITIERUNG GEWUENSCHT? (Y/N) y

WAEHLE TERMINAL-TYP

A - HAZELTINE 1500

C - HEATH 89

E - ADM-3A

G - VDP-80

I - GNAT SYSTEM 10

K - APPLE

M - SUPERBRAIN

O - OSBORNE

Q - CROMEMCO 3102

S - ADDS VIEWPOINT

U - NEC

B - SOROC, TELEVIDEO

D - PERKIN ELMER 1100

F - ADM-31

H - INTERCOLOR

J - TRS-80 II (P&T)

L - VEKTOR GRAPHICS

N - VISUAL 100

P - H 2621, HP-125

R - TRS-80 (FMG)

T - XEROX 820

Y - WECHSEL/VERAENDERE BEREITS EINGERICHTETES TERMINAL

Z - BENUTZERDEFINIERTES TERMINAL-EIGENSCHAFTEN

k

Wenn Sie eines der aufgeführten Terminals gewählt haben, fragt dBASE II Sie anschließend, welches Zeichen Sie zur Anzeige von Makro-Befehlen verwenden wollen (beschrieben im Abschnitt 4, verwendet in Kapitel 4/ADAENDRG.CMD und ausführlich definiert in Kapitel 3 dieses Handbuchs). Siehe Abb. 1.2. Wenn das Und-Zeichen (&) von Ihrem Texteditor nicht anderweitig verwendet wird, geben Sie <RETURN> ein. Andernfalls tippen Sie das Symbol ein, das Sie stattdessen verwenden wollen.

Anfangs werden Sie den fehlerkorrigierenden Dialog benutzen wollen (error correcting dialogue), geben Sie daher <RETURN> auf die entsprechende Frage ein. Dies gestattet es Ihnen, Eingabefehler zu verbessern, ohne daß Sie den gesamten Befehl neu eingeben müssen (siehe Abschnitt 2). Später können Sie auf diese Möglichkeit verzichten, indem Sie den "Y-CHANGE/MODIFY"-Befehl benutzen (siehe oben).

Abb. 1.2a:

ENTER A CHARACTER TO BE USED FOR INDICATING MACROS
OR A RETURN FOR DEFAULT CHARACTER OF AMPERSAND (&)
<RETURN>

TYPE A RETURN IF THE ERROR CORRECTION DIALOGUE IS
TO BE USED OR ANY OTHER KEY IF NO DIALOGUE IS WANTED:
<RETURN>

ENTER OPERATING SYSTEM

A - CP/M 1.4 OR CP/M 2.2

B - CDOS SYSTEM

C - CROMIX-SYSTEM

D - MP/M II SYSTEM

a

TYPE "Y" TO SAVE, ANY OTHER CHAR TO ABORT INSTALL
y

SAVING INSTALLATION PARAMETERS

Abb. 1.2b:

GEBEN SIE EIN ZEICHEN ZUR ANZEIGE VON MAKROS EIN
ODER RETURN FÜR STANDARDMAESSIGES UND-ZEICHEN (&)
<RETURN>

GEBEN SIE RETURN EIN WENN DER FEHLERKORRIGIERENDE DIALOG
VERWENDET WERDEN SOLL ODER IRGEND EIN ANDERES ZEICHEN, WENN NICHT:
<RETURN>

WÄHLE BETRIEBSSYSTEM

A - CP/M 1.4 OR CP/M 2.2

B - CDOS SYSTEM

C - CROMIX-SYSTEM

D - MP/M II SYSTEM

a

DRUECKE "Y", UM EINRICHTUNG ABZUSPEICHERN, ODER EINE ANDERE
TASTE, UM VORGANG ABZUBRECHEN
y

PARAMETER DER ANPASSUNG WERDEN ABGESPEICHERT

Am Ende der Anpassungs-Prozedur können Sie diese gültig machen, indem Sie 'y' eingeben, oder Sie können die Anpassung abbrechen, indem Sie eine andere Taste drücken und damit zu der Terminal-Konfiguration zurückkehren, die vor diesem Arbeitsgang eingestellt war.

Wenn Ihr Terminal nicht in der Liste aufgeführt war und Sie daher 'Z' eingegeben haben, so listet dBASE II die Terminal-Befehle auf, die zur Vervollständigung der Einrichtungs-Prozedur für Ihr Terminal/Ihren Bildschirm benötigt werden. Sie können diese Anpassungs-Prozedur auch wählen, um die normalen Werte für das von Ihnen gewählte Terminal zu modifizieren (beispielsweise die Ausgabe invertierter Texte, d.h. schwarz auf weiß, durch bestimmte Code-Folgen).

Abb. 1.3a:

USER SUPPLIED SPECS ROUTINE

FOR THIS METHOD, YOU WILL NEED THE HEX OR DECIMAL CODES THAT CAN BE SENT FROM YOUR COMPUTER TO THE VIDEO TERMINAL TO CONTROL IT

THE CODES OR SEQUENCES THAT YOU WILL NEED ARE:

DELETE A CHAR SEQUENCE
DIRECT CURSOR POSITIONING SEQUENCE
CLEAR SCREEN COMMAND
HOME CURSOR COMMAND
(CLEAR AND HOME CAN BE COMBINED)
OPTIONAL: BRIGHT/DIM COMMANDS OR VIDEO/REVERSE COMMANDS

TYPE "Y" IF YOU WISH TO CONTINUE

y

Abb. 1.3b:

EINGABE VOM BENUTZER DEFINIERTER CODEFOLGEN

FUER DIESES VERFAHREN MUESSEN SIE DIE HEXADEZIMALEN ODER DEZIMALEN CODES KENNEN, DIE IHR COMPUTER AN IHR VIDEO-TERMINAL SENDEN KANN, UM ES ZU STEUERN

DIE CODES ODER CODE-FOLGEN DIE SIE BENOETIGEN SIND:

FOLGE ZUM LOESCHEN EINES ZEICHENS
FOLGE ZUR DIREKTEN POSITIONIERUNG DES CURSORS
BILDSCHIRM-LOESCH-CODE
CODE FUER CURSOR HOME (NACH LINKS OBEN)
(LOESCHEN UND HOME KOENNEN KOMBINIERT SEIN)
WAHLWEISE: BEFEHLE ZUR HELL/DUNKEL-STEUERUNG ODER NEGATIVEN SCHRIFTDARSTELLUNG

DRUCKEN SIE "Y" UM FORTZUFAHREN

y

EINFÜHRUNG...8

Wenn Sie die Codes kennen, die Ihr Terminal zu den oben genannten Operationen veranlassen, dann geben Sie 'y' ein. dBASE II fordert Sie dann zur Eingabe der jeweiligen Codefolge auf. Das unten gezeigte Beispiel betrifft ein IBM 3101/12 Terminal. Dieses Terminal besitzt keine Hervorhebung durch hellere (bright) oder negative (reverse video) Schrift, daher wurde bei den betreffenden Fragen <RETURN> eingegeben.

dBASE II zeigt jeweils die alten Werte der Steuerzeichen an, wir haben die neu eingetippten Werte in zwei einfache Anführungszeichen (') eingeschlossen. BITTE GEBEN SIE DIESE ZEICHEN (') NICHT EIN!

Abb. 1.4a

WILL YOU BE ENTERING COMMANDS AS HEX OR
DECIMAL? TYPE "D" FOR DECIMAL OR
"H" FOR HEXADECIMAL

h

COMMANDS ARE ENTERED AS A SEQUENCE OF NUMBERS
TYPE A CARRIAGE RETURN TO END A SEQUENCE

NOW ENTER THE CODES FOR CHARACTER DELETION THIS IS THE SEQUENCE "BACKSPACE, SPACE,
BACKSPACE"

ON MOST TERMINALS IF THIS IS TRUE FOR YOUR TERMINAL, THEN TYPE "Y"

y

Abb. 1.4b

MOECHTEN SIE DIE CODES IN HEX ODER DEZIMAL EINGEBEN?
DRUECKEN SIE "D" FUER DEZIMAL ODER
"H" FUER HEXADEZIMAL

h

DIE BEFEHLE WERDEN ALS FOLGE VON ZAHLEN EINGEGEBEN
DRUECKEN SIE RETURN UM EINE FOLGE ZU BEENDEN

GEBEN SIE NUN DIE CODES ZUM LOESCHEN EINES ZEICHENS EIN
AUF DEN MEISTEN TERMINALS IST DIES DIE BEFEHLSFOLGE:
"BACKSPACE, SPACE, BACKSPACE"

WENN DAS FUER IHR TERMINAL ZUTRIFFT, DRUECKEN SIE "Y"

y

“Backspace” ist der Code, der den Cursor ein Zeichen rückwärts bewegt. Die Taste dafür kann auf Ihrem Rechner unterschiedlich bezeichnet sein, z. B. “BSP”, “RUB”, “RUBOUT” oder mit einem Links-Pfeil “<--”. Falls Sie mehrere dieser Bezeichnungen auf verschiedenen Tasten finden, haben sie etwas unterschiedliche Funktionen - am besten probieren Sie es aus!

“Space” bezeichnet das Leerzeichen (BLANK), das von der Leertaste Ihrer Tastatur erzeugt wird.

Abb. 1.5a

- DIRECT CURSOR POSITIONING -

THE CURSOR CONTROL SEQUENCE IS
USUALLY A 3 TO 4 BYTE SEQUENCE. THE
FIRST ONE OR TWO BYTES ARE USUALLY
FIXED AND THE REMAINING BYTES CONTAIN
THE LINE AND COLUMN NUMBERS

FIRST, ENTER THE POSITION IN THE
SEQUENCE THAT HOLDS THE COLUMN
NUMBER

'4' <RETURN>

NEXT, ENTER THE POSITION IN THE SEQUENCE THAT HOLDS THE LINE NUMBER

'3' <RETURN>

MANY TERMINALS ADD A CONSTANT TO THE
LINE AND COLUMN NUMBERS. ENTER THE
CONSTANT BIAS FOR YOUR TERMINAL

'20' <RETURN>

NOW ENTER THE SKELETON FOR THE DIRECT
CURSOR COMMAND. ENTER A ZERO IN
THE PLACES WHERE COLUMN AND LINE
NUMBERS GO
(11 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 03 '1B' <RETURN>

ENTER CONTROL CODE BYTE 2: 00 '59' <RETURN>

ENTER CONTROL CODE BYTE 3: 00 '0' <RETURN>

ENTER CONTROL CODE BYTE 4: 00 '0' <RETURN>

ENTER CONTROL CODE BYTE 5: 00 <RETURN>

IS THIS CORRECT (Y/N)? y

Abb. 1.5b

- DIREKTE CURSOR POSITIONIERUNG -

DIE STEUERFOLGE FÜR DEN CURSOR
BESTEHT ÜBlicherweise AUS 3 ODER 4 BYTES.
DIE ERSTEN EIN ODER ZWEI BYTES SIND IN DER REGEL
KONSTANT UND DIE ÜBRIGEN BYTES ENTHALTEN
DIE ZEILEN- UND SPALTENNUMMER

GEBEN SIE ZUERST DIE POSITION IN DER FOLGE
AN, WELCHE DIE SPALTENNUMMER ENTHÄLT
'4' <RETURN>
NUN GEBEN SIE DIE POSITION IN DER FOLGE
AN WELCHE DIE ZEILENNUMMER ENTHÄLT
'3' <RETURN>

VIELE TERMINALS ADDIEREN EINE KONSTANTE
ZU DEN ZEILEN- UND SPALTENNUMMERN.
GEBEN SIE DEN KONSTANTEN VERSATZWERT FÜR IHR TERMINAL AN:
'20' <RETURN>

NUN GEBEN SIE DAS GERÜST FÜR EINEN DIREKTEN
CURSOR POSITIONIERBEFEHL EIN. GEBEN SIE EINE NULL
AN DEN STELLEN EIN, AN DIE SPALTEN- UND ZEILENNUMMER
GEOHÖREN
(11 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 03 '1B' <RETURN>
ENTER CONTROL CODE BYTE 2: 00 '59' <RETURN>
ENTER CONTROL CODE BYTE 3: 00 '0' <RETURN>
ENTER CONTROL CODE BYTE 4: 00 '0' <RETURN>
ENTER CONTROL CODE BYTE 5: 00 <RETURN>

IST DIES KORREKT (Y/N)? y

Abb. 1.6a

- DIM/BRIGHT VIDEO/REVERSE VIDEO -

ENTER THE COMMAND THAT WILL SWITCH TO
HIGH INTENSITY OR NORMAL VIDEO
(5 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 1D <RETURN>

IS THIS CORRECT (Y/N)? y

ENTER THE COMMAND THAT WILL SWITCH TO LOW INTENSITY
OR REVERSE VIDEO
(5 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 1D <RETURN>

IS THIS CORRECT (Y/N)? y

Abb. 1.6b

- HELL/DUNKEL POSITIV/NEGATIV-DARSTELLUNG -

GEBEN SIE DIE BEFEHLSFOLGE EIN DIE AUF
HELLE BZW. POSITIVE DARSTELLUNG UMSCHALTET
(5 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 1D <RETURN>

IST DIES KORREKT (Y/N)? y

GEBEN SIE DIE BEFEHLSFOLGE EIN DIE AUF DUNKLE BZW.
NEGATIVE DARSTELLUNG UMSCHALTET
(5 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 1D <RETURN>

IST DIES KORREKT (Y/N)? y

Abb. 1.7a

- CLEAR AND HOME COMMAND(S) -

ENTER THE COMMAND(S) THAT WILL CLEAR
THE SCREEN AND PLACE THE CURSOR IN
THE UPPER LEFT CORNER OF THE TERMINAL
(11 BYTE MAX)

ENTER CONTROL CODE 1: 0C '1B' <RETURN>
ENTER CONTROL CODE 2: 00 '4C' <RETURN>
ENTER CONTROL CODE BYTE 3: 00 <RETURN>

IS THIS CORRECT (Y/N)? y

Abb. 1.7b

- BILDSCHIRM LOESCHEN UND CURSOR HOME BEFEHL(E) -

GEBEN SIE DIE BEFEHLSFOLGE(N) EIN, DIE DEN
BILDSCHIRM LOESCHEN UND DEN CURSOR IN DIE
LINKE OBERE ECKE DES TERMINALS SCHICKEN
(11 BYTE MAX)

ENTER CONTROL CODE 1: 0C '1B' <RETURN>
ENTER CONTROL CODE 2: 00 '4C' <RETURN>
ENTER CONTROL CODE BYTE 3: 00 <RETURN>

IST DIES KORREKT (Y/N)? y

Abb. 1.8a

ENTER THE COMMANDS TO BE ISSUED WHEN
ENTERING THE FULL-SCREEN EDITING MODE
(IF ANY)
(11 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 00 <RETURN>

IS THIS CORRECT (Y/N)? y

Abb. 1.9a

ENTER THE COMMANDS TO BE ISSUED WHEN
LEAVING THE FULL-SCREEN EDITING MODE

SUGGESTION: USE DIRECT CURSOR POSI-
TIONING TO PUT CURSOR ON THE BOTTOM
LINE OF THE SCREEN
(11 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 1D '1B' <RETURN>
ENTER CONTROL CODE BYTE 2: 17 '59' <RETURN>
ENTER CONTROL CODE BYTE 3: 03 '31' <RETURN>
ENTER CONTROL CODE BYTE 4: 00 '20' <RETURN>
ENTER CONTROL CODE BYTE 5: 2E <RETURN>

IS THIS CORRECT (Y/N)? y

Abb. 1.9b

GEBEN SIE DIE BEFEHLE EIN, DIE BEIM VERLASSEN DER
BILDSCHIRM-ORIENTIERTEN EDITIERUNG AUSGEGEBEN WERDEN SOLLEN

VORSCHLAG: VERWENDEN SIE DIREKTE CURSOR-POSITIONIERUNG
UM DEN CURSOR IN DIE UNTERSTE BILDSCHIRMZEILE ZU BRINGEN
(11 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 1D '1B' <RETURN>
ENTER CONTROL CODE BYTE 2: 17 '59' <RETURN>
ENTER CONTROL CODE BYTE 3: 03 '31' <RETURN>
ENTER CONTROL CODE BYTE 4: 00 '20' <RETURN>
ENTER CONTROL CODE BYTE 5: 2E <RETURN>

IST DIES KORREKT (Y/N)? y

Abb. 1.10a

ENTER THE COMMAND THAT WILL SWITCH TO
STANDARD INTENSITY OR NORMAL VIDEO
TO RESET THE SCREEN AFTER FULL SCREEN
OPERATIONS
(5 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 1D <RETURN>

IS THIS CORRECT (Y/N)? y

Abb. 1.10b

GEBEN SIE DEN BEFEHL EIN DER NACH DER BILDSCHIRM-ORIENTIERTEN
EDITIERUNG AUF NORMALE HELLIGKEIT BZW. POSITIVE DARSTELLUNG
ZURUECKSCHALTET
(5 BYTE MAX)

ENTER CONTROL CODE BYTE 1: 1D <RETURN>

IST DIES KORREKT (Y/N)? y

Abb. 1.11a

ENTER THE SCREEN WIDTH

38 <RETURN>

NEXT, ENTER THE NUMBER OF LINES ON THE SCREEN

24 <RETURN>

Abb. 1.11b

GEBEN SIE DIE BILDSCHIRMBREITE EIN

38 <RETURN>

GEBEN SIE NUN DIE ANZAHL ZEILEN AUF DEM BILDSCHIRM AN

24 <RETURN>

Abb. 1.12a

ENTER A CHARACTER TO BE USED FOR INDICATING
MACROS OR A RETURN FOR THE DEFAULT CHARACTER OF
AMPERSAND (&) <RETURN>

TYPE A RETURN IF THE ERROR CORRECTION DIALOGUE IS
TO BE USED OR ANY OTHER KEY IF NO DIALOGUE IS
WANTED: <RETURN>

ENTER OPERATING SYSTEM

A - CP/M 1.4 OR CP/M 2.2

B - CDOS SYSTEM

C - CROMIX-SYSTEM

D - MP/M II SYSTEM

etc.

TYPE "Y" TO SAVE, AND OTHER CHAR TO ABORT INSTALL

y
SAVING INSTALLATION PARAMETERS

Abb. 1.12b

GEBEN SIE EIN ZEICHEN EIN, DAS ZUR ANZEIGE VON MAKROS
BENUTZT WERDEN SOLL ODER RETURN, WENN STANDARDMAESSIG
DAS UND-ZEICHEN (&) BENUTZT WERDEN SOLL <RETURN>

DRUECKEN SIE RETURN WENN DER FEHLERKORRIGIERENDE DIALOG
VERWENDET WERDEN SOLL ODER EINE ANDERE TASTE, WENN KEIN
DIALOG GEWUENSCHT WIRD: <RETURN>

WAEHLEN SIE IHR BETRIEBSSYSTEM

A - CP/M 1.4 OR CP/M 2.2

B - CDOS SYSTEM

C - CROMIX-SYSTEM

D - MP/M II SYSTEM

etc.

DRUECKEN SIE "Y" ZUR ABSPEICHERUNG, EINE ANDERE TASTE FUEHRT ZUM ABBRUCH DER
ANPASSUNG

y
ANPASSUNGS-PARAMETER WERDEN ABGESPEICHERT

Abb. 1.13a

MODIFY EXISTING SPECS ROUTINE

FOR THIS METHOD, YOU WILL NEED THE HEX OR DECIMAL CODES THAT CAN BE SENT FROM YOUR COMPUTER TO THE VIDEO TERMINAL TO CONTROL IT

TYPE "Y" IF YOU WISH TO CONTINUE

y

WILL YOU BE ENTERING COMMANDS AS HEX OR DECIMAL?

TYPE "D" FOR DECIMAL OR "H" FOR HEXADECIMAL

h

COMMANDS ARE ENTERED AS A SEQUENCE OF NUMBERS

TYPE A CARRIAGE RETURN TO END A SEQUENCE

- 1 - DELETE A CHAR SEQUENCE
- 2 - DIRECT CURSOR POSITIONING SEQUENCE
- 3 - CLEAR AND HOME SCREEN COMMAND
- 4 - BRIGHT/STD VIDEO COMMANDS
- 5 - DIM/REVERSE VIDEO COMMANDS
- 6 - INITIALIZATION SEQUENCE
- 7 - EXIT SEQUENCE
- 8 - RESET TO STANDARD VIDEO MODE
- 9 - ALTER SCREEN SIZE

SELECT ITEM TO CHANGE

ANY CHAR OTHER THAN 1-9 TERMINATES SESSION

'7'

Abb. 1.13b

MODIFIZIERE VORHANDENE TERMINAL-ROUTINEN

BEI DIESER METHODE BRAUCHEN SIE DIE HEX- ODER DEZIMALCODES
DIE IHR RECHNER AN DAS TERMINAL SENDEN KANN,
UM ES ZU STEUERN

DRUECKEN SIE "Y" UM FORTZUFAHREN

y

MOECHTEN SIE DIE CODES HEXADEZIMAL ODER DEZIMAL EINGEBEN?
DRUECKEN SIE "D" FUER DEZIMAL ODER "H" FUER HEXADEZIMAL

h

BEFEHLE WERDEN ALS EINE FOLGE VON ZAHLEN EINGEGEBEN
DRUECKEN SIE RETURN UM EINE FOLGE ZU BEENDEN

- 1 - FOLGE ZUM LOESCHEN EINES ZEICHENS
- 2 - FOLGE ZUR DIREKTEN CURSOR-POSITIONIERUNG
- 3 - BILDSCHIRMLOESCH- UND CURSOR-HOME-BEFEHL
- 4 - HELLE/POSITIVE DARSTELLUNG
- 5 - DUNKLE/NEGATIVE DARSTELLUNG
- 6 - INITIALISIERUNGS-FOLGE BILDSCHIRMEDITIERUNG
- 7 - FOLGE BEI AUSTRITT AUS BILDSCHIRMEDITIERUNG
- 8 - RUECKSETZEN AUF NORMALE SCHRIFTDARSTELLUNG
- 9 - AENDERE BILDSCHIRM-GROESSE

WAEHLEN SIE DIE ZU AENDERNDE GRUPPE
IRGENDEIN ANDERES ZEICHEN ALS 1-9 BEENDET DIE SITZUNG
'7'

Wenn Sie ein bereits eingerichtetes dBASE II-System modifizieren wollen, geben Sie "install" dann 'y' oder 'n' als Antwort auf die Frage nach der bildschirmorientierten Editierung (FULL SCREEN EDITING), dann wählen Sie die 'y' Alternative in der Liste der angebotenen Terminals (siehe Abb. 1.1). Im folgenden Beispiel wollten wir die Codefolge ändern, die beim Verlassen der bildschirmorientierten Arbeitsweise den Cursor auf die 23. statt die 17. Zeile setzen sollte. (Sie werden dies in später in diesem Handbuch besprochenen dBASE-Anweisungen erklärt finden.)

Beachten Sie, daß die Zahlen im hexadezimalen Format eingegeben werden und daß die Zeilen von 0 bis 23, die Spalten von 0 bis 79 numeriert sind.

Abb. 1.14a

ENTER COMMANDS TO BE ISSUED WHEN LEAVING THE FULL-
SCREEN EDITING MODE

SUGGESTION: USE DIRECT CURSOR POSITIONING TO PUT
CURSOR ON THE BOTTOM LINE OF THE SCREEN

(11 BYTE MAX)

CURRENT SEQUENCE:

1B

59

31

20

IS THIS CORRECT (Y/N)? n

ENTER CONTROL CODE BYTE 1: 1B '1B' <RETURN>

ENTER CONTROL CODE BYTE 2: 59 '59' <RETURN>

ENTER CONTROL CODE BYTE 3: 31 '36' <RETURN>

ENTER CONTROL CODE BYTE 4: 20 '20' <RETURN>

ENTER CONTROL CODE BYTE 5: 00 <RETURN>

IS THIS CORRECT (Y/N)? y

1 - DELETE A CHAR SEQUENCE

2 - DIRECT CURSOR POSITIONING SEQUENCE

3 - CLEAR AND HOME SCREEN COMMAND

4 - BRIGH/STD VIDEO COMMANDS

5 - DIM/REVERSE VIDEO COMMANDS

6 - INITIALIZATION SEQUENCE

7 - EXIT SEQUENCE

8 - RESET TO STANDARD VIDEO MODE

9 - ALTER SCREEN SIZE

SELECT ITEM TO CHANGE

ANY CHAR OTHER THAN 1-9 TERMINATES SESSION

<RETURN>

ENTER A CHARACTER TO BE USED FOR INDICATING MACROS
OR A RETURN FOR DEFAULT CHARACTER OF AMPERSAND (&)

<RETURN>

TYPE A RETURN IF THE ERROR CORRECTION DIALOGUE IS
TO BE USED OR ANY OTHER KEY IF NO DIALOGUE IS WANTED:

<RETURN>

ENTER OPERATING SYSTEM

A - CP/M 1.4 OR CP/M 2.2

B - CDOS SYSTEM

C - CROMIX-SYSTEM

D - MP/M II SYSTEM

etc.

TYPE "Y" TO SAVE, ANY OTHER CHAR TO ABORT INSTALL

y

SAVING INSTALLATION PARAMETERS

Abb. 1.14b

GEBEN SIE DIE BEFEHLE EIN DIE BEIM VERLASSEN DER
BILDSCHIRM-ORIENTIERTEN EDITIERUNG AUSGEGEBEN WERDEN SOLLEN

VORSCHLAG: VERWENDEN SIE DIREKTE CURSOR-POSITIONIERUNG UM DEN
CURSOR IN DIE UNTERSTE ZEILE DES BILDSCHIRMS ZU BRINGEN
(11 BYTE MAX)

DERZEITIGE FOLGE:

1B
59
31
20

IST DIES KORREKT (Y/N)? n

ENTER CONTROL CODE BYTE 1: 1B '1B' <RETURN>
ENTER CONTROL CODE BYTE 2: 59 '59' <RETURN>
ENTER CONTROL CODE BYTE 3: 31 '36' <RETURN>
ENTER CONTROL CODE BYTE 4: 20 '20' <RETURN>
ENTER CONTROL CODE BYTE 5: 00 <RETURN>

IST DIES KORREKT (Y/N)? y

- 1 - FOLGE ZUM LOESCHEN EINES ZEICHENS
- 2 - FOLGE ZUR DIREKTEN CURSOR-POSITIONIERUNG
- 3 - BILDSCHIRMLOESCH- UND CURSOR-HOME-BEFEHL
- 4 - HELLE/POSITIVE DARSTELLUNG
- 5 - DUNKLE/NEGATIVE DARSTELLUNG
- 6 - INITIALISIERUNGS-FOLGE BILDSCHIRMEDITIERUNG
- 7 - FOLGE BEI AUSTRITT AUS BILDSCHIRMEDITIERUNG
- 8 - RUECKSETZEN AUF NORMALE SCHRIFTDARSTELLUNG
- 9 - AENDERE BILDSCHIRM-GROESSE

WAEHLEN SIE DIE ZU AENDERNDE GRUPPE
IRGEND EIN ANDERES ZEICHEN ALS 1-9 BEENDET DIE SITZUNG

<RETURN>

GEBEN SIE EIN ZEICHEN EIN DAS ZUR ANZEIGE VON MAKROS BENUTZT WERDEN SOLL ODER
RETURN FUER DAS STANDARDMAESSIGE UND-ZEICHEN (&)

<RETURN>

DRUECKEN SIE RETURN WENN DER FEHLERKORRIGIERENDE DIALOG
BENUTZT WERDEN SOLL ODER IRGEND EINE ANDERE TASTE WENN KEIN
DIALOG GEWUENSCHT WIRD

<RETURN>

WAEHLEN SIE IHR BETRIEBSSYSTEM

- A - CP/M 1.4 OR CP/M 2.2
- B - CDOS SYSTEM
- C - CROMIX-SYSTEM
- D - MP/M II SYSTEM
- etc.

DRUECKEN SIE "Y" ZUR ABSPEICHERUNG ODER EINE ANDERE TASTE
ZUM ABRUCH DER ANPASSUNG

y

ANPASSUNGS-PARAMETER WERDEN ABGESPEICHERT

Das dBASE II-System ist jetzt fertig auf Ihrem Rechner angepaßt und Sie können es sofort benutzen. Starten Sie dBASE II, indem Sie 'dbase' eingeben.

Eine Antwortzeile fragt nach dem Datum. Wenn Sie ein Datum eingeben, so wird es jedesmal, wenn Sie etwas einfügen oder löschen, in Ihren Dateien als letzter Zugriff vermerkt. Das kann recht nützlich sein, um die Übersicht über Veränderungen Ihrer Dateien zu behalten. Wenn Sie das Datum ignorieren wollen, drücken Sie einfach <RETURN>.

dBASE II lädt sich in den Speicher Ihres Rechners, gibt eine Meldung aus und schreibt den "Prompt"-Punkt (.), um anzuzeigen, daß es bereit für Ihre Kommandos ist.

Um Ihnen zu zeigen, wie leistungsfähig und leicht zu bedienen dBASE II tatsächlich ist, werden wir als allererstes eine Datenbank erzeugen und Daten in ihr abspeichern.

Das wird nur ein paar Minuten beanspruchen.

Hinweise für die Nutzer von 16-Bit-Mikrocomputern

Da dBASE II ursprünglich für 8-Bit-Rechner mit dem Betriebssystem CP/M-80 entwickelt wurde, sind alle Angaben und Beschreibungen im vorliegenden Handbuch auf diese Konfiguration ausgerichtet. Für 16-Bit-Systeme ergeben sich Abweichungen in zwei Punkten:

1. hinsichtlich der Hard- und Software-Umgebung (vgl. S. 11):

Benötigt wird:

- ein Mikrocomputersystem auf der Basis einer 8086- oder 8088-CPU,
- mindestens 128 KBytes Arbeitsspeicher,
- ein 16-Bit-Betriebssystem (z.B. CP/M-86, Concurrent CP/M-86, MSDOS, PC-DOS usw.).

Für die Ausstattung mit Peripheriegeräten gelten die Angaben für 8-Bit-Systeme.

2. hinsichtlich der Bezeichnung von Dateien:

Die meisten 16-Bit-Betriebssysteme verwenden zur Kennzeichnung von Maschinenprogrammdateien den Dateityp .CMD, der in der 8-Bit-Version zur Identifizierung von dBASE-Befehlsdateien dient. Aus diesem Grunde erhalten dBASE-Befehlsdateien in der 16-Bit-Version den Dateityp .PRG. Alle anderen Dateibezeichnungskonventionen entsprechen den Angaben für 8-Bit-Systeme (siehe Kapitel 2, Abschnitt 1). Die Unterschiede sind noch einmal in untenstehender Tabelle zusammengefaßt:

DATEITYP		
	8-Bit-Version (im Handbuch verwendete Bezeichnung)	16-Bit-Version
Maschinenprogrammdateien	.COM	.CMD
dBASE-Befehlsdateien	.CMD	.PRG

Neben diesen Abweichungen enthält die 16-Bit-Version von dBASE II folgende Erweiterungen gegenüber der 8-Bit-Version:

1. Beeinflussung der Farbdarstellung am Bildschirm

Bei den Sichtgeräten IBM PC (oder kompatible), Victor, TI Professional können die Bildschirmfarben durch einen zusätzlichen Befehl beeinflusst werden (Voraussetzung ist, daß eines dieser Sichtgeräte bei der Installation angegeben wurde):

SET COLOR TO <n1>,<n2>

Dabei entspricht <n1> der Farb- und/oder Attributauswahl für die Negativdarstellung bzw. Darstellung in einer anderen Helligkeitsstufe, während <n2> die Farb- und/oder Attributauswahl für die Standardausgabe darstellt.

Folgende Werte können für <n1> und <n2> angegeben werden:

<n1> bzw. <n2>	Farbbildschirm	Schwarz-Weiß-Bildsch.
0	schwarz	Anzeige unterdrückt
1	blau	Anzeige unterstrichen
2	grün	
3	cyan	
4	rot	
5	magenta	
6	gelb	
7	weiß	
112	Negativdarstellung (schwarz auf weiß) bei Farb- und S/W-Bildschirm	

Voreinstellung von <n1> und <n2> ist

<n1> = 112 - Negativdarstellung mit weißem Hintergrund,

<n2> = 7 - weiß auf schwarzem Hintergrund

Beispiel: SET COLOR TO 4,1

Anzeige bei Negativdarstellung bzw. einer anderen Helligkeitsstufe in rot, Anzeige bei Normaldarstellung in blau

2. Belegung der Funktionstasten

Bei den unter 1. aufgeführten Sichtgeräten werden deren Funktionstasten standardmäßig bestimmten dBASE-Befehlen zugeordnet. Die Belegung der Funktionstasten kann jedoch jederzeit geändert werden durch den Befehl

SET F<n> TO <Belegung>

Dabei gibt <n> die Nummer der Funktionstaste an und <Belegung> den neuen Befehl, der der Funktionstaste zugeordnet werden soll. Dabei muß der Befehl mit einem Semikolon (;) abgeschlossen werden und komplett in Hochkomma (') oder Anführungszeichen (") eingeschlossen sein. Das Semikolon ersetzt hier die RETURN- bzw. ENTER-Taste. Soll der Befehl noch durch eine Eingabe von der Tastatur aus ergänzt werden, entfällt das Semikolon. Falls einer Funktionstaste eine Folge von Befehlen zugeordnet werden soll, müssen alle Einzelbefehle bis auf den letzten durch Semikolon abgeschlossen werden.

Voreinstellung der Funktionstastenbelegung:

Funktionstaste	Belegung
F1	'HELP'
F2	'DISPLAY;'
F3	'LIST;'
F4	'LIST FILES;'
F5	'LIST STRU;'
F6	'LIST STAT;'
F7	'LIST MEMO;'
F8	'CREATE;'
F9	'APPEND;'
F10	'EDIT *;'

Beispiele: SET F3 TO "BROWSE;"

Ordnet der Funktionstaste F3 den Befehl BROWSE zu (Befehl wird sofort ausgeführt, da ";" der ENTER-Taste entspricht).

SET F8 TO "DISPLAY RECORD"

Bei Betätigung der Funktionstaste F8 erscheint der Befehl DISPLAY RECORD auf dem Bildschirm; der Anwender muß noch die Nummer des gewünschten Satzes angeben und die ENTER-Taste betätigen.

3. LOCK und UNLOCK für Concurrent CP/M-86

Um beim Mehrbenutzerbetrieb unter Concurrent CP/M-86 die gleichzeitige Änderung eines Datensatzes durch verschiedene Benutzer zu verhindern, stehen unter diesem Betriebssystem die Funktion LOCK und der Befehl UNLOCK zur Verfügung, deren Wirkungsweise im folgenden kurz beschrieben wird.

LOCK()

Die Funktion LOCK sperrt den aktuellen Datensatz für den Zugriff durch andere Benutzer, falls er nicht bereits durch einen anderen Benutzer zuvor gesperrt wurde. Sie wird durch die Befehle "?" und "IF" ausgelöst. Ist der aktuelle Satz noch nicht gesperrt (d.h. durch einen anderen Benutzer belegt), so liefert die Funktion den Wahrheitswert "wahr" (.T.), wobei gleichzeitig der Satz für andere Benutzer gesperrt wird. Andernfalls liefert die Funktion den Wahrheitswert "falsch" (.F.). Die Klammern dienen lediglich der Unterscheidung von gleichnamigen Variablen.

UNLOCK

Dieser Befehl gibt einen gesperrten Satz wieder für alle Benutzer frei.

Beispiel: (Auszug aus einer Befehlsdatei)

```
IF LOCK( )  
  <Anweisungen, falls Satz verfügbar >  
  UNLOCK  
ELSE  
  <Anweisungen, falls Satz nicht verfügbar >  
ENDIF
```

2. Kapitel: Die Anwendung von dBASE II

Inhaltsverzeichnis

Abschnitt 1: Grundsätzliches über dBASE

Datenbank Grundlagen.....	25
Kurze Einführung in die Organisation einer Datenbank.....	28
Datensätze, Dateien und Daten-Typen.....	29
Feld-Namen.....	30
Dateitypen.....	31
Zusammenfassung der Operatoren.....	33
Zusammenfassung der Funktionen.....	34
Zusammenfassung der Befehle.....	35
Befehle nach Funktionsgruppen geordnet.....	40
Datei-Struktur.....	40
Datei-Bearbeitung.....	41
Sortierbefehle.....	42
Bearbeitung zweier Datenbanken.....	42
Editieren, Aktualisieren und Verändern von Daten.....	43
Gebrauch von Variablen.....	44
Interaktive Eingabe (Dialoge).....	44
Suchen.....	45
Ausgabe.....	45
Programmierung.....	45

Abschnitt 2: Wie man eine Datenbank erzeugt und mit ihr umgeht

Erläuterung einiger Begriffe.....	47
Wie man eine Datenbank erzeugt (CREATE).....	50
Eingabe von Daten in Ihre neue Datenbank.....	54
Ändern von Daten mit dem EDIT-Befehl.....	57
Allgemeine bildschirmorientierte Tastatur-Befehle.....	58
CONTROL-Funktionen	
Der fehlerkorrigierende Dialog.....	60
Befehls-Erweiterung mit Auswahlkriterien am Beispiel LIST.....	61
Ausgabe von Daten mit dem DISPLAY-Befehl.....	64
Aufsuchen eines Datensatzes mit GO oder GOTO und SKIP.....	66
Der interaktive Frage-Befehl "?".....	68
Eingeben weiterer Datensätze mit APPEND und INSERT.....	70
Datenbankpflege mit DELETE, RECALL, PACK.....	73
Zusammenfassung.....	77

Abschnitt 3: Arithmetische Funktionen und Manipulationen der Datenbank

Der Gebrauch von Auswahlkriterien.....	79
Konstante und Variable (STORE).....	80
Konstante	80
Variable	81
Logische Operatoren.....	85
arithmetische Operatoren.....	85
Vergleichsoperatoren	86
logische Operatoren.....	87
logische \$-Substring-Such-Operatoren.....	90
String-Operatoren	92
Verändern einer leeren Datenbank-Struktur (MODIFY).....	94
Kopieren von Datenbanken und Datenstrukturen (COPY).....	95
Hinzufügen und Löschen von Feldern mit Daten in der Datenbank.....	99
Austausch von dBASE-Dateien und CP/M-Dateien mit COPY, APPEND.....	102
Umbenennen von Datensatz-Feldern mit COPY und APPEND.....	104
Veränderung von Daten mit REPLACE, CHANGE.....	106
Organisation Ihrer Datenbank mit SORT und INDEX.....	110
Suchen von Daten mit FIND und LOCATE.....	115
Berichtsauswertung der Daten mit REPORT.....	118
Automatisches Zählen und Aufsummieren mit COUNT und SUM.....	122
Summieren von Daten und Ausblenden unwichtiger Details (TOTAL).....	123
Zusammenfassung von Abschnitt 3.....	125

Abschnitt 4: Die Programmierung der Datenbank

Schreiben Sie Ihr erstes dBASE-Programm.....	127
MODIFY COMMAND (-file-)	
Auswahl zwischen Alternativen und Fällen von Entscheidungen	
mit IF..ELSE.....	130
Wahl zwischen verschiedenen Alternativen.....	130
Wahl zwischen zwei Alternativen.....	131
Mehrfachauswahlen.....	131
Mehrfachauswahlen mit CASE.....	132
Wiederholen eines Vorganges mit DO WHILE.....	133
Das Aufrufen von Unterprogrammen (Prozeduren).....	135
Die Eingabe von Daten in ein laufendes Programm	
mit WAIT, INPUT, ACCEPT.....	136
Hinweise, wann was zu benutzen ist.....	137
Wie Sie mit @...SAY...GET Daten und Mitteilungen auf	
dem Bildschirm an eine gewünschte Stelle bringen.....	138
Ein Programm, das zusammenfaßt, was wir bisher kennen.....	143
Arbeiten mit mehreren Datenbanken gleichzeitig.....	147
SELECT PRIMARY/SECONDARY	
Allgemein nützliche Systembefehle und Funktionen.....	152
Einige Worte über Programmierung und wie Sie Ihre Anwendung	
analysieren sollten.....	153

Abschnitt 5: Zusatzfunktionen zur Programmierung und Manipulation der Datenbank

Weitere Befehle zur Datenmanipulation.....	157
Verändern von dBASE II-Standard-Werten (SET ...).	164
Mischen von Datensätzen aus zwei Dateien mit UPDATE.....	166
Zusammenfügen ganzer Datenbanken mit JOIN.....	167
Vollständige Bildschirm-Formatierung und Editiermöglichkeiten mit @...SAY...GET... PICTURE.....	168
Formatieren von Druckseiten mit SET FORMAT TO PRINT, @ ...SAY ...USING.....	170
Entwurf und Druck eines Formulars.....	171
Rückblick.....	173

Abschnitt 6: ZIP - der Bildschirm- und Druck-Maskengenerator

Welche Möglichkeiten bietet ZIP.....	175
Anpassen von ZIP an Ihren Rechner/Ihr Terminal.....	176
Wenn Ihr Terminal nicht aufgeführt ist.....	176
Einstellen einer benutzerspezifischen Terminalanpassung.....	178
Auswechseln von Befehlsabkürzungen und Markierungssymbolen.....	179
Kurzer Überblick, was ZIP kann.....	181
Arbeiten mit ZIP.....	183
Zeichnen von horizontalen/vertikalen Linien.....	185
Hilfstafel, Standard-Werte verändern.....	186
Abspeichern Ihres Entwurfs in einer Datei.....	187
Beginnen eines neuen Entwurfs, Quit.....	189
Nie wieder @...SAY...GET programmieren.....	190
Einfügen von dBASE II-Anweisungen in die Formular-Tafeln.....	191
Probleme mit kurzen Feldern und langen Namen.....	192
Zusammenfassung der ZIP-Befehle.....	193
Dynamische Voreinstellungen (Standard-Werte).....	194
Übrige ZIP-Befehle und Symbole.....	194
Display-Unterprogramme für ZWEIDAT.CMD (Beispiel 4.14).....	196

Beachte Kapitel 4 dieses Handbuches, das Auszüge aus einem deutschen Demonstrationspaket für dBASE II enthält. Gelegentlich wird in diesem Kapitel auf diese Programmbeispiele verwiesen.

Abschnitt 1: Grundsätzliches über dBase II

Inhalt

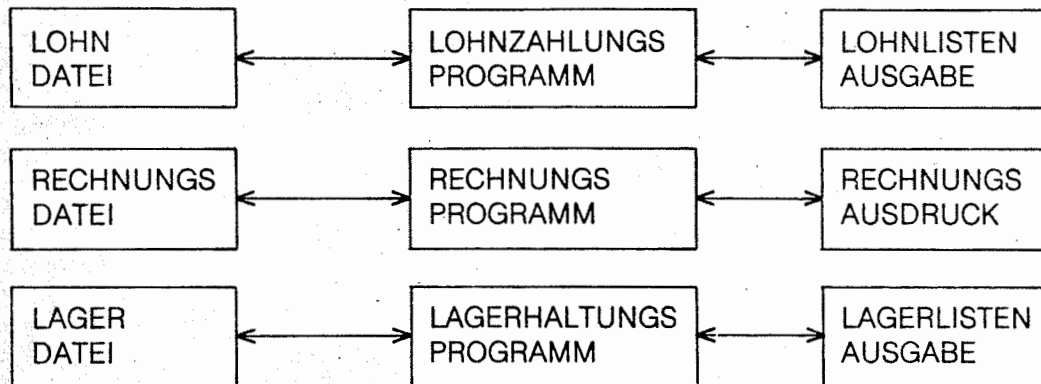
Datenbank Grundlagen.....	25
Kurze Einführung in die Organisation einer Datenbank.....	28
Datensätze, Dateien und Daten-Typen.....	29
Feld-Namen.....	30
Dateitypen.....	31
Zusammenfassung der Operatoren.....	33
Zusammenfassung der Funktionen.....	34
Zusammenfassung der Befehle.....	35
Befehle nach Funktionsgruppen geordnet.....	40
Datei-Struktur.....	40
Datei-Bearbeitung.....	41
Sortierbefehle.....	42
Bearbeitung zweier Datenbanken.....	42
Editieren, Aktualisieren und Verändern von Daten.....	43
Gebrauch von Variablen.....	44
Interaktive Eingabe (Dialoge).....	44
Suchen.....	45
Ausgabe.....	45
Programmierung.....	45

Datenbank-Grundlagen

Ein Datenbank Management System (DBMS) wie dBASE II unterscheidet sich beträchtlich von einem in üblicher Programmierung erstellten Programmkomplex.

Ein herkömmliches Programm ist üblicherweise etwa so strukturiert:

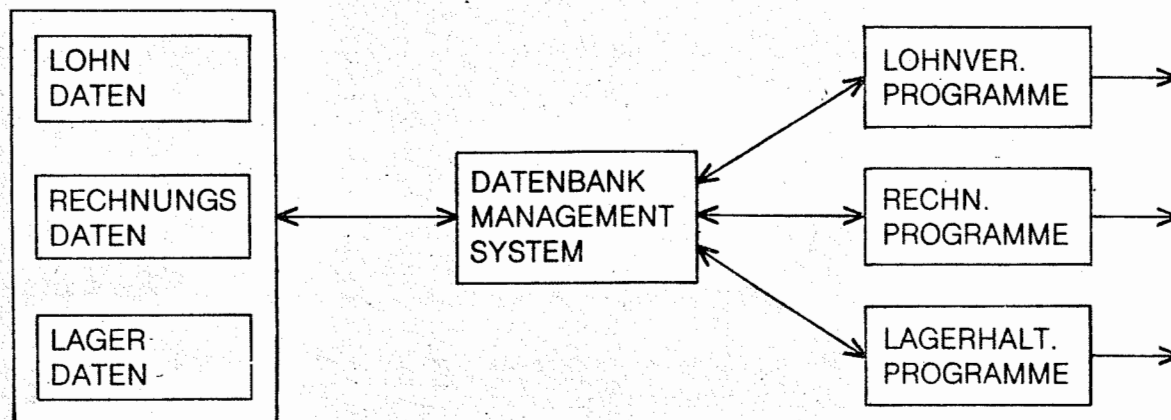
Abb. 1.1:



Das Lohnzahlungs-Programm bearbeitet die Lohn-Datei. Das Rechnungs-Schreib-Programm bearbeitet die Rechnungs-Dateien. Und das Lagerhaltungs-Programm bearbeitet die Lager-Datei. Um Statistiken zu erstellen, die Daten von diesen verschiedenen Dateien verarbeiten, müßte ein neues Programm geschrieben werden, das nicht notwendigerweise auch funktioniert: Die Daten sind vielleicht in den einzelnen Files in so unterschiedlicher Form gespeichert, daß sie nicht nach einem gemeinsamen Schema verarbeitet werden können, oder sie sind so sehr mit anderen Programmen verbunden, daß es der Mühe nicht wert ist, sie von diesen zu isolieren.

Ein Datenbank Management System integriert die Daten und macht es viel einfacher, nützliche Informationen aus Ihren Aufzeichnungen zu gewinnen, anstatt nur Berge von Datensätzen und Zahlen zu produzieren. Prinzipiell ist ein DBMS etwa so konzipiert:

Abb. 1.2:



Der Zugriff zu den Daten und ihre Bearbeitung erfolgt durch das DBMS, nicht durch individuelle Anwendungs-Programme, die erst noch mühsam erstellt werden müssen. Alle Anwendungsprogramme haben Zugriff zu allen Daten. In einem herkömmlichen Programm würde dies eine Vielzahl von mehrfach gespeicherten Daten bedeuten. Neben der Gefahr von Fehleingaben ist die Integrität (Übereinstimmung) der Daten äußerst schwierig zu erreichen, wenn die selben Daten in verschiedenen Dateien mehrfach vorhanden sind: Sie ist fast nie gewährleistet.

Um eine neue Auswertung der Daten in einem Programmsystem zu erreichen, müssen ein neues Programm geschrieben und neue Dateien erzeugt werden. In einem DBMS wird nur ein neues Auswerte-Programm geschrieben, das auf die bereits vorhandenen Datenbanken zugreift. Die Daten müssen nicht umgespeichert werden - das DBMS kümmert sich um die neue Interpretation.

Wenn zu einem Datensatz neue Datenfelder hinzugefügt werden müssen (beispielsweise über die Gehaltsentwicklung in einer Personaldatei), müssen in einem File-Verwaltungs-System die Programme neu geschrieben werden. In einem DBMS haben Erweiterungen und Veränderungen keinen Einfluß auf bestehende Programme, die sich nicht mit der neuen Information befassen: Sie "sehen" sie nicht und merken gar nicht, daß sie existiert.

Es gibt zwei Arten von Datenbank-Management-Systemen: Hierarchische und Relationale. Diese Begriffe beziehen sich auf die Art und Weise, wie die DBMS Daten ordnen.

Ein hierarchisches System tendiert dazu, äußerst kompliziert und schwierig in der Handhabung zu werden, weil die Beziehungen zwischen den Datenelementen mit Mengen, verketteten Listen und Zeigern verwaltet werden. Recht bald gehen Sie unter in Listen von Listen von Listen und Zeigern, die auf Zeiger deuten, die auf Zeiger deuten.

Ein relationales Datenbank-System wie dBASE II ist in seiner Anwendung sehr viel einfacher. Die Information (Daten) wird so dargestellt, wie Sie es gewohnt sind. Die Beziehung zwischen den Datenelementen kann durch zweidimensionale Tabellen dargestellt werden:

Beispiel 1.3:

Rechnungs Nummer	Lieferant	Artikel	Betrag	Nummer
2386	Graphik & Co.	Drucke	23.00	BBQ-747
78622	Braun Gravuren	Druckplatten	397.42	TFS-901
M1883	Luftfracht GmbH.	Versand	97.00	SPT-233

Jede Zeile in der Tabelle nennt man **Datensatz (Record)**. Jede Spalte in der Zeile nennt man **Feld (field)**. Jeder Eintrag in der Tabelle besteht aus einzelnen Werten (nicht aus Mengen oder sonstigen Strukturen). Alle Einträge in einer Spalte (alle Zeilen) müssen vom gleichen Typ sein. Jeder Datensatz (Zeile) ist einmalig, die Anordnung der Datensätze (Zeilen) spielt keine Rolle.

Grundsätzlich wird eine Datenbank mit dBASE immer so wie in Beispiel 1.3 aufgebaut. Sie werden später noch einige praxisnahe Beispiele finden, an Hand derer Sie die Anwendung von dBASE II lernen werden.

Kurze Einführung in die Datenbank-Organisation

Nachdem Sie eine Datenbank eingerichtet haben, wollen Sie Ihre Daten geordnet wieder abrufen.

Bei einigen Datenbanken bestimmt die Reihenfolge, in der Sie die Daten eingeben, die Reihenfolge, in der sie ausgegeben werden können. In vielen Fällen aber werden Sie die Daten in einer davon verschiedenen Weise ordnen wollen.

Mit dBASE II können Sie Ihre Daten mit dem SORT-Befehl oder auch mit der INDEX-Anweisung sortieren. (Beide werden ausführlicher behandelt).

Der Befehl SORT verschiebt ganze Datensätze, um Ihre Datenbank in aufsteigender oder absteigender Ordnung nach einem beliebigen, von Ihnen bestimmten, Feld zu sortieren (Name, Postleitzahl usw.). Dieses Feld wird in der Datenbank-Sprache Schlüssel (key) genannt.

Eine Sortierung wird oft vermieden, da die Einträge in eine Datenbank für eine Anwendung nach dem einen, für eine andere Anwendung nach einem anderen Schlüsselfeld geordnet erfolgen müssen. Weiterhin muß für später angefügte Datensätze jedesmal ein zeitaufwendiger Sortier-Vorgang erfolgen, wenn die Daten in richtiger Reihenfolge ausgegeben werden sollen.

Weiterhin dauert das Auffinden von bestimmten Informationen verhältnismäßig lange, weil die sortierte Datenbank in der Reihe der Einträge nach durchsucht werden muß.

Einen eleganten Ausweg aus diesen Nachteilen bietet die INDIZIERUNG (INDEXING).

Bei der Indizierung wird eine Datei erzeugt, die lediglich die Schlüsselfelder, für die Sie sich interessieren, enthält, nicht aber die gesamten Daten der Datenbank. Ein Schlüssel ist ein Datensatz-Feld (oder eine Kombination von Feldern), welches wesentliche Merkmale des Datensatzes enthält, auf das die Datenbank häufig zugreift. In einem Lagerhaltungs-Sytem kann dies die Teile-Nummer sein, während vorhandene Stückzahl, Preis, Aufbewahrungs-Ort usw. beschreibende Informationen des Teiles sind. In einer Personal-Datei sind wahrscheinlich Name oder Angestellten-Nummer die besten Schlüssel.

In einer indizierten Datenbank werden nur die Schlüssel geordnet, die jeweils auf den Datensatz hinweisen. dBASE II benutzt für die Indizes eine Struktur, die man "Präfix-B-Baum" nennt. Sie nutzt den Speicher effektiv aus und arbeitet schnell. Eine FIND-Anweisung benötigt bei einer mittleren bis großen Datenbank typischerweise zwei Sekunden zum Auffinden eines Datensatzes.

Wenn Sie Ihre Daten für verschiedene Anwendungen nach unterschiedlichen Schlüsselfeldern organisieren wollen, so können Sie für diesen Zweck verschiedene Index-Dateien erzeugen (eine für jedes der Schlüssel-Felder). Sie benutzen dann einfach jeweils die entsprechende Index-Datei. Sie können Index-Dateien nach Lieferanten, Kundennummern, Postleitzahlen oder irgendeinem anderen Schlüssel ordnen - alles für ein und dieselbe Datenbank.

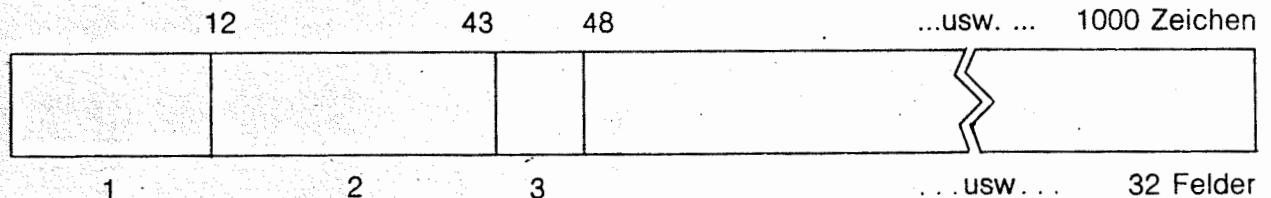
Neue Einträge zu einer Datenbank werden automatisch in den jeweils verwendeten Schlüsseldatei eingetragen.

Datensätze, Dateien und Datentypen in dBASE II

dBASE II wurde speziell für Mikro-Computer entworfen, seine Fähigkeiten erstrecken sich daher nicht ins Unendliche, aber Sie werden sicher einige Zeit dafür aufwenden müssen, um seine Grenzen auszuschöpfen.

dBASE II beschränkt die Anzahl von Datensätzen in einer Datenbank auf 65 535, aber bei den für Mikrocomputern überhaupt verfügbaren Haupt- und Massenspeichern (Disketten oder Festplatten) dürfte dies kaum eine praktische Einschränkung bedeuten.

Ein Datensatz kann aus bis zu 32 Feldern oder 1000 Zeichen bestehen (je nachdem, welches Maß zuerst erschöpft wird).



Sie können sich das als ein 1000 Zeichen langes Band vorstellen, das Sie in beliebiger Weise unterteilen können (bis zu den angegebenen Maximalgrößen), oder verkürzen, wenn Sie nur einen Teil davon benötigen. Sie können vier Felder definieren, die zusammen alle 1000 Zeichen ausnutzen (höchstens 254 Zeichen pro Feld). Oder einen Datensatz, der nur ein Zeichen lang ist (und nur ein Feld besitzt). Oder irgendeine dazwischen liegende Anordnung.

In unserem Beispiel 1.3 hatte jeder Datensatz fünf Felder, die gesamte Länge des Datensatzes betrug 58 Zeichen:

GRUNDLAGEN...30

Rechnungs Nummer			Lieferant		Artikel		Betrag		Nummer	
1	9	10	28	29	43	44	51	52	58	

Daten-Typen

Wie wir früher schon gesehen haben, muß jedes Datenfeld einen bestimmten Typ von Eintragung enthalten. In dBASE II stehen dafür zur Verfügung:

Zeichen (Character):

Alle druckbaren ASCII-Zeichen, einschließlich der Ziffern, Sonderzeichen und des Leerzeichens. Nicht druckbare Zeichen können Sie auf Ihrer Tastatur ebenfalls erzeugen, z.B. sind <RETURN> oder <BACKSPACE> sowie die meisten durch die <CTL>- (<CONTROL>-TASTE) erzeugten Zeichen nicht druckbar. Sie sind hier nicht zugelassen.

Numerische:

Positive und negative Zahlen von 1.0×10^{63} bis herab zu 1.0×10^{-63} . Die Genauigkeit beträgt zehn Stellen oder bis auf den Pfennig genau bei Beträgen, die nicht größer als DM 99 999 999.99 sind.

Logische:

Das sind die Werte "wahr" und "falsch" oder "ja"/ "nein". dBASE II erkennt T, t (true = wahr) oder Y und y (yes = ja) als "wahr" sowie F, f (false = falsch) oder N und n (no = nein) als "falsch".

Feld-Namen

Jedes Feld in einem Datensatz hat einen Namen, an dem dBASE II es erkennt, wenn Sie danach suchen. Feldnamen dürfen bis zu 10 Zeichen lang sein (das Leerzeichen darf nicht enthalten sein) und sie müssen mit einem Buchstaben anfangen, können aber dann Ziffern und eingefügte Doppelpunkte enthalten:

A	(gültig)
A123456789	(gültig)
Auftr:Numr	(gültig, Groß- und Kleinschreibung erlaubt)
A123,B456	(Komma nicht erlaubt)
Lesend:	(ungültig: Doppelpunkt ist nicht eingebettet)

Vorschlag: Verwenden Sie so viele Buchstaben wie nötig, um einen Namen verständlich zu machen. "Auftr:Numr" ist viel besser als "No" und unendlich besser als "A". Wenn Sie nur 9 Zeichen verwenden, lassen sich leicht entsprechende Namen für Speicher-Variable bilden (durch voranstellen von S wie Speicher oder M wie Memory - siehe später).

Noch ein Vorschlag: Wenn Sie anfangen Programme zu schreiben, indem Sie Befehle in einer Datei speichern, werden Sie es nützlich finden, Schlüsselworte von dBASE II mit Großbuchstaben zu bezeichnen und Groß- und Kleinschreibung für Felder, Variable usw. die Sie beliebig benennen. Es wird Ihnen einleuchten, wenn Sie zum erstenmal ein Programm wiederlesen, um es zu ändern.

Dateitypen in dBASE II

Datei-Namen sind auf 8 Zeichen und 3 Zeichen für die Namen-Erweiterung (Extension) nach einem Punkt beschränkt. Sie können den Doppelpunkt in einem Dateinamen verwenden, aber dann können Sie diese Dateien nur unter dBASE II bearbeiten: CP/M wird die Dateien speichern und Ihnen die richtigen Namen im DIR-Befehl (Inhaltsverzeichnis d. Diskette) ausgeben, aber es wird sie nicht erkennen, wenn Sie sie mit einem CP/M-Befehl wie PIP ansprechen wollen. Das ist eine der Anomalien, wie sie manchmal in Betriebssystemen vorkommen: Dateinamen, wie sie auch durch Bedienfehler entstehen können, die sich nicht mehr löschen lassen (außer manchmal durch bestimmte Tricks). Auch zehn Zeichen lange Dateinamen können gebildet werden: CP/M beschneidet sie einfach auf die ersten 8 Zeichen. Wenn Sie Groß- und Kleinschreibung zur Benennung von Dateien benutzen, so verwandelt CP/M die Namen in Großbuchstaben, aber sie sehen in dBASE II Programmen nach wie vor besser aus.

Eine dBASE II-Datei ist einfach eine Sammlung von Dateneinträgen gleicher Struktur unter einem bestimmten Namen, so etwa wie ein riesiger Stapel zusammengeklebter, ziehharmonika-artig gefalteter Karteikarten. dBASE II arbeitet mit sechs verschiedenen Dateitypen, die nachstehend beschrieben werden:

- .DBF** Datenbank Files: In dieser Form werden alle Ihre Daten gespeichert, die Namen-Erweiterung wird von dBASE automatisch angefügt, wenn Sie eine Datei mit CREATE erzeugen. Jede .DBF-Datei kann bis zu 65 535 Datensätze speichern. Bearbeiten Sie solche Dateien nicht mit einem Texteditor!
- .FRM** Report Form Files: Dateien, die das Format von Berichten (Auswertungen) enthalten. Sie werden automatisch erzeugt, wenn Sie den Dialog mit der REPORT-Anweisung ausführen. Sie enthalten Überschriften, Zusammenfassungen, Spalten- und Summen-Inhalte usw. Sie können mit einem Texteditor bearbeitet werden, aber wir raten Ihnen davon entschieden ab: Führen Sie Änderungen unter dBASE II durch.

- .CMD/** Command Files: Programm-Dateien, die eine Folge von dBASE II Anweisungen enthalten, um Programm-Folgen auszuführen, die Sie programmieren können. Sie können den Umfang eines ganzen Lohnabrechnungs-Systems haben. Man erzeugt sie mit einem Texteditor oder dBASE-Generatoren wie ZIP (ZIP ist auf der dBASE-Programmdiskette abgespeichert und in Abschnitt 6 in diesem Handbuch beschrieben), Autocode oder dBASE-Window. Fragen Sie Ihren Händler nach diesen wirkungsvollen und nützlichen dBASE-Zusatzprogrammen.
- .PRG** um Programm-Folgen auszuführen, die Sie programmieren können. Sie können den Umfang eines ganzen Lohnabrechnungs-Systems haben. Man erzeugt sie mit einem Texteditor oder dBASE-Generatoren wie ZIP (ZIP ist auf der dBASE-Programmdiskette abgespeichert und in Abschnitt 6 in diesem Handbuch beschrieben), Autocode oder dBASE-Window. Fragen Sie Ihren Händler nach diesen wirkungsvollen und nützlichen dBASE-Zusatzprogrammen.
- .NDX** Index Files: Sie werden automatisch durch den INDEX-Befehl erzeugt. Die Indizierung bewirkt die Sortierung nach einem Schlüsselfeld und ermöglicht so den sehr schnellen Zugriff auf Daten in größeren Datenbanken.
- .MEM** Memory Files: Werden automatisch erzeugt, wenn Sie die Ergebnisse von Berechnungen, Konstante oder Variable mit der Anweisung SAVE speichern. Sie können bis zu 64 Einträge abspeichern, jeder bis zu 254 Zeichen lang, und sie später mit RESTORE wieder in den Speicher laden und weiterverwenden.
- .TXT** Text-Ausgabe-Dateien: Solche Dateien werden erzeugt, wenn Sie die Anweisung SET ALTERNATE benutzen, um alles zu speichern, was auf dem Bildschirm erscheint. Diese Informationen können später als Systemprotokoll editiert und gedruckt bzw. aufbewahrt werden. Text-Dateien werden auch durch COPY ... SDF erzeugt.

Zusammenfassung der Operatoren von dBASE II

Feldinhalte können durch folgende mathematischen Funktionen miteinander verknüpft werden. Die Ergebnisse befinden sich in jeweils zu definierenden Feldern.

Arithmetische Operatoren (erzeugen rechnerische Resultate)

()	: Klammern, Zusammenfassung für Vorrang-Regelung
*	: Multiplikation
/	: Division
+	: Addition
-	: Subtraktion

Relationale (vergleichende) Operatoren (erzeugen logische Resultate)

<	: kleiner als
>	: größer als
=	: gleich
<>	: ungleich, nicht gleich
<=	: kleiner oder gleich
>=	: größer oder gleich

Logische Operatoren (ergeben logische Werte Richtig/Falsch)

()	: Klammern zur Vorrangregelung
.NOT.	: logische Verneinung (einstelliger Operator)
.AND.	: logisches UND
.OR.	: logisches ODER
\$	: Substring-Suchoperator (ist string1 in string2 enthalten?)

String-Operatoren (ergeben Strings (Zeichenketten) als Ergebnis)

+	: String-Verkettung durch einfaches Anhängen
-	: String-Verkettung, wobei Leerzeichen nach rechts verschoben werden

Zusammenfassung der Funktionen von dBASE II

Eine ausführliche und systematische Darstellung aller Funktionen finden Sie in Kapitel 3 des Handbuches.

Allgemeine Begriffe:

# (Nummern-Kreuz)	Datensatz-Nummer
*	gelöschter Datensatz
EOF	Ende einer Datei erreicht
!(<variable/string>)	Konversion in Großbuchstaben
TYPE (<ausdruck>)	Datentyp ('N' = numerisch, 'C' = Zeichen (character), 'L' = logisch, 'U' = undefiniert).
INT (<variable/ausdruck>)	Umwandlung in Integer
VAL (<variable/string/teilstring>)	Umwandlung von String in Integer
STR (<ausdruck/variable/zahl> , <länge> , <stellen>)	Integer in String verwandeln
LEN (<variable/string>)	Länge eines Strings
\$ (<ausdruck/variable/string> , <start> , <länge>)	Herauskopieren eines Teilstrings
@ (<variable1/string1> , <variable2/string2>)	Teilstring-Suche (Position)
CHR (<zahl>)	Zahl in ASCII umwandeln
&<var>	Makro-Aufruf
FILE (<"dateiname"/var/ausdruck>)	Abfrage, ob Datei existiert
TRIM (<var/string>)	nachfolgende Leerzeichen entfernen
RANK (<string/var>)	ASCII-Wert eines Zeichens
TEST (<aus>)	Zulässigkeit eines Ausdrucks prüfen

Zusammenfassung der Befehle von dBASE II

In dieser Zusammenfassung werden folgende Abkürzungen benutzt:

<aus>	=	Ausdruck
<var>	=	Variable
<str>	=	String
<koord>	=	Koordinate.

Diese <Klammerung> zeigt symbolische Einträge an, die vom Benutzer durch konkrete Angaben zu ersetzen sind. Diese [Klammerung] schließt wahlweise Angaben ein (die auch fortgelassen werden können). In einigen Fällen sind Wahlmöglichkeiten geschachtelt, d.h. sie enthalten weitere Wahlmöglichkeiten.

? <aus [liste]>

Zeigt einen Ausdruck (oder eine Liste, durch Kommata getrennt) an.

@ <koord> [SAY <aus> [USING 'bild']]
[GET <var> [PICTURE 'bild']]

Formatiert Bildschirm oder Drucker-Ausgabe.

ACCEPT ['nachricht'] **TO** <var>

Eingabe einer Zeichenkette von der Tastatur, ohne Anführungszeichen.

APPEND [BLANK]

APPEND FROM <dateiname> [SDF] [FOR <aus>]
[DELIMITED [WITH begrenzer]]

Fügt Datensätze an eine Datenbank an.

CANCEL

Abbruch der Ausführung einer Programm-Datei.

CHANGE [bereich] **FIELD** <liste> [FOR <aus>]

Führt mehrfache Änderungen in einer Datenbank aus.

CLEAR

Rücksetzen (Löschen) der eröffneten Dateien und erzeugten Speichervariablen in dBASE II.

CONTINUE

Setze eine LOCATE-Anweisung fort.

COPY [bereich] TO <dateiname> [STRUCTURE] [FIELD <liste>]
[FOR <aus>] [SDF] [DELIMITED [WITH begrenzer]]
Kopiert Daten aus einer Datenbank in eine andere Datei.

COPY TO <dateiname> STRUCTURE EXTENDED
Erzeugt eine neue Datei, deren Datensätze die Struktur der alten Datei beschreiben (siehe auch CREATE <neue datei> FROM <alte datei>).

COUNT [bereich] [FOR <aus>] [TO <var>]
Zählt Datensätze, die der angegebenen Bedingung entsprechen.

CREATE [<dateiname>]
Erzeugt eine neue Datenbank.

CREATE <neue datei> FROM <alte datei>
Erzeugt die <neue datei> automatisch mit der Datenstruktur, die in den Datensätzen der <alten datei> beschrieben wird.
(Siehe auch COPY STRUCTURE EXTENDED).

DELETE [bereich] [FOR <aus>]
Markiere bestimmte Datensätze als gelöscht.

DELETE FILE <dateiname>
Löscht eine Datei vom System aus.

DISPLAY [bereich] [<feld> [liste]]
Zeigt nur die gewählten Datenfelder.

DISPLAY STRUCTURE
Zeigt die Datenstruktur der bearbeiteten Datenbank.

DISPLAY MEMORY
Zeigt die Inhalte der Speicher-Variablen.

DISPLAY FILES [ON <laufwerk>] [LIKE <dateimaske>]
Zeigt ein Disk-Inhaltsverzeichnis (Directory).

DISPLAY STATUS
Zeigt den aktuellen Systemzustand an.

DO <dateiname>
Führt eine dBASE-Programm aus.

DO CASE <Bedingung>
Führt für jede Bedingung eine oder mehrere Anweisungen aus.

DO WHILE <aus>
Führt eine Gruppe von Anweisungen mehrfach aus.

EDIT

Bearbeiten von Daten in einer Datenbank.

EDIT [nummer]

Nimmt einen bestimmten Datensatz in Bearbeitung.

EJECT

Führt einen Blattvorschub auf dem Drucker aus.

ELSE

Alternativer Ausführungspfad in einer IF-Anweisung.

ENDCASE

Begrenzungs-Zeichen für den Anweisungsblock in einer DO CASE-Anweisung.

ENDDO

Begrenzungs-Zeichen für den Anweisungs-Block in einer DO WHILE-Anweisung.

ENDIF

Begrenzungs-Zeichen für den Anweisungs-Block in einer IF-Anweisung.

ENDTEXT

Begrenzungs-Zeichen für den Text-Block in einer TEXT-Anweisung.

ERASE

Lösche Bildschirm/Terminal-Bildschirm.

FIND <schlüssel>

Sucht einen Datensatz in einer indizierten Datenbank nach dem Wert des Schlüssels (Zeichen-Schlüssel ohne Anführungszeichen angeben).

HELP

Anzeige von Kurzinformationen zu einzelnen dBASE-Befehlen

GO oder **GOTO** [RECORD], oder [TOP], oder [BOTTOM], <n>

Gehe zu einer bestimmten Position (Datensatznummer) in einer Datenbank.

IF <aus>

Bedingte Befehls-Ausführung.

INDEX

Fügt einen Verweis auf den aktuellen Satz in die aktivierten Schlüsseldateien ein.

INDEX ON <schlüssel> TO <dateiname>

Erzeugt eine Index-Datei für die bearbeitete Datenbank.

INPUT ['nachricht'] TO <var>

Nimmt Eingaben des Benutzers entgegen und legt sie in Speicher-Variablen ab. Nachricht an den Benutzer wahlweise.

INSERT [BEFORE] [BLANK]

Fügt einen neuen Datensatz zwischen anderen Datensätzen in einer Datenbank ein.

JOIN TO <dateiname> FOR <aus> [FIELDS <liste>]

Erzeugt eine Datenbank, die aus den einander entsprechenden (gleichen) Datensätzen zweier verschiedener Datenbanken besteht.

LIST

Listet Datensätze aus.

LOCATE [bereich] [FOR <aus>]

Findet den Datensatz, der mit einer Bedingung übereinstimmt.

LOOP

Mechanismus zum Abbrechen einer Befehlsfolge in DO WHILE.

MODIFY COMMAND <dateiname>

Erlaubt die Editierung eines Programms (einer Datei) direkt aus dBASE II.

MODIFY STRUCTURE

Ändert die Struktur der Datensätze in einer Datenbank. Achtung: Löscht alle Daten in der Datenbank.

NOTE oder *

Kommentar-Einleitung in einem Programm, Kommentar wird beim Programmlauf nicht angezeigt.

PACK

Entfernt alle als gelöscht markierten Datensätze endgültig.

QUIT [TO <liste von CP/M-Befehlen oder .COM-Files>]

Beendet dBASE-Dialogmodus und führt eine Kette von Programmen aus. Jeder Befehl muß in Anführungszeichen stehen, mehrere Befehle müssen durch Kommata getrennt sein.

READ [NOUPDATE]

Liest ein Bildschirmformular bei formatiertem Bildschirm, nimmt die Daten aus GET-Anweisungen entgegen.

RECALL [bereich]

Macht die Lösch-Markierung von Datensätzen rückgängig.

RELEASE [<var> [,liste]] oder [ALL [LIKE/EXCEPT <dateimaske>]]

Löscht nicht mehr benötigte Speicher-Variable.

REMARK

Ein Kommentar, der auf dem Bildschirm ausgeschrieben wird, wenn das Programm läuft.

RENAME <alter dateiname> TO <neuer dateiname>

Gibt einer Datei einen neuen Namen.

REPLACE [bereich] <feld> WITH <aus> [, <feld> WITH <aus>...]

Ändert Daten in einer Datenbank. Vergewissern Sie sich, daß Sie eine Sicherheitskopie der Datenbank haben, denn dBASE II tut genau, was Sie ihm angewiesen haben, auch wenn Sie es sich anders vorgestellt hatten.

REPORT [bereich] [FORM <dateiname>] [TO PRINT] [PLAIN] [FOR <aus>]

Erzeugt Auswertungen der Datenbank-Informationen.

RESET [<laufwerk>]

Teilt CP/M mit, daß ev. eine Diskette ausgetauscht wurde.

RESTORE FROM <dateiname> [ADDITIVE]

Lädt früher abgespeicherte Variable wieder in den Arbeitsspeicher. Löscht alle zuvor vorhandenen Variablen.

RETURN

Beendet die Ausführung einer Programm-Datei und gibt die Kontrolle an das aufrufende Programm zurück.

SAVE TO <dateiname> [ALL LIKE <dateimaske>]

Schreibt Speichervariable für späteren Gebrauch in die Datei

SELECT [PRIMARY] oder [SECONDARY]

Schaltet Arbeitsbereich zwischen Vorder- und Hintergrund um.

SET <parameter> [ON] oder [OFF]

Dynamische Umschaltung verschiedener dBASE-Parameter.

SKIP [[-/+] <aus/nummer>]

Springt eine entsprechende Anzahl von Datensätzen in der Datenbank vor- oder rückwärts.

SORT ON <schlüselfeld> TO <dateiname> [ASCENDING]
[DESCENDING]

Erzeugt eine Datenbank, die nach dem Schlüssel-Feld sortiert ist.

STORE <aus> TO <var>

Legt einen Wert in einer Speichervariablen ab.

SUM [bereich] <feld [, liste]> [TO <var [, liste]>] [FOR <aus>]

Bildet Gesamtsummen der Felder in einer Datenbank.

TEXT

Bringt den folgenden Textblock zur Anzeige.

TOTAL TO <dateiname> ON <schlüssel> [FIELDS <feld [, liste]>]

Erzeugt eine Datenbank mit (Teil)-Summen von Datensätzen.

UPDATE FROM <dateiname> ON <schlüssel> [ADD <feld [,liste]>]
[REPLACE <feld [,liste]>]

Modifiziert eine Datenbank mit Daten aus einer anderen Datenbank.

USE <dateiname> [INDEX <dateiname>]

Eröffnet eine Datenbank für nachfolgende Bearbeitung.

USE Schließt alle zuvor eröffneten Datenbanken, wenn kein Dateiname angegeben wird.

WAIT [TO <var>]

Das Programm wartet auf einen Tastendruck [speichert die Eingabe].

Befehle von dBASE II nach Funktionsgruppen geordnet

Befehle zur Bearbeitung der Datei-Struktur

CREATE

Erzeugt eine neu zu strukturierende Datenbank.

CREATE <neue datei> FROM <alte datei>

Erzeugt die <neue datei> automatisch mit der Datenstruktur, die in den Datensätzen der <alten datei> beschrieben wird. (Siehe auch COPY STRUCTURE EXTENDED).

USE <alte datei>

COPY [bereich] TO <dateiname> [STRUCTURE] [FIELD <liste>]
[FOR <aus>]

Kopiert Daten aus einer Datenbank in eine andere Datei.

USE <alte datei>

COPY TO <dateiname> STRUCTURE EXTENDED

Erzeugt eine neue Datei, deren Datensätze die Struktur der alten Datei beschreiben (siehe auch CREATE <neue datei> FROM <alte datei>).

CREATE <neue datei> FROM <alte datei>

Erzeugt die <neue datei> mit der Datenstruktur der Datensätze der <alten datei>. (Siehe auch COPY STRUCTURE EXTENDED).

DISPLAY STRUCTURE LIST STRUCTURE

Beide Anweisungen zeigen die Struktur der gerade bearbeiteten Datei.

MODIFY STRUCTURE

Ändert die Struktur der Datensätze in einer Datenbank (Feldnamen, Größen und Gesamtstruktur). Löscht alle Daten in der Datenbank.

Veränderung einer Datenbankstruktur mit Daten in der Datenbank:

```
USE <alte datei>
COPY TO <neue datei>
USE <neue datei>
MODIFY STRUCTURE
APPEND FROM <alte datei>
COPY TO <alte datei>
USE <alte datei>
DELETE FILE <neue datei>
```

Umbenennen von Datenfeldern mit Daten in der Datenbank:

```
USE <alte Datei>
COPY TO <neue Datei> SDF
MODIFY STRUCTURE
APPEND FROM <neue Datei>.TXT SDF
DELETE FILE <neue Datei>
```

Befehle zur Dateibearbeitung:

USE <dateiname>
eröffnet eine Datei und nimmt sie in Bearbeitung.

USE <neue datei>
schließt die alte Datei.

USE
schließt alle Dateien.

RENAME <alter Name> TO <neuer Name>
Ändert den Namen einer Datei. NICHT auf die gerade bearbeitete Datei anwenden!

COPY TO <dateiname>
erzeugt eine (identische) Sicherheitskopie.

CLEAR

schließt alle Dateien und löscht alle Speicher-Variablen.

SELECT [PRIMARY][SECONDARY]

Erlaubt zwei Dateien, zur gleichen Zeit unabhängig voneinander eröffnet zu sein und abwechselnd bearbeitet zu werden (Vorder- und Hintergrund-Arbeitsfeld). Daten können zwischen den Dateien mit dem Namen-Präfix "P." bzw. "S." übertragen werden.

DISPLAY FILES [ON <I>]

listet die Datenbanken, die auf dem eingerasteten (logged in) oder dem angegebenen Laufwerk gespeichert sind. Statt dessen kann auch der Befehl LIST benutzt werden.

DISPLAY FILES LIKE <jokerzeichen> [ON <I>]

zeigt andere Arten von Dateien auf den Laufwerken.

DELETE FILE <dateiname>

löscht eine Datei auf der Diskette.

QUIT

schließt beide Arbeitsfelder (Vorder- und Hintergrund), alle Dateien und beendet die Arbeit mit dem dBASE System.

Sortier-Befehle

SORT ON <schlüssel> TO <neue datei>

Sortieren der Datensätze in einer Datei nach einem Schlüssel. Der Befehl verändert die Reihenfolge der Sätze in der sortierten Datei.

INDEX ON <schlüssel> TO <neue datei>

Indizieren von Datensätzen in einer Datei (erzeugen eines Index-Files), auch mit mehrfachen Schlüsseln. Der Befehl verändert die Satznummer nicht.

Befehle zur Bearbeitung zweier Datenbanken

COPY TO <neue datei>

erzeugt ein Duplikat der gerade bearbeiteten Datei.

APPEND FROM <andere datei>

fügt Datensätze zu der in Bearbeitung befindlichen Datei.

UPDATE FROM <andere datei> ON <schlüssel>

summiert zu Übersichten oder ersetzt die Daten in der in Bearbeitung genommenen Datei aus der anderen Datei. Beide Dateien müssen nach dem Schlüssel sortiert sein.

JOIN

erzeugt aus zwei Dateien eine dritte Datei.

Befehle zum Editieren, Aktualisieren, Verändern von Daten:**DISPLAY, LIST, BROWSE**

zeigen den Inhalt von Datensätzen an.

DELETE

markiert einen Datensatz, der gelöscht werden soll.

RECALL

macht die Lösch-Markierung (DELETE) rückgängig.

PACK

entfernt Datensätze mit Lösch-Markierung (DELETE) endgültig aus der Datenbank.

EDIT

erlaubt verändern der Einträge bestimmter Datensätze.

REPLACE <feld WITH daten>

globales Ersetzen von Daten in bestimmten Feldern, kann mit Bedingungen erweitert werden wie die meisten dBASE Befehle.

CHANGE ... FIELD

Editierung auf der Grundlage von Feldern statt Datensätzen.

@ <koord> GET <var>

READ stellt die Variablen dar und erlaubt die Eingabe neuer Werte.

INSERT [BEFORE] [BLANK]

fügt einen Datensatz in eine Datenbank ein.

UPDATE FROM <andere datei> ON <schlüssel>

summiert zu Übersichten oder ersetzt die Daten in der in Bearbeitung genommenen Datei aus der anderen Datei. Beide Dateien müssen nach dem Schlüssel sortiert sein.

MODIFY COMMAND <dateiname>

erlaubt das Bearbeiten Ihrer Programmdatei, ohne daß Sie dazu Ihren Texteditor aus dem Betriebssystem aufrufen müssen.

Befehle zum Gebrauch von Variablen:

(Es können bis zu 64 Speicher-Variable und eine beliebige Zahl von Feldnamen benutzt werden.)

LIST MEMORY, DISPLAY MEMORY

beide Anweisungen geben die erzeugten Variablen, ihre Datentypen und ihren Inhalt aus.

& ersetzt den Inhalt einer Variablen vom Typ Zeichen durch ihren Inhalt (d.h. erzeugt den Effekt, als stünde anstelle des Variablen-Namens der Wortlaut des Textes, den die Variable speichert (Makro-Aufrufe etc.).

STORE <wert> TO <var>

erzeugt Variable oder ändert ihren Inhalt.

RELEASE <var>

löscht die genannte Variable.

SAVE MEMORY TO <dateiname>

speichert Variable in der angegebenen Datei (mit der Namen-Erweiterung .MEM).

RESTORE FROM <dateiname>

liest Variable in den Speicher zurück (und löscht dabei zuvor vorhandene Speichervariable).

Befehle für interaktive Eingabe (Dialoge):

WAIT

hält Programmausführung, Vorbeirollen des Bildschirminhaltes usw. an, bis irgendeine Taste gedrückt wird.

WAIT TO <var>

speichert das Zeichen in einer Speichervariablen.

INPUT ['nachricht'] TO <var>

akzeptiert alle Datentypen und speichert sie in einer Variablen (die erzeugt wird, falls sie noch nicht existiert), Zeichen-Eingaben müssen in Anführungszeichen eingeschlossen werden.

ACCEPT ['nachricht'] TO <var>

wie INPUT, aber ohne Anführungszeichen bei textueller Eingabe (Zeichenketten).

**@ <koord> SAY ['nachricht'] GET <var> [PICTURE]
READ**

stellt Variable auf dem Bildschirm dar und liest neue Werte in sie ein.

Suchbefehle:**SKIP** [+ <aus>]

springt eine bestimmte Anzahl von Datensätzen vor- oder rückwärts.

GO [TO] <zahl>, **GO TOP**, **GO BOTTOM**bringt Sie zu einem bestimmten Datensatz, dem ersten Datensatz (**GO TOP**), oder zum letzten Datensatz (**GO BOTTOM**) in der Datei.**FIND** <str>

sucht einen Datensatz in einer indizierten Datenbank nach dem Wert des Schlüssels.

LOCATE FOR <aus>**CONTINUE**

Findet den Datensatz, der mit der Bedingung <aus> übereinstimmt.

Ausgabe-Befehle:**?, DISPLAY, LIST**

zeigen (Werte von) Ausdrücken, Datensätzen, Variable, Strukturen.

REPORT [FORM <formatname>] erzeugt Datenbankauswertungen**@** <koord> **SAY** <var/aus/str>

formatiert Ausgaben auf dem Bildschirm oder dem Drucker.

[USING <format>] kann hinzugefügt werden, um das PICTURE-Format auf dem Drucker zu erzeugen.

Programm-Befehle für Befehls-Dateien:

(dBASE-Programme werden in sog. "command files" (Befehls-Dateien) gespeichert, deren Namen die Erweiterung ".CMD" besitzt.)

DO <dateiname>

startet die Ausführung eines Programms.

IF <bedingungen>

auszuführende Befehle

ELSE

alternativ auszuführende Befehle

ENDIF

Trifft einfache Entscheidungen (ein oder zwei Wege) oder mehrfache, wenn verschachtelt.

DO CASE

CASE <ausdruck>
auszuführende Befehle

...

OTHERWISE

ENDCASE

Ähnlich wie IF ... ENDIF, aber ohne Verschachtelung.

DO WHILE <bedingungen>

zu wiederholende Anweisungen

ENDDO

Schleifen-Anweisung. Die "bedingungen" müssen durch irgendeine Anweisung in der Schleife verändert werden, damit die Schleife nicht "ewig" läuft.

Abschnitt 2: Wie man eine Datenbank erzeugt und mit ihr umgeht

Inhalt

Erläuterung einiger Begriffe.....	47
Wie man eine Datenbank erzeugt (CREATE).....	50
Eingabe von Daten in Ihre neue Datenbank.....	54
Ändern von Daten mit dem EDIT-Befehl.....	57
Allgemeine bildschirmorientierte Tastatur-Befehle.....	58
CONTROL-Funktionen	
Der fehlerkorrigierende Dialog.....	60
Befehls-Erweiterung mit Auswahlkriterien am Beispiel LIST.....	61
Ausgabe von Daten mit dem DISPLAY-Befehl.....	64
Aufsuchen eines Datensatzes mit GO oder GOTO und SKIP.....	66
Der interaktive Frage-Befehl “?”.....	68
Eingeben weiterer Datensätze mit APPEND und INSERT.....	70
Datenbankpflege mit DELETE, RECALL, PACK.....	73
Zusammenfassung	77

In diesem Abschnitt beginnen wir mit der praktischen Anwendung von dBASE II. Wir erzeugen eine Datenbank und geben Daten in sie ein. Wir machen Sie außerdem mit einigen dBASE II-Befehlen bekannt, die in den übrigen Abschnitten des Handbuches ergänzt und vertieft behandelt werden. Für die vollständige Definition eines Befehls schlagen Sie bitte im 3. Kapitel nach.

Sollten Sie an einer Stelle nicht weiterkommen, lesen Sie bitte in Ruhe den Rest dieses Abschnittes - vermutlich finden Sie an anderer Stelle Hinweise, die Ihnen helfen. Beachten Sie die Tafeln mit Erklärungen über die Tastatur. Beginnen Sie dann noch einmal von Vorne.

Sollten Sie dann immer noch nicht weiterkommen, dann wenden Sie sich an Ihren Händler oder das Unternehmen, bei dem Sie dBASE gekauft haben.

Erläuterung wichtiger Begriffe

Wir benutzen in diesem Handbuch immer wieder die Begriffe Datei, Datenbank und File.

Während Sie Daten über die Tastatur eingeben oder auf dem Bildschirm sehen, müssen diese Daten zwischenzeitlich irgendwo gespeichert werden, auch wenn der Computer einmal ausgeschaltet wird. Früher geschah das, indem man die "Daten" auf Karteikarten geschrieben hat und diese in einen Karteikasten einsortierte (oder durch Abheften von Akten in einem Ordner).

Die Speicherung im Computer ist davon gar nicht sehr verschieden. Statt eines Karteikastens wird ein Eintrag in magnetischer Form auf einer Diskette benutzt, den man File nennt. So ein File ist einfach eine Sammlung von gleichartigen Datengruppen - was früher auf einer einzelnen Karteikarte stand, heißt jetzt "Datensatz", ein Stapel Karten, der zu einer bestimmten Rubrik gehört, heißt Datei oder auch Datenbank. Auch Datensätze können alphabetisch sortiert werden - doch davon später. Statt File sagen wir auch Datei. Wie ein Karteischränk viele verschiedene Karteien enthalten kann, mögen sich auf einer Diskette mehrere unterschiedliche Dateien befinden. Die auf einer Diskette gespeicherten Dateien (manche davon enthalten Programme, andere Datenbanken, wieder andere Texte (z. B. Briefe)) können Sie in der Regel von Ihrem Betriebssystem aus durch Befehle wie DIR oder DIR A:, DIR B: usw. auslisten lassen (DIR steht für "Directory", was nichts anderes als "Inhaltsverzeichnis" bedeutet).

Mit dem Wort Datenbank wollen wir ausdrücken, daß die in einer Datei gespeicherten Daten in vielfältiger Weise abgefragt und manipuliert werden können. Diese Daten sind nicht wie in einem Brief oder einem Roman willkürlich, sondern nach einem standardisierten Schema abgespeichert. Dieses Schema erlaubt eine sehr schnelle und vielfältige Abfrage, kann aber auch gewisse Einschränkungen mit sich bringen. Genauso wie eine ordentliche Kartei systematisch aufgebaut ist, so daß jede Karte das gleiche Erscheinungsbild zeigt, besteht auch eine Datenbank aus einer Vielzahl gleichartig geordneter Datensätze. Was die übersichtliche Anordnung der Beschriftung auf der Karteikarte war, heißt nun "Datenstruktur". Wie eine Karteikarte spalten- oder auch zeilenweise systematisch in verschiedene Plätze aufgeteilt wurde, wird ein einzelner Datensatz in einer Datenbank in verschiedene Felder aufgeteilt.

Wenn Sie schnell mal vorblättern bis zum "Beispiel 2.1", so sehen Sie ein solches Schema, das man ebensogut auf einer Karteikarte unterbringen könnte. Sie werden sehen, wie daraus im Handumdrehen ein "Datensatz" aus 6 Feldern mit den Namen "Name", "Adresse", "Stadt", "Postleitzahl", "Telefon" und "Bundesland" wird. Das ist nichts anderes, als wenn man bestimmte Stellen für die Eintragung entsprechender Informationen auf einer Karteikarte vorsehen würde.

Nun können auf so einer Diskette (einer kreisrunden Plastikfolie, die mit einer Magnetschicht wie ein Tonband versehen ist, in einer Hülle steckt und in das sogenannte "Laufwerk" oder Floppy-Disk-Drive Ihres Computers eingesetzt wird) viele verschiedene Dateien (oder Datenbanken) gleichzeitig gespeichert werden. Zu einer Datenbank können auch mehrere Hilfsdateien gehören (zum Beispiel, um für eine Bibliothek die Bücher gleichzeitig nach Autoren, Titeln, Sachgebieten usw. alphabetisch für sekundenschnelle Abfrage geordnet bereitzuhalten). Um sich dabei zurechtzufinden, vergibt der Computer für jede Datei oder Hilfsdatei einen sogenannten Dateinamen oder auch Filenamen. Er besteht in der Regel aus nicht mehr als 8 Zeichen (und meistens drei weiteren Zeichen, die vom Hauptnamen durch einen Punkt getrennt sind, und die besondere technische Art der Datei angeben).

Wenn Sie schnell mal vorblättern bis zum "Beispiel 2.1", so sehen Sie ein solches Schema, das man ebensogut auf einer Karteikarte unterbringen könnte. Sie werden sehen, wie daraus im Handumdrehen ein "Datensatz" aus 6 Feldern mit den Namen "Name", "Adresse", "Stadt", "Postltzahl", "Telefon" und "Bundesland" wird. Das ist nichts anderes, als wenn man bestimmte Stellen für die Eintragung entsprechender Informationen auf einer Karteikarte vorsehen würde.

Nun können auf so einer Diskette (einer kreisrunden Plastikfolie, die mit einer Magnetschicht wie ein Tonband versehen ist, in einer Hülle steckt und in das sogenannte "Laufwerk" oder Floppy-Disk-Drive Ihres Computers eingesetzt wird) viele verschiedene Dateien (oder Datenbänke) gleichzeitig gespeichert werden. Zu einer Datenbank können auch mehrere Hilfsdateien gehören (zum Beispiel, um für eine Bibliothek die Bücher gleichzeitig nach Autoren, Titeln, Sachgebieten usw. alphabetisch für sekundenschnelle Abfrage geordnet bereitzuhalten). Um sich dabei zurechtzufinden, vergibt der Computer für jede Datei oder Hilfsdatei einen sogenannten Dateinamen oder auch Filenamen. Er besteht in der Regel aus nicht mehr als 8 Zeichen (und meistens drei weiteren Zeichen, die vom Hauptnamen durch einen Punkt getrennt sind, und die besondere technische Art der Datei angeben).

Sie begegnen auch immer wieder den Begriffen "Betriebssystem" und "dBASE II". Stellen Sie sich einmal vor, Ihr Computer entspräche einem Verwaltungsgebäude (z.B. einer Versicherung) mit zahllosen Gängen und Zimmern, in denen die unterschiedlichsten Akten lagern. Wenn nachts die Angestellten nach Hause gegangen sind, dann ist dieses Gebäude ein riesiges Labyrinth, in dem man tagelang umherirren könnte, ohne sich zurechtzufinden. Das Gebäude (die "Hardware") ist auch zu keinerlei Leistung imstande - es kann Ihnen keine Auskünfte geben oder Anträge bearbeiten.

Erst am Morgen können Sie sich vom Pförtner aus zu dem Zimmer durchfragen, in dem ein Beamter sitzt, der für Ihren Fall zuständig ist. Die Angestellten, welche das Gebäude beleben, entsprechen dem sogenannten Betriebssystem Ihres Computers. Das Betriebssystem hat mit Ihrer Problemlösung ebenso wenig zu tun wie das Privatleben der Angestellten mit Ihrem Versicherungsfall - aber wie das Versicherungsgebäude ohne Angestellte nur ein lebloses Gemäuer ist, so ist ein Computer ohne Betriebssystem nur ein nutzloser Haufen Draht und Platinen.

Ihr persönlicher Vorgang aber, nach dem Sie sich erkundigen und über den Sie sich mit einem zuständigen Sachbearbeiter unterhalten können, stellt Ihr besonderes Programm dar. Wenn der Vergleich auch hinkt - ich hoffe, er macht Ihnen Unterschied und Bedeutung von Hardware (der Computer, der vor Ihnen auf dem Tisch steht und den Sie anfassen können), Betriebssystem-Software (z.B. "CP/M") und Anwender-Programm (dBASE II, Ihre Datenbänke und speziellen Datenbank-Programme), die beide auf Diskette gespeichert werden, klar.

So ist dBASE II ein bestimmtes Programm, das auf Ihrem Computer läuft und der Vermittlung durch das Betriebssystem bedarf (CP/M oder MS-DOS usw.). So wie Sie wissen müssen, was ein Pförtner ist, was Ihre Versicherungsnummer ist und was die elementaren Umgangsformen mit Behörden sind, um beim Betreten des Versicherungspalastes eine Chance zu haben, je zum Ziele zu kommen, so müssen Sie ein bißchen etwas über den Umgang mit dem Betriebssystem Ihres Computers wissen, um das System aufzurufen.

Sobald Sie aber zu Ihrem Sachbearbeiter vorgedrungen sind, müssen Sie Ihr Wissen über Ihren "Fall" anwenden bzw. durch gezielte Fragen ergänzen oder durch Aufträge dafür sorgen, daß Ihre Anliegen erfüllt werden. So müssen Sie auch ein wenig über dBASE, im Besonderen wissen - und dieses Handbuch wird Ihnen dieses Wissen vermitteln.

Ein weiteres Problem für viele Neulinge stellt die Tastatur eines Computers dar. Ich will Sie nicht langweilen - vielleicht kennen Sie sich schon damit aus. Vielleicht aber auch nicht. Eine moderne Computer-Tastatur ähnelt der Tastatur einer Schreibmaschine - mit einigen Unterschieden. Es gibt eine Reihe von besonderen Tasten wie z. B. "RETURN" (oder auch "ENTER"), "BACKSPACE", "CONTROL" oder "CTRL" oder ähnlich usw. Im Text weise ich in besonderen Kästen auf die spezielle Bedeutung einiger dieser Tasten und ihre Verwendung hin. Normalerweise erscheint, wenn Sie eine Taste wie "A" (zweite Reihe von unten, fast ganz links) betätigen, das entsprechende Zeichen auf dem Bildschirm Ihres Computers, etwa wie auf dem Papier, das in eine Schreibmaschine eingespannt ist. An der Stelle, wo dieses Zeichen erscheinen wird (bzw. rechts von dem Platz, an dem es gerade erschienen ist) sehen Sie ein ev. blinkendes Leuchtzeichen (oder Strichlein, jedenfalls irgendetwas, das all Ihren Eingaben voraneilt). Dieses Zeichen nennen wir CURSOR. Es soll gewissermaßen die Spitze des Bleistifts oder Filzstiftes ersetzen, den Ihnen der Computer aus der Hand nimmt, damit Sie dennoch wissen: Wo wird denn nun etwas geschrieben.

Weil der Computer nicht von vornherein wissen kann, wann Sie mit Ihrem Schreiben zu Ende sind, beschließen Sie alle Ihre Eingaben (von seltenen Ausnahmen abgesehen) durch Drücken der mit <ENTER> oder <RETURN> bezeichneten Taste. Näheres dazu und zu weiteren Sondertasten weiter unten.

Jede Schreibmaschine und die Tastatur jedes Computers, auf dem Sie dBASE II benutzen können, besitzt unten eine ganz breite Taste, die wir im Deutschen als "Leertaste" kennen - wenn man auf sie drückt, rückt der Cursor um eine Stelle weiter (oder der Wagen der Schreibmaschine um eine Position nach links). Was Ihnen vielleicht nicht auf Anhieb klar ist: Auf dem Computer bedeutet dieser Zwischenraum ebenfalls ein Zeichen, geradeso wie "A" oder "B". Wir nennen es Leerzeichen (englisch: Space), schreiben es manchmal " " oder ' '. Es ist keinesfalls "Nichts", sondern eben ein bestimmtes Zeichen, das Platz einnimmt, aber nicht gedruckt wird, sondern - auf dem Papier - weiß oder - auf dem Bildschirm - schwarz bleibt und entweder zur Lesbarkeit von Worten beiträgt, die es trennt, oder Datenfelder (Plätze auf der Karteikarte), die von den Eintragungen nicht vollständig in Anspruch genommen werden, auffüllt.

Tatsächlich - obwohl Sie das zunächst nicht zu kümmern braucht - erzeugt auch die <RETURN>-Taste ein Zeichen (u. Umständen sogar zwei, nämlich "Wagenrücklauf" und "Zeilenvorschub"), das man nicht sieht. Es gibt, und das ist u. U. etwas verwirrend, noch mehr Tasten, die "unsichtbare Zeichen" erzeugen. Zum Beispiel <BACKSPACE>, <ESCAPE> u. a., siehe unten.

Nur die Tasten <SHIFT> (Groß-Buchstaben-Umschaltung) und <CONTROL> oder <CTRL> sowie (falls vorhanden) <LOCK> (Feststellen von <SHIFT>) oder <CAPS> (Feststellen von Großbuchstaben, ohne alle sonst durch <SHIFT> erzielten Zeichen wie z. B. "ö" oder "&" aufzurufen) erzeugen, für sich betätigt, keine Zeichen. Sie ändern nur die von anderen Tasten ausgesandten Zeichen, wenn Sie mit diesen gleichzeitig gedrückt werden. Auch die <REPEAT> (Wiederholungs-)Taste, falls Ihr Computer eine besitzt, sendet selbst kein Zeichen aus. Wenn man sie gleichzeitig mit einer anderen Taste betätigt, bewirkt sie die fortlaufend wiederholte Aussendung des Zeichens, das die andere Taste erzeugt. Auf manchen Computern wird das Gleiche durch das langdauernde Niederdrücken einer Taste bewirkt.

Sie brauchen sich das jetzt nicht alles auf einmal einzuprägen, denn ich weise im Text mehrmals darauf hin, ich wollte Sie nur auf einige Merkwürdigkeiten so einer Computer-Tastatur vorbereiten.

Wie man eine Datenbank erzeugt

Wir werden damit anfangen, eine Datei mit Namen für ein Adress-System zu erzeugen. Jeder Eintrag in die Datei soll die folgenden Informationen enthalten:

Beispiel 2.1: Datensatz (Record) unserer Datenbank

Name:	bis zu 20 Zeichen lang
Adresse:	bis zu 20 Zeichen lang
Postltzahl:	4 Zeichen lang
Stadt:	bis zu 20 Zeichen lang
Telefon:	bis zu 10 Zeichen lang
Bundesland:	3 Zeichen lang

Jede Zeile (z. B. "Name") stellt ein sog. Feld des Datensatzes dar! Eine Datenbank kann nahezu beliebig viele solcher "Datensätze" enthalten (bis zu 65 535).

Als erstes geben Sie 'create' ein (to create = erzeugen) - vgl. Abb. 2.1a u. 2.1b.

Dies setzt voraus, daß Sie - wie auf der letzten Seite vom Kapitel 1 erklärt - das fertig angepaßte Datenbank-System von Ihrem Betriebssystem aus mit "dbase <RETURN>" gestartet haben. Den Unterschied erkennen Sie folgendermaßen: Nach Einschalten Ihres Rechners (POWER ON bzw. NETZ EIN) und Einsetzen einer vorbereiteten CP/M-Diskette meldet sich das Betriebssystem mit "A>". Nun können Sie beliebige Programme starten (sofern diese sich auf der Diskette befinden). Um ggf. auf das zweite Laufwerk umzuschalten, befehlen Sie "B: <RETURN>". Das "A>" bzw. "B>" wird der Prompt von CP/M genannt (ein "Hier-bin-ich-was-willst-Du"-Zeichen). Nach

dbase <RETURN>

wird das Datenbank-System gestartet und meldet sich mit seinem Prompt (.), dem Punkt. Von diesem Zustand gehen wir in allen Abbildungen aus. Mit "QUIT" können Sie dBASE verlassen und ins Betriebssystem zurückkehren.

Wenn Sie ein anderes Betriebssystem als CP/M benutzen, lesen Sie bitte die jeweilige Anleitung oder lassen Sie sich von einer kundigen Person den Umgang mit dem Computer erklären. Machen Sie sich ev. Notizen über die wichtigsten Handgriffe (z. B. wie herum die Disketten einzusetzen sind, wo sich die Netzschalter befinden, welches Laufwerk A und welches Laufwerk B ist usw.).

dBASE II antwortet mit:

ENTER FILENAME:

(to enter = eingeben).

Geben Sie einen Dateinamen (Filenamen) an, unter dem die Datei auf der Diskette abgespeichert werden soll. Er muß mit einem Buchstaben anfangen und darf bis zu 8 Zeichen lang sein (diese Begrenzung besteht im CP/M-Betriebssystem). Er darf keine Doppelpunkte und keine Leerzeichen enthalten. Da diese Datei Namen enthalten soll, wählen Sie eine Bezeichnung, die für einen Menschen eine sinnvolle Bedeutung hat: Tippen Sie 'Namen'.

Falls auf der Diskette, welche die dBASE - Files enthält, kein Platz mehr ist, können Sie auch die Bezeichnung eines anderen Laufwerks voranstellen, z. B.:

'B:Namen'

Sie können auch - wohlgermerkt auf Ihrer Arbeitskopie! - alle nicht benötigten Files, deren Namen auf ".DBF" (Datenbank-File), ".FRM" (Report-File), ".CMD" (Command-File) oder ".NDX" (Index-File) löschen. Sie gehören zu mitgelieferten Datenbank-Demonstrationen, die Sie in der Einführung nicht benötigen.

Sobald Sie danach <RETURN> (oder <ENTER>) drücken, erzeugt dBASE II eine Datei mit dem Namen "NAMEN.DBF". Der Teil dieser Bezeichnung nach dem Punkt ist die CP/M-Filenamen-Ergänzung ("extension") und ist eine Abkürzung für "Datenbank File" (siehe Abschnitt 1, File-Typen). Diese Ergänzung des Filenamens, um die Sie sich jetzt nicht weiter zu kümmern brauchen, fügt dBASE automatisch an - auch, wenn Sie später die Datenbank wieder aufrufen, geben Sie nur "namen" ein.

Mit <RETURN> ist gemeint, daß Sie diese Taste (auf manchen Tastaturen heißt sie <ENTER>, manchmal findet man auch beide Bezeichnungen) drücken. Es ist in der Regel eine besonders breite Taste, die sich rechts über oder neben der rechten SHIFT-Taste (Groß-Buchstaben-Umschaltung) befindet. Sie entspricht weitgehend der Taste für Zeilenvorschub bei der elektrischen Schreibmaschine. Auf einer Computer-Tastatur hat sie noch eine weitere Bedeutung: Sie zeigt das Ende einer Befehlseingabe an. Bevor Sie <RETURN> drücken, können Sie Ihre Befehlseingabe noch korrigieren, danach versucht der Computer, den Befehl auszuwerten. Sie sollen also niemals etwa das Wort "RETURN" oder "ENTER" eingeben. Auch in anderen Fällen, in denen ein Wort wie <TASTE> geschrieben ist, sollen Sie nicht etwa "Taste" eintippen, sondern eine mit dem Begriff <TASTE> gekennzeichnete einzelne Taste drücken.

In einem DatenbankSystem wird jede Dateneinheit, die wir in einen einzelnen Datensatz eingeben wollen, ein Feld genannt, und der Datensatz wird als Record bezeichnet (vgl. Beispiel 2.1 sowie Abschnitt 1, Datenbank-Grundlagen). In unserem Beispiel hat jeder Datensatz 6 Felder. dBASE muß von jedem Feld den Namen kennen, den Typ der Daten, den es enthält, wie lang es ist und wieviele Vor- und Nachkomma-Stellen es enthält, falls sein Typ numerisch ist. (Datentypen sind zum Beispiel Alpha-Zeichen wie in "Johannes" oder eine Zahl wie in DM "34.80").

Abb. 2.1a (englischer Original-Dialog auf Ihrem Bildschirm)

```
. create <RETURN>
ENTER FILENAME: namen <RETURN>
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD NAME, TYPE, WIDTH, DECIMAL PLACES
001
```

Abb. 2.1b (eingedeutschte Erläuterung)

```
. create <RETURN>                (= erzeuge Datenbank)
GIB FILENAMEN EIN: namen <RETURN>
GIB DATENSATZ-STRUKTUR WIE FOLGT EIN:
FELD NAME, TYP, LAENGE, NACHKOMMA-STELLEN
001
```

In den folgenden Beispielen werde ich, außer in besonderen Fällen, in den Abbildungen das Wort **<RETURN>** nicht mehr anschreiben. In der Regel müssen aber alle Eingaben an den Computer durch das Drücken der **<RETURN>** oder **<ENTER>**-Taste "abgeschickt" werden.

Unter dem Wort "FIELD" (FELD) zeigt das System mit der Zahl "001" an, daß es die Beschreibung des ersten Feldes in unserem Datensatz-Schema (Record) erwartet. Wir sollen jetzt also, entsprechend der vorangegangenen Versinnbildlichung, den Aufbau unserer Karteikarte beschreiben.

Feldnamen dürfen bis zu 10 Zeichen lang sein, sie können in Groß- oder Kleinschreibung eingegeben werden. Umlaute und "ß" werden jedoch nicht akzeptiert (später bei den Eintragungen schon). Sie können also sehr wohl in das Feld "Name" den Wert "Müller" eintragen. Ein Feld namens "Erläut" (Erläuterung) müßte aber - leider - den Namen "ERLAEUT" bekommen). Der Name muß mit einem Buchstaben beginnen und darf keine Zwischenräume (Leertaste) enthalten, Zahlen und eingefügte Doppelpunkte dürfen aber vorkommen. Kürzen Sie nicht stärker ab als nötig: Der Computer wird verstehen, was Sie meinen, die Leute, welche die Datenbank benutzen, aber vielleicht nicht.

Der Datentyp in einem Feld wird durch einen einzelnen Buchstaben bezeichnet: C für Zeichen (character = Zeichen, Text bzw. "String"), N fuer Zahlen (numerisch) und L für logisch (boolean). In unserem Beispiel enthalten alle Felder Characters (Zeichen).

Die Länge eines Feldes kann beliebig bis zu 254 Zeichen gewählt werden. Wenn ein Feld numerisch ist, beachten Sie bei der Angabe der Anzahl der Dezimalstellen, daß der Dezimalpunkt (der das im deutschen Sprachraum übliche Komma vertritt) ebenfalls eine Zeichen-Stelle beansprucht (und zwar bei den Nachkomma-Stellen).

Wir wissen (aus Beispiel 2.1), welche Namen wir unseren Datenfeldern geben wollen, den Datentyp, den sie enthalten sollen (nämlich Zeichen oder "Characters", also "C"), und ihre Länge. Geben Sie daher diese Informationen jetzt ein. Hier sehen Sie, wie Ihr Bildschirm aussehen soll, wenn sie fertig sind (ich verzichte jetzt auf die "deutsche" Version des Dialoges, vgl. hierzu Abb. 2.1a und 2.1b):

Abb. 2.2

```

create
ENTER FILENAME: namen
ENTER RECORD STRUCTURE AS FOLLOWS:
  FIELD    NAME, TYPE, WIDTH, DECIMAL PLACES
  001      name,c,20
  002      adresse,c,20
  003      postltzahl,c,4
BAD NAME FIELD
  003      postltzahl,c,4
  004      stadt,c,20
  005      telefon,c,10
  006      bundesland,c,3
  007      <RETURN>
INPUT DATA NOW?
y

```

Beachten Sie, was in Feld 3 geschehen ist: Ich habe einen Fehler gemacht, indem ich in den Feldnamen ein Leerzeichen eingefügt habe. dBASE II hat mich darauf hingewiesen, daß der Feldname "schlecht" (bad) ist und gab mir danach die Möglichkeit, ihn neu einzugeben.

Beachten Sie auch, daß der Datentyp für das Feld "Postltzahl" mit "character" (Zeichen) angegeben wurde, obwohl wir üblicherweise dabei an Zahlen denken.

Dies wurde so gewählt, weil gewisse dBASE II-Befehle wie TOTAL alle numerischen Felder in einem Datensatz für die Erzeugung von Auswertungen addieren (ohne sie im einzelnen auszulisten). Es wäre reine Zeitverschwendung, dies mit dem Postleitzahlen-Feld zu tun. Wir können aber ohne weiteres die Vergleichsoperatoren wie "größer als", "kleiner als", "gleich mit" oder "ungleich mit" auch auf textuelle Daten anwenden. So stört dies nicht, wenn wir später z. B. Einträge nach Postleitzahlen sortieren wollen. Mehr davon später.

Als uns das System nach der Kennzeichnung eines siebten Feldes fragte, haben wir <RETURN> gedrückt (oder ev. <ENTER> auf Ihrer Tastatur), um die Datensatzdefinition abzuschließen. dBASE II speichert nun die Struktur des Datensatzes und fragt uns, ob wir Daten in die neu definierte Datenbank eingeben wollen (INPUT DATA NOW?).

Die Datei <namen.dbf> ist sogleich für die Erfassung von Daten bereit, drücken Sie daher 'y' (Yes = Ja). Auf der nächsten Seite zeigen wir Ihnen, wie die Eingabe von Daten funktioniert.

In der Einleitung dieses Abschnittes habe ich davon gesprochen, daß Daten in einem bestimmten Schema gespeichert werden. dBASE II speichert die Beschreibung des Schemas sowie die Daten, die später gemäß dem Schema in die Datenbank einzutragen sind, in ein und derselben Datei. Es ist aber, wie wir später sehen werden, möglich, nur die Daten oder auch nur das Schema einer Datenbank herauszufiltern und in eine andere Datei zu übertragen.

Eingabe von Daten in Ihre neue Datenbank

Falls Sie auf Ihrem Terminal die bildschirmorientierte Editierung nicht verwenden können (weil es keine direkte Cursoradressierung besitzt, siehe Kapitel 1 "Anpassung und Installationshinweise") so werden die Datensatz-Nummer (record number) und die Feldnamen jeweils für sich unter dem, was bis dahin auf dem Bildschirm eingegeben wurde, erscheinen. Die Länge jedes Feldes wird durch ein Paar von Doppelpunkten angezeigt, der Cursor befindet sich an der Stelle, an der Sie Ihre Eingaben beginnen sollen. Sobald Sie das Feld vollständig ausgefüllt oder <RETURN> (oder <ENTER>) gedrückt haben, erscheint das nächste Feld. Nachdem das letzte Feld in einem Datensatz ausgefüllt (oder durch <RETURN> ignoriert) wurde, können Sie mit dem nächsten Datensatz beginnen.

Um die Eingabe von Daten zu beenden, drücken Sie <RETURN>, wenn sich der Cursor auf der ersten Zeichenposition im ersten Feld eines neuen Datensatzes befindet.

Mit CURSOR bezeichnen wir das Leuchtzeichen, das auf dem Bildschirm die Stelle markiert, an der ein neues Zeichen erscheinen wird, wenn Sie eine Taste drücken. Es kann durch Unterstreichen eines Buchstabens oder einer Position dargestellt sein oder durch negative Darstellung des Zeichens. Oft blinkt der Cursor. Er hat noch eine weitere Bedeutung: Der Computer kann den Cursor auch selbst positionieren, z.B. ans Ende einer Frage, um Sie aufzufordern, diese zu beantworten. Auf einem modernen Terminal kann der Cursor jederzeit auf jede beliebige Stelle des Bildschirms gefahren werden, z.B. mit Hilfe von Pfeil-Tasten (unter dBASE II mit besonderen <CTL>-Zeichen, siehe Kasten weiter unten). Wir nennen das bildschirmorientierte Editierung.

Falls Sie dBASE II mit der bildschirmorientierten Editierung eingerichtet haben, so wird der Bildschirm gelöscht, dann erscheinen die Datensatz-Nummer (record number) und alle Felder des Datensatzes in der linken oberen Ecke des Bildschirms mit dem Cursor auf der ersten Zeichenposition des ersten Feldes.

(Falls Sie eines der Standard-Terminals auf der Einrichtungs-Liste (siehe Kapitel 1) gewählt haben, so erscheinen die Feld-Namen möglicherweise in negativer Darstellung oder halber Helligkeit. Falls Sie das später ändern wollen, so können Sie es abstellen, indem Sie die Möglichkeit "Y - CHANGE/MODIFY" in der Anpassungsprozedur wählen (siehe Kapitel 1).

Abb. 2.3:

```

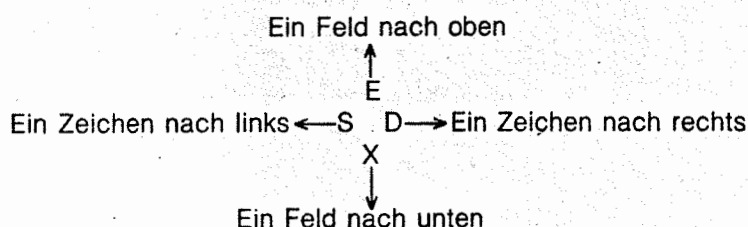
RECORD # 00001
NAME      :
ADRESSE   :
POSTLTZAH :
STADT     :
TELEFON   :
BUNDESLAND:
    
```

BEMERKUNG: Wenn es auf Ihrem Bildschirm nicht so aussieht, dann haben Sie vielleicht einen Fehler bei der Anpassung gemacht. Wiederholen Sie bitte die in Kapitel 1 (Anpassung und Installationshinweise) beschriebene INSTALL-Prozedur.

Die Feldlängen werden durch ein Paar von Doppelpunkten angezeigt. Sobald ein Feld ganz ausgefüllt wird oder Sie <RETURN> drücken, springt der Cursor zum nächsten Feld nach unten. Der Cursor kann auf ein vorhergehendes Feld zurückgeführt werden, indem Sie die CONTROL (oder CTRL)-Taste niedergedrückt halten und einmal die Taste "E" drücken: '<CONTROL>-E', abgekürzt zu '<CTL>-E'.

Die CONTROL-Taste befindet sich meistens links, neben oder über der SHIFT-Taste (Groß-Klein-Umschaltung) auf der Tastatur. Sie ist oft nur mit CTRL oder CTL beschriftet. Sie stellt so etwas wie eine Super-SHIFT-Taste dar, d.h. eine zweite Umschaltung, die, wenn man sie gleichzeitig mit anderen Tasten betätigt, diesen eine neue Bedeutung verleiht! Wenn Sie ein solches Zeichen eingeben sollen, bezeichnen wir es mit der Beschriftung der entsprechenden Taste und dem abgekürzten Vorzeichen "<CTL>". Beispielsweise besagt "<CTL>-C", daß Sie die CONTROL-Taste drücken, festhalten und gleichzeitig <C> drücken sollen.

Besonders wichtig sind die <CTL>-Zeichen, die dazu dienen, den Cursor auf dem Bildschirm zu bewegen, wenn Sie sich in der Arbeitsweise "bildschirmorientierte Editierung" befinden! Falls Ihr Rechner zu diesem Zweck Pfeiltasten besitzt, versuchen Sie zuerst diese. Es kann aber sein, daß sie unter dBASE II nicht funktionieren. Dann benutzen Sie die folgenden Tasten, die durch ihre Kreuz-Anordnung auf der Tastatur die jeweilige Bewegungs-Richtung versinnbildlichen, zusammen mit der <CTL>-Taste:



Weiter unten in diesem Abschnitt, in Tabelle 1.1, finden Sie eine Übersicht über weitere <CTL>-Zeichen für die bildschirmorientierte Editierung.

Wenn Sie die Eingabe im letzten Feld beendet haben, so präsentiert dBASE II den nächsten leeren Datensatz.

Geben Sie die folgenden Namen und Adressen ein. Wir werden sie bald benutzen, um Ihnen einige der Möglichkeiten von dBASE II zu demonstrieren.

Beispiel 2.2:

Adams, Peter	Bergstraße 15	8000 München 2	12 34 56 BAY
Bergmüller, Hans	Feldweg 3	4200 Oberhausen	45 23 98 NRW
Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
Faltmann, Gisela	Däumlingsweg 8	8049 Dirnbach	13 29 8 BAY
Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY

Bitte machen Sie mich nicht dafür haftbar, falls Ihr Nachbar so heißt oder wenn die Straßennamen nicht existieren.

Sollten Ihnen bei der Eingabe irgendwelche Fehler passieren, die nicht einfach durch **<BACKSPACE>** und Überschreiben berichtigt werden können, so lesen Sie bitte die beiden nächsten Seiten über die Editierung bevor Sie mit dem nächsten Datensatz fortfahren.

<BACKSPACE> oder **<BS>** oder **<DELETE>** oder **** oder **<RUBOUT>** oder **<RUB>**, manchmal auch nur mit einem Linkspfeil "**<-**" bezeichnet, ist eine Taste, mit der Sie mit dem Cursor ein Zeichen zurückgehen und dieses auslöschen können. Wenn Sie mehr als eine der genannten Tasten haben, kann es sein, daß eine davon den Cursor zurückbewegt, ohne das letzte Zeichen zu löschen. Probieren Sie es aus!

Sollte im später besprochenen Editmodus keine dieser Tasten die gewünschte Wirkung haben (oder Sie finden keine entsprechende Taste auf Ihrem Rechner), so überprüfen Sie noch die Möglichkeit, **<CTL>-"**, d.h. die CONTROL-Taste und gleichzeitig den Bindestrich, was dem ASCII-Code von **<DELETE>** entspricht, oder **<CTL>-H** (Code von **BACKSPACE**).

Sollten Sie unbeabsichtigt zum dBASE II Prompt (dem Punkt) zurückgeschaltet werden, so geben Sie bitte ein:

'use namen'	benutze (die Datei) "namen"
'append'	hänge an

und fahren dann mit Ihren Eingaben fort. (Wir erklären diesen Vorgang später in diesem Handbuch). Um die Eingabe von Daten zu beenden, drücken Sie bitte **<RETURN>** nachdem Sie das letzte Bundesland eingegeben haben und während Sie sich noch auf dem ersten Zeichen des ersten Feldes im nächsten Datensatz befinden. Falls Sie bereits irgendwelche Daten eingegeben oder den Cursor bewegt haben, so halten Sie die **<CONTROL>**-Taste niedergedrückt und betätigen gleichzeitig die Taste **<Q>** ('**<CTL>-Q**').

dBASE verläßt dann die Dateneingabe und schreibt seinen Prompt (.) aus, um anzuzeigen, daß es auf Ihre weiteren Befehle wartet.

Wenn Sie hier mit der Arbeit aufhören wollen, geben Sie einfach 'quit' ein. (quit = "quittiere" den Dienst, brich dBASE-Dialog-Status ab.)

'quit' muß jedesmal eingegeben werden, wenn Sie eine Arbeitssitzung mit dBASE II beenden! Dadurch werden automatisch alle Dateien korrekt abgeschlossen. Falls Sie dies versäumen, zerstören Sie möglicherweise Ihre Datenbank!

Ändern von Daten mit dem EDIT-Befehl

(Die folgenden Ausführungen beziehen sich darauf, daß Sie noch im dBASE-System sind oder dieses erneut durch 'dbase <RETURN>' gestartet haben, d.h. daß das System mit dem Punkt-Prompt (.) auf Ihre Anweisungen wartet.)

Falls Sie bei den Dateneinträgen irgendwelche Fehler gemacht haben, so können Sie diese mit der bildschirmorientierten Editierung leicht verbessern. Geben Sie ein:

'use namen' verwende (die Datei) namen
'edit <nummer>' editiere (Datensatz Nr. :) <nummer>

wobei "nummer" die Datensatz-Nummer eines der Records in der Datenbank sein sollte.

dBASE stellt den gesamten Datensatz auf dem Bildschirm dar. Sie können nun mit den bildschirmorientierten Editier-Befehlen einige oder alle Daten in dem Datensatz verändern. Um zum nachfolgenden Datensatz zu gelangen geben Sie '<CTL>-C' ein. Den vorhergehenden Datensatz erreichen Sie mit '<CTL>-R'. Probieren Sie es aus, indem Sie 'edit 3<RETURN>' eingeben.

Abb. 2.4:

RECORD 00003		DELETED (GELÖSCHT)
NAME	:Camphausen, Eva	:
ADRESSE	:Muellerstr. 7	:
POSTLEITZAHL	:1000:	
STADT	:Berlin 2	:
TELEFON	:63 78 12 :	
BUNDESLAND	:BLN:	

Wenn Sie einen Datensatz als "gelöscht" markieren, indem Sie '<CTL>-U' eingeben, so erscheint oben auf dem Bildschirm das Wort "DELETED" (GELÖSCHT). Wenn Sie erneut '<CTL>-U' eingeben, so verschwindet das Wort und die Löschung wird rückgängig gemacht.

Wenn Sie Ihre Datenbank mit 'list' oder 'display' ansehen, dann werden Sie bei allen Datensätzen, die als gelöscht markiert sind, ein Sternchen sehen (näheres über Löschung später in diesem Abschnitt). Um die bildschirmorientierte Editierung zu verlassen, verwenden Sie '<CTL>-Q'. In diesem Fall werden die Änderungen, die Sie gemacht haben, nicht abgespeichert!

Wenn Sie Ihre Änderungen dauerhaft machen wollen, so verlassen Sie die bildschirmorientierte Editierung mit '<CTL>-W' (oder '<CTL>-O' - dem Buchstaben "O" - auf dem Superbrain).

Allgemeine bildschirmorientierte Tastatur-Befehle:

- <CTL>-X bewegt den Cursor nach unten zum nächsten Feld
- <CTL>-F bewegt den Cursor nach unten zum nächsten Feld
- <CTL>-E bewegt den Cursor zurück zum vorhergehenden Feld
- <CTL>-A bewegt den Cursor zurück zum vorhergehenden Feld
- <CTL>-D bewegt den Cursor um ein Zeichen vorwärts.
- <CTL>-S bewegt den Cursor um ein Zeichen zurück.
- <CTL>-V schaltet um zwischen Überschreiben und Einfügen. Normalerweise überschreiben neu eingegebene Zeichen diejenigen, auf denen sich der Cursor gerade befindet. Nach <CTL>-V erscheint das Wort INSERT über dem Datensatz, nun schieben neu eingegebene Zeichen bereits rechts vom Cursor stehende nach rechts, was sehr nützlich ist, um z. B. einen ausgelassenen Buchstaben in ein Wort einzufügen.
- <CTL>-G löscht das Zeichen unter dem Cursor.
- <RUBOUT> löscht das Zeichen links vom Cursor.
(Statt <RUBOUT> kann auch eine der anderen Tastenbezeichnungen vorkommen, die Sie im Kasten über <BACKSPACE> (S. 66) finden! Vielleicht müssen Sie auch <CTL>-“-“, d.h. CONTROL-Taste und Bindestrich, verwenden! Dies war z.B. auf dem Rechner des Übersetzers der Fall.
- <CTL>-P schaltet Ihren Drucker EIN oder AUS.
- <CTL>-Q bricht die Editierung ab und kehrt zur dBASE II-Kommandoebene zurück, ohne daß Änderungen erhalten bleiben (auch nicht im Modify-Modus).

'modify'-Befehle:

- <CTL>-T löscht das Feld, auf dem sich der Cursor befindet, und zieht alle folgenden Felder nach oben. Dies funktioniert nicht im EDIT- und APPEND-Modus!
- <CTL>-Y löscht das laufende Feld, indem es mit Leerzeichen gefüllt wird, beläßt aber die übrigen Felder an ihrem Platz. Dies funktioniert auch im EDIT- und APPEND-Modus.
- <CTL>-N bewegt die Felder um eine Position nach unten, um an der Position des Cursors Platz für neue Einträge zu schaffen. Dies funktioniert nicht im EDIT- und APPEND-Modus!

'append'-Befehle:

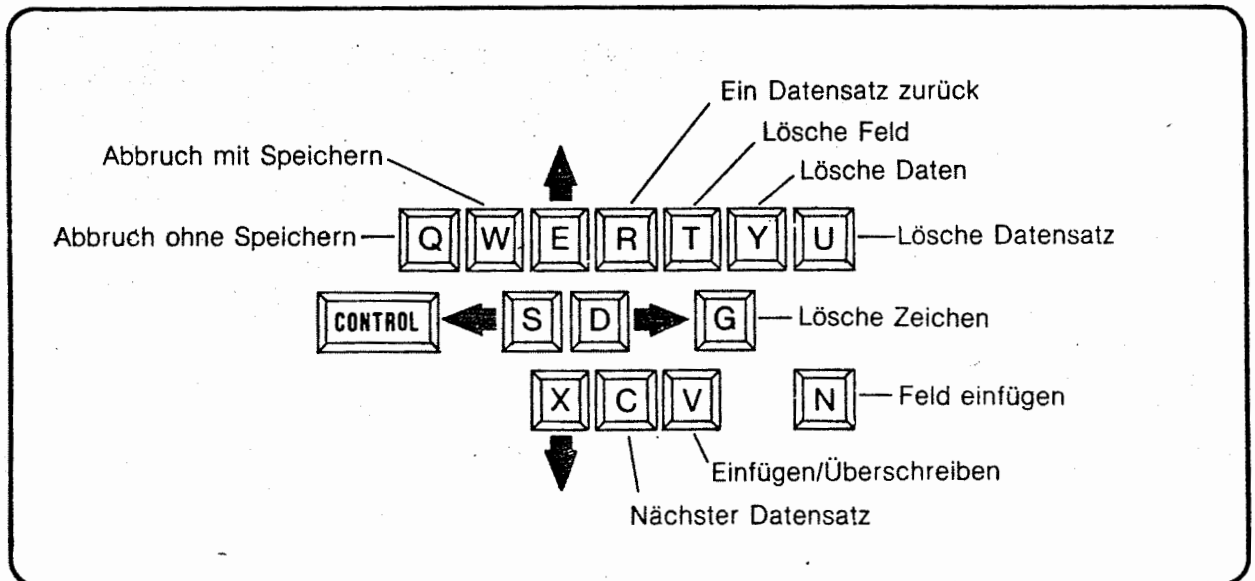
- <CTL>-C schreibt den Datensatz auf die Diskette und geht zum nächsten Datensatz über.
- <CTL>-R schreibt den Datensatz auf die Diskette und geht zum nächsten Datensatz über.
- <RETURN> während sich der Cursor auf der ersten Position eines neuen Datensatzes befindet, kehrt zur dBASE II-Kommandoebene zurück.
- <CTL>-Q löscht den Datensatz und kehrt zur dBASE II Kommandoebene zurück.
- <CTL>-Y löscht das laufende Feld, indem es mit Leerzeichen gefüllt wird, beläßt aber die übrigen Felder an ihrem Platz.

'edit'-Befehle: (Nicht im 'append'-Modus benutzen!)

- <CTL>-C schreibt den Datensatz auf die Diskette und geht zum nächsten Datensatz über.
- <CTL>-R schreibt den Datensatz auf die Diskette und bringt den vorhergehenden Datensatz zur Anzeige.
- <CTL>-Y löscht das laufende Feld, indem es mit Leerzeichen gefüllt wird, beläßt aber die übrigen Felder an ihrem Platz.
- <CTL>-U schaltet die Löschmarkierung für einen Datensatz ein/aus.
- <CTL>-W speichert irgendwelche ausgeführten Änderungen ab und kehrt zur dBASE II Kommandoebene zurück. Auf dem Superbrain verwenden Sie stattdessen '<CTL>-O' (den Buchstaben O).
- <CTL>-Q bricht jegliche Änderungen in dem Datensatz, an dem Sie gerade arbeiten, ab und gibt den Koordinaten-Prompt (mit der Datensatz-Nummer) aus. Drücken Sie <RETURN> um zur dBASE II Kommandoebene zurückzukehren.

Die erste Befehls-Gruppe funktioniert in allen Editier-Betriebsarten. Beachten Sie, daß die Befehle, die unter MODIFY stehen, zum Teil nicht funktionieren, wenn Sie durch Eingabe von 'edit' in den Edit-Modus gegangen sind usw.!

Abb. 2.5 - Control-Funktionen auf der Tastatur im Überblick:



Der fehlerkorrigierende Dialog (USE, LIST, DISPLAY)

dBASE II-Befehle bestehen in der Regel aus Worten. Sie geben diese Worte über die Tastatur ein, wenn Sie auf dem Bildschirm den Punkt-Prompt (.) von dBASE sehen. Wenn Sie sich vertippt haben, können Sie mit den entsprechenden Tasten die letzten Buchstaben löschen und korrigieren. Sobald Sie dann <RETURN> drücken, versucht dBASE, Ihre Eingabe als Befehl auszuwerten.

Um dem dBASE-System mitzuteilen, mit welcher Datei (Datenbank-File) Sie arbeiten wollen, geben Sie ein:

'use <filename>' (benutze <filename>).

Um den Datensatz zu sehen, auf dem Sie gerade stehen, sagen Sie:

'display' (zeige)

Um alle Datensätze in der Datei zu sehen, schreiben Sie:

list (liste)

(Um das Vorbeigehen auf dem Bildschirm anzuhalten und fortzusetzen drücken Sie '<CTL>-S').

dBASE II - Befehle können auf vier Buchstaben abgekürzt werden. Wenn Sie aber mehr als vier Buchstaben des Befehlswortes eingeben, so müssen diese korrekt sein.

('display', 'disp' und 'displa' sind gültige Befehle, 'dispray' hingegen nicht!)

Wenn Sie bei der Einrichtung von dBASE II auf Ihrem Rechner den fehlerkorrigierenden Dialog aktiviert haben (siehe Kapitel 1), dann wird die Eingabezeile mit Ihren Befehlen (command line) vom System nach Fehlern durchgekämmt. Wenn es dabei Fehler entdeckt, so teilt es dies auf dem Bildschirm mit. Sie erhalten dann die Möglichkeit, die command line (die gerade eingegebene Befehlszeile) zu verbessern, ohne daß Sie das Ganze neu eintippen müßten.

Geben Sie (fälschlich) ein: 'edut 3'.

Abb. 2.6

edut 3	
*** UNKNOWN COMMAND	(unbekannter Befehl)
edut 3	
CORRECT AND RETRY (Y/N)?y	(korrigieren?)
CHANGE FROM: u	(ändere von: u)
CHANGE TO: i	(zu: i)
edit 3	
MORE CORRECTIONS (Y/N)? n	(weitere Änderungen?)

Wenn dBASE II einen Befehl nicht erkennt, so wiederholt es ihn mit einem entsprechenden Hinweis. Wenn Sie ihn korrigieren wollen, brauchen Sie nicht den ganzen Befehl neu einzugeben. Als Antwort auf "CHANGE FROM:" (ändere von:) geben Sie nur ein genügend langes Stück der fehlerhaften Stelle ein, so daß die Buchstabenfolge in dem Befehlswort eindeutig identifiziert werden kann, dann drücken Sie <RETURN>.

Als Antwort auf "CHANGE TO:" (ändere zu:) geben Sie die Zeichenfolge an, welche die fehlerhafte ersetzen soll.

In diesem Beispiel (Abb. 2.6) haben wir nur einen einzelnen Buchstaben korrigiert. Sie werden diese Möglichkeit besonders dann nützlich finden, wenn Sie lange Befehlszeilen (command lines) ausprobieren und von Fehlern befreien wollen.

Hinweis: Der Befehl 'erase' löscht den Bildschirm und bringt den Prompt-Punkt in die obere linke Ecke, so daß Sie neue Befehle "auf einer sauberen Tafel" eingeben können.

Befehlserweiterungen mit Auswahlkriterien am Beispiel LIST

Eine der wichtigsten Eigenschaften von dBASE II ist die Möglichkeit, Befehle durch Anhängsel zu erweitern und dadurch in ihrer Wirkung zu verfeinern. Sie können zu den meisten Befehlen Ergänzungen und Bedingungs-Ausdrücke hinzufügen, die detaillierter beschreiben, was die Befehlszeile bewirken soll. Befehle können in Groß- oder Kleinschreibung eingegeben werden und dürfen bis zu 254 Zeichen lang sein. Um eine Zeile über die Breite Ihres Bildschirms hinaus auszudehnen geben Sie ein Semikolon (;) als letztes Zeichen in der Zeile ein (kein Leerzeichen nach dem ";"). dBASE II wird dann die nachfolgende Zeile ebenfalls als Bestandteil der Befehlszeile auffassen.

Da dBASE II ein relationales Datenbank-System ist, verarbeitet es leistungsfähige Operatoren zur Formulierung von Zugriffs-Bedingungen. Sehr oft werden Sie die relationalen (Vergleichs-) Operatoren verwenden:

- < : weniger (kleiner, kommt vor) als
- > : mehr (größer, kommt nach) als
- <= : kleiner oder gleich wie
- >= : größer oder gleich wie

Diese Anweisungen bedeuten genau das, was die Erläuterung auf der rechten Seite besagt. Sie ergeben einen logischen Wert (Richtig oder Falsch, im Englischen True (Wahr) oder False (Falsch). Wenn der Ausdruck richtig ist, dann wird der Befehl ausgeführt. Wenn das Ergebnis des Ausdrucks falsch ist, so wird der Befehl nicht ausgeführt.

Wir erwähnten bereits, daß der LIST-Befehl alle Datensätze in der Datenbank zeigt (um das Vorbeirollen anzuhalten und wieder zu starten drücken Sie '<CTL>-S'). Die vollständige Form dieses Befehls lautet:

"LIST [OFF] [FOR <ausdruck>]"

Die "[" und "]" meinen die öffnende und schließende eckige Klammer und besagen, daß die so geklammerten Teile des Befehls entfallen dürfen. Geben Sie die Klammern nicht ein - siehe Beispiel 2.3. Die "<" und ">" meinen die öffnende und schließende spitze Klammer, besagen, daß in dieser Klammer ein symbolischer Begriff steht, den Sie bei Ihrer Eingabe durch konkrete Information - z.B. einen bestimmten Ausdruck - ersetzen sollen. Geben Sie die Klammern nicht ein.

Wir machen immer wieder von dieser Art Gebrauch, ein Erweiterungsschema für Befehle anzugeben. In den untenstehenden Beispielen sehen Sie einige konkrete Ausführungen dieses Schemas für den LIST-Befehl.

(to list = auflisten, off = aus (Datensatz-Nummern nicht anzeigen), for = für (Liste alle Datensätze, für welche der Bedingungs-Ausdruck zutrifft).)

Wenn die Option (Wahlmöglichkeit) OFF verwendet wird, dann werden die Datensatz-Nummern nicht dargestellt.

Wenn die optionale FOR-Bedingung benutzt wird, dann wird dBASE II nur diejenigen Datensätze auflisten, für welche der <ausdruck> wahr ist. Geben Sie die Befehle aus Beispiel 2.3 ein. Verwenden Sie einfache oder doppelte Anführungszeichen um die aus Zeichen bestehenden Daten einzuschließen (mehr über Datentypen in Abschnitt 3):

Beispiel 2.3:

'use namen'	(verwende die Datei "namen")
'list'	(liste alle Datensätze aus)
'list off'	(liste ohne Datensatz-Nummern)
'list for postltzahl = "7"'	(für alle mit PLZ = '7...')
'list off for postltzahl < '8' '	(spitze Klammer "<"!)
'list for name = "Gruber"'	(für Name = "Gruber")

Beachten Sie: Wenn Sie nur einen Teil des Inhalts eines Datenfeldes eingeben (vgl. Beispiel 2.1), dann vergleicht dBASE II nur mit diesem Teil des Feldes. Beispielsweise brauchten wir nicht Herrn Grubers vollständigen Namen angeben. Wir hätten das allerdings tun können, falls unsere Datenbank mehrere "Gruber" enthält, von denen wir nur einen ansprechen wollten.

Abb. 2.7

. list			
00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY
00002 Bergmueller,	Hans Feldweg 3	4200 Oberhausen	45 23 98 NRW
00003 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
00004 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
00005 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
00006 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY
00007 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
00008 Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
. list off			
Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY
Bergmueller, Hans	Feldweg 3	4200 Oberhausen	45 23 98 NRW
Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY
Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
. list for postltzahl = '7'			
00004 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
00007 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
. list off for postltzahl < '8'			
Bergmueller, Hans	Feldweg 3	4200 Oberhausen	45 23 98 NRW
Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
. list for name = 'Gruber'			
00007 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W

Neben der präzisen Abfrage von Daten aus Ihrer Datenbank kann der LIST-Befehl auch dazu benutzt werden, System-Informationen zu gewinnen.

'list structure' zeigt Ihnen die Struktur der Datenbank, mit der Sie gerade arbeiten. Vergleichen Sie Abb. 2.7 mit Abb. 2.2!

'list files' gibt die Namen aller Datenbank-Files ".DBF" aus, die sich auf der eingerasteten Diskette befinden (logged-in). Die "eingerastete Diskette" ist diejenige, mit deren Prompt sich das CP/M-Betriebssystem meldet (z. B. "A>" oder "B>"), wenn es auf Ihre Kommandos wartet. Um Dateien auszulisten, die sich auf einer anderen Diskette befinden, benutzen Sie die Form: list files on <drive>. Ersetzen Sie <drive> beispielsweise durch A. Lassen Sie bitte den unter CP/M üblichen Doppelpunkt weg!!!

Abb. 2.8

. use namen
. list structure

STRUCTURE FOR FILE: NAMEN.DBF

NUMBER OF RECORDS: 00008

DATE OF LAST UPDATE: 11/25/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ADRESSE	C	020	
003	POSTLTZahl	C	004	
004	STADT	C	020	
005	TELEFON	C	010	
006	BUNDESland	C	003	
** TOTAL **			00078	

. list files

DATABASE	FILES	# RCDS	LAST UPDATE
TEST	DBF	00002	00/00/00
ADRESSEN	DBF	00097	11/23/82
NAMBAK	DBF	00010	11/01/82
AUSGABEN	DBF	00000	10/19/82
AUFTRAG	DBF	00000	10/19/82
NAMEN	DBF	00008	11/25/82

Die Ausgabe von Daten mit dem DISPLAY-Befehl

Der Befehl 'display' ähnelt 'list'. Der Hauptunterschied besteht darin, daß "DISPLAY" allein nur einen (den sog. gerade laufenden) Datensatz ausgibt, während LIST ähnliches wie DISPLAY ALL bewirkt - nämlich alle Datensätze der Datei auszugeben. Die vollständige Form von DISPLAY lautet:

```
[ ALL ]
DISPLAY [ RECORD n ] [ OFF ] [FOR <ausdruck> ]
[ NEXT n ]
```

(Die übereinanderstehenden Ergänzungs-Möglichkeiten sind Alternativen).
(to display = anzeigen, all = alles, next = nächste)

Damit haben Sie die Möglichkeit, den Darstellungsbereich (scope) eines DISPLAY-Befehls anzugeben (wie auch für LIST).

Bei der Angabe von 'RECORD n' wird nur der Datensatz mit der Nummer "n" dargestellt. "NEXT n" gibt die nächsten "n" Datensätze aus, einschließlich des gerade bearbeiteten Datensatzes. DISPLAY ALL bewirkt das Gleiche wie LIST, nur mit dem Unterschied, daß nach LIST alle Datensätze der Datenbank schnell über den Bildschirm rollen, während Ihnen DISPLAY ALL den Inhalt der Datei in Gruppen von 15 Datensätzen zeigt (durch Drücken einer beliebigen Taste erscheinen die nächsten 15 Einträge).

Geben Sie folgendes ein:

'display all'	zeige alles
'display record 3'	zeige datensatz 3
'display next 4'	zeige die nächsten 4 (datensätze)

Das Ergebnis sehen Sie in Abb. 2.9.

Abb. 2.9

. display all			
00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY
00002 Bergmueller,	Hans Feldweg 3	4200 Oberhausen	45 23 98 NRW
00003 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
00004 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
00005 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
00006 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY
00007 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
00008 Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
. display record 3			
00003 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
. display next 4			
00003 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
00004 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
00005 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
00006 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY

Wie bei LIST können Sie eine FOR-Bedingung einfügen, um durch einen logischen Ausdruck bestimmte Daten auszuwählen, für die diese Bedingungen zutreffen.

Der DISPLAY-Befehl kann, ebenso wie LIST, nicht nur zum Abruf von Daten aus einer Datei, sondern auch von System-Informationen verwendet werden.

DISPLAY STRUCTURE entspricht LIST STRUCTURE
DISPLAY FILES entspricht LIST FILES

Sowohl 'list' als auch 'display' können verschiedene Arten von Files auf einer Diskette auslisten, indem Sie die "wild cards" von CP/M verwenden (soviel wie "Joker-Zeichen", Zeichen, die für beliebig lautende Teile eines Filenamens stehen). Beispielsweise zeigt Ihnen 'display files like *.com on B' alle Files auf der Diskette im "B:"-Laufwerk, deren Namen auf ".COM" enden. Falls Ihnen das nicht vertraut ist, schlagen Sie bitte in Ihrem CP/M-Handbuch nach und benutzen Sie dieses Schema:

DISPLAY FILES LIKE <Joker-Zeichen>.

Mit anderen Worten: Das Sternchen "*" steht für Teile eines Dateinamens, die Sie nicht näher festlegen wollen.
(like = wie, die aussehen wie...)

Aufsuchen eines Datensatzes mit GO oder GOTO und SKIP

Sobald Sie eine Datenbank mit USE in Bearbeitung genommen haben, können Sie leicht und rasch von Datensatz zu Datensatz wechseln. Geben Sie folgendes ein und vergleichen Sie mit Abb. 2.10:

'use namen'	verwende die Datei "namen"
'go top'	gehe zum Anfang der Datei
'display'	zeige den Datensatz
'go bottom'	gehe zum Ende der Datei
'display'	zeige den Datensatz
'goto 5'	gehe zum 5. Datensatz
'display'	zeige den Datensatz
'8'	nun zum 8.
'display'	zeige den Datensatz

Abb. 2.10

```
. use namen
. go top
. display
```

00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY
--------------------	----------------	-----------------	--------------

```
. go bottom
. display
```

00008 Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
---------------------	---------------	---------------------	-------------

```
. goto 5
. display
```

00005 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
------------------------	--------------	-----------------	--------------

```
. 8
. display
```

00008 Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
---------------------	---------------	---------------------	-------------

Mit 'go top' oder auch 'goto top' gehen Sie zum ersten Datensatz der Datei. 'go bottom' bringt Sie ans Ende der Datei. Sie können zu einem beliebigen Datensatz gehen, indem Sie 'goto <nummer>' oder 'go <nummer>' befehlen. Sie können sogar das GO weglassen und nur die Datensatz-Nummer angeben.

'skip' ("überspringe") bringt Sie zum nachfolgenden Datensatz. Mit "skip + n" können Sie "n" Datensätze vor- oder rückwärts springen. Sie können auch die Form "skip + <variable/ausdruck>" gebrauchen, dabei wird die Anzahl Datensätze, die Sie überspringen, durch den Wert der Variablen oder des Ausdrucks bestimmt (beides wird später erklärt werden).

Geben Sie folgendes ein (und vergleichen mit Abb. 2.11):

```
'display'
'skip -3'
'display'
'skip'
'display'
```

Abb. 2.11

```
. display

00008 Haberer, Karl      Teufelsweg 13      8047 Schrobenhausen      12 45 7  BAY

. skip -3
RECORD: 00005
. display

00005 Eberhart, Tobias    Marktstr. 19      5013 Ebenhausen      34 56 23 NRW

. skip
RECORD: 00006
. display

00006 Faltmann, Gisela   Daeumlingsweg 8    8049 Dirnbach      13 29 8  BAY
```

Der interaktive Frage-Befehl "?"

Sie können dBASE II auch als Tischrechner benutzen. Geben Sie einfach ein Fragezeichen (?) ein, dann ein Leerzeichen, und danach die Größe oder mathematische Formel, deren Wert Sie wissen möchten. dBASE wird Ihnen in der nächsten Zeile die Antwort ausgeben. Wenn Sie ?? verwenden, so erscheint die Antwort in der gleichen Zeile.

Geben Sie folgendes ein (und vergleichen Sie mit Abb. 2.12):

NM'? 73/3.0000'
'? 73.00/3'
'? 73/3'

Abb. 2.12

. ? 73/3.0000
24.3333
. ? 73.00/3
24.33
. ? 73/3
24

Der '?'-Befehl ergibt die Lösung einer mathematischen Operation auf so viele Stellen genau wie maximal in einer der eingegebenen Zahlen vorkommen.

Sie können '?' auch in dem Sinne von "Was enthält ..." verwenden, wobei die Pünktchen für einen Ausdruck, eine Variable (einen Feldnamen oder eine Speichervariable), eine dBASE II - Funktion oder eine Liste solcher Dinge, durch Kommata getrennt, stehen.

Probieren Sie folgendes aus (Abb. 2.13):

'use namen'	verwende die Datei "namen"
'6'	für "GOTO 6", gehe zum 6. Datensatz
'? postltzahl'	wie lautet die Postleitzahl?
'? name'	wie lautet der Name?
'go bottom'	gehe zum letzten Datensatz
'? stadt'	wie heißt die Stadt?

Abb. 2.13

. use namen
. 6
. ? postltzahl
8049 . ? name
Faltmann, Gisela
. go bottom
. ? stadt
Schrobenhausen

Im Abschnitt 5 über Funktionen und Befehle werden wir Ihnen zeigen, wie Sie mit Hilfe des '?' weitere dBASE II Funktionen benutzen können und wie Sie damit auf dem Bildschirm Anweisungen an die Bedienkraft von einer Befehls-Datei ausgeben können.

Eingeben weiterer Datensätze mit APPEND und INSERT

Sie können zu einer beliebigen Datenbank mit einem Ein-Wort-Befehl sehr einfach und rasch weitere Daten hinzufügen. Wählen Sie zunächst die Datei, zu der Sie weitere Daten hinzufügen wollen, indem Sie `USE <filename>` eingeben, dann tippen Sie das Kommando `APPEND` (hänge an):

```
'use namen'  
'append'
```

Abb. 2.14

```
. use namen  
. append  
  
RECORD #:00009  
  
NAME      :  
ADRESSE   :  
POSTLTZAHL :  
STADT     :  
TELEFON   :  
BUNDESLAND:
```

(to append = anhängen)

dBASE II antwortet, indem es die Datensatznummer anzeigt, die dem letzten Datensatz in der Datei folgt, sowie die Felder, die für diese Datenbank definiert wurden. Wenn Sie, wie am Anfang dieses Abschnitts erklärt, diesen Datensatz ausfüllen, so wird er an das Ende der Datei angehängt ("appended" im Englischen).

Die Bildschirmdarstellung enthält die Namen der Felder, ihre Länge wird durch Doppelpunkte angezeigt. Der Cursor befindet sich auf der ersten Position des ersten Feldes, wo Sie sogleich mit der Eingabe von Daten beginnen können. Wenn Sie das ganze Feld (so lang, wie es definiert ist) ausfüllen, so springt der Cursor automatisch zum nächsten Feld. Wenn Ihre Eingabe kürzer ist, erreichen Sie das Weiterspringen dadurch, daß Sie `<RETURN>` (oder `<ENTER>`) drücken.

Wenn Sie in ein Feld nichts eintragen wollen, gehen Sie einfach mit `<RETURN>` zum nächsten Feld über. Felder mit textueller Information ("C" oder "character"-Felder) werden dabei automatisch mit Leerzeichen aufgefüllt, numerische Felder erhalten den Wert Null. Bei der Eingabe von numerischen Daten ist es nicht nötig, den Dezimalpunkt zu tippen, wenn es keine Stellen nach dem "Komma" gibt. dBASE II fügt automatisch den Dezimalpunkt und die nötige Anzahl folgender Nullen ein.

Sie können Datensätze auch, statt mit APPEND nur ans Ende des Files, an einer beliebigen Stelle einfügen (etwa, um die alphabetische Reihenfolge einzuhalten), indem Sie folgendes Schema verwenden:

'INSERT [BEFORE] [BLANK]'

(to insert = einfügen, before = vor, blank = leer)

Wenn Sie das Wort INSERT allein gebrauchen, so wird der neue Datensatz unmittelbar nach dem laufenden Datensatz eingefügt. Die Angabe von BEFORE fügt den neuen Datensatz vor dem laufenden Datensatz ein. In beiden Fällen erscheint der gleiche Bildschirm-Dialog wie bei den APPEND und CREATE-Befehlen. Wenn das Wort BLANK in Ihrem Befehl vorkommt, dann wird ein leerer Datensatz eingefügt (und es erscheint kein Eingabe-Dialog zum Füllen der Datenfelder).

Fügen Sie folgende Namen an alphabetisch richtiger Stelle in Ihre "namen.dbf"-Datenbank ein:

Beispiel 2.4:

Dirschedel, Fritz	Magnolienweg 20	8000 München 90	97 81 13 BAY
Imhof, Bernhard	Gärtnerstr. 9	4703 Marktdorf	123 NRW
Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY

Die nötige Befehlsfolge sieht so aus:

'use namen'

'4'

'insert before' (Geben Sie die Daten zum 1. Namen ein)

'append' (Geben Sie die Daten der letzten Namen ein)

Abb. 2.15

. use namen
. list off all

Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY
Bergmueller, Hans	Feldweg 3	4200 Oberhausen	45 23 98 NRW
Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY
Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY
Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123 NRW
Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY

. quit

*** END RUN

dBASE II

In der Befehlsart INSERT kehrt dBASE, nachdem Sie das letzte Feld ausgefüllt haben, zur Hauptbefehlsebene zurück (mit dem Punkt-Prompt).

Um aus der APPEND-Betriebsart herauszukommen, bringen Sie den Cursor auf den Anfang des ersten Feldes in einem neuen Datensatz und drücken dann <RETURN> oder 'control-Q'. Sie können jede Befehlsart auch während der Arbeit an einem Datensatz abbrechen, indem Sie '<CTL>-W' ('<CTL>-O' auf dem Superbrain) eingeben. Damit werden die bisherigen Eingaben abgespeichert und sie kehren zur Hauptbefehlsebene mit dem Punkt zurück. Wenn Sie die Eingaben nicht speichern wollen, brechen Sie durch <CTL>-Q ab.

Die Datenbankpflege mit DELETE, RECALL, PACK

Sie können direkt von der Hauptbefehlsebene in dBASE II oder vom EDIT-Mode aus Daten löschen.

Um den laufenden Datensatz (den Sie zuletzt bearbeitet oder durch ein Positionier-Kommando angesprochen haben) zu löschen, geben Sie einfach 'delete' ein.

Um einen Datensatz, den Sie gerade mit der bildschirmorientierten Editierung (nach EDIT oder MODIFY z.B.) bearbeiten, zu löschen, geben Sie <CTL>-Q ein.

Um mehr als einen Datensatz zu löschen benutzen Sie die Form: 'delete <bereich>'. Die Bereichsangabe wird genauso wie bei anderen dBASE-Befehlen formuliert: all, record n oder next n. Um bedingte Löschungen vorzunehmen, die also alle Datensätze mit einer bestimmten Eigenschaft betreffen, verwenden Sie die Form:

'DELETE [bereich] [FOR <ausdruck>']

(to delete = löschen)

wobei "ausdruck" eine Bedingung oder eine Anzahl von Bedingungen ist, denen die zu löschenden Einträge entsprechen müssen. (Wir gehen diesbezüglich im 3. Abschnitt mehr ins Detail.)

Um eine Datei zu löschen (d.h. einen File-Eintrag im Directory Ihres Betriebssystems) geben Sie ein:

'DELETE FILE <drive>:<filename>'.

Wenn Sie dies tun, sind die betreffenden Daten für alle Zeiten verloren - seien Sie also vorsichtig!

Im Gegensatz dazu werden einzelne Datensätze in einer Datei lediglich als "gelöscht" markiert. Daher können sie wiedergewonnen ("ungelöscht") werden. Sie sehen nach dem LIST oder DISPLAY-Befehl bei den als "gelöscht" markierten Datensätzen einen Stern. dBASE II ignoriert diese Einträge dann, so als wären sie gar nicht mehr vorhanden, und bezieht sie nicht mehr in irgendwelche Arbeitsvorgänge ein.

Um diese Einträge wiederzugewinnen verwenden Sie den Befehl:

'RECALL [bereich] [FOR <ausdruck>'].

(recall = "rufe zurück", scope = "Bereich").

Diese Operation funktioniert in der entsprechend umgekehrten Weise wie DELETE (LÖSCHE) mit der Möglichkeit, auf Wunsch einen Gültigkeitsbereich und Bedingungen einzufügen. Wenn Sie einen Bedingungs-Ausdruck verwenden braucht dieser nicht der gleiche zu sein wie der, mit dem die Datensätze gelöscht worden waren.

Irgendwann aber werden Sie die Datensätze wirklich aus dem File löschen wollen, sei es, um die Darstellung mit LIST oder DISPLAY von unnötigem Ballast zu befreien, sei es, um auf Ihrer Diskette Platz zu schaffen. Das erreichen Sie mit dem Befehl:

'pack'. (packe, komprimiere)

Er entfernt alle Datensätze, die als "gelöscht" markiert waren und gibt anschließend aus, wieviele Einträge sich nun in der Datenbank befinden.

Achtung: Sobald Sie diesen Befehl ausgeführt haben, sind die betreffenden Datensätze wirklich und für immer verschwunden (es sei denn, Sie geben sie neu ein)!

Um zu sehen, wie diese Befehle arbeiten, versuchen Sie folgende Eingaben:

'use namen'	(verwende die Datei "namen")
'list'	(Liste alle Datensätze aus)
'delete record 2'	(Lösche den Datensatz Nr. 2)
'delete record 4'	(Lösche den Datensatz Nr. 4)
'list'	(Liste alle Datensätze aus)
'recall record 4'	(Mache Löschung von DS Nr. 4 rückgängig)
'list'	
'pack'	(Komprimiere die Datei)
'list'	

Die Bildschirm-Wiedergabe in Abb. 2.16 auf der nächsten Seite zeigt die ersten paar Einträge in unserer Datei <namen.dbf> während der Ausführung dieser Befehle.

Abb. 2.16

. use namen

. list

00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY
00002 Bergmueller, Hans	Feldweg 3	4200 Oberhausen	45 23 98 NRW
00003 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
00004 Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY
00005 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
00006 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
00007 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY
00008 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
00009 Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
00010 Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123 NRW
00011 Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY

. delete record 2

00001 DELETION(S)

. delete record 4

00001 DELETION(S)

. list

00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY
00002 *Bergmueller, Hans	Feldweg 3	4200 Oberhausen	45 23 98 NRW
00003 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
00004 *Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY
00005 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
00006 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
00007 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY
00008 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
00009 Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
00010 Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123 NRW
00011 Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY

. recall record 4

00001 RECALL(S)

. list

00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY
00002 *Bergmueller, Hans	Feldweg 3	4200 Oberhausen	45 23 98 NRW
00003 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
00004 Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY
00005 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
00006 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
00007 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY
00008 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
00009 Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
00010 Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123 NRW
00011 Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY

DATENBANKERZEUGUNG...76

. pack

PACK COMPLETE, 00010

RECORDS COPIED

. list

00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY
00002 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN
00003 Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY
00004 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W
00005 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW
00006 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY
00007 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W
00008 Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY
00009 Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123 NRW
00010 Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY

Zusammenfassung

Jetzt haben Sie einen Eindruck davon gewonnen, welche Manipulationen Ihrer Daten das relationale Datenbanksystem dBASE II bietet.

Sie sind jetzt in der Lage, mit CREATE (ERZEUGE) eine neue Datenbank einzurichten und in wenigen Minuten Daten in sie einzuspeichern.

Wenn Sie die Daten verändern wollen, gelingt Ihnen dies leicht mit den Befehlen EDIT (EDITIERE, BEARBEITE), DELETE (LÖSCHE), RECALL (RUFE ZURÜCK) und PACK (KOMPRIMIERE).

Mit APPEND (HÄNGE AN) oder INSERT (FÜGE EIN) können Sie nach Bedarf weitere Daten einfügen und sich mit LIST und DISPLAY einen Überblick über eine ganze Datei oder präzise ausgewählte Datensätze verschaffen. Weiter können Sie sich mit GOTO (GEHE ZU) und SKIP (ÜBERSRPRINGE) frei in Ihrer Datenbank bewegen.

Weiterhin kann dBASE II als ein interaktiver, leistungsfähiger Tischrechner (und mehr) verwendet werden, dazu benutzen Sie den "?"-Befehl.

Wir haben Sie mit Auswahlkriterien bekannt gemacht und, wie mit ihrer Hilfe die Leistung von dBASE II - Befehlen verfeinert werden kann. Im nächsten Abschnitt werden wir dieses Thema vertiefen und Ihnen zeigen, wie Sie schnell und leicht nützliche Informationen aus Ihrer Datenbank gewinnen können.

Zwei weitere Datenbanken

Zuvor aber erzeugen Sie mit create die folgenden beiden Dateien, die wir für die weiteren Beispiele benötigen werden!

Abb. 2.17

```
. create
  ENTER FILENAME: ausgaben
  ENTER RECORD STRUCTURE AS FOLLOWS:
  FIELD          NAME,TYPE,WIDTH,DECIMAL PLACES
  001             scheck:dat,c,8
  002             scheck:nr,c,5
  003             kunde,c,9
  004             auftrnumr,n,3
  005             name,c,20
  006             beschr,c,20
  007             betrag,n,9,2
  008             rech:dat,c,8
  009             rech:nr,c,7
  010             stunden,n,6,2
  011             angest:nr,n,3
  012
  INPUT DATA NOW? N
```

Die Struktur der zweiten Datei sehen Sie auf der folgenden Seite.

Abb. 2.18

```
. create
ENTER FILENAME: auftraeg
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD      NAME,TYPE,WIDTH,DECIMAL PLACES
001        kundnumr,c,9
002        artikel,c,20
003        menge,n,4
004        preis,n,7,2
005        betrag,n,9,2
006        ruecksend,l,1
007        auftrdat,c,6
008
INPUT DATA NOW? N
```

Abschnitt 3: Arithmetische Funktionen und Datenmanipulation

Inhalt

Der Gebrauch von Auswahlkriterien.....	79
Konstante und Variable (STORE).....	80
Konstante	80
Variable	81
Logische Operatoren.....	85
arithmetische Operatoren.....	85
Vergleichsoperatoren.....	86
logische Operatoren.....	87
logische \$-Substring-Such-Operatoren.....	90
String-Operatoren	92
Verändern einer leeren Datenbank-Struktur (MODIFY).....	94
Kopieren von Datenbanken und Datenstrukturen (COPY).....	95
Hinzufügen und Löschen von Feldern mit Daten in der Datenbank.....	99
Austausch von dBASE-Dateien und CP/M-Dateien mit COPY, APPEND.....	102
Umbenennen von Datensatz-Feldern mit COPY und APPEND.....	104
Veränderung von Daten mit REPLACE, CHANGE.....	106
Organisation Ihrer Datenbank mit SORT und INDEX.....	110
Suchen von Daten mit FIND und LOCATE.....	115
Berichtsauswertung der Daten mit REPORT.....	118
Automatisches Zählen und Aufsummieren mit COUNT und SUM.....	122
Summieren von Daten und Ausblenden unwichtiger Details (TOTAL).....	123
Zusammenfassung von Abschnitt 3.....	125

In diesem Abschnitt werden wir uns näher mit Bedingungs-Ausdrücken beschäftigen. Das Erweitern von Befehlen mit solchen Bedingungen gehört zu den wichtigsten Merkmalen von dBASE II, mit denen Sie sich vertraut machen sollten, damit Sie wirklich vollen Nutzen aus dem System ziehen können.

Die Befehle von dBASE II sind leicht zu merken, da sie der menschlichen Sprache ähneln, und das Lernen eines neuen Befehls entspricht der Erweiterung Ihres Wortschatzes um einen neuen Begriff. - Der deutsche Leser benötigt aber trotz der vorgegebenen englischen Befehle keine Englischkenntnisse, um mit dBASE II arbeiten zu können. Sie werden sich die wenigen Befehls-Worte rasch aneignen, die zudem wiederholt auf deutsch erklärt sind. Alle übrigen Begriffe wurden, sofern es möglich und nicht zu ungebräuchlich erschien, "eingedeutscht".

Durch die Erweiterung der dBASE-Befehle mit logischen Auswahlkriterien erzielen Sie die jeweils nötige Feinkontrolle über Ihre Daten, wie sie für Ihren besonderen Anwendungsfall nötig ist. Sobald Sie den Gebrauch dieser Ausdrücke gelernt haben, brauchen Sie sich nur mehr zwei weitere Techniken der dBASE-Programmierung anzueignen, um selbständig Ihre Anwendungs-Programme schreiben zu können. (Es handelt sich um das Benutzen von logischen Entscheidungen und die automatische Wiederholung von Anweisungen. Wir werden dies im Abschnitt 4 ausführlicher behandeln).

Der Gebrauch von Auswahl-Kriterien

Wir haben uns in Abschnitt 2 bereits kurz mit Vergleichsoperatoren beschäftigt, die zusammen mit den dort besprochenen Befehlen benutzt werden können.

Wie Sie gesehen haben, stellen diese Operatoren eine wirkungsvolle Methode zur Erweiterung der Befehle und der leichten und raschen Manipulation Ihrer Daten dar. Wenn Sie das Stichwortverzeichnis zu Rate ziehen, so werden Sie finden, daß viele der dBASE II-Befehle nach folgendem Schema erweitert werden können:

"<befehl> [FOR <ausdruck>]"

Sie werden im Handbuch immer wieder derartige Schemata finden. Alles was in <spitzen Klammern> steht, bezeichnet einen allgemeinen Begriff. Hier sollen Sie sich statt <befehl> einen der konkreten Befehle des dBASE-Systems vorstellen.

Alles was in [eckigen Klammern] steht, kann entweder fortgelassen oder dem Befehl hinzugefügt werden. Wenn es hinzugefügt wird, muß dies in der durch das Schema angezeigten Form geschehen. Das heißt (beim augenblicklichen Beispiel), daß viele Befehle durch das Schlüsselwort FOR und einen logischen "ausdruck" erweitert werden können.

Um diese Möglichkeiten auszunutzen, müssen Sie sich den Umgang mit Ausdrücken und Operatoren und deren Kombinationsmöglichkeiten vertraut machen, sowie das Zusammensetzen so erweiterter Befehle zu sog. "command files" oder Programmen, die bestimmte Aufgaben erfüllen sollen.

Die folgenden Seiten werden Ihnen den Einstieg leicht machen. Probieren Sie alles aus und denken Sie daran: Eigene Erfahrung ist der beste Lehrmeister.

Was übrig bleibt: Wir versuchen Ihnen bei der Erläuterung diverser Befehle im Text einiges von dem zu zeigen, was Sie mit Ihrer Datenbank anfangen können. Wir können aber nicht alle Möglichkeiten eines Befehls auf einmal erklären. Um sich einen umfassenden Überblick über die Anwendungsmöglichkeiten eines Befehls zu verschaffen, benutzen Sie bitte die Zusammenfassung im ersten Abschnitt des Kapitels 2 und die genaue Beschreibung der einzelnen Befehle im Kapitel 3.

Konstante und Variable (STORE)

Um Daten in Ihrer Datenbank zu manipulieren (vgl. "DISPLAY") werden bestimmte "Datenformen" verwendet, die Konstante oder Variable heißen.

Konstante sind Dateneinheiten, die sich nicht verändern, gleichgültig, ob sie sich in einer Datei oder im Computer befinden. Sie heißen Literale (etwa: "wörtlich" selbsterklärende Werte) und bedeuten genau das, was ihrem Wortlaut entspricht. Im Gegensatz dazu stehen Variablen-Namen stellvertretend für einen veränderlichen Inhalt. Wenn man auf einen Zettel "Komme gleich" schreibt, dann bedeutet der Text, was sein Inhalt dem Leser sagt, er ist eine Konstante. Wenn man an ein Schubfach schreibt "Nachricht" und jemand mitteilt, er solle die Nachricht im Schubfach "Nachricht" lesen, so besteht die Nachricht nicht aus dem Text "Nachricht", sondern aus dem, was im Schubfach mit der Bezeichnung "Nachricht" drinliegt - und das können verschiedene Nachrichten sein. Der Empfänger kann die Nachricht auch herausnehmen und durch eine andere ersetzen. In dBASE sind Konstante z. B. in Anführungszeichen geschriebene Texte. Variablennamen (ohne Anführungszeichen geschrieben) verweisen auf ein "Schubfach", d.h. eine Stelle im Computerspeicher oder in einer Datenbank, in dem sich verschiedene und wechselnde Informationen befinden können.

Characters (Zeichen) und character strings (Zeichenketten, Sätze usw.), bestehen aus allen druckbaren Zeichen einschließlich dem Zwischenraum, sie können ebenfalls Konstante sein, müssen aber etwas anders behandelt werden. Sie bestehen aus Zeichen und werden als Zeichenketten oder "Strings" bezeichnet:

"Strings" sind einfach eine Aneinanderfügung von Zeichen (einschließlich Zwischenraum, Ziffern und Sonderzeichen), die bearbeitet oder als Daten benutzt werden können. Ein "substring" ist ein Teilstring, also ein Teil irgendeines bestimmten Strings.

Wenn ein Zeichen (Character) oder eine Zeichenkette (String) als String Konstante behandelt werden soll, muß es (sie) in einfache oder doppelte Anführungszeichen oder in eckige Klammern eingeschlossen werden, damit der Computer versteht, daß die Zeichen für sich selbst stehen sollen. Um zu sehen, wie das gemeint ist, starten Sie dBASE II auf Ihrem Computer und nehmen die Datei "namen" in Bearbeitung. Geben Sie ein:

Beispiel 3.1:

```
'dbase'
'use namen'
'? "name"'
'? name'
```

Auf den ersten "was ist..."-Befehl (das ?) antwortet der Computer mit "name", weil das der Wert der Konstanten 'name' ist. Wenn Sie die einfachen Anführungszeichen weglassen, dann prüft der Computer zunächst, ob es sich bei dem Wort "name" um einen Befehl handelt. Da dies nicht der Fall ist, prüft er anschliessend, ob es sich vielleicht um den Namen einer Variablen handelt (das ist hier der Fall, denn es ist der Name des Namen-Feldes).

Variablen sind Datenobjekte, deren Inhalt sich verändern kann. Häufig sind sie Namen von Datenfeldern, deren Inhalt sich wandeln kann. In diesem Fall fand der Computer, daß unser Datensatz (in der Datei "namen") ein Feld namens "name" hat, daher gab er uns den gegenwärtigen Inhalt dieses Feldes aus. Geben Sie folgendes ein:

```
'skip 3'
'? name'
```

Abb. 3.1

```
dbase
. use namen
. ? 'name'
name
. ? name
Adams, Peter
. skip 3
RECORD: 00004
. ? name
Dollinger, Erwin
. use
. ? name

*** SYNTAX ERROR ***
?
? NAME
CORRECT AND RETRY (Y/N)? N
```

Geben Sie nun 'use' ein. Da wir keinen Dateinamen angeben, schließt das System einfach alle Dateien.

Wenn Sie nun wieder '? name' tippen, so erhalten Sie eine Fehlermeldung. In diesem Fall haben Sie versucht, eine Variable zu benutzen, die nicht existiert, weil keine Datei mehr im Gebrauch war, in der ein Feld mit einem korrespondierenden Feldnamen vorkommt.

Variable können außer Feldnamen einer Datenbank auch Speichervariable sein. dBASE II reserviert einen Speicherbereich in Ihrem Computer, in dem bis zu 64 Variable gespeichert werden können. Jede davon darf maximal 254 Zeichen lang sein, alle Variablen zusammen dürfen aber nicht mehr als 1536 Zeichen belegen.

Sie können sich 64 Schubfächer vorstellen, in denen Sie vorübergehend Daten ablegen können, während Sie an einem Problem arbeiten.

Variablen-Namen können aus jedem gültigen dBASE II-Bezeichner bestehen (d.h. mit einem Buchstaben anfangen, bis zu zehn Zeichen lang sein und eingefügte Doppelpunkte und Ziffern, aber keine Zwischenräume enthalten). Wie gesagt, entsprechen Variablen-Namen Etiketten, die man auf die Schubfächer geklebt hat, und sind nicht mit dem Inhalt des Schubfaches zu verwechseln, das sie bezeichnen.

Sie können Speichervariable verwenden, um vorübergehend Daten zu speichern oder um Eingabedaten getrennt von Datenfeldern zu behandeln. Beispielsweise können wir während einer Sitzung (mit dem dBASE-System) das Datum in so einem Schubfach ablegen, welches den Namen "datum" erhält. Später können wir es wieder abfragen, indem wir nach dem Inhalt von "datum" fragen, und ihn in irgendein Datenfeld in einer beliebigen Datenbank einspeichern, ohne daß wir dabei das jeweilige Datum neu eingeben müßten.

Um Daten (Zeichen, Zahlen oder logische Werte) in eine Speichervariable zu bringen können Sie den Befehl "STORE" (SPEICHERE) verwenden. Die vollständige Form lautet:

"STORE <ausdruck> TO <speicher variable>"

Geben Sie folgendes ein:

Beispiel 3.2:

```
'store "Wie geht's denn so?" to nachricht'  
'store 10 to stunden'  
'store 17.35 to stden:satz'  
'? stden:satz * stunden'  
'? nachricht'
```

Abb. 3.2

```

. store "Wie geht's denn so." to nachricht
Wie geht's denn so.
. store 10 to stunden
10
. store 17.35 to stden:satz
17.35
. ? stden:satz * stunden
173.50
. ? nachricht
Wie geht's denn so.

```

Beachten Sie, daß wir doppelte Anführungszeichen zum Einklammern des Strings (der eine Konstante darstellt) in der ersten Zeile von Beispiel 3.2 verwendet haben, weil wir das einfache Anführungszeichen als Apostroph im Inneren des Strings verwenden wollten.

Sollte Ihnen das noch nicht klar sein, so experimentieren Sie ein wenig mit und ohne Anführungszeichen, um sich den Unterschied zwischen Variablen und Konstanten klarzumachen.

Fangen Sie beispielsweise so an:

Beispiel 3.3:

```

'store 99 to variable'
'store 33 to andere'
'store variable/andere to dritte'
'store '99' to konstante'
'? variable/andere'
'? variable/3'
'? konstante/3'
'display memory'

```


Abb. 3.3

```

. store 99 to variable
99
. store 33 to andere
33
. store variable/andere to dritte
3
. store '99' to konstante
99
. ? variable/andere
3
. ? variable/3
33
. ? konstante/3

*** SYNTAX ERROR ***
?
? KONSTANTE/3
CORRECT AND RETRY (Y/N)? N
. display memory
NACHRICHT      (C)  Wie geht's denn so.
STUNDEN        (N)  10
STDEN:SATZ     (N)  17.35
VARIABLE       (N)  99
ANDERE         (N)  33
DRITTE         (N)  3
KONSTANTE      (C)  99
** TOTAL **    07  VARIABLES USED 00051 BYTES USED

```

Wenn Sie in eine Variable Daten einspeichern, entnimmt dBASE der Form dieser Daten automatisch den Datentyp. Von diesem Augenblick an können Sie die Datentypen nicht mehr mischen (etwa, indem Sie eine Zeichenkette (String) durch eine Zahl zu teilen versuchen, oder indem Sie versuchen, in eine bereits bestehende Variable Daten eines anderen als des ursprünglichen Typs einzuspeichern). Wir sagen, eine Variable wird bei ihrem erstmaligen Gebrauch "erzeugt". Danach bleibt sie, auch wenn wir sie nicht mehr verwenden, so lange bestehen, bis wir sie mit einem unten beschriebenen Befehl vernichten.

Namen, also Worte, die mit einem Zeichen beginnen, werden entweder als Befehl oder als Variablen-Name interpretiert. Worte, die in Anführungszeichen oder eckige Klammern eingeschlossen sind, werden als Texte (String-Konstanten) behandelt (auch, wenn sie nur Ziffern enthalten). "Worte", die mit einer Zahl anfangen, werden als numerische Konstante behandelt, sie dürfen nur weitere Ziffern und ev. einen Dezimalpunkt enthalten.

VORSCHRIFTEN: Zeichenketten, die in Ausdrücken auftreten, müssen beidseitig übereinstimmend in doppelte oder einfache Anführungszeichen oder in eckige Klammern eingeschlossen werden. Zeichenketten dürfen jedes druckbare Zeichen (einschließlich des Leerzeichens) enthalten. Wenn Sie das Und-Zeichen (&) verwenden wollen, dann muß es zwischen zwei Leerzeichen stehen, da es sonst die Bedeutung des dBASE II-Makrozeichens (siehe Abschnitt 5) hat.

Die letzte Anweisung in Abb. 3.3 zeigt eine weitere Ergänzung des DISPLAY-Befehls (Sie können auch LIST MEMORY verwenden).

Sie können eine einzelne Variable vernichten, indem Sie 'release <variablen-name>' befehlen, oder Sie können alle Speichervariablen löschen durch 'release all'.

Geben Sie folgendes ein (ev. löschen Sie zuerst mit 'erase' den Bildschirm):

'display memory'
'release andere'
'display memory'
'release all'
'display memory'

Vorschläge: Machen Sie Variablen-Namen so lang, daß ihre Bedeutung jedermann verständlich ist (soweit im Deutschen mit zehn Zeichen möglich, die Angelsachsen tun sich hier leichter).

Wenn Sie Datenfeldnamen nur neun Zeichen lang machen, können entsprechende Namen als Speichervariable verwenden, indem Sie einfach ein "M" (für Memory, oder, wenn Sie wollen, ein "S" für Speicher) vor den Namen setzen. Besonders wenn Sie später einmal Ihr Programm überarbeiten ist das viel einleuchtender und klarer als wenn Sie völlig neue Namen für verwandte Zwecke erfinden!

Die logischen Operatoren

Mit Hilfe der logischen Operatoren manipulieren Sie Ihre Daten. Sie werden durch die vier Grund-Rechenzeichen und die Vergleichs-Operatoren ausgedrückt. Die meisten werden Ihnen vertraut sein, an einige müssen Sie sich vielleicht erst gewöhnen.

Arithmetische Operatoren sind am bekanntesten. Sie ergeben arithmetische (rechnerische) Ergebnisse.

()	: Klammern zur Zusammenfassung von Ausdrücken
*	: Multiplikation
/	: Division
+	: Addition
-	: Subtraktion

Die arithmetischen Operatoren werden in der Reihenfolge ihres Vorrangs ausgewertet. Die Rangordnung ist (mit abnehmendem Vorrang): Klammern, Multiplikation und Division, Addition und Subtraktion. Wenn Operatoren gleichen Vorrang haben, werden sie von links nach rechts ausgewertet. Hier einige Beispiele:

$17/33*72 + 8 = 45.09$	(Division, Multiplikation und dann Addition)
$17/(33*72 + 8) = 0.00644$	(Multiplikation, Addition und schließlich Division)
$17/33*(72 + 8) = 41.21$	(Division, Addition und dann Multiplikation)

Vergleichs-Operatoren (relationale Operatoren) vergleichen und ergeben ein logisches Resultat (wahr oder falsch).

< : kleiner als
 > : größer als
 = : gleich
 <> : ungleich
 <= : kleiner oder gleich
 >= : größer oder gleich

Geben Sie folgendes ein:

'use namen'
 'list for postltzahl <= '7000'
 'list for telefon <> '123'
 'list for name = 'Haberer''

Sie haben es sicher schon bemerkt - "Größen-Vergleiche" zwischen Zeichenketten beziehen sich auf die lexikalische Reihenfolge. 'A' < 'B' ist z. B. wahr, 'C' > 'D' ist falsch. '0' (Null) < '1' ist deshalb wahr, weil man die Zahlzeichen so aufzählt: '0', '1', '2', '3'... usw. Genauer gesagt, kommt es hier auf die Reihenfolge der sogenannten ASCII-Codes an, durch welche die Zeichen im Computer dargestellt werden.

Abb. 3.4 - siehe Listing

. use namen
 . list for postltzahl <= '7000'

00002 Camphausen, Eva	Muellerstr. 7	Berlin 2	1000 63 78 12 BLN
00005 Eberhart, Tobias	Marktstr. 19	Ebenhausen	5013 34 56 23 NRW
00009 Imhof, Bernhard	Gaertnerstr. 9	Marktdorf	4703 123 NRW
00010 Jahn, Alois	Huehnersteig 3	Oberhof	6142 456 BAY

. list for telefon <> '123'

00001 Adams, Peter	Bergstrasse 15	Muenchen 2	8000 12 34 56 BAY
00002 Camphausen, Eva	Muellerstr. 7	Berlin 2	1000 63 78 12 BLN
00003 Dirschedel, Fritz	Magnolienweg 20	Muenchen 90	8000 97 81 13 BAY
00004 Dollinger, Erwin	Hanfstr. 43	Unterdorf	7432 12 45 B-W
00005 Eberhart, Tobias	Marktstr. 19	Ebenhausen	5013 34 56 23 NRW
00006 Faltmann, Gisela	Daeumlingsweg 8	Dirnbach	8049 13 29 8 BAY
00007 Gruber, Otto	Mozartstr. 29	Pfaffenhausen	7080 46 28 B-W
00008 Haberer, Karl	Teufelsweg 13	Schrobenhausen	8047 12 45 7 BAY
00010 Jahn, Alois	Huehnersteig 3	Oberhof	6142 456 BAY

. list for name = 'Haberer'

00008 Haberer, Karl	Teufelsweg 13	Schrobenhausen	8047 12 45 7 BAY
---------------------	---------------	----------------	------------------

Die logischen Operatoren erweitern die Möglichkeiten der Manipulation von Daten, Datensätzen und Datenbanken wesentlich.

Es würde den Rahmen dieses Handbuches sprengen, tiefer in die Regeln der formalen Logik einzudringen. Das dürfte auch nicht nötig sein. Weiter unten finden Sie ein paar Beispiele, die Ihnen sicher den Umgang mit den logischen Operatoren klarmachen.

Die logischen Operatoren erzeugen, wie die relationalen (Vergleichs-) Operatoren, logische Ergebnisse (wahr oder falsch). Sie sind in Beispiel 3.4 in der Reihenfolge ihres Vorrangs aufgezählt (.NOT. wird vor .AND. angewendet usw.):

Beispiel 3.4:

()	: Klammern zur Zusammenfassung von Ausdrücken
.NOT.	: logisches NICHT (einstelliger Operator)
.AND.	: logisches UND
.OR.	: logisches ODER
\$	: Suche nach einem Teilstring (Test auf Enthaltensein)

Die Eingabe:

Beispiel 3.5:

```
'list for (arbnnummer = 730 .or. arbnnummer = 731);
      .and. (rech:dat >= '791001'.and.;
            (rech:dat <= '791031'))'
```

gibt alle Datensätze vom Oktober 1979 aus, die Rechnungen für die Auftrags-Nummern 730 und 731 betreffen. (Beachten Sie, wie die Befehls-Zeile mit Hilfe des Semikolons auf mehrere Bildschirmzeilen erweitert wurde!)

AUSDRÜCKE...88

Falls Ihnen noch nicht klar ist, woraus sich das ergibt, mögen Ihnen folgende Überlegungen helfen: Jeder logische Operator verknüpft zwei logische Ausdrücke, die selbst den Wert "wahr" oder "falsch" besitzen, und produziert daraus wiederum einen Wahrheitswert. Zum Beispiel ist von den beiden Aussagen

"es regnet"

und

"die Sonne scheint"

höchstens eine wahr (wenn wir einmal von besonderen Wetterphänomenen absehen). Angenommen, heute scheint die Sonne, dann ist "es regnet" falsch, der Ausdruck

Ausdruck I

"es regnet .ODER. die Sonne scheint"

ist dennoch wahr, da es für den .ODER.-Operator genügt, daß einer der beiden Ausdrücke wahr ist. Dagegen ist

Ausdruck II

"es regnet .UND. die Sonne scheint"

immer falsch, egal, wie das Wetter ist.

Wenn wir aber - beim Auftreten eines Regenbogens - annehmen, daß es regnet und die Sonne scheint, dann sind beide Aussagen und damit auch beide Ausdrücke wahr, d.h. das .ODER. ist ein "ODER/UND", kein "ENTWEDER ODER".

Dagegen ist der Ausdruck "es ist kalt" .UND. "die Sonne scheint" beispielsweise an schönen Wintertagen wahr, an den - ohnehin seltenen - Sommertagen aber falsch. In einem mit .UND. (im Dialog ".AND.") verknüpften Ausdruck müssen stets beide Teilausdrücke wahr sein, damit der ganze Satz wahr ist.

Der .NOT. oder .NICHT.-Operator schließlich kehrt den Wahrheitswert eines Ausdrucks um. Wenn es z. B. heute regnet, dann ist "die Sonne scheint" falsch, ".NOT. (die Sonne scheint)" ist aber wahr. Interessant ist, daß

Ausdruck III

"(.NOT. (die Sonne scheint) .OR. (die Sonne scheint))"

immer wahr ist, ob die Sonne nun scheint oder nicht, denn entweder ist die Aussage wahr, oder aber ihre Verneinung ist wahr, der ODER-verknüpfte Ausdruck ist aber immer wahr, wenn mindestens eine seiner Teilaussagen wahr ist.

Nun noch einmal zu unserem Beispiel 3.5:

dBASE II entscheidet zunächst über die Wahrheit folgender relationaler Ausdrücke:

- 1.) Ist die Arbeitsnummer gleich 730 (wahr oder falsch)
- 2.) Ist die Arbeitsnummer gleich 731 (wahr oder falsch)
- 3.) Ist das Rechnungsdatum größer oder gleich '791001' (w o. f)
- 4.) Ist das Rechnungsdatum kleiner oder gleich '791031' (w o. f)

Anschließend wird Ausdruck 1 mit Ausdruck 2 durch .OR. (ODER) verknüpft, es entsteht also der Wert "wahr", wenn eine der beiden Fragen (oder beide) bejaht werden kann (können). Dieses Ergebnis wird dann mit dem zweiten Klammersausdruck durch .AND. (UND) verknüpft, der seinerseits fordert, daß das Rechnungsdatum größer oder gleich '791001' UND zugleich kleiner oder gleich '791031' ist.

Die Klammern werden, wie in rechnerischen Ausdrücken, verwendet, um die Reihenfolge der Auswertung zu bestimmen. Das .AND. zwischen den beiden Klammern soll sich auf die Aussage (Arbeitsnummer = 730 ODER Arbeitsnummer = 731) und die zweite Klammer beziehen, nicht etwa, wie es die Vorrang-Regeln in Beispiel 3.4 bestimmen würden, wenn man die Klammern fortließe, auf die Feststellung: "Arbeitsnummer = 731 UND Rechnungsdatum >= '791001'".

dBASE II prüft diese Bedingungen für jeden einzelnen Datensatz in der Datei und listet nur diejenigen aus, für welche der gesamte zusammengesetzte Ausdruck wahr ist.

Wir wollen das nun noch mit unserer Adress-Datei ausprobieren. Geben Sie folgendes ein:

Beispiel 3.6:

```
'use namen'
'display all for postltzahl > '5' .und. postltzahl < '8' '
'display all for name < 'F' '
'display all for telefon > '40' .OR. telefon < '60' '
```

Abb. 3.5

```
. use namen
. display all for postltzahl > '5' .and. postltzahl < '8'
00004 Dollinger, Erwin      Hanfstr. 43      7432 Unterdorf      12 45      B-W
00007 Gruber, Otto      Mozartstr. 29      7080 Pfaffenhausen      46 28      B-W
00010 Jahn, Alois      Huehnersteig 3      6142 Oberhof      456      BAY

. display all for name < 'F'
00001 Adams, Peter      Bergstrasse 15      8000 Muenchen 2      12 34 56 BAY
00002 Camphausen, Eva      Muellerstr. 7      1000 Berlin 2      63 78 12 BLN
00003 Dirschedel, Fritz      Magnolienweg 20      8000 Muenchen 90      97 81 13 BAY
00004 Dollinger, Erwin      Hanfstr. 43      7432 Unterdorf      12 45      B-W
00005 Eberhart, Tobias      Marktstr. 19      5013 Ebenhausen      34 56 23 NRW

. display all for telefon > '40' .or. < '60'
00001 Adams, Peter      Bergstrasse 15      8000 Muenchen 2      12 34 56 BAY
00002 Camphausen, Eva      Muellerstr. 7      1000 Berlin 2      63 78 12 BLN
00003 Dirschedel, Fritz      Magnolienweg 20      8000 Muenchen 90      97 81 13 BAY
00004 Dollinger, Erwin      Hanfstr. 43      7432 Unterdorf      12 45      B-W
00005 Eberhart, Tobias      Marktstr. 19      5013 Ebenhausen      34 56 23 NRW
00006 Faltmann, Gisela      Daeumlingsweg 8      8049 Dirnbach      13 29 8 BAY
00007 Gruber, Otto      Mozartstr. 29      7080 Pfaffenhausen      46 28      B-W
00008 Haberer, Karl      Teufelsweg 13      8047 Schrobenhausen      12 45 7 BAY
00009 Imhof, Bernhard      Gaertnerstr. 9      4703 Marktdorf      123      NRW
00010 Jahn, Alois      Huehnersteig 3      6142 Oberhof      456      BAY
```

Beachten Sie, was bei dem letzten Befehl passiert ist: Alle Datensätze wurden aufgelistet. Das ist ganz klar - egal wie die Telefon-Nummer lauten mag, wenn sie nicht "größer als '40'" auf den ersten beiden Stellen ist, dann ist sie aber doch "kleiner als '60'", und wenn sie nicht "kleiner als '60'" ist, dann aber doch "größer als '40' ". Achten Sie darauf, daß Sie keine solchen sinnlosen Auswahlkriterien formulieren, die in Wahrheit gar nichts auswählen, weil sie auf jeden denkbaren Inhalt der Datensätze zutreffen.

Der logische \$-Substring-Such-Operator ist sehr hilfreich durch seine Fähigkeit, zu entscheiden, ob eine bestimmte Zeichenfolge in einem String vorkommt. Sein Format lautet:

"<teilstring> \$ <string>"

Der Operator sucht den links stehenden Teilstring in dem String auf der rechten Seite. Einer oder beide Terme können String-Variable oder String-Konstante sein. Das Ergebnis der Operation ist "wahr", wenn der Teilstring im rechts stehenden Vergleichsstring gefunden wird, "falsch", wenn er nicht darin vorkommt. Verwechseln Sie den logischen String-Suchoperator, der die Frage "ist ein Teilstring in einem anderen String enthalten" mit wahr oder falsch beantwortet, nicht mit den nachfolgend besprochenen String-Operatoren, die Strings als Ergebnis liefern.

Um zu sehen, wie das funktioniert, geben Sie folgendes ein:

Beispiel 3.7:

```
'use namen'
'list for 'll' $ name'
'list for '1' $ adresse'
'list for 'NRW' $ bundesland'
'? 'oo' $ 'Hollywood'
'go 6'
'display'
'? bundesland $ 'BAYERN' '
```

Abb. 3.6

```
. use namen
. list for 'll' $ name

00004 Dollinger, Erwin      Hanfstr. 43      7432 Unterdorf      12 45      B-W

. list for '1' $ adresse
00001 Adams, Peter         Bergstrasse 15    8000 Muenchen 2     12 34 56 BAY
00005 Eberhart, Tobias     Marktstr. 19      5013 Ebenhausen     34 56 23 NRW
00008 Haberer, Karl       Teufelsweg 13     8047
                               Schrobenhausen     12 45 7      BAY

. list for 'NRW' $
bundesland
00005 Eberhart, Tobias     Marktstr. 19      5013 Ebenhausen     34 56 23 NRW
00009 Imhof, Bernhard     Gaertnerstr. 9    4703 Marktdorf      123        NRW

. ? 'oo' $ 'Hollywood'
.T.
. go 6
. display

00006 Faltmann, Gisela    Daeumlingsweg 8   8049 Dirnbach       13 29 8      BAY

. ? bundesland $
'BAYERN'
.T.
```

Sie erhalten für zwei der Ausdrücke auf dem Bildschirm das Ergebnis ".T.". Dies und ".F." sind "logische Konstante", die für die Werte wahr (true, daher ".T.") und falsch (false, daher ".F.") stehen.

Unter Berücksichtigung dieser Funktion hätten wir die Datenstruktur unserer Namen-Datei auch vereinfachen können. Beispielsweise hätte das Bundesland Teil des Datenfeldes "stadt" sein können. Um Datenfelder aufzurufen, die zu einem bestimmten Bundesland gehören, könnten wir dann einfach folgendes eingeben (dabei ist XXX stellvertretend für das jeweils gewünschte Bundesland):

'<befehl> for 'XXX' \$ stadt'

String-Operatoren liefern Strings als Ergebnis.

- + = String-Verkettung (exakt)
- = String-Verkettung (Leerzeichen ans Ende verschoben)

"Verkettung" bedeutet ganz einfach, daß eine Zeichenkette an eine andere angehängt wird. Geben Sie ein:

Beispiel 3.8

```
'use namen'
'? name + adresse'
'? name - adresse'
'? 'Der Name in diesem Datensatz ist' + name;
- ' und die Adresse lautet ' + adresse'
```

Abb. 3.7a

```
. use namen
. ? name + adresse
Adams, Peter          Bergstrasse 15
. ? name - adresse
Adams, Peter Bergstrasse 15
. ? 'Der Name in diesem Datensatz ist' + name;
- 'und die Adresse lautet' + adresse
Der Name in diesem Datensatz ist Adams, Peter und die Adresse lautet          Bergstrasse 15
```

Das sieht allerdings noch nicht ganz so aus, wie man es sich wünschen möchte. Woher kommt der unschöne lange Abstand zwischen den Worten "lautet" und "Bergstraße"? Die Erklärung findet man, wenn man ein wenig über die Beschreibung dessen nachdenkt, was die beiden String-Operatoren tun: "+" verknüpft Strings, wie sie vorliegen, "-" schiebt nachfolgende Leerzeichen ans Ende. Daher sammeln sich in Strings, die mehrmals mit "-" verknüpft wurden, alle nachfolgenden Leerzeichen an - es werden immer mehr. Es hülfe auch nichts, zwischen '...die Adresse lautet' und "+ adresse" ein "-" zu setzen - dann werden die beiden Strings ganz ohne Zwischenraum aneinandergeklatscht. Stattdessen hilft die "Trim"-Funktion weiter, die alle nachfolgenden Leerzeichen löscht. Nachdem das geschehen ist, kann man kontrolliert wieder genau ein Leerzeichen einfügen, wie die Abb. 3.7b zeigt:

Abb. 3.7b

```
. use namen
. ? trim ('Der Name in diesem Datensatz ist' + name;
- 'und die Adresse lautet') + ' ' + adresse
```

Der Name in diesem Datensatz ist Adams, Peter und die Adresse lautet Bergstrasse 15

Sowohl "+" als auch "-" hängen zwei Zeichenketten (Strings) aneinander. Das "plus"-Zeichen verbindet die beiden Strings genau so, wie sie vorliegen. Das "minus"-Zeichen dagegen schiebt nachfolgende Leerzeichen im linken String an das Ende des neuen Strings. Wie Sie aus Abb. 3.7 ersehen, ergibt das eine ansprechendere Darstellung beispielsweise bei der Verkettung von Datenfeldern, deren Einträge durch Leerzeichen bis zur maximal zulässigen Eintragslänge aufgefüllt sind. Die Leerzeichen sind zwar nach der Verkettung nicht verschwunden, aber für die meisten Zwecke genügt das Verschieben ans Ende, da die überflüssigen Leerzeichen in der Darstellung nicht mehr sichtbar sind (dafür bleibt die entsprechende Gesamtlänge der Felder erhalten).

Wenn Sie die Leerzeichen ganz beseitigen wollen, so können Sie dies durch die Funktion "TRIM" (trimme) erreichen. Dies geschieht durch die Eingabe von:

```
store trim ( <variable> ) to <variable>
```

Beispielsweise hätten wir eingeben können: "store trim (name) to name" um die Leerzeichen, die auf die Buchstaben des Namens folgen, zu entfernen.

Um alle nachfolgenden Leerzeichen in Beispiel 3.8 zu beseitigen hätten wir auch eingeben können: 'store trim (name - adresse) to beispiel'.

Nachdem wir Sie jetzt mit Ausdrücken und dBASE II-Operatoren vertraut gemacht haben, fahren wir mit weiteren dBASE II-Befehlen fort. Sie werden später beim Entwerfen von Programmen einige Erfahrung im Umgang mit Ausdrücken und Operatoren erwerben.

Verändern einer leeren Datenbank-Struktur (MODIFY)

WARNUNG: Der Befehl "MODIFY" löscht den Inhalt Ihrer Datenbank! Bitte beachten Sie die folgenden Anweisungen genau!

Wenn sich in einer Datenbank noch keine Daten befinden, dann ist der MODIFY-Befehl der schnellste und leichteste Weg, um die Struktur einer Datenbank zu ergänzen, Datenfelder daraus zu entfernen, sie umzubenennen, ihre Größe oder andere Eigenschaften zu verändern.

Dadurch werden jegliche Daten in dieser Datei zerstört, verwenden Sie den Befehl also nicht, nachdem Sie bereits Daten eingegeben haben!

Etwas weiter unten werden wir Ihnen einen Weg zeigen, Dateien, die bereits Daten enthalten, zu verändern, ohne daß die Daten verloren gehen.

Die Datei "ausgaben.dbf" enthält noch keine Daten, daher werden wir sie zur Demonstration benutzen. Eine sinnvolle Änderung könnte darin bestehen, das Feld "AUFTRNUMR" in "AUFTR:NR" umzutaufen, so daß seine Abkürzung dem gleichen Schema entspricht wie bei "ANGEST:NR" und "RECH:NR". Geben Sie dazu folgendes ein:

```
'use ausgaben'
'list structure'
'modify structure'
'y'
```

Abb. 3.8

```
. use ausgaben
. list structure
STRUCTURE FOR FILE: AUSGABEN.DBF
NUMBER OF RECORDS: 00000
DATE OF LAST UPDATE: 11/25/82
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	SCHECK:DAT	C	007	
002	SCHECK:NR	C	005	
003	KUNDE	C	003	
004	AUFTRNUMR	N	003	
005	NAME	C	020	
006	BESCHR	C	020	
007	BETRAG	N	009	002
008	RECH:DAT	C	007	
009	RECH:NR	C	007	
010	STUNDEN	N	006	002
011	ANGEST:NR	N	003	
** TOTAL **			00091	

```
. modify structure
MODIFY ERASES ALL DATA RECORDS ... PROCEED? (Y/N) Y
. quit
*** END RUN          dBASE II          ***
```

dBASE II löscht den Bildschirm und stellt die ersten 16 (oder weniger) Datenfelder der Datenstruktur dar. Um mit dem Cursor ein Feld nach unten zu gehen, benutzen Sie <CTL-X>. Geben Sie dann einfach den gewünschten neuen Feldnamen über den alten ein (benutzen Sie ggf. die Leertaste, um übriggebliebene Buchstaben zu löschen, oder den Befehl <CTL>-Y, um ein ganzes Feld zu löschen, oder <CTL>-G, um ein Zeichen (vorwärts) zu löschen.

Sie können die Betriebsart "MODIFY" auf zweierlei Weisen abbrechen: Mit <CTL-W> (= Write (schreibe)) (<CTL-0> auf dem Superbrain), wird die auf Diskette gespeicherte Datenstruktur entsprechend Ihren Änderungen revidiert, und Sie kehren zum dBASE II-Prompt zurück.

Mit <CTL-Q> (= Quit ("quittiere den Dienst")) kehren Sie in das dBASE II zurück ohne daß Ihre Änderungen abgespeichert werden. Auch Ihre Daten werden in diesem Fall nicht überschrieben (falls die Datei bereits welche enthielt), sondern bleiben erhalten. Auf den nächsten Seiten wird gezeigt, wie Sie diese durch eine Sicherheitskopie retten und in eine geänderte Datenstruktur übertragen können.

Kopieren von Datenbanken und Datenstrukturen (COPY)

Es ist recht einfach, eine Datei (File) zu kopieren, ohne daß Sie dazu in das Betriebssystem Ihres Computers zurückkehren müssen (Sie brauchen also das dBASE II-System dafür nicht zu verlassen). Geben Sie folgendes ein:

'use namen'	(benutze "namen")
'copy to temp'	(kopiere nach "temp")
'use temp'	(benutze "temp")
'display structure'	(stelle die Datenstruktur dar)
'list'	(liste die Datensätze aus)

Abb. 3.9

```
. use namen
. copy to temp
00010 RECORDS COPIED
. use temp
. display structure
STRUCTURE FOR FILE: TEMP.DBF
NUMBER OF RECORDS: 00010
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ADRESSE	C	020	
003	POSTLTZAHL	C	004	
004	STADT	C	020	
005	TELEFON	C	010	
006	BUNDESLAND	C	003	
** TOTAL **			00078	

```
. list
```

00001	Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56	BAY
00002	Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12	BLN
00003	Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13	BAY
00004	Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45	B-W
00005	Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23	NRW
00006	Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8	BAY
00007	Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28	B-W
00008	Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7	BAY
00009	Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123	NRW
00010	Jahn, Alois	Huehnersteig 3	6142 Oberhof	456	BAY

Warnung: Wenn Sie mit "COPY" eine Datei auf einen Filenamen kopieren, der schon benutzt ist, d.h. unter dem bereits eine Datei existiert, dann wird die alte Datei überschrieben und ihre Daten gehen verloren!

"COPY TO TEMP" erzeugt eine neue Datenbank unter dem Namen "temp.dbf". Sie ist eine identische Kopie von "namen.dbf", besitzt also die gleiche Struktur und enthält die gleichen Daten.

Der Befehl kann aber auch erweitert werden:

COPY TO <dateiname> [STRUCTURE] [FIELD Liste von Datenfeldern]

Mit dieser Form des Befehls können Sie lediglich die Datenstruktur oder Teile der Datenstruktur in eine neue Datei kopieren. Geben Sie folgendes ein:

'use namen'

'copy to temp structure'

'use temp'

'display structure'

• (kopiere (nur) die Datenstruktur)

Abb. 3.10

```
. use namen
. copy to temp structure
. use temp
. display structure
```

STRUCTURE FOR FILE: TEMP.DBF

NUMBER OF RECORDS: 00000

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ADRESSE	C	020	
003	POSTLTZahl	C	004	
004	STADT	C	020	
005	TELEFON	C	010	
006	BUNDESland	C	003	
** TOTAL **			00078	

Wir können auch nur einen Teil der Datenstruktur kopieren, indem wir (nur) die Felder aufzählen, die wir in die neue Datenbank übernehmen wollen:

```
'use namen'
'copy to temp structure fields name, bundesland'
'use temp'
'display structure'
```

Abb. 3.11

```
. use namen
. copy to temp structure fields name, bundesland
. use temp
. display structure
```

STRUCTURE FOR FILE: TEMP.DBF

NUMBER OF RECORDS: 00000

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	BUNDESland	C	003	
** TOTAL **			00024	

Für Fortgeschrittene Programmierer: COPY kann auch verwendet werden, um Ihrem Programm den Zugriff auf eine Datenbank-Struktur zu ermöglichen. Geben Sie ein:

```
'use namen'
'copy to neu structure extended'
'use neu'
'display structure'
'list'
```

Abb. 3.12

```
. use namen
. copy to neu structure extended
00006 RECORDS COPIED
. use neu
. display structure
STRUCTURE FOR FILE: NEU.DBF
NUMBER OF RECORDS: 00006
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	FIELD:NAME	C	010	
002	FIELD:TYPE	C	001	
003	FIELD:LEN	N	003	
004	FIELD:DEC	N	003	
** TOTAL **			00018	
. list				
00001	NAME	C	20	0
00002	ADRESSE	C	20	0
00003	POSTLTZahl	C	4	0
00004	STADT	C	20	0
00005	TELEFON	C	10	0
00006	BUNDESLAND	C	3	0

Die Datensätze der Datenbank "neu.dbf" beschreiben die Datenbank "namen", indem sie selbst aus den vier Feldern "NAME", "TYPE", "WIDTH" (LÄNGE) und "DEC" (Nachkommastellen) besteht. Das sind gerade die Begriffe, für welche Sie beim Definieren von Datenfeldern Werte eingeben. Die sechs Datensätze von "neu.dbf" enthalten also die Beschreibung der sechs Datenfelder von "namen.dbf"! Dadurch kann ein Anwendungs-Programm auf die Beschreibung dieser Feldstruktur zugreifen.

Andererseits kann ein File mit der gleichen Struktur wie "neu.dbf" in ein Programm eingefügt werden, so daß eine Bedienungskraft die Struktur einer neu anzulegenden Datei eingeben kann, ohne daß sie die Bedienung von dBASE I lernen muß. Das Programm würde dann an ihrer Stelle die Datenbank erzeugen, und zwar unter dem folgenden Befehl:

```
"CREATE <Datenfile> FROM <Datenstrukturfile>"
```

Der Benutzer braucht nur zuvor den Namen der Datei, die erzeugt werden soll, und den Namen des Files, der die Struktur der Datei beschreibt, eingeben. Wie man das erreichen kann, besprechen wir später.

Hinzufügen und Löschen von Feldern mit Daten in der Datenbank

Wenn Sie dBASE I in der Praxis einsetzen, werden Sie wahrscheinlich gelegentlich den Wunsch haben, eine Datenbank um neue Datenfelder zu erweitern oder überflüssige aus ihr zu entfernen.

Die alleinige Anwendung des Befehls MODIFY STRUCTURE würde alle Daten in Ihrer Datenbank löschen, in Verbindung mit den Anweisungen COPY und APPEND aber können Sie nach Belieben Felder hinzufügen oder entfernen.

Die Vorgehensweise besteht darin, zunächst die Struktur der Datenbank, die Sie ändern wollen, auf einen temporären (vorübergehend eingerichteten) File zu kopieren. Auf diesem File führen Sie dann die gewünschten Strukturänderungen aus. Anschließend füllen Sie die neu strukturierte Datenbank mit den Informationen aus der alten.

Wir werden als Beispiel unsere Datei "namen" benutzen und ihr ein Feld für die Kundennummer hinzufügen, welches dem Feld "KUNDNUMR" in der Datei "auftraeg" entspricht. Um das zu erreichen, ohne die Daten zu zerstören, die wir bereits gespeichert haben, geben Sie folgendes ein:

'use namen'	(benutze "namen")
'copy to temp structure'	(kopiere die Struktur nach "temp")
'use temp'	(benutze "temp")
'modify structure'	(verändere die Struktur)
'y'	(Ja (auf die Rückfrage))

Bringen Sie den Cursor nach unten bis zum ersten leeren Feld (mit <CTL-X>, geben Sie dann in der entsprechenden Spalte die Erweiterungen ein (der Feldname ist "KUNDNUMR", der Datentyp "C", die Länge 9). Mit <CTL-W> (bzw. <CTL-O> auf dem Superbrain) speichern Sie Ihre Änderungen ab und kehren zur Hauptbefehlsebene in dBASE II zurück.

Befehlen Sie 'display structure', um sich zu vergewissern, daß alles in Ordnung ist. Wenn das der Fall ist, können wir die Daten aus "namen" einspeichern:

'append from namen'	(Ergänze aus "namen")
---------------------	-----------------------

Wir hätten auch die Feldgrößen ändern können: Der Befehl APPEND überträgt Daten zwischen Feldern mit entsprechenden Namen.

Abb. 3.13

```
. use namen
. copy to temp structure
. use temp
. modify structure
MODIFY ERASES ALL DATA RECORDS ... PROCEED? (Y/N) Y
. display structure
STRUCTURE FOR FILE: TEMP.DBF
NUMBER OF RECORDS: 00000
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ADRESSE	C	020	
003	POSTLTZAHL	C	004	
004	STADT	C	020	
005	TELEFON	C	010	
006	BUNDESLAND	C	003	
007	KUNDNUMR	C	009	
** TOTAL **			00087	

```
. append from namen
00010 RECORDS ADDED
. list
```

00001	Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56	BAY
00002	Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12	BLN
00003	Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13	BAY
00004	Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45	B-W
00005	Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23	NRW
00006	Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8	BAY
00007	Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28	B-W
00008	Haberer, Karl	Teufelsweg 13	8047 Schrobhausen	12 45 7	BAY
00009	Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123	NRW
00010	Jahn, Alois	Huehnersteig 3	6142 Oberhof	456	BAY

Unsere neue Datei "temp" sollte jetzt das gewünschte neue Feld besitzen sowie die alten Daten aus dem File "namen". Geben Sie 'display structure' und 'list' ein, um sich zu vergewissern, daß nicht durch eine Netzstörung oder eine fehlerhafte Stelle auf der Diskette ein Übertragungsfehler aufgetreten ist.

Wenn die Daten einwandfrei übertragen wurden, können wir die Sache folgendermaßen zum Abschluß bringen:

```
'copy to namen'
'use namen'
```

Der Befehl 'COPY' überschreibt Struktur und Daten der alten Datei. Nachdem Sie den neuen File "namen" überprüft haben, können Sie mit 'delete file temp' die nun überflüssige Zwischendatei löschen.

Abb. 3.14

```
. copy to namen
00010 RECORDS COPIED
. use namen
. display structure
STRUCTURE FOR FILE: NAMEN.DBF
NUMBER OF RECORDS: 00010
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ADRESSE	C	020	
003	POSTLTZahl	C	004	
004	STADT	C	020	
005	TELEFON	C	010	
006	BUNDESLAND	C	003	
007	KUNDNUMR	C	009	
** TOTAL **			00087	

```
. list
00001 Adams, Peter      Bergstrasse 15      8000 Muenchen 2      12 34 56 BAY
00002 Camphausen, Eva  Muellerstr. 7       1000 Berlin 2        63 78 12 BLN
00003 Dirschedel, Fritz Magnolienweg 20     8000 Muenchen 90     97 81 13 BAY
00004 Dollinger, Erwin Hanfstr. 43         7432 Unterdorf       12 45   B-W
00005 Eberhart, Tobias  Marktstr. 19        5013 Ebenhausen     34 56 23 NRW
00006 Faltmann, Gisela Daeumlingsweg 8     8049 Dirnbach        13 29 8  BAY
00007 Gruber, Otto     Mozartstr. 29       7080 Pfaffenhausen   46 28   B-W
00008 Haberer, Karl    Teufelsweg 13       8047 Schrobenhausen  12 45 7  BAY
00009 Imhof, Bernhard  Gaertnerstr. 9      4703 Marktdorf       123     NRW
00010 Jahn, Alois      Huehnersteig 3      6142 Oberhof         456     BAY
```

Zusammengefaßt können Sie mit folgendem Verfahren Datenfelder in einer Datenbank löschen oder zu ihr hinzufügen:

```
'use <alterfile>'
'copy to <neuerfile> structure'
'use <neuerfile>'
'modify structure'
'append from <alterfile>'
'copy to <alterfile>'
```

```
verwende <alterfile>
kopiere Struktur auf <neuerfile>
verwende <neuerfile>
verändere Struktur
füge Daten aus <alterfile> hinzu
kopiere nach <alterfile> zurück
```

Austausch von dBASE-Dateien und CP/M-Dateien mit COPY und APPEND

Mit dBASE II gespeicherte Informationen können in eine Form umgewandelt werden, die es ermöglicht, sie Texteditoren oder verschiedenen Programmiersprachen zugänglich zu machen (BASIC, PASCAL, FORTRAN, PL/1 usw.). dBASE II kann auch Dateien lesen, die von diesen Editoren oder Sprachen erzeugt worden sind.

Unter CP/M beinhaltet das Standard-Daten-Format (abgekürzt als SDF in dBASE II) nach jeder Textzeile die Zeichen carriage return (Wagenrücklauf) und line feed (Zeilenvorschub). Um einen übertragbaren Datenfile, beispielsweise für die Bearbeitung mit einem Texteditor, aus einer Ihrer Datenbanken zu erzeugen, verwenden Sie eine weitere Form des "COPY"-Befehls.

Geben Sie ein:

```
'use namen'
'copy to sysdata sdf'
```

Diese Anweisung erzeugt einen File unter dem Namen "sysdata.txt". Verlassen Sie nun durch 'quit' das dBASE II System und benutzen Sie Ihren Texteditor (wenn Sie gerade keinen zur Hand haben sollten, verwenden Sie einfach das type-Kommando in CP/M) um sich den File anzuschauen. Verlassen Sie dazu dBASE mit "QUIT" und geben ein: TYPE SYSDATA.TXT. Sie werden feststellen, daß sie gradeso damit arbeiten können, als hätten Sie die Datei unter CP/M erzeugt.

Das Standard-Daten-Format erlaubt dBASE II auch, mit Daten aus CP/M-Files zu arbeiten. Die Daten müssen aber in die Struktur der Datenbank hineinpassen, unter der sie benutzt werden sollen!

Wenn Sie mit Hilfe eines Texteditors einen File namens "neudat.txt" erzeugt haben, können Sie ihn mit der untenstehenden Befehlsfolge zur Datenbank "namen.dbf" hinzufügen. Beachte: Die Daten müssen mit so vielen Leerzeichen aufgefüllt werden, wie es der Datenbankstruktur entspricht. Wenn der File "neudat.txt" folgende Informationen enthält:

Beispiel 3.9:

Freitag, Johanna	Münchbergstr. 854	8500Nürnberg	19 87 36 BAY
Goldig, Nicole	Radweg 73	8400Regensburg	29 42 31 BAY
Peters, Alice	Wagnerstr. 676	2190Cuxhafen	34 25 32 NDS
Grün, Frank	Spitzwegstr. 41	7400Tübingen	85 63 24 B-W
(20)	(20)	-(4) (20)	*** (10) *** (3)

können Sie ihn zur Datei "namen" hinzufügen, indem Sie folgendes eingeben:

```
'use namen'
'append from neudat.txt sdf'
```

Das Anhängen von Daten an eine schon bestehende Datei aus einem Betriebssystem-File dauert nur ein paar Sekunden.

Es wird Ihnen an Beispiel 3.9 auffallen, daß die Postleitzahlen in sehr unschöner Weise (ohne Zwischenraum) an die Stadtnamen angrenzen. Dies könnte man verschönern, indem man die Bezeichnung des Bundeslandes dazwischenstellt (dann müßte natürlich die Datensatz-Struktur entsprechend aufgebaut sein), aber dann wäre Ihnen dieser Umstand vielleicht gar nicht so aufgefallen. Vollständig ausgefüllte Datenfelder grenzen eben so aneinander.

Mit 'LIST' oder 'DISPLAY' fügt dBASE II beim Darstellen der Datensätze hier automatisch ein Leerzeichen ein, wir müssen aber die Felder in dem Textfile durch die genaue Anzahl Druckstellen identifizierbar machen!

Ähnlich gestaltet sich der Arbeitsvorgang, wenn Ihre "Fremddateien" unterschiedliche Begrenzungszeichen verwenden. Ein gebräuchliches Dateiformat benutzt Kommata zwischen den Datenfeldern und einfache Anführungszeichen, um Strings zu kennzeichnen. Diese Begrenzungszeichen werden im Englischen delimiter genannt. Um ein solches Format zu erzeugen oder zu benutzen, verwenden Sie das Wort DELIMITED statt SDF. Um zu sehen, wie das funktioniert, geben Sie ein:

'copy to temp delimited'

kehren dann ins Betriebssystem ('quit') zurück und sehen sich Ihre Daten an (mit einem Texteditor oder mit TYPE temp.txt in CP/M)

Wenn Ihr System andere Begrenzungszeichen benutzt, können Sie diese in dem Befehl "DELIMITED [WITH <begrenzungszeichen>]"

angeben (die spitzen Klammern "<" und ">" sollen Sie NICHT eingeben). Wenn Ihr System nur Kommata und keine Zeichen zur Eingrenzung von Strings verwendet, dann geben Sie an: 'delimited with ,'.

Die vollständige Syntax für COPY und APPEND beim Umgang mit Betriebssystem-Files lautet:

COPY [bereich] TO <filename> [FIELD liste]
[SDF]
[STRUCTURE] [FOR <ausdruck>]
[DELIMITED [WITH <begrenzungszeichen>]]

APPEND FROM <filename> [SDF] [FOR <ausdruck>]
[DELIMITED [WITH <begrenzungszeichen>]]

Die "[...]"-Klammern bedeuten, daß das, was darin steht, wahlweise benutzt oder fortgelassen werden kann.

Beide Befehle können durch einen Bedingungs-Ausdruck selektiv gemacht werden, und der Bereich (scope) von COPY kann wie bei anderen dBASE II-Befehlen angegeben werden (ALL, RECORD n oder NEXT n).

BEACHTEN: Obwohl dBASE II automatisch Namenerweiterungen (extensions) für die Dateien, die es selbst erzeugt, einsetzt, müssen sie die Filenamen-Ergänzung "T" angeben, wenn Sie von einem Betriebssystem-File mit APPEND Daten zu einer Datenbank hinzufügen.

BEACHTEN: Beim Befehl APPEND müssen alle Datenfelder, die in einem <ausdruck> angegeben werden, in der Datenbank, zu welcher die Daten hinzugefügt werden, auch tatsächlich existieren.

Umbenennen von Datensatz-Feldern mit COPY und APPEND

Wie wir schon gesagt haben, überträgt APPEND Daten aus einer Datei in eine andere, sofern die Datenfelder in den Datensätzen übereinstimmen. Wenn ein Feldname aus dem FROM-File (woher die Daten kommen) in dem File, der mit USE in Bearbeitung genommen wurde, nicht vorkommt, werden Daten aus diesem Feld nicht übernommen.

Mit der im vorigen Abschnitt besprochenen erweiterten Form des Befehls können Sie ja auch die nackten Daten (ohne Bezug auf Feldnamen) übertragen, und diese Möglichkeit können wir dazu benutzen, Feldnamen in einer Datenbank zu verändern. Nehmen wir an, Sie wollen das Feld "KUNDNUMR" in "namen.DBF" umbenennen zu "KUNDCODE", so geben Sie ein:

'use namen'	
'copy to temp sdf'	(nackte Daten nach TEMP.TXT)
'modify structure'	(verändere Datensatz-Struktur)
'append from temp.txt sdf'	(Rückholen der Daten nach Änd.)

Wenn Sie sich jetzt mit 'display structure' die Datenstruktur anschauen, hat das letzte Feld den Namen "KUNDCODE" angenommen. Vergessen Sie nicht, auch den Namen des "KUNDNUMR"-Feldes in unserer Datei "auftrag" zu ändern, so daß die Felder einander entsprechen.

Abb. 3.15

```
. use namen
. copy to temp sdf
00014 RECORDS COPIED
. modify structure
MODIFY ERASES ALL DATA RECORDS ... PROCEED? (Y/N) Y
. append from temp.txt sdf
00014 RECORDS ADDED
. display structure
STRUCTURE FOR FILE: NAMEN.DBF
NUMBER OF RECORDS: 00014
DATE OF LAST UPDATE: 11/30/82
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ADRESSE	C	020	
003	POSTLTZAHL	C	004	
004	STADT	C	020	
005	TELEFON	C	010	
006	BUNDESLAND	C	003	
007	KUNDCODE	C	009	
** TOTAL **			00087	

Daten in einem <.TXT>-File, der mit der Option (wahlweisen Angabe) SDF (oder DELIMITED) erzeugt worden ist, werden in Spalten gespeichert, die so angeordnet und mit Leerzeichen aufgefüllt (und ev. mit Trennzeichen markiert) sind, wie es dem Datenbank-File entspricht. Sie können einen <.TXT>-File mit Ihrem Texteditor bearbeiten, aber das birgt einige Gefahren:

Warnung: Ändern Sie die Position oder Größe von Datenfeldern nicht: Die Daten, die Sie abgespeichert haben, werden durch ihre Position, nicht durch Feldnamen identifiziert! Wenn Sie die Größe eines Feldes verändern, während Sie in der Datenbank die Strukturbeschreibung ändern, dann wird Ihre Datenbank zerstört, wenn Sie die ausgelagerten Daten zurückholen wollen.

Wenn Sie Daten mit COPY in eine <.TXT>-Datei kopieren, können Sie die vollständige erweiterte Form des Befehls benutzen, um den Bereich (scope), Felder (fields) und Bedingungen (conditions) anzugeben (siehe die früheren Erklärungen hierzu).

Verändern von Daten mit REPLACE, CHANGE

Sie können einige oder alle Datensätze schnell verändern, indem Sie folgenden Befehl benutzen:

```
"REPLACE [bereich] <feld> WITH <daten> [, <feld> WITH <daten> ...]
  [FOR <ausdruck>.]"
```

(replace = ersetze, with = durch).

Dieser Befehl ist sehr leistungsfähig, denn er ersetzt den Inhalt in einem <Feld-das-Sie-benennen> durch <was-auch-immer-Sie-hier-eintragen>. Sie können mehr als nur ein Feld verändern, indem Sie nach der ersten Kombination ein Komma setzen und weitere Felder und Daten auflisten, wie dies in den mittleren Klammern angedeutet wurde.

Die "Daten" können bestimmte neue Informationen sein (einschließlich Leerzeichen), oder eine Operation, zum Beispiel das Hinzuzählen der Mehrwertsteuer zu all Ihren Rechnungsbeträgen: "REPLACE ALL Betrag WITH Betrag * 1.13".

Sie können auch Bedingungen für diesen Austausch angeben, indem Sie die Ergänzung FOR und die entsprechenden Eigenschaften aller zu verändernden Datensätze im <ausdruck> angeben.

Um Ihnen zu zeigen, wie das funktioniert, müssen wir zu den Datenbanken "NAMEN" und "AUF-TRAEG" einige Daten hinzufügen.

Nehmen Sie zuerst durch 'use namen' die Datei "namen" in Bearbeitung, geben Sie dann ein: 'edit 1'. Schreiben Sie nun '1001' in das Feld "KUNDCODE" (bringen Sie dazu den Cursor mit <CTL>-X nach unten in das entsprechende Feld). Mit <CTL>-C gehen Sie zum nächsten Datensatz über, sobald Sie die Kodenummer des Kunden eingetragen haben. Die Kundennummern sollten mit vier Stellen eingegeben werden, mit der Datensatz-Nummer als letzten beiden Ziffern (1001, 1002, 1003 usw.).

Tippen Sie dann 'USE auftraeg' ein und fügen Sie mit dem Befehl "APPEND" die folgenden Auftragsdaten ein (natürlich ohne die Spaltenüberschriften):

(Kunde)	(Artikel)	(Menge)	(Preis)
1012	38567	5	.83
1003	83899	34	.12
1009	12829	7	.17
1012	73833	23	1.47

'use auftraeg'
'replace all betrag with menge*preis'
'list'

Abb. 3.16

```
. use auftraeg
. replace all betrag with menge * preis
00004 REPLACEMENT(S)
. list
00001      1012      38567      5      0.83      4.15 .F.
00002      1003      83899     34      0.12      4.08 .F.
00003      1009      12829      7      0.17      1.19 .F.
00004      1012      73833     23      1.47     33.81 .F.
```

Sie werden den Befehl "REPLACE" auch in Programmen nützlich finden, um etwas in einen leeren Datensatz einzutragen, den sie Ihrer Datei hinzugefügt haben (appended). Sie werden häufig Daten aus Speicher-Variablen dazu benutzen, die zunächst leeren Datenfelder zu füllen.

Auch das Ändern von einigen Datenfeldern in einer großen Zahl von Datensätzen kann sehr rasch durchgeführt werden, indem Sie den CHANGE-Befehl benutzen:

"CHANGE [bereich] FIELD <liste> [FOR <ausdruck>]"

Der "Bereich" ist wieder genauso zu formulieren wie bei den anderen dBASE II - Befehlen (ALL, RECORD n, NEXT n). Mindestens ein Feld muß in der "liste" genannt werden, es können aber mehrere, durch Kommata getrennte, Felder aufgezählt werden. Wenn die Ergänzung mit FOR angefügt wurde, so sucht dieser Befehl den ersten Datensatz auf, der den Bedingungen im "Ausdruck" entspricht, und zeigt dann die Bezeichnung und den Inhalt des Datensatzes mit einer Frage: "CHANGE?" ("VERÄNDERE?"). Geben Sie nun den Teil des Feldinhaltes ein, den Sie verändern wollen, und dann die gewünschte neue Information. dBASE II zeigt Ihnen anschließend, wie der Feldinhalt jetzt aussieht und fragt wieder CHANGE? Wenn Sie also noch weitere Änderungen vornehmen wollen oder wenn das Resultat nicht Ihren Erwartungen entspricht, dann wiederholt sich das Spiel.

Wenn Sie den Inhalt so stehenlassen wollen, wie er angezeigt wird, drücken Sie einfach <RETURN> (oder <ENTER>). Dann erscheint das nächste Feld (wenn Sie mehrere Felder angegeben hatten) oder der nächste Datensatz (wenn Sie einen Bereich angegeben hatten, der mehrere Datensätze umfaßt).

Wenn das Feld leer ist und Sie wollen Daten einschreiben, drücken Sie zuerst die Leertaste, um dem System mitzuteilen, daß sie das leere Feld ändern wollen!

Sobald Sie alle aufgelisteten Felder in dem Datensatz geprüft u. ggf. geändert haben, wird Ihnen das erste Feld im nächsten Datensatz, der den angegebenen Bedingungen genügt, präsentiert. Um in die Hauptbefehlsebene von dBASE II zurückzukehren, drücken Sie bitte die Taste <ESC> ("ESCAPE").

Abb. 3.17

. use namen
. change all field kundcode

RECORD: 00001

KUNDCODE:
CHANGE? ' ' (Eingabe eines Leerzeichens, um leeres Feld zu ändern)
TO 1001

KUNDCODE: 1001
CHANGE? <RETURN> (RETURN, um zum nächsten Datensatz zu gelangen)

RECORD: 00002

KUNDCODE:
CHANGE? ' '
TO 1002

KUNDCODE: 1002
CHANGE? <RETURN>

Abb. 3.18

. use testdat
. change all field name, adresse, stadt

RECORD: 00001

NAME: Adams, Peter

CHANGE? Adams

TO Adam

NAME: Adam, Peter

CHANGE? <RETURN>

ADRESSE: Bergstrasse 15

CHANGE? 15

TO 30

ADRESSE: Bergstrasse 30

CHANGE? <RETURN>

STADT: Muenchen 2

CHANGE? Muenchen

TO München

STADT: München 2

CHANGE? <RETURN>

RECORD: 00002

NAME: Camphausen, Eva

CHANGE? Cam

TO Kam

NAME: Kamphausen, Eva

CHANGE? <RETURN>

ADRESSE: Muellerstr. 7

CHANGE? Mue

TO Mü

ADRESSE: Müllerstr. 7

CHANGE? <RETURN>

STADT: Berlin 2

CHANGE? 2

TO 3

STADT: Berlin 3

CHANGE? <RETURN>

RECORD: 00003

NAME: Dirschedel, Fritz

CHANGE? Fritz

TO Bernd

NAME: Dirschedel,

Bernd CHANGE? <RETURN>

ADRESSE: Magnolienweg 20

CHANGE? <ESCAPE>

Sicherlich wird Sie das an die Korrekturmöglichkeiten des fehlerkorrigierenden Dialoges erinnern. Sie sehen, es ist recht bequem, so die Einträge in Feldern zu ändern. Sie müssen nur genügend von dem zu ändernden Eintrag angeben, um die Stelle eindeutig zu kennzeichnen. Anschließend können Sie eine andere, auch längere oder kürzere, Zeichenfolge eingeben, welche die zuvor genannte ersetzen soll.

Die Organisation Ihrer Datenbank mit SORT und INDEX

Daten werden oft in zusammenhangloser Reihenfolge erfaßt, also z. B. nicht von vornherein in alphabetischer Ordnung, wie es in unserer Datei "namen" geschehen ist. Dadurch sind die Datensätze nicht notwendigerweise in der Reihenfolge, in der Sie sie gerne hätten - daher stellt Ihnen dBASE II Mittel zur Verfügung, um Ihre Daten durch Sortieren (SORTING) und Indizieren (INDEXING) zu organisieren.

INDIZIERTE Dateien erlauben den schnellen Zugriff auf Datensätze (typischerweise innerhalb von zwei Sekunden, auch mit Floppy-Disk-Laufwerken (statt schnelleren Plattenlaufwerken)). Dateien können in aufsteigender oder absteigender Reihenfolge sortiert werden. Das vollständige Befehlsschema lautet:

"SORT ON <feldname> TO <filename> [DESCENDING]"

Der "Feldname" bestimmt den Schlüssel ("key"), nach welchem die Datei sortiert wird, er kann numerisch oder textuell (Character-String) sein (nicht vom logischen Typ). Normalerweise erfolgt die Sortierung aufsteigend (ascending), aber Sie können dies umkehren, indem Sie den absteigenden Modus durch Angabe von DESCENDING (ABSTEIGEND) verwenden.

Wenn Sie nach mehreren Merkmalen (Schlüsseln) sortieren wollen, fangen Sie mit einer Sortierung nach dem niederwertigsten Schlüssel an und sortieren Sie dann in weiteren Durchgängen nach jeweils höherrangigen Schlüsseln. dBASE II bewegt nur so viele Datensätze wie nötig, so daß die niederwertigere Sortierung jeweils erhalten bleibt.

Um dies mit unserer "namen"-Datei zu zeigen, sortieren wir sie zunächst so, daß die Städte in alphabetischer Reihenfolge stehen. Geben Sie ein:

```
'use namen'  
'sort on stadt to temp'  
'use temp'  
'list'  
'copy to namen'
```

Abb. 3.19

. use namen
 . sort on stadt to temp
 SORT COMPLETE
 . use temp
 . list off

Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN 1002
Peters, Alice	Wagnerstr. 676	2190 Cuxhafen	34 25 32 NDS 1013
Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY 1006
Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW 1005
Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123 NRW 1009
Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY 1001
Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY 1003
Freitag, Johanna	Münchbergstr. 854	8500 Nürnberg	19 87 36 BAY 1011
Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY 1010
Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W 1007
Goldig, Nicole	Radweg 73	8400 Regensburg	29 42 31 BAY 1012
Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY 1008
Grün, Frank	Spitzwegstr. 41	7400 Tübingen	85 63 24 B-W 1014
Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W 1004

. use namen
 . sort on name to temp
 SORT COMPLETE
 . use temp
 . list off

Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY 1001
Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN 1002
Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY 1003
Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W 1004
Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW 1005
Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY 1006
Freitag, Johanna	Münchbergstr. 854	8500 Nürnberg	19 87 36 BAY 1011
Goldig, Nicole	Radweg 73	8400 Regensburg	29 42 31 BAY 1012
Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W 1007
Grün, Frank	Spitzwegstr. 41	7400 Tübingen	85 63 24 B-W 1014
Haberer, Karl	Teufelsweg 13	8047 Schrobenhausen	12 45 7 BAY 1008
Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123 NRW 1009
Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY 1010
Peters, Alice	Wagnerstr. 676	2190 Cuxhafen	34 25 32 NDS 1013

. copy to namen
 00014 RECORDS COPIED

WARNUNG: Sortieren Sie eine Datei nicht auf sich selbst, geben Sie nach "TO" eine temporäre (zwischenzeitliche) Datei an! Eine Netzstörung oder ein anderes unvorhergesehenes Ereignis könnte Ihre gesamte Datenbank zerstören, wenn es im ungünstigen Moment auftritt!

Sortieren Sie stattdessen auf einen vorübergehend benutzten File und kopieren dann - nach einer Überprüfung - die Daten in die originale Datenbank zurück.

Eine Datenbank kann auch INDIZIERT werden, so daß sie als sortiert erscheint. Die Form des INDEX-Befehls lautet:

"INDEX ON <key (variable/ausdruck)> TO <indexfile-name>"

Dadurch wird ein File mit dem neuen Namen und der Namens-Erweiterung <X> erzeugt. Nur die Daten im "Schlüssel" werden sortiert, obwohl es so aussieht, als wäre die ganze Datenbank sortiert worden. Der Schlüssel kann ein Variablen-Name oder ein komplizierter Ausdruck (bis zu 100 Zeichen lang) sein. Er darf aber nicht ein logisches Feld sein.

Vielleicht ist Ihnen noch nicht ganz klar, was der Unterschied zwischen Sortieren und Indizieren ist. Er besteht in mehreren Punkten:

Bei der Indizierung werden die Datensätze in der Datenbank auf ihren Plätzen belassen. Stattdessen wird eine Hilfsdatei (Indexfile) erzeugt, welche den angegebenen Schlüssel in sortierter Reihenfolge und zu jedem Schlüsselwert einen "Zeiger" auf die Position des Datensatzes in der Datenbank besitzt. Zusätzlich können Indexfiles sehr viel effizienter gehandhabt werden. Man findet daher, unabhängig von der Größe der Datenbank, sehr schnell den gewünschten Schlüssel und kann durch den Zeiger sofort den entsprechenden Datensatz in der Datenbank ansprechen (ohne die Datenbank erst durchsuchen zu müssen).

Das Schöne ist, Sie brauchen sich nicht um die interne Organisation dieser Mechanismen zu kümmern.

Um unsere Kundendatei nach Postleitzahlen zu ordnen, geben Sie ein:

'use namen'

'index on postltzahl to postleit'

'use namen index postleit'

'list'

Abb. 3.20

```
. use namen
. index on postltzahl to postleit
00014 RECORDS INDEXED
. use namen index postleit
. list
```

00002 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN 1002
00014 Peters, Alice	Wagnerstr. 676	2190 Cuxhafen	34 25 32 NDS 1013
00012 Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123 NRW 1009
00005 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW 1005
00013 Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY 1010
00009 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W 1007
00010 Grün, Frank	Spitzwegstr. 41	7400 Tübingen	85 63 24 B-W 1014
00004 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W 1004
00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY 1001
00003 Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY 1003
00011 Haberer, Karl	Teufelsweg 13	8047	
		Schrobenhausen	12 45 7 BAY 1008
00006 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY 1006
00008 Goldig, Nicole	Radweg 73	8400 Regensburg	29 42 31 BAY 1012
00007 Freitag, Johanna	Münchbergstr. 854	8500 Nürnberg	19 87 36 BAY 1011

Wir können unsere Datei auch nach zwei oder mehr Schlüsseln indizieren, indem wir eingeben:

'index on bundesland + kundcode to mehrfach'

Abb. 3.18b

```
. use namen
. index on bundesland + kundcode to mehrfach
00014 RECORDS INDEXED
. use namen index mehrfach
. list
```

00004 Dollinger, Erwin	Hanfstr. 43	7432 Unterdorf	12 45 B-W 1004
00009 Gruber, Otto	Mozartstr. 29	7080 Pfaffenhausen	46 28 B-W 1007
00010 Grün, Frank	Spitzwegstr. 41	7400 Tübingen	85 63 24 B-W 1014
00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY 1001
00003 Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY 1003
00006 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY 1006
00011 Haberer, Karl	Teufelsweg 13	8047	
		Schrobenhausen	12 45 7 BAY 1008
00013 Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY 1010
00007 Freitag, Johanna	Münchbergstr. 854	8500 Nürnberg	19 87 36 BAY 1011
00008 Goldig, Nicole	Radweg 73	8400 Regensburg	29 42 31 BAY 1012
00002 Camphausen, Eva	Muellerstr. 7	1000 Berlin 2	63 78 12 BLN 1002
00014 Peters, Alice	Wagnerstr. 676	2190 Cuxhafen	34 25 32 NDS 1013
00005 Eberhart, Tobias	Marktstr. 19	5013 Ebenhausen	34 56 23 NRW 1005
00012 Imhof, Bernhard	Gaertnerstr. 9	4703 Marktdorf	123 NRW 1009

Sie sehen, daß die Datensätze zunächst nach den Bundesländern geordnet aufgelistet wurden. Innerhalb eines Bundeslandes aber sind sie nach dem Kundencode sortiert.

Wenn hierbei numerische Felder verwendet werden, müssen sie in Zeichen (characters) umgewandelt werden. Wenn Kundcode ein numerisches Feld mit 5 Stellen vor und 2 nach dem Komma (Dezimalpunkt) wäre, würde die Funktion STR (die später beschrieben wird), wie folgt angewandt:

'index on name + STR (kundcode,5,2) + bundesland to mehrfach'

Um die wesentlich erhöhte Zugriffsgeschwindigkeit durch Index-Dateien auszunutzen, müssen Sie im USE-Befehl den gewünschten Indexfile angeben:

'USE <datenbank name> INDEX <index file name>'

Wenn Sie in dieser Weise einen Indexfile in Gebrauch genommen haben (to use = benutzen), dann springen die Positionier-Befehle (GO, GO BOTTOM usw.) zu den entsprechenden Positionen im Index-File, nicht im Datenbank-File selbst. Zum Beispiel bringt Sie 'go bottom' (Gehe zum Ende) auf den letzten Datensatz, der im Indexfile steht, nicht auf den am Ende des Datenbank-Files gespeicherten Datensatz.

Wenn Sie mit APPEND, EDIT, REPLACE oder PACK Veränderungen an den Schlüsselfeldern einer Datenbank vornehmen, so betreffen diese auch einen gleichzeitig benutzten Indexfile.

Andere Index-Dateien, die zu Ihrer Datenbank gehören, können Sie "updaten" (auf den neuen Stand bringen), indem Sie eingeben:

"SET INDEX TO <index-file 1>, <index-file 2>, ... <index-file n>".

Führen Sie dann Ihre APPEND, EDIT usw. -Operationen aus. Alle aufgezählten Indexdateien befinden sich dann auf dem Laufenden. Sie können bis zu sieben Indexdateien angeben.

Sie können auch bei USE eine oder mehrere (bis zu sieben) Indexdateien angeben. Jeweils die erste Indexdatei gilt als Hauptindex und wird bei den Befehlen FIND, SKIP, GOTO usw. als Grundlage benutzt. Die übrigen Indexdateien werden mit aktualisiert, wenn Sie die Datenbank mit APPEND, EDIT usw. verändern. Sie können aber auch andere Indexdateien leicht auf den neuesten Stand bringen, indem Sie einfach neu indizieren - das dauert nicht lange.

Ein wesentlicher Vorzug einer indizierten Datei ist, daß sie es Ihnen gestattet, den anschließend beschriebenen FIND-Befehl dazu zu verwenden, selbst in großen Datenbanken innerhalb von Sekunden bestimmte Datensätze aufzufinden.

Suchen von Daten mit FIND und LOCATE

Wenn Sie einen Datensatz suchen, von dem Sie den Wert des Schlüsselfeldes kennen, können Sie den Befehl FIND (FINDE) verwenden (aber nur dann, wenn Ihre Datenbank nach dem entsprechenden Schlüsselfeld indiziert wurde und der Indexfile in Gebrauch genommen, d.h. beim "USE"-Befehl mit angegeben wurde). Typischerweise dauert es mit einem Floppy-Disk System mit FIND zwei Sekunden, einen Datensatz zu finden, gleichgültig, wie groß die Datenbank ist. In Abb. 3.22 ist das Postleitzahlen-Feld der Schlüssel.

Geben Sie einfach FIND <zeichenkette> (ohne Anführungszeichen oder Klammern) ein, wobei die "zeichenkette" der gesamte oder ein Teil des Inhalts eines Datenfeldes ist.

Diese Zeichenkette (string) können Sie so kurz wählen, wie es Ihnen beliebt, aber sie sollte lang genug sein, um sie eindeutig zu machen. Zum Beispiel kommt "Ch" in einer großen Zahl von Worten vor, "Chem" ist sehr viel eindeutiger.

Geben Sie folgendes ein:

```
'use namen index postleit'
'find 10'
'display'
'find 8'
'display'
'display next 3'
```

Abb. 3.22

```
. use namen index postleit
. find 10
. display

00002 Camphausen, Eva      Muellerstr. 7      1000 Berlin 2      63 78 12 BLN 1002

. find 8
. display

00001 Adams, Peter        Bergstrasse 15      8000 Muenchen 2      12 34 56 BAY 1001

. display next 3

00001 Adams, Peter        Bergstrasse 15      8000 Muenchen 2      12 34 56 BAY 1001
00003 Dirschedel, Fritz   Magnolienweg 20     8000 Muenchen 90     97 81 13 BAY 1003
00011 Haberer, Karl       Teufelsweg 13       8047 Schrobenhausen 12 45 7 BAY 1008
```

Wenn der Schlüssel nicht eindeutig ist, dann findet dBASE II den ersten Datensatz, der Ihrer Angabe entspricht. Dies kann derjenige sein, nachdem Sie suchen, oder auch nicht. Sie können dann mit SKIP zum nächsten Datensatz springen oder gleich durch DISPLAY NEXT n eine Anzahl Datensätze anschauen. Das ist z.B. nützlich, wenn Sie nur den Anfangsbuchstaben eines Namens wissen oder wenn der Nachname "Huber" lautet, es aber mehrere Huber in Ihrer Datenbank gibt. Wenn kein Datensatz mit einem Schlüssel existiert, der mit dem identisch ist, nach dem Sie suchen, so meldet dBASE II NO FIND (NICHTS GEFUNDEN).

FIND kann auch mit Dateien benutzt werden, die mit mehrfachen Schlüsseln indiziert wurden. Der Nachteil eines mehrfachen Schlüssels (was bei Ihrer Anwendung vielleicht kein Nachteil ist) liegt darin, daß seine Bestandteile von links nach rechts paßgenau angegeben werden müssen. Das heißt, bezogen auf unser Beispiel in Abb. 3.21, daß Sie die Daten durch Angabe des Bundeslandes im FIND-Befehl, oder durch das Bundesland und den "Kundcode" finden können, nicht aber durch Angabe des Kundencodes allein. Dabei muß die Anzahl ev. vorhandener Leerstellen innerhalb des Schlüssels genau stimmen - in Abb. 3.23 gibt es nach dem Bundesland kein Leerzeichen, da das Feld genau 3 Zeichen lang ist.

Abb. 3.23

```
. use namen
. set index to mehrfach
. find BAY
. display next 5
```

00001 Adams, Peter	Bergstrasse 15	8000 Muenchen 2	12 34 56 BAY 1001
00003 Dirschedel, Fritz	Magnolienweg 20	8000 Muenchen 90	97 81 13 BAY 1003
00006 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY 1006
00011 Haberer, Karl	Teufelsweg 13	8047	
		Schrobenhausen	12 45 7 BAY 1008
00013 Jahn, Alois	Huehnersteig 3	6142 Oberhof	456 BAY 1010

```
. find 1006
NO FIND
. find BAY 1006
NO FIND
. find BAY1006
. display
```

00006 Faltmann, Gisela	Daeumlingsweg 8	8049 Dirnbach	13 29 8 BAY 1006
------------------------	-----------------	---------------	------------------

Um isolierte Informationen in Feldern zu finden, über die kein Indexfile erzeugt wurde (z. B. nur mit der Angabe "Kundencode" in unserem Beispiel), müssen Sie entweder das (nachstehend beschriebene) LOCATE-Kommando benutzen oder Sie müßten einen anderen Indexfile erzeugen, der z.B. den Kundencode als primären (ersten) Schlüssel benutzt.

Wenn Sie nach einer bestimmten Art von Daten suchen, die einer logischen Bedingung entsprechen, benutzen Sie

"LOCATE [bereich] [FOR <ausdruck>]"

(to locate = lokalisieren).

Diesen Befehl verwenden Sie auch, wenn Sie nach bestimmten Informationen in einer Datei suchen, die nicht nach dem Schlüssel indiziert ist, für den Sie sich interessieren (Beispiel: Die Datei ist zwar nach Postleitzahlen indiziert, Sie suchen aber nach Bundesländern usw.).

Wenn Sie die gesamte Datenbank durchsuchen wollen, dann brauchen Sie "bereich" nicht anzugeben, da LOCATE bei dem ersten Datensatz zu suchen anfängt. Um nur einen Teil der Datei zu durchsuchen, verwenden Sie 'LOCATE NEXT <anzahl>'. Die Suche startet dann bei dem laufenden Datensatz und erstreckt sich auf die "anzahl" nächsten Datensätze. Falls dies den Datensatz-Zeiger über das Ende der Datei hinauschieben würde, inspiziert LOCATE nur jeden Datensatz vom augenblicklichen bis zum Ende des Files.

Wenn Sie nach Daten in einem Zeichenfeld suchen, dann sollten Ihre Angaben in einfache Anführungszeichen eingeschlossen sein. Probieren Sie:

```
'use namen'
'locate for name = 'Gol' '
'display'
'locate for postltzahl > '6' .and. name < 'F' '
'display name, postltzahl'
```

Wenn ein Datensatz (Record) gefunden wurde, der den von Ihnen im "ausdruck" angegebenen Bedingungen entspricht, so zeigt ihn dBASE II durch RECORD n an. Sobald der Datensatz aufgefunden ("lokalisiert" = located) wurde, können Sie ihn mit DISPLAY darstellen oder mit "EDIT <recordnummer>" editieren. Wenn es mehr als einen Datensatz geben könnte, der Ihren Angaben entspricht, so geben Sie CONTINUE (to continue = fortfahren) ein, um die nächste Datensatznummer zu erhalten.

```
'continue'
(display)
'continue'
(display)
'continue'
usw.
```

Falls dBASE II Ihren Datensatz innerhalb des "bereichs", den Sie angegeben haben, nicht finden kann, so meldet es: END OF LOCATE oder END OF FILE ENCOUNTERED (ENDE DES FILES ERREICHT).

Abb. 3.24

```
. use namen
. locate for name = 'Gol'
RECORD: 00008
. display

00008 Goldig, Nicole      Radweg 73      8400 Regensburg      29 42 31 BAY 1012

. locate for postltzahl > '6' .and. name < 'F'
RECORD: 00001
. display name, postltzahl
00001 Adams, Peter      8000
. continue
RECORD: 00003
. display name, postltzahl
00003 Dirschedel, Fritz 8000
. continue
RECORD: 00004
. display name, postltzahl
00004 Dollinger, Erwin  7432
. continue
END OF FILE ENCOUNTERED
```

Die Berichtsauswertung der Datenbank mit REPORT

FIND und LOCATE sind gut zu gebrauchen, um einzelne Datensätze und Dateneinträge zu finden. Oft aber werden Sie Zusammenfassungen oder Auswertungen von Datensätzen wünschen, die bestimmten Bedingungen entsprechen. Hierfür brauchen Sie den REPORT-Befehl (report = Bericht).

Falls Sie in Ihrem Drucker einzelne Papierblätter (anstatt Endlosformularen) verwenden, dann befehlen Sie zuerst 'set eject off', um den sonst vorausgehenden Formularvorschub abzuschalten.

Wählen Sie nun die Datenbank, von der Sie einen Bericht anfertigen wollen und erzeugen das bei Ihnen übliche bzw. gewünschte Format für solche Übersichten, indem Sie eingeben:

```
'set eject off'
'use <datenbank>'
'report'
```

dBASE II führt Sie dann durch einen Dialog, mit dem Sie das gewünschte Berichts-Format einstellen. Sie geben an, welche Datenfelder der Datenbank Sie berücksichtigen wollen, Haupt- und Spaltenüberschriften, welche Spalten aufsummiert (totalled) werden sollen usw. Dazu kann es sehr hilfreich sein, sich zuvor die Struktur der Datenbank auszudrucken und das Blatt dann bei der Definition des Berichts-Formates neben den Rechner zu legen (Sie erreichen dies, indem Sie zunächst befehlen "SET PRINT ON" (dadurch werden alle Bildschirmausgaben auf dem Drucker mitprotokolliert) und dann "DISPLAY STRUCTURE". Danach schalten Sie den Drucker mit "SET PRINT OFF" wieder aus.)

Der übliche Standard für den Druckspiegel besteht aus einer 8 Spalten betragenden Einrückung vom linken Rand des Papiers, 56 Zeilen pro Seite (amerikanisches Papierformat), und einer Seitenbreite von 80 Zeichen.

Sie können das mit den Dateien ausprobieren, die Sie auf der Versuchs-Diskette erzeugt haben, aber die Dateien "namen" und "auftrag", die wir bisher als Beispiele benutzt haben, enthalten zu wenig Daten, um Ihnen die volle Leistungsfähigkeit von dBASE zu zeigen.

In den folgenden Beispielen werde ich die Datenbank "ausgaben.dbf" benutzen. Ich habe zuvor folgende Daten eingegeben:

Abb. 3.25

. use ausgaben					
. list					
00001	03/12/82	01234	1011	103 Mueller & Co	Rundkopfschrauben 192.80
00002	12/04/82	23456	1234	98 Hieberl und Söhne	Flachstangen 380.00
00003	29/06/82	34567	2134	718 Lutz & Farrenkopf	Nylon Farbband 58.90
00004	29/11/79	21323	2134	718 Meyers Papiermühle	Endlospapier weiß 50.00
00005	14/08/81	43564	2134	718 Schnelldruck GmbH	Matrixdrucker 2489.89
00006	21/09/82	46532	103	103 Mueller & Co	Flachkopfschrauben 17.80
00007	17/08/82	32598	1234	98 Hieberl und Söhne	Rundstangen 479.80
00008	29/11/82	32243	1011	103 Mueller & Co	Zylinderkopfschraub. 210.13
00009	13/04/79	33245	1234	98 Hieberl und Söhne	Winkelisen 210.78

In Abb. 3.26 sehen Sie, daß zuerst für eine Spalte des Bericht-Formulars die Anzahl Druckstellen und dann ein gültiger Feldname der Datenbank, für die der Report angefertigt werden soll, anzugeben ist. Nach <RETURN> werden Sie dann noch aufgefordert, eine Spaltenüberschrift einzugeben, die auf jeder Seite ausgedruckt wird.

Abb. 3.26

report	(Bericht erstellen)
ENTER REPORT FORM NAME: auftrko	(Name des Berichtes)
ENTER OPTIONS, M = LEFT MARGIN, L = LINES/PAGE, W = PAGE WIDTH L = 70	(M = linker Papierrand, L = Zeilen pro Seite, W = Breite der Seite)
PAGE HEADING? (Y/N) y	(Seitenüberschrift?)
ENTER PAGE HEADING: Kosten-Übersicht	
DOUBLE SPACE REPORT? (Y/N) n	(zweifacher Zeilenabstand?)
ARE TOTALS REQUIRED? (Y/N) y	(Zusammenfassungen erwünscht?)
SUBTOTALS IN REPORT? (Y/N) n	(Zwischenübersichten erwünscht?)
COL WIDTH, CONTENTS	(Spalte 1, Breite, Inhalt (Feldname))
001 10,scheck:dat	
ENTER HEADING: Datum	(Spaltenüberschrift)
002 22,name	
ENTER HEADING: Lieferant	
003 22,beschr	
ENTER HEADING: Beschreibung	
004 4,auftr:nr	
ENTER HEADING: ANr.	
ARE TOTALS REQUIRED? (Y/N) n	(Zusammenfassung erwünscht?) (Diese Frage wird bei allen numerischen Feldern einzeln gestellt)
005 12,betrag	
ENTER HEADING: Betrag	
ARE TOTALS REQUIRED? (Y/N) y	
006 <RETURN>	(Beenden der Report-Definition)

PAGE NO. 00001
12/04/82

		Kosten-Übersicht		
Datum	Lieferant	Beschreibung	ANr.	Betrag
03/12/82	Mueller & Co	Rundkopfschrauben	103	192.80
12/04/82	Hieberl und Söhne	Flachstangen	98	380.00
29/06/82	Lutz & Farrenkopf	Nylon Farbband	718	58.90
29/11/79	Meyers Papiermühle	Endlospapier weiß	718	50.00
14/08/81	Schnelldruck GmbH	Matrixdrucker	718	2489.89
21/09/82	Mueller & Co	Flachkopfschrauben	103	17.80
17/08/82	Hieberl und Söhne	Rundstangen	98	479.80
29/11/82	Mueller & Co	Zylinderkopfschraub.	103	210.13
13/04/79	Hieberl und Söhne	Winkelisen	98	210.78
** TOTAL **				

Sobald Sie alle Bestandteile des gewünschten Reports definiert haben, drücken Sie <RETURN> (oder <ENTER>), wenn die nächste Datenfeld-Nummer erscheint (wenn Sie nicht alle Felder der Datenbank in den Report einbeziehen wollen). dBASE II beginnt sofort mit der Anfertigung des Reports und durchkämmt dabei die gesamte Datenbank, wenn Sie es lassen. Um den Report zu stoppen, drücken Sie die Taste <ESCAPE> (<ESC>).

Gleichzeitig speichert dBASE II das Berichts-Format in einem File mit der Namen-Erweiterung ".FRM", so daß Sie es wieder verwenden können, ohne erneut durch den ganzen Dialog zu gehen. Die ausführliche Form des Befehls lautet:

```
"REPORT FORM <formatbezeichnung> [bereich] [FOR <ausdruck>]
[TO PRINT]"
```

Durch Eingabe von

```
'report form auftrko for auftrnumr = '718''
```

erhalten wir eine Aufstellung aller Kosten für den Auftrag Nummer 718, ohne daß wir das Format neu einzugeben bräuchten.

Abb. 3.27

```
. report form auftrko for auftr:nr = 718
```

```
PAGE NO. 00001
12/04/82
```

Kosten-Übersicht				
Datum	Lieferant	Beschreibung	ANr.	Betrag
29/06/82	Lutz & Farrenkopf	Nylon Farbband	718	58.90
29/11/79	Meyers Papiermühle	Endlospapier weiß	718	50.00
14/08/81	Schnelldruck GmbH	Matrixdrucker	718	2489.89
** TOTAL **				2598.79

Sie können die Information in der Überschrift austauschen, indem Sie eingeben: "SET HEADING TO zeichenkette" (= setze Überschrift zu - Sie können bis zu 60 Zeichen und Leerzeichen, keine Anführungszeichen, angeben). Der "bereich" des REPORT-Befehls erstreckt sich auf "all" (alle Datensätze), wenn nichts anderes angegeben wurde.

Der "ausdruck" hätte um weitere Bedingungen ergänzt werden können, und der ganze Bericht hätte als "Hardcopy" (Ausdruck auf Papier) ausgegeben werden können, wenn man die Angabe "TO PRINT" (ZUM DRUCKER) an das Ende des Befehls gesetzt hätte.

Diese Möglichkeiten der Berichts-Gestaltung können für jede Art von Auswertungen benutzt werden, angefangen mit der Berechnung von Zahlungen (FOR scheck:nr = ' ') über die anfallenden Kosten (FOR auftrnumr = '4 ') bis zu allem, was Sie wünschen.

Automatisches Zählen und Aufsummieren mit COUNT und SUM

Bei einigen Anwendungen werden Sie nicht die einzelnen Datensätze einsehen wollen, sondern Sie wollen nur wissen, wie viele Datensätze bestimmten Bedingungen entsprechen, oder was der Gesamtbetrag für eine bestimmte Rubrik ist (Wie viele Teile sind auf Lager? Wie viele sind bestellt. Welche Forderungen sind noch offen?) Dafür gibt es je einen Befehl zum Zählen von Datensätzen, die einer bestimmten Bedingung entsprechen, sowie zum Summieren der Werte von Feldern in bestimmten Datensätzen.

Der Befehl zum Zählen von Datensätzen lautet:

"COUNT [bereich] [FOR bedingungen] [TO speichervariable]"

Der Befehl kann ohne, mit einigen oder allen Erweiterungen benutzt werden.

Ohne irgendwelche Angaben zählt er einfach alle Datensätze in der Datenbank. Der <bereich> kann auf einen oder eine bestimmte Anzahl von Datensätzen beschränkt werden, und die <bedingungen> können ein beliebig komplizierter logischer Ausdruck sein (siehe den früheren Abschnitt über Ausdrücke). Das Ergebnis der Zählung kann in einer Speicher-Variablen festgehalten werden, die erzeugt wird, sobald der Befehl ausgeführt wird, falls sie nicht schon existiert.

Um Gesamtsummen (Totalübersichten) zu erhalten, geben Sie folgenden Befehl ein:

"SUM feld(er) [bereich] [FOR bedingung] [TO speichervariable(n)]"

Sie können bis zu 5 numerische Felder einer gerade bearbeiteten Datenbank angeben, von denen eine Gesamtsumme gebildet werden soll. Mehrere Feldnamen werden durch Kommata getrennt. Die zu summierenden Datensätze können durch die Angabe eines <bereichs> und/oder von <bedingungen> nach dem Wort "FOR" speziell ausgewählt werden (Kunde <> 'Sem' .AMD. betrag > 10...).

Wenn Sie mehrere Felder in Speicher-Variable summieren wollen, so geben Sie eine entsprechende Liste von Variablen an (natürlich mit verschiedenen Namen). Die Zuordnung zwischen den Feldern in der Feldliste zu den Variablen geschieht aufgrund der Position in der Aufzählung (das erste Feld in die zuerst genannte Variable, das zweite Feld in die zweitgenannte Variable usw.). Dabei dürfen zwischen den aufgezählten Feldern keine Lücken auftreten.

Wenn Sie also die letzten Felder in einem Datensatz nicht in Speichervariablen abspeichern wollen, aber über die ersten Felder eine Summen-Übersicht haben wollen, ist dies kein Problem: Zählen Sie einfach die ersten Variablen, die Sie bearbeiten wollen, auf. Wenn dabei ein Loch vorhanden ist (zum Beispiel, weil Sie das erste, dritte und vierte von sechs Feldern aufsummieren wollen), geben Sie bitte Speichervariable für die ersten vier Felder an. Nachdem die Summierung vorgenommen worden ist, löschen Sie einfach die zweite Variable mit RELEASE.

Abb. 3.28

```
. use ausgaben
. count for betrag < 100 to klein
COUNT = 00003
. sum betrag for auftr:nr = 718 to kosten
2598.79
. display memory
KLEIN (N) 3
KOSTEN (N) 2598.79
** TOTAL ** 02 VARIABLES USED 00012 BYTES USED
```

Summieren von Daten und Ausblenden unwichtiger Details (TOTAL)

TOTAL arbeitet ähnlich wie die Zwischen-Summen-Funktion im REPORT-Befehl, ausgenommen, daß die Ergebnisse, anstatt ausgedruckt zu werden, in einer Datenbank abgelegt werden:

TOTAL ON <schlüssel> TO <datenbank> [FIELDS liste]
[FOR bedingungen]

BEACHTEN: Die Datenbank, von welcher die Informationen kommen, muß vorsortiert oder nach dem Schlüssel indiziert sein, der in diesem Befehl benutzt wird.

Dieser Befehl ist besonders nützlich, wenn es darum geht, unwichtige Einzelheiten zu unterdrücken und Zusammenfassungen zu gewinnen. Abbildung 3.29 zeigt, was mit unserer Datenbank "ausgaben" passiert:

Beispiel 3.10:

```
'use ausgaben'
'index on auftrnumr to posten'
'total on auftrnumr to temp fields betrag for auftrnumr >699;
.and. auftrnumr <800'
'use temp'
'list'
```

Die neue Datei hat für jede Auftragsnummer einen Eintrag, und eine Zusammenfassung für alle Kosten, bezogen auf die Auftragsnummer in unserer Datenbank "ausgaben". Ein Problem mit der neuen Datenbank liegt darin, daß nur zwei der Felder brauchbare Information enthalten. Dies kann mit einer weiteren Befehlszeile überwunden werden.

"TOTAL" überträgt alle Datenfelder aus der Quelldatei, wenn die angegebene Zieldatei noch nicht existiert (sie wird dabei erzeugt), wobei der Befehl die Struktur der Quelldatei abbildet. In Beispiel 3.10 hätte man die Anzahl der Felder, die in die Zusammenfassungs-Datei übertragen werden, reduzieren können. Dies erreicht man dadurch, daß die Zieldatei zuvor erzeugt wird, und zwar nur mit den Datenfeldern, deren Zusammenfassung von Interesse ist.

ACHTUNG: Das kann auch zu mysteriösen Fehlern führen, wenn - ohne daß Sie daran denken - sich eine Datei mit dem Namen, den Sie als Ziel angeben, auf der Diskette befindet, diese Datei aber eine andere (unverträgliche) Struktur hat. Wenn Sie also unerklärliche Fehlermeldungen bei diesem Befehl erhalten, prüfen Sie, ob nicht schon eine Datei unter dem angegebenen Namen existiert und löschen Sie diese oder wählen Sie einen anderen Namen.

'COPY TO temp FIELD auftrnumr, betrag'

Wenn wir jetzt den Befehl TOTAL mit Angabe der Zieldatei "temp" benutzen, so wird die neue Datei lediglich die Auftragsnummern und Beträge enthalten. Versuchen Sie es mit Ihrer Datenbank! Die gleiche Technik können Sie anwenden, um die Anzahl von Teilen, ausstehende Zahlungen oder irgendeine andere geordnete (SORTierte oder INDEXierte) Information zu summieren.

Abb. 3.29

```
. use ausgaben
. index on auftr:nr to posten
00009 RECORDS INDEXED
. use ausgaben index posten
. total on auftr:nr to temp field betrag for auftr:nr > 90;
                                   .and. auftr:nr < 800

00003 RECORDS COPIED
. use temp
. list

00001 12/04/82 23456 1234 98 Hieberl und Söhne      Flachstangen      1070.58
00002 03/12/82 01234 1011 103 Mueller & Co          Rundkopfschrauben 420.73
00003 29/06/82 34567 2134 718 Lutz & Farrenkopf     Nylon Farbband    2598.79

. use
. delete file temp.dbf
FILE HAS BEEN DELETED
. use ausgaben index posten
. copy to temp field auftr:nr, betrag
00009 RECORDS COPIED
. total on auftr:nr to temp field betrag
00003 RECORDS COPIED
. use temp
. list
00001      98      070.58
00002     103      420.73
00003     718     2598.79
```

Zusammenfassung von Abschnitt 3:

Dieser Abschnitt hat Ihre Kenntnis davon, was man mit dBASE anfangen kann, erweitert. Wir haben Ihnen gezeigt, wie verschiedene Operatoren (arithmetische, vergleichenden (relationale) und String-Operatoren) dazu verwendet werden können, dBASE II-Befehle zu verfeinern und zu erweitern.

Weil Datenstrukturen die Grundlage jedes Datenbanksystems sind, haben wir eine Anzahl von Möglichkeiten behandelt, wie Datenstrukturen gebildet und verändert werden können. Dies sowohl beim Aufbau neuer Datenstrukturen (wobei die Datei außer der gespeicherten Beschreibung der Datenstruktur noch gar keine Dateneinträge enthält), als auch beim Überführen von Datenbanken, die bereits Informationen enthalten, in neue (erweiterte oder veränderte) Strukturen.

Wir haben auch gezeigt, wie Sie die speziellen Informationen eingeben, ändern und wiederfinden können, für die Sie sich gerade interessieren. Wir haben ferner Befehle kennengelernt, mit denen Sie isolierte Daten in zusammenfassende Information überführen können (COUNT, SUM, REPORT, TOTAL).

Im nächsten Abschnitt werden wir zeigen, wie man in dBASE II Programme schreibt (die nichts anderes als Dateien aus aneinandergereihten Befehlen sind), mit denen Sie Ihre Informationsverarbeitung automatisieren können.

Abschnitt 4: Die Programmierung der Datenbank

Inhaltsverzeichnis

Schreiben Sie Ihr erstes dBASE-Programm.....	127
MODIFY COMMAND (-file-)	
Auswahl zwischen Alternativen und Fällen von Entscheidungen	
mit IF..ELSE.....	130
Wahl zwischen verschiedenen Alternativen.....	130
Wahl zwischen zwei Alternativen.....	131
Mehrfachauswahlen.....	131
Mehrfachauswahlen mit CASE.....	132
Wiederholen eines Vorganges mit DO WHILE.....	133
Das Aufrufen von Unterprogrammen (Prozeduren).....	135
Die Eingabe von Daten in ein laufendes Programm	
mit WAIT, INPUT, ACCEPT.....	136
Hinweise, wann was zu benutzen ist.....	137
Wie Sie mit @...SAY...GET Daten und Mitteilungen auf	
dem Bildschirm an eine gewünschte Stelle bringen.....	138
Ein Programm, das zusammenfaßt, was wir bisher kennen.....	143
Arbeiten mit mehreren Datenbanken gleichzeitig.....	147
SELECT PRIMARY/SECONDARY	
Allgemein nützliche Systembefehle und Funktionen.....	152
Einige Worte über Programmierung und wie Sie Ihre Anwendung	
analysieren sollten.....	153

Wenn Sie verstanden haben, wie man Ausdrücke schreibt, dann fehlt Ihnen nicht mehr viel dazu, Programme zu schreiben.

Es gibt **vier grundlegende Programm-Strukturen**, mit denen man einen Computer dazu bringt, zu tun, was man von ihm will:

- * Abfolge (von Befehlen)
- * Auswahl und Entscheidung
- * Wiederholung (Schleifen)
- * Prozeduren (Unterprogramme)

Sie sind ja schon damit vertraut, daß dBASE II Ihre Befehle der Reihe nach - wie Sie sie eingegeben haben - ausführt. Wenn Sie diese Befehle, statt sie von Hand einzugeben, in einer Datei ablegen - haben Sie ein Programm.

In diesem Abschnitt werden Sie lernen, wie der Computer die Wahl zwischen zwei Alternativen trifft (IF ... ELSE), wie man ihn eine bestimmte Folge von Befehlen wiederholen läßt (DO WHILE) und wie man bestimmte Aktivitäten in Unterdateien zu Unterprogrammen zusammenfaßt.

Dann werden wir Ihnen zeigen, wie Sie diese einfachen Techniken dazu verwenden können, Befehls-Dateien (Programme) für Ihre Anwendungs-Probleme zu schreiben.

Schreiben Sie Ihr erstes dBASE-Programm

So nützlich und leistungsfähig die Befehle sind, die wir bisher kennengelernt haben, so haben sie Ihnen doch nur eine Vorahnung davon gegeben, was dBASE II leisten kann, wenn Befehle in Programmdateien aneinandergekettet werden.

Wenn Sie Befehle in einer Datei ablegen, schreiben Sie ein Computer-Programm, aber das ist viel einfacher, als es sich anhört, da dBASE II der menschlichen Sprache ähnliche Befehle benutzt (allerdings in englischen Begriffen). Sie haben nichts mit Bits und Bytes und diesen schrecklichen Innereien eines Computers zu tun, sondern nur mit Daten und Informationen, die anschaulich von jedermann zu verstehen sind.

Um eine Programm-Datei zu erhalten, schreiben Sie die Befehle, deren Ausführung Sie wünschen, einfach in einen CP/M-File mit der Namen-Erweiterung (extension) ".CMD". Sie können das mit Ihrem gewohnten Texteditor oder jedem Textverarbeitungs-System tun (aber auch ohne einen solchen, wie Sie bei der Behandlung des dBASE-Befehls MODIFY COMMAND etwas weiter unten sehen werden).

dBASE II fängt am Anfang der Liste an wie ein Mensch, der einen Einkaufszettel mit dem Einkaufswagen in einem Supermarkt "abarbeitet". Es führt einen Befehl nach dem anderen aus, bis es am Ende der Liste ankommt.

Andere Computersprachen arbeiten ganz genauso. In BASIC ist die Abfolge der Befehle besonders hervorgehoben durch die Zeilennummern. In anderen Sprachen (wie auch in dBASE II) beruht diese Abfolge lediglich aus der Aufzählung der Befehle in der ersten, zweiten, dritten usw. Zeile auf der Seite, ohne daß es Zeilennummern gibt. (Der Vorteil liegt darin, daß Sie bei nachträglichen Einfügungen, Umstellungen usw. nicht in Konflikte mit den ursprünglich erteilten Zeilennummern kommen.) Einige Programmiersprachen verwenden besondere Trennzeichen (wie den Doppelpunkt) zwischen den einzelnen Befehlen, dBASE II verwendet einfach das Ende der Zeile (nach <RETURN> bzw. <ENTER>, den sog. Wagen-Rücklauf-Code (der unsichtbar das Ende der Zeile markiert).

Die einzige Ausnahme, bei welcher der Computer von dieser einfachen Abfolge abweicht, liegt vor, wenn er auf eine Anweisung stößt, die ihn an eine andere Stelle des Programms schickt, um dort weiterzuarbeiten. In der Regel hängt dies von besonderen Bedingungen ab, so daß der Computer aufgrund von Ausdrücken oder logischen Bedingungen eine Entscheidung treffen muß, die Sie im Programm vorbereitet haben. Wir kommen darauf noch zurück.

Jetzt wollen wir einfach einmal ein einfaches Programm mit dem Namen "Test" schreiben. Sie können dies mit einem Texteditor tun, aber es geht auch ganz einfach mit dBASE II. Geben Sie ein:

```
'MODIFY COMMAND Test'
```

dBASE II bietet Ihnen daraufhin einen gelöschten Bildschirm an, auf den Sie nun mit der früher behandelten bildschirmorientierten Editierung schreiben können. Geben Sie so das kurze Programm ein, das Sie am Anfang der nächsten Seite finden.

ANMERKUNG: Wenn Ihr Terminal über die Hervorhebung durch negative Darstellung verfügt, wird nun der ganze Bildschirm weiß - mit Ausnahme der Zeile, in welcher sich der Cursor befindet. Falls Sie sich dadurch geblendet fühlen, schalten Sie vor dem Aufruf von MODIFY COMMAND die Hervorhebung einfach aus - mit der Anweisung SET INTENSITY OFF. Wenn Sie nachher, z.B. für die Hervorhebung von Masken, die Hervorhebung wieder einschalten wollen, tun Sie dies durch SET INTENSITY ON. Dauerhaft abschalten können Sie die Hervorhebung auch in der Anpassungs-Prozedur (Kapitel 1).

Das Ende jeder Zeile markiert das Ende eines Befehls (außer, Sie verwenden den Strichpunkt (;), um die Zeile zu erweitern), geben Sie also die Befehle genau in der gezeigten Anordnung ein.

Beispiel 4.1

```
USE namen
COPY STRUCTURE TO temp FIELD name, pltzahl
USE temp
APPEND FROM namen
COUNT FOR name = 'G' TO G
DISPLAY MEMORY
? 'Mein erstes Programm ist gerade erfolgreich gelaufen.'
```

Benutzen Sie die Cursor-Befehle (Zeile nach oben, unten, Zeichen nach rechts, links, löschen, einfügen usw.). Wenn Sie fertig sind, benutzen Sie <CTL>-W (<CTL>-O auf dem Superbrain), um den Text abzuspeichern und zum dBASE II Punkt-Prompt zurückzukehren. Verlassen Sie dann das dBASE-System ("quit"). Dann geben Sie ein:

'dbase test'

(die in CP/M übliche Form, ein Programm zu starten und ihm dabei zugleich eine Datei zu übergeben, den es bearbeiten soll. Diese Datei ist unser erstes dBASE II - Programm.)

Wenn Sie das Programm genau so eingegeben haben, wie es oben steht, wird es mit einer Fehlermeldung anhalten. Befehlen Sie nun wieder 'MODIFY COMMAND test' und korrigieren Sie den Feldnamen in der zweiten Zeile zu "postltzahl".

Wenn Sie einmal größere Programme schreiben, werden Sie feststellen, daß dieser eingebaute Editor einer der größten Vorzüge von dBASE II ist, da Sie damit Programme schreiben, korrigieren und verändern können, ohne ins Betriebssystem zurückkehren und einen Texteditor starten zu müssen. Gegenwärtig kann dieser eingebaute Editor maximal 4000 Bytes bearbeiten.

Das Programm selbst ist recht trivial, aber es zeigt, wie Sie eine **Folge von Befehlen** mit einer Anweisung aus einer Datei abrufen können. Das ähnelt der Art und Weise, wie Sie sonst ".COM"-Dateien in Ihrem Betriebssystem (CP/M oder ähnliche) aufrufen.

Wenn Sie dBASE bereits gestartet haben (Punkt-Prompt ist sichtbar), rufen Sie ein Programm durch folgende Befehlsform auf:

. 'DO <programmname>'

wobei der <programmname> die Namens-Ergänzung ".CMD" hat!

Vorschlag: Vielleicht wollen Sie den Hauptfile "dbase.com" umbenennen in "do.com", so daß Sie einfach eingeben:

'do <programmname>', ohne erst darüber nachzudenken, ob Sie das Programm von CP/M oder von dBASE II aus starten. Um dies in CP/M zu erreichen, schreiben Sie (nach dem Verlassen des dBASE II - Systems natürlich):

A> 'REN DO.COM=DBASE.COM'

Natürlich müssen Sie dann in jedem Fall das dBASE-System mit "do" starten (anstatt wie bisher mit "dbase").

Auswahl zwischen Alternativen und Fällen von Entscheidungen mit IF ... ELSE

Die Wahl zwischen verschiedenen Alternativen wird in dBASE durch die Schlüsselworte IF...ELSE...ENDIF getroffen. (if = wenn, else = andernfalls). Das geht nicht viel anders als in einfachem Deutsch:

```
WENN ich hungrig bin, dann esse ich,
ANDERNFALLS
    lasse ich es sein,
ENDE DER ALTERNATIVEN.
```

```
IF    ich hungrig bin,
      dann esse ich,
ELSE
      lasse ich es sein
ENDIF.
```

Auf dem Computer benutzen wir das gleiche Schema, aber wir müssen es exakt mit den Worten und in der Form tun, die das dBASE-System versteht:

Beispiel 4.2:

```
IF bedingung [ .AND. bedingung2 .OR. bedingung3 ... ]
    führe-diesen-befehl-aus
    [führe-den-nächsten-befehl-aus]
    [..]
ENDIF
```

Hier haben wir für den Fall, daß die Bedingung nicht zutrifft, noch keine alternative Aktion vorgesehen. Der ELSE-Zweig kann also auch entfallen!

Die "bedingung" kann aus einer ganzen Reihe von Ausdrücken (bis zu 245 Zeichen lang) bestehen, deren logische Auswertung als Resultat "wahr" oder "falsch" ergibt (vgl. die "logischen Operatoren" in Abschn. 3). Verwenden Sie die logischen Operatoren, um mehrere Ausdrücke zu verknüpfen. Wenn wir unsere Datei "ausgaben" verwenden, können wir die folgende Entscheidung formulieren:

```
IF Auftr:Nr = '730' .AND. Betrag > 99.99;
    .OR. Lieferant = 'Magische Schalter';
    .OR. Rech:Dat > '791231'
    führe-diesen-befehl-aus
    [befehl2]
    [..]
ENDIF
```

Wenn alle nötigen Bedingungen erfüllt sind, um den logischen Ausdruck "wahr" zu machen, dann führt der Computer die Befehle aus, die zwischen IF und ENDIF aufgezählt sind, anschließend fährt er mit der nächsten Anweisung, die auf ENDIF folgt, fort. Ist die Bedingung nach IF nicht erfüllt, springt er gleich zur nächsten Anweisung nach ENDIF!

Wahl zwischen zwei Alternativen: Wenn es zwei alternative Arbeitsgänge gibt, von denen einer abhängig von der (den) Bedingung(en) zu wählen ist, dann verwenden Sie die IF ... ELSE-Anweisung nach folgendem Schema:

Beispiel 4.3:

```
IF bedingung(en)
    führe-befehl(e)-1-aus
ELSE
    führe-befehl(e)-2-aus
ENDIF
```

Beachten Sie, daß das Schlüsselwort ELSE freistehen muß, d.h. alles, was nach ELSE (oder ENDIF) in der gleichen Zeile steht wird wie ein Kommentar behandelt. Die Anweisungen im ELSE-Zweig müssen, ebenso wie die im IF-Zweig, in einer neuen Zeile beginnen. Pascal-Programmierer können leicht darüber stolpern. Nur die BEDINGUNGEN gehören zu der Befehlszeile, die mit IF eingeleitet wird, stehen also in der gleichen Zeile (die notfalls durch ";" am Ende verlängert werden kann).

Der Computer führt entweder die Befehlsfolge 1 oder Befehlsfolge 2 aus, dann springt er zur Anweisung, die nach dem ENDIF folgt.

Mehrfachauswahlen: Gelegentlich müssen Sie eine Entscheidung aus einer längeren Liste von Alternativen treffen. Ein Beispiel dafür ist ein "Menue" auf dem Bildschirm, also ein Angebot von mehreren Befehlen, aus denen der Benutzer einen auswählen kann. In diesem Fall benutzen Sie ein geschachteltes IF...ELSE...IF-Schema.

Das ist eigentlich das gleiche IF..ELSE-Schema, das wir schon kennen, nur in mehreren Lagen wie chinesische Puppen ineinandergesteckt oder, wie man sagt, "verschachtelt". In Beispiel 4.4 sehen Sie das.

Beispiel 4.4:

```
IF <bedingung 1>
    <befehl 1>
ELSE
    IF <bedingung 2>
        <befehl 2>
    ELSE
        IF <bedingung 3>
            <befehl 3>
        ELSE
            .
        .
    ENDIF 3
ENDIF 2
ENDIF 1
```

Die Verschachtelung kann so weit wie nötig fortgesetzt werden, das heißt, bis für jede der gewünschten Alternativen eine IF-Entscheidung vorhanden ist.

Beachten Sie, daß jedes einleitende IF sein zugehöriges ENDIF haben muß - sonst "stürzt Ihr Programm ab".

Hinweis: dBASE II liest den Rest der (gleichen) Zeile nach einem ELSE oder ENDIF nicht mehr, daher können Sie dort Kommentare anbringen (wie wir es oben durch die Zahlen 1, 2, 3 getan haben). Das hilft sehr, die Übersicht über solche Strukturen zu behalten.

Mehrfachauswahlen mit CASE:

Die geschachtelte Abfrage durch IF ... ELSE ... IF ... ELSE aus Beispiel 4.4 wirkt weder sehr elegant noch übersichtlich, besonders bei umfangreichen Menüabfragen verliert man leicht den Überblick. Aus diesem Grunde wurde eine weitere Anweisung eingeführt, die, ins Deutsche übersetzt, folgendermaßen aussieht:

Beispiel 4.5:

```
TUE IM FALLE DASS
  FALLS heute Montag ist
    sieh auf der Speisekarte unter Montag nach
  FALLS heute Dienstag ist
    sieh auf der Speisekarte unter Dienstag nach
  FALLS heute Mittwoch ist
    sieh auf der Speisekarte unter Mittwoch nach
  FALLS heute Donnerstag ist
    sieh auf der Speisekarte unter Donnerstag nach
  FALLS heute Freitag ist
    sieh auf der Speisekarte unter Freitag nach
  ANDERNFALLS
    ist die Kantine geschlossen
ENDE DER FALLUNTERSCHIEDUNG
```

Sie verstehen nun, warum diese Anweisung im Deutschen als "Fallunterscheidung" bezeichnet wird. Im englischen Dialog wird sie mit dem Schlüsselwort CASE (case = Fall, Lage) bezeichnet.

Beispiel 4.6 zeigt das konkrete Schema, wie Sie es in dBASE programmieren müssen (vergleiche mit Beispiel 4.4):

Beispiel 4.6

```
DO CASE
    CASE (bedingung 1)
        (befehl 1)
    CASE (bedingung 2)
        (befehl 2)
    CASE (bedingung 3)
        (befehl 3)
    ...
    OTHERWISE
        (befehl, falls keine Bedingung zutrifft)
ENDCASE
```

Die Vorkehrung für den Fall, daß keine der genannten Bedingungen zutrifft, die mit dem Wort OTHERWISE getroffen wird, kann auch entfallen, dann wird nichts getan, wenn keine der aufgezählten Bedingungen zutrifft. Im Beispiel 4.5 könnte zum Beispiel unter "ANDERNFALLS" stehen: "Gehe ins Restaurant um die Ecke". In einer Firma, in der es keine Wochenendarbeiter gibt, kann diese Anweisung dagegen ganz entfallen.

Wiederholen eines Vorganges mit DO WHILE ...

Unermüdliche **Wiederholungen** einer Aktivität sind eines der kennzeichnendsten Leistungs-Merkmale eines Computers. Er vermag die gleiche Aufgabe wieder und wieder auszuführen, ohne sich zu langweilen oder durch Ermüdung Fehler zu machen, egal wie stupide (oder kompliziert) sie sein mag. In den meisten Programmiersprachen wird dies durch die DO WHILE-Konstruktion angewiesen:

```
DO WHILE bedingung(en)
    befehle
ENDDO
```

Solange die Bedingungen, die Sie angeben, logisch wahr bleiben, werden die aufgelisteten Befehle immer wieder ausgeführt.

Hinweis: Bedenken Sie, daß diese Befehle irgendwann die fraglichen Bedingungen verändern müssen, sonst läuft die Schleife bis in alle Ewigkeit!

Wenn Sie wissen, wie oft Sie den Auftrag wiederholen lassen wollen, dann können Sie dies nach folgendem Schema formulieren:

Beispiel 4.7:

STORE 1 TO Index	* Starte Zähler mit 1
DO WHILE Index < 11	* Bearbeite 10 Datensätze
IF Eintrag = ''	* Wenn keine Daten eingetragen
SKIP	* Überspringe den Datensatz und
LOOP	* gehe zurück zu DO WHILE
ENDIF nichts	* ohne Auftrag-A auszuführen
DO AuftragA	* Führe Auftrag-A.CMD-Datei aus
STORE Index + 1 TO Index	* Erhöhe Zähler um 1
ENDDO zehnmal	* Ende des DO-Blocks

Wenn sich in dem Datenfeld "Eintrag" irgendwelche Daten befinden, so wird der Computer irgendwelche Anweisungen ausführen, die sich in einem anderen Programmfile namens "AuftragA" befinden, danach kehrt er in den Programmfile Beispiel 4.3 an die Stelle unmittelbar nach diesem Aufruf zurück. Das heißt, er stößt nun auf die Anweisung, den Zähler "Index" um eins zu erhöhen (z.B. von 1 auf 2). Nun wird der Test am Anfang des DO-Blocks wiederholt. Wenn der Zählerinhalt noch kleiner als "11" ist (äußerstenfalls also "10"), dann wird die Schleife erneut durchlaufen. Nachdem der Zähler einen größeren Inhalt (also 11) erreicht hat, überspringt der Computer die Schleife und führt die Instruktion aus, die nach ENDDO im Programm steht (falls es eine solche gibt, sonst ist das Programm zu Ende).

Die Anweisung LOOP wird verwendet, um eine Befehlsfolge abubrechen. Sie veranlaßt den Computer, sofort zum Anfang des DO WHILE-Blocks zurückzuspringen, in dem sich diese Anweisung befindet.

In diesem Fall, wenn also das Datenfeld "Eintrag" leer ist, wird der betreffende Datensatz nicht bearbeitet, da die Ausführung zur Bedingungsabfrage DO WHILE Index < 11 zurückspringt. Der Datensatz mit dem leeren Eintrag wird nicht gezählt, da wir die Programmzeile, welche 1 zum Index hinzuzählt, ausgelassen haben.

Ein Problem mit der LOOP-Anweisung ist, daß sie den Programmfluß sozusagen kurzschließt (indem sie einen Rücksprung verursacht), was es sehr erschweren kann, den Überblick über die Ablauflogik des Programms zu behalten (insofern ähnelt LOOP der berühmt-berüchtigten GOTO-Anweisung, vielleicht haben Sie schon einmal vom Streit der Experten darüber gehört). Man muß also sehr genau aufpassen, wo die Programmausführung fortsetzt, wenn eine LOOP-Anweisung ausgeführt wird, und welche Anweisungen vielleicht außer Acht gelassen werden (in unserem Fall das Weiterschalten des "Index"-Zählers). Am besten verzichtet man ganz auf die LOOP-Anweisung (außer, wenn man sie halt unbedingt braucht).

Das Aufrufen von Unterprogrammen (Prozeduren)

Die Möglichkeit, in einer Sprache bestimmte Aufgaben oder Prozeduren standardisieren zu können, vereinfacht die Programmierung von Computern beträchtlich.

In BASIC nennt man diese Prozeduren "Unterprogramme", in moderneren Sprachen wie PL/I oder Pascal werden sie als "procedure" (Prozedur) bezeichnet. In dBASE II werden sie durch "command files" (genau wie die Programm-Dateien, die wir schon kennen) realisiert, die aus anderen Programmen (ebenfalls "command files") aufgerufen werden können. Durch den Dateinamen haben wir die Möglichkeit, dem Unterprogramm einen sinnvollen Namen zu geben, aus dem ersehen werden kann, was das Unterprogramm überhaupt macht.

In Beispiel 4.7 haben wir ein Unterprogramm aufgerufen, indem wir DO "AuftragA" geschrieben haben. "AuftragA" ist eine weitere Programm-Datei (deren Name die Ergänzung ".CMD" aufweist). Der Inhalt dieser Datei könnte so lauten:

Beispiel 4.8:

```
IF Zustand = B
  DO zahlBar
ELSE
  IF Zustand = S
    DO zahlScheck
  ELSE
    IF Zustand = U
      DO zahlUeberweisung
    ENDIF
  ENDIF
ENDIF
RETURN
```

Erneut haben wir weitere Unterprogramme aufgerufen, die ihrerseits wieder andere Programm-Dateien aufrufen könnten. Bis zu 16 "command files" (Programm-Dateien) können gleichzeitig "eröffnet" sein (d.h. in Gebrauch durch Aufrufe). Wenn eine Datei durch "USE" eröffnet wurde, so bleiben also noch 15 Dateien übrig, die gleichzeitig bearbeitet werden können. Einige Befehle benutzen weitere Dateien, (REPORT, INSERT, COPY, SAVE, RESTORE und PACK brauchen eine weitere Datei, SORT benutzt zwei zusätzliche Dateien). - Der Grund liegt darin, daß der Computer nur einen bestimmten Speicherplatz dafür reserviert, sich die Namen von Dateien zu merken, sowie ihre Position auf der Diskette und weitere Dinge, die zur Bearbeitung einer Datei nötig sind. Daraus geht übrigens hervor, daß die "Verschachtelungstiefe" von Unterprogrammen, die andere Unterprogramme aufrufen, die wiederum Unterprogramme aufrufen... nie größer als 16 (abzüglich gleichzeitig bearbeiteter Datenbanken) sein kann.

Dadurch entsteht aber nur selten ein Engpaß, denn Sie können eine beliebige Anzahl von Dateien bearbeiten, wenn Sie sie jeweils wieder schließen und nie mehr als 16 gleichzeitig eröffnen.

Eine Datei wird "geschlossen", wenn - im Falle einer Programmdatei - das Ende der Datei erreicht wird oder wenn eine RETURN-Anweisung ausgeführt wird. Die Anweisung RETURN gibt die Kontrolle an diejenige Programm-Datei zurück, von der aus die Programmdatei aufgerufen wurde, welche diese Anweisung enthält (oder auch an die Tastatur, falls der "command file" direkt von dort aufgerufen worden war).

Die Anweisung RETURN ist nicht in jedem Fall unbedingt erforderlich, da die Kontrolle in jedem Fall am Ende einer Programm-Datei zum aufrufenden Programm zurückgegeben wird, aber es ist eine gute Programmier-Praxis, sie am Ende jeder Programm-Datei anzufügen.

***Wichtiger Hinweis*:** Sie sehen, daß die Befehls-Zeilen in unseren Beispielen jeweils eingerückt sind. Das ist technisch für die Programmausführung keineswegs notwendig, aber es erhöht die Klarheit der Programme außerordentlich, besonders dann, wenn sie verschachtelte Strukturen enthalten. Der Gebrauch von Großschreibung für die Befehle und Groß- und Kleinschreibung für Variable ist ebenfalls eine Maßnahme, welche die Übersicht erleichtert.

Vermutlich warten Sie jetzt schon ungeduldig auf weitere konkrete Programmbeispiele, die Sie eingeben und ausprobieren können. Einige Seiten weiter finden Sie solche Beispiele. Um diese aber sinnvoller gestalten zu können, führen wir zunächst ein paar neue Befehle ein, die eine sehr flexible Ein- und Ausgabe von Daten ermöglichen.

Die Eingabe von Daten in ein laufendes Programm mit WAIT, INPUT, ACCEPT

In vielen Fällen benötigen Programme zusätzliche Daten, die vom Benutzer (der Bedienungskraft) eingegeben werden müssen und die (noch) nicht in der bearbeiteten Datenbank vorhanden sind.

Sie können Ihre Programme so einrichten, daß sie den Benutzer durch eine Nachricht anleiten, wann und welche Art von Information benötigt wird. Ein gutes Beispiel ist ein Menü, d.h. eine Liste von möglichen Arbeitsgängen, die auf dem Bildschirm ausgeschrieben werden und von denen einer auszuwählen ist. In einem anderen Fall mag es darum gehen, sicherzustellen, daß Rechnungs-Daten in der richtigen Weise eingegeben werden. Nachstehend die Befehle, mit denen dies erreicht werden kann.

WAIT [TO Speicher-variable]

stoppt die Ausführung eines Programms und wartet auf die **Eingabe eines einzelnen Zeichens** von der Tastatur, wobei die Nachricht WAITING (ICH WARTEN) ausgegeben wird. Die Ausführung des Programms wird fortgesetzt, sobald irgendeine Taste gedrückt wird (wie beim DISPLAY-Befehl von dBASE II).

Wenn außerdem ein Variablenname angegeben wird, dann wird das eingegebene Zeichen in dieser Variablen gespeichert. Wenn die Eingabe aus einem nicht-druckbaren Zeichen bestand (beispielsweise <RETURN> oder <ENTER>, ein <CONTROL>-Code usw.), so wird ein Leerzeichen in der Variablen gespeichert.

INPUT ['prompt'] TO speicher-variable

akzeptiert **jeden beliebigen Datentyp** vom Terminal (der Tastatur), der in einer geeigneten Speicher-Variablen abgelegt wird (die Variable wird erzeugt, falls sie noch nicht existiert).

Wenn die Möglichkeit, eine "Prompt"-Nachricht anzugeben (in einfachen oder doppelten Anführungszeichen, aber auf beiden Seiten jeweils die gleichen), ausgenutzt wurde, so erscheint auf dem Bildschirm diese Nachricht, gefolgt von einem Doppelpunkt, um anzuzeigen, wo die Daten einzugeben sind. Der Datentyp der Variablen (Zeichen, numerisch oder logisch) wird aus der Form der eingegebenen Daten bestimmt.

Zeichenketten müssen mit umschließenden Anführungszeichen oder eckigen Klammern eingegeben werden.

Die übrigen Datentypen sind numerisch (mit einer Ziffer beginnend) oder logisch (die Zeichen T (true = wahr), F (false = falsch), Y (Yes = wahr) und N (No = false)).

ACCEPT ['prompt'] TO speicher-variable

akzeptiert Zeichen-Daten (Texte, Strings), ohne die Notwendigkeit, sie in Anführungszeichen einzuschließen. Das ist sehr nützlich bei der Eingabe langer Zeichenketten (langer Texte) bzw. macht die Eingabe bequemer, wenn grundsätzlich ein Text einzugeben ist.

Wie bei INPUT kann vor der Dateneingabe eine Erläuterung für die Bedienkraft ausgegeben werden: 'ACCEPT 'Erläuterung' TO <variable>'.

Hinweise, wann was zu benutzen ist:**WAIT**

kann bei schnellen Eingaben (reagiert sofort auf das Antippen einer Taste) benutzt werden. Sollte nicht verwendet werden, wenn eine falsche Eingabe ernsthaften Schaden anrichten kann. Es ist nicht nötig, nach dem eingegebenen Zeichen <RETURN> oder <ENTER> zu drücken (knopfdruckartige Funktion).

ACCEPT

ist nützlich bei der Eingabe langer Zeichenfolgen, da es nicht das Einschließen in Anführungszeichen verlangt. Sollte auch für die Eingabe eines einzelnen Zeichens benutzt werden, wenn die Notwendigkeit, danach <RETURN> oder <ENTER> zu drücken die Zuverlässigkeit der Eingabe erhöhen kann. Im Gegensatz zu INPUT auf textuelle Daten spezialisiert.

INPUT

gestattet die Eingabe von numerischen und logischen Werten ebenso wie die von Zeichen(ketten), kann ansonsten wie ACCEPT benutzt werden.

Wie Sie mit @..SAY..GET Daten und Mitteilungen auf dem Bildschirm an eine gewünschte Stelle bringen

Achtung! Wichtiger Hinweis: Das Zeichen "@", das in dBASE II den Rang eines Befehls-Wortes besitzt, soll der sogenannte "Klammeraffe" oder "atsig" sein. Dieses Zeichen, das man auf jeder amerikanischen Computer-Tastatur (aber niemals der einer europäischen Schreibmaschine) findet, hat das Aussehen eines kleinen Schreibschrift-"a" mit einem Kringel drumrum.

Sollten Sie eine europäische Tastatur besitzen, auf der sich nur das Paragraph-Zeichen "§" befindet, so verwenden Sie bitte dieses anstelle des "Klammeraffen"!

Die Befehle "?", ACCEPT und INPUT können alle dazu benutzt werden, um Nachrichten (Prompts) an den Benutzer/Bediener auf dem Bildschirm auszugeben.

Sie haben aber den Nachteil, daß die Nachrichten immer gerade an der zufälligen Stelle nach der letzten Zeile, die auf den Bildschirm geschrieben wurde, erscheinen. Damit kann man natürlich arbeiten, aber es gibt eine vielseitigere Technik, mit der Sie Ausgaben und Eingaben exakt an jeder gewünschten Stelle auf dem Bildschirm abwickeln können.

Wenn Ihr Terminal/Bildschirm über die sogenannte X-Y-Adressierung des Cursors verfügt, dann können Sie Ihre Nachrichten mit einem besonderen dBASE II - Befehl jeweils auf eine beliebige Position auf dem Bildschirm schreiben:

```
"@ <koordinaten> [SAY <'nachricht'>]"
(to say = sagen)
```

Dies positioniert die "nachricht" (in Anführungszeichen oder eckigen Klammern angegeben) auf den Bildschirm-Koordinaten, die Sie angeben. Diese Koordinaten bestehen aus der Zeilen- und Spaltennummer, wobei die Position 0, 0 links oben (die "Cursor Home"-Position) ist. Wenn wir als Koordinaten "9, 34" angeben, dann fängt unsere Nachricht in der 10ten Zeile in der 35ten Spalte an.

Beachte: Wenn Sie bei der Anpassung von dBASE II (Kapitel 1) die Hervorhebung mit halber Helligkeit bzw. negativer (schwarz auf weiß) Darstellung installiert haben, so werden solche Nachrichten mit halber Helligkeit oder negativ ausgegeben. Falls Sie dies nicht wünschen, wiederholen Sie die Anpassungs-Prozedur und benutzen die Möglichkeit "Modify/Change", oder führen Sie den Befehl SET INTENSITY OFF aus. Im letzteren Fall können Sie beliebig zwischen Hervorhebung und nicht-Hervorhebung umschalten (SET INTENSITY ON), wie wir es in unserem späteren Programmbeispiel "zweidat.CMD" tun.

Das SAY... kann auch weggelassen werden, da dieser Befehl auch dazu verwendet werden kann, eine Zeile (oder einen Teil einer Zeile) auf dem Bildschirm zu löschen.

Sie können die folgenden Beispiele direkt eingeben. Dabei entsteht aber ein unschönes Durcheinander von Programm-Eingabe und den von den Programm-Beispielen erzeugten Ausgaben. Schöner wird es, wenn Sie so vorgehen:

```
'MODIFY COMMAND beispX/Y'
(Eingabe des Beispiels...)
<CTL>-W          (Abspeichern)
'DO beispX/Y'     (Ausführen des Programms)
usw...
```

Beginnen Sie mit der direkten Eingabe, und wechseln Sie dann probeweise zum Schreiben eines Programms in eine Datei über, das Sie anschließend aufrufen!

Starten Sie dBASE II und geben Sie ein:

Beispiel 4.9:

ACHTUNG: Nach dem "@" muß ein Leerzeichen folgen!!!

```
'ERASE'
'@ 20,30 SAY 'Was?' '
'@ 5,67 SAY 'Hier...'
'@ 11,11 SAY "Das ist alles." '
'@ 20, 0'
'@ 5, 0'
'@ 11,16'
```

(Bitte denken Sie an meinen Hinweis zum Zeichen "@" am Anfang dieses Unterabschnittes!)

Anstatt nur eine Nachricht (Textkonstante) auszugeben, kann der Befehl auch dazu benutzt werden, den Wert eines Ausdrucks aus einer oder mehreren Variablen auszugeben. Das Schema lautet:

```
"@ <koordinaten> [SAY <ausdruck>]"
```

Der Befehl kann ferner erweitert werden, um die Inhalte von verwendeten Variablen (Felder einer Datenbank oder Speichervariable) an jeder beliebigen Bildschirmposition auszugeben. Neben "SAY" (sage) gibt es die Anweisung "GET" (hole), mit der ein neuer Wert in eine Variable eingelesen wird. Geben Sie unter dBASE II folgendes ein:

Beispiel 4.10:

BEMERKUNG: Wenn Sie bei Ihren Versuchen "Salat" auf dem Bildschirm erzeugt haben, geben Sie ERASE ein!

```
'USE namen'
'@ 13,9 SAY Pleit:zahl'
'@ 13,6 SAY Bundesland'
'SKIP 3'
'@ 23, 5 SAY Name + ", " + Adresse'
```

Die Variablen, die von SAY oder GET benutzt werden, müssen bereits existieren, bevor sie durch diese Befehle angesprochen werden, sonst erhalten Sie eine Fehlermeldung.

Nach der Ausgabe durch SAY können Sie gleich einen neuen Wert vom Benutzer eingeben lassen, indem Sie die Erweiterung GET benutzen:

```
"@ <koordinaten> [SAY <ausdruck>] [GET <variable>]"
```

Um zu sehen, wie das funktioniert, geben Sie Beispiel 4.11 ein (verlassen Sie danach dBASE II noch nicht, es kommt noch etwas!):

Beispiel 4.11:

```
'ERASE'
'USE Namen'
'@ 15, 5 SAY "Postleitzahl" GET Postltzahl'
'@ 10,17 GET Stadt'
'@ 5, 0 SAY "Name" GET Name'
```

(bleiben Sie in dBASE II)

Diese Befehle haben die Werte von Variablen (mit und ohne Nachrichten) auf verschiedenen Stellen des Bildschirms positioniert. Damit können Sie individuelle Eingabeformulare gestalten, so daß der Bildschirm für die Bedienungskraft geradeso aussieht wie die alten Papier-Formulare, welche sie zuvor benutzt hat.

Um nun neue Daten in die dargestellten Variablen an den entsprechenden Stellen auf dem Bildschirm einzugeben, benutzen Sie den Befehl:

```
'READ'
```

Wenn Sie Beispiel 4.11 direkt eingeben, merken Sie, daß zuerst das gesamte Formular (die "Bildschirmmaske") aufgebaut wird. Erst mit der Ausführung der Anweisung READ beginnt die eigentliche Daten-Eingabe!

Der Cursor springt von selbst auf das erste Feld, das Sie angegeben haben. Sie können dort jetzt neue Daten eingeben oder den alten Inhalt stehenlassen, indem Sie einfach <RETURN> drücken. Wenn Sie das erste Feld verlassen, geht der Cursor zum nächsten Feld, das Sie angegeben haben.

Mit <CTL>-E ("Ein Feld nach oben") können Sie zum vorherigen Feld zurückkehren, wenn Sie nachträglich bemerken, daß Sie eine fehlerhafte Eingabe gemacht haben. Die Anordnung der Felder auf dem Bildschirm spielt dabei keine Rolle, maßgeblich ist, welches Feld zuerst mit GET genannt worden war!

Verändern Sie die Daten in den übrigen beiden Feldern. Wenn Sie mit dem letzten Feld fertig sind, landen Sie automatisch wieder im dBASE II-System. Geben Sie nun 'DISPLAY' ein. Der Datensatz der Datenbank "namen.DBF" enthält nun die neuen Daten, die Sie eben eingegeben haben.

Sie sehen, daß GET ähnlich arbeitet wie die Befehle INPUT und ACCEPT. GET ist aber viel mächtiger als die anderen Anweisungen, da es die formatierte Eingabe zahlreicher Variablen erlaubt, d.h. gleichzeitig eine automatische Cursorsteuerung ausführt.

Eine Datenbank mag pro Datensatz zwei oder ein Dutzend (bis zu 32) Felder haben, aber bei einem gegebenen Daten-Erfassungsvorgang wollen Sie vielleicht nur in die Hälfte davon Daten eingeben. Anstatt nun APPEND zu verwenden, was alle Felder in der Datenbank auslistet, können Sie mit APPEND BLANK zunächst einen leeren (blank = leer) Datensatz anfügen und dann mit GET die individuelle Eingabe der momentan zu erfassenden Daten gestalten.

Unsere Datei "Namen" ist kein gutes Beispiel (das Programm-Beispiel am Ende dieses Teils des Handbuches ist ein besseres), aber wir können sie dazu benutzen, um das Prinzip zu zeigen, nach dem man in einer Datenbank mit einer umfangreichen Struktur in bestimmte ausgewählte Felder neue Daten eingeben kann.

Damit Sie mehr Übung im Umgang mit Programmen erhalten, erzeugen sie einen "command file" unter dem Namen "versuch.CMD", der die folgenden Befehle enthält (verwenden Sie Ihren Texteditor oder MODIFY COMMAND. Bei MODIFY COMMAND geben Sie die Datei-Namen-Erweiterung ".CMD" nicht an, da dBASE dies selbst tut!!!):

Beispiel 4.12:

```

ERASE
? "Diese Prozedur erlaubt Ihnen, neue Datensätze"
? "zu der Datei NAMEN.DBF hinzuzufügen und dabei nur in"
? "ausgewählte Felder Daten einzutragen. Wir werden jetzt nur"
? "den Namen und die Postleitzahl eingeben."
?
? "Drücken Sie <S> um die Prozedur zu stoppen,"
? "<RETURN> um fortzufahren."
WAIT TO fortsetz

USE Namen
DO WHILE fortsetz <> 'S' .AND. fortsetz <> 's'
    APPEND BLANK          (Füge leeren Datensatz an)
    ERASE
    @ 10, 0 SAY "NAME" GET Name
    @ 10,30 SAY "POSTLEIT-ZAHL" GET PostltZahl
    READ

    ? " <S> bricht die Eingabe ab,"
    ? " <RETURN> setzt sie fort."
    WAIT TO fortsetz
ENDDO
RETURN

```

Wenn Sie ins CP/M (oder Ihr Betriebssystem) zurückgekehrt sind, geben Sie ein "dBASE versuch" (oder, wenn Sie die "dbase.com"-Datei wie vorgeschlagen umbenannt haben "do versuch"). Geben Sie dann Daten in einige Datensätze ein. Anschließend LISTen Sie die Datei, um zu sehen, was Sie angefügt haben.

Besonders, wenn Ihr Programm fehlerhaft ist, können Sie es auch jederzeit unterbrechen, indem Sie die Taste <ESCAPE> drücken (die auf Ihrer Tastatur auch mit <ESC> oder ähnlich bezeichnet sein kann).

Wie Sie sehen, ist die Eingabe von Daten mit "@..SAY..GET" einfach und übersichtlich zu gestalten. Sie können den Bildschirm durch die beliebige Anordnung von Nachrichten und Variablen-Eingabefeldern Ihrem persönlichen Geschmack und Ihrer besonderen Anwendung anpassen.

HINWEIS: Sie müssen nach jeweils 64 Eingaben mit GET entweder ERASE oder CLEAR GETS aufrufen (damit der intern reservierte Speicherplatz für diese Eingabefelder wieder frei wird). Wenn Sie damit nicht zugleich den Bildschirm löschen wollen, benutzen Sie die zweite Anweisung.

Ein Programm, das zusammenfaßt, was wir bisher kennengelernt haben

Bevor Sie weiterlesen, können Sie das folgende Programm laufen lassen, um zu sehen, was es macht. Starten Sie es mit 'dBASE beispiel', wenn Sie im CP/M sind, oder mit 'DO beispiel', wenn Sie im dBASE II sind. Antworten Sie auf die Nachrichten des Programms. Nachdem Sie es ausprobiert haben, können Sie sich die Beschreibung ("Dokumentation") ansehen. Sie bildet eine Zusammenfassung und Erläuterung des meisten Stoffes, den wir bisher behandelt haben.

Beispiel 4.13:

***** Beispiel.CMD *****

- * Dieses Programm informiert den Benutzer durch Bildschirm-Nachrichten, nimmt Daten entgegen,
- * um sie in Speicher-Variablen aufzubewahren und führt dann ein vom Benutzer ausgewähltes Unter-
- * programm aus. Es handelt sich nur um ein fragmentarisches Programm, aber es funktioniert
- * und demonstriert die Programmier-Technik in dBASE II.
- *
- * Wir haben die Unterprogramme noch nicht geschrieben, die aufgerufen werden sollen, sondern
- * sie durch bloße Nachrichten simuliert, die anzeigen, welchen Weg die Programmausführung
- * genommen hat.
- * Üblicherweise meldet dBASE die Ausführung von Befehlen (z. B. SKIP, d.h. Weiterschalten auf
- * einen neuen Datensatz) auf dem Bildschirm. Dies kann verwirrend sein, daher eingeben wir SET
- * TALK OFF (schalte "Gesprächigkeit" aus).

SET TALK OFF

USE ausgaben

- * Um das Programm so lange zu Wiederholen, bis sein Abbruch gewünscht wird,
- * betten wir es in eine DO WHILE-Schleife ein

STORE "4" TO WAHL

DO WHILE WAHL <> "0"

ERASE

- * Es ist ein guter Grundsatz, auf dem Bildschirm mit ERASE aufzuräumen, bevor irgendwelche
- * neuen Daten dargestellt werden. Dies erhöht nicht nur die Übersicht, sondern erregt die Aufmerksamkeit
- * der Bedienkraft.
- * Ein Unterprogramm kann zeitweilig durch eine Bildschirmausgabe wie das folgende Menue
- * ersetzt werden:

```

?
?
?
?
? " Menue für Kontoführung"
?
?
? " 0 = Ende"
? " 1 = Übersicht über die Außenstände"
? " 2 = Neue Rechnungen schreiben"
? " 3 = Zahlungen eingeben"
?
? " Wählen Sie durch Eingabe einer Zahl"
WAIT TO Wahl
ERASE

```

- * Da wir die Unterprogramme noch nicht entworfen haben, welche diese drei Aufgaben abwickeln
- * sollen, müssen wir den Computer veranlassen, verschiedene Kommentare auszugeben, abhängig
- * davon, welche Wahl der Benutzer getroffen hat.

```

IF Wahl = "1"
    @ 0, 20 SAY "EINS"
ELSE
    IF Wahl = "2"
        @ 1, 20 SAY "ZWEI"
    ELSE
        IF Wahl = "3"
            @ 2, 20 SAY "DREI"
        ELSE
            @ 7, 20 SAY '
            @ 8, 20 SAY 'Jedes andere Zeichen als 0, 1, 2 oder 3 '
            @ 9, 20 SAY 'veranlaßt dieses Programm, nach der Ausgabe '
            @ 10, 20 SAY 'dieser Nachricht, die Ausführung abubrechen. '
            @ 11, 20 SAY 'Beachten Sie, daß die Ziffern in den oben '
            @ 12, 20 SAY 'stehenden "IF"-Anweisungen in Anführungszeichen '
            @ 13, 20 SAY 'geschrieben werden müssen, weil der Befehl '
            @ 14, 20 SAY ' "WAIT" nur Zeichen-Eingaben annimmt. '
            @ 15, 20 SAY '

```

- * Die Ausgabe von "SAY ' ' ", d.h. einiger Leerzeichen, ist dann sinnvoll, wenn Sie auf Ihrem
- * Rechner die Hervorhebung durch negative Darstellung eingestellt haben. Dann erscheint nämlich
- * ein entsprechender weisser Balken.

```

ENDIF 3
ENDIF 2
ENDIF 1

```

- * Jedes IF muss ein zugehöriges ENDIF haben! Wir haben außerdem hinter jedes ENDIF als
- * Kommentar eine Zahl geschrieben, die anzeigt, zu welchem IF es gehört. So können wir
- * sichergehen, daß wir alle Blöcke geschlossen haben.
- * Die folgende DO WHILE-Schleife erzeugt eine Verzögerung von ein paar Sekunden, um die
- * letzte Nachricht lange genug auf dem Bildschirm stehenzulassen, daß man sie vor dem Abbruch
- * des Programms lesen kann. Dies werden Sie sicher in vielen Programmen, die Sie schreiben,
- * nützlich finden. Um die Verzögerungszeit zu verändern, tauschen Sie entweder die Obergrenze oder
- * die Schrittweite (+ 1) gegen einen anderen Wert.

```
STORE 1 TO X
DO WHILE X < 200
    STORE X + 1 TO X
ENDDO
```

```
    IF Wahl <> "0" .AND. Wahl <> "1" .AND. Wahl <> "2" .AND. Wahl <> "3"
    RETURN
ENDIF
```

```
ENDDO
```

- * Hier ist unsere Haupt-Schleife zu Ende. Es folgt eine Verabschiedung
- * des Programms?

```
?
?
?
```

```
INPUT 'Wollen Sie fortfahren (Yes oder No)?' TO Entscheid
```

- * Die Eingabe der Buchstaben "Y", "N", "T" oder "F" bewirkt die Eingabe eines
- * Wahrheitswertes. Es bewirken: "Y", "T": TRUE (wahr)
- * "N", "F": FALSE (falsch)

```
ERASE
```

```
IF Entscheid
```

```
    INPUT "Okay, geben Sie mir rasch eine Zahl." TO zahl
```

```
    IF TYPE (zahl) = "N"
```

```
        @ 10, 20 SAY "Das war die Zahl: "
```

```
        @ 10, 38 SAY zahl
```

```
    ELSE
```

```
        IF TYPE (zahl) = "C"
```

```
            @ 10, 20 SAY "Das war keine Zahl, sondern ein TEXT!"
```

```
    ELSE
```

```
        IF TYPE (zahl) = "L"
```

```
            @ 10, 20 SAY "Das war keine Zahl, sondern ein logischer Wert!"
```

```
        ENDIF logischer Wert
```

```
    ENDIF Texteingabe
```

```
ENDIF numerische Eingabe
```

```
ELSE
```

```
    @ 10, 20 SAY " Warum nicht? "
```

```
    WAIT
```

```
ENDIF Entscheid
```

```
@ 20, 20 SAY " Jetzt ist endgültig Schluß. Auf Wiedersehen. "
```

- * Die folgende DO WHILE-Schleife erzeugt eine Verzögerung von ein paar Sekunden, um die letzte
- * Nachricht lange genug auf dem Bildschirm stehenzulassen, daß man sie vor dem Abbruch des
- * Programms lesen kann. Dies werden Sie sicher in vielen Programmen, die Sie schreiben, nützlich
- * finden. Um die Verzögerungszeit zu verändern, tauschen Sie entweder die Obergrenze oder die
- * Schrittweite (+ 1) gegen einen anderen Wert.

```
STORE 1 TO X
DO WHILE X < 300
  STORE X + 1 TO X
ENDDO
ERASE
RETURN
```

* Ende Beispiel 4.13

Vielleicht wollen Sie das Programm noch einmal laufen lassen. Probieren Sie alle Alternativen aus, versuchen Sie dann, Eingaben zu machen, die eindeutig falsch sind. Sie werden daran sehen, wie das Programm arbeitet und wie dBASE II mit auftretenden Bedien-Fehlern umgeht.

Obwohl es nur ein fragmentarisches Programm ist, das keine wirklich nützliche Arbeit leistet, macht "Beispiel.CMD" einige Dinge deutlich:

- 1.) Der regelmäßige Gebrauch der Anweisung ERASE schafft auf einfache Weise Ordnung. Einerseits wird dadurch der Bildschirm gelöscht, so daß neue Meldungen auf einer "sauberen Tafel" erscheinen. Andererseits wird der für maximal 64 GET-Anweisungen reservierte Speicherplatz wieder frei gemacht (mit einem READ können maximal 64 mal Daten eingelesen werden).
- 2.) Das Einrücken hilft, die Arbeitsweise des Programms für den Leser klarzustellen. Aus dem gleichen Grund verwenden wir Groß- und Kleinschreibung. Der Computer sieht alles (außer Texten ("Strings") oder Zeichen ("C")), als wäre es groß geschrieben, aber für den Menschen erhöht dies die Lesbarkeit.
- 3.) Die "?"-Anweisung kann dazu verwendet werden, Leerzeilen auf dem Bildschirm einzufügen und Zeichenketten (in Anführungszeichen oder eckigen Klammern) als Nachrichten auszugeben.
- 4.) Die WAIT-Anweisung wartet auf ein einzelnes Zeichen, bevor die Programmausführung fortgesetzt wird. Die Eingabe muß dann als Character (Zeichen) behandelt werden, in der Weise, wie wir es in den geschachtelten IF-Anweisungen getan haben, indem wir die Werte, nach denen wir suchen, in Anführungszeichen eingeschlossen haben.
- 5.) Der Befehl INPUT wartet auf und akzeptiert jeden Datentyp, aber Zeichen oder Zeichenketten (Texte) müssen in einfache oder doppelte Anführungszeichen oder in eckige Klammern eingeschlossen werden. Wenn in Ihrer Eingabe ein Apostroph vorkommt, dann benutzen Sie die doppelten Anführungszeichen oder die eckigen Klammern, um den String zu kennzeichnen, sonst kommt der Computer durcheinander.

- 6.) Sie brauchen Variable nicht vorher zu definieren (wie zum Beispiel in Pascal). Verwenden Sie einfach einen neuen Namen, wenn Sie einen benötigen (bis zu maximal 64 gleichzeitig). Aber achten Sie darauf, daß Sie nicht einen Namen mehrfach für verschiedene Zwecke benutzen und an der einen Stelle den anderswo wieder benötigten Inhalt überschreiben.
- 7.) Logische Werte können in abgekürzter Form benutzt werden. "IF Entscheidung" im Programmbeispiel funktioniert so, als hätten wir gesagt: "IF Entscheidung = T" (True = wahr).
- 8.) Die Anweisung RETURN am Ende des Programms ist nicht notwendig, wurde aber angefügt, weil sie nötig wäre, wenn das Beispiel als Unterprogramm in einem anderen "command file" aufgerufen würde. Dadurch wird dem Computer gesagt, daß er am Ende des (Unter-) Programms zu der Stelle zurückkehren soll, von wo dieses aufgerufen worden war, anstatt einfach mit der Ausführung aufzuhören.

Wir haben einmal RETURN eingefügt, um die Programmausführung abubrechen, wenn im ersten Menue ein ungültiges Zeichen eingegeben wurde.

- 9.) Das System kann mit der Funktion "TYPE <ausdruck>" den Datentyp einer Eingabe erkennen. Das ist wichtig für die Behandlung von Eingabefehlern. Wenn das Programm sagt: "Gib mir rasch eine Zahl", erkennt es, ob wirklich eine Zahl eingegeben wurde. Gibt man z.B. "Hannelore" oder "123" ein, so protestiert es: "Das war ein Text".

Es erkennt auch die Zeichen T, F, Y, N als logische Werte. Die Eingabe von Hannelore (ohne Anführungszeichen) dagegen verursacht eine Fehlermeldung:

SYNTAX ERROR, RE-ENTER
(SYNTAX-FEHLER, BITTE NEU EINGEBEN)

In diesem Fall wird das Programm also nicht abgebrochen. Das Wort ohne Anführungszeichen kann für dBASE II höchstens einen Variablen-Namen bedeuten, in eine Variable kann aber kein Variablen-Name eingegeben werden.

Arbeiten mit mehreren Datenbanken gleichzeitig (PRIMARY, SECONDARY, SELECT)

Wie wir gesehen haben, gibt man, wenn man mit dBASE II zu arbeiten beginnt, USE <dateiname> ein, um dem System zu sagen, mit welcher Datei man arbeiten will, anschließend kann man in diese Datenbank Daten eingeben, sie editieren usw.

Um mit einer anderen Datenbank zu arbeiten, gibt man USE <NeueDatei> ein. dBASE II schließt dann die erste Datei und eröffnet die zweite, ohne Rücksicht darauf, ob das unseren Wünschen entspricht. (Der Computer benötigt eine gewisse interne Verwaltung, um mit einer Datei umzugehen, die ihm z. B. sagt, wo die einzelnen Datensätze tatsächlich auf der Platte stehen. Außerdem muß dafür gesorgt werden, daß Veränderungen an einer Datei stets auch wirklich auf die Platte (Diskette) geschrieben werden, bevor man den Computer ausschaltet - das ist der Grund, warum wir einmal dringend darauf hingewiesen haben, daß Sie immer QUIT eingeben sollen, wenn Sie eine Sitzung mit dBASE II beenden wollen. Das veranlaßt das System, alle eventuell noch im Speicher befindlichen Informationen tatsächlich auf die Platte zu schreiben und die Datei korrekt als "geschlossen" zu markieren, so daß die Daten später wieder gelesen werden können.)

Sie können auf diese Weise eine beliebige Anzahl von Dateien (nacheinander) benutzen, sowohl direkt vom Terminal als auch von Programmen aus. Sie können eine Datei auch schließen, ohne gleichzeitig eine neue zu eröffnen, indem Sie USE eingeben (ohne eine neue Datei zu benennen).

Wenn Sie eine Datei mit USE eröffnen, "spult" das dBASE II-System sie auf den Anfang zurück und stellt einen inneren Merker auf den ersten Datensatz ein. In den meisten Fällen wird das genau das sein, was Sie wünschen. Bei manchen Anwendungen aber werden Sie zwischenzeitlich eine andere Datei bearbeiten wollen, ohne die gerade bearbeitete Position in der ersten zu verlieren. dBASE II besitzt eine sehr nützliche Vorkehrung, um mit zwei aktiven Positionen in zwei Dateien gleichzeitig zu arbeiten. Sie können es "VORDERGRUND-DATEI" und "HINTERGRUND-DATEI" nennen oder - den englischen Schlüsselworten entsprechend - primäre und sekundäre Datei: PRIMARY und SECONDARY. Zwischen beiden schalten Sie mit der SELECT-Anweisung um.

Wenn Sie zu arbeiten anfangen und mit USE eine Datei eröffnen, wird diese automatisch als primäre (PRIMARY) Datei behandelt. Um nun mit einer anderen Datei zu arbeiten, ohne die Position in der ersten zu verlieren, schalten Sie zuerst auf die Hintergrundbearbeitung um durch Eingabe von: SELECT SECONDARY, dann USE <NeueDatei>. Um in die Vordergrundbearbeitung zurückzukehren, geben Sie einfach die Anweisung SELECT PRIMARY und fahren mit der Bearbeitung der primären Datenbank fort.

Die beiden Arbeitsbereiche können unabhängig voneinander benutzt werden. Jegliche Befehle, die Datenfelder oder Datensätze bewegen, beziehen sich ausschließlich auf den gewählten Arbeitsbereich.

Sie können aber Informationen zwischen dem Vordergrund- (P.) und Hintergrund- (S.) Bereich transportieren, indem Sie P. oder S. als Präfix (Vorsatz) von Variablennamen verwenden. Wenn Sie im PRIMARY (Vordergrund)-Bereich arbeiten, verwenden Sie "S." (um Datenobjekte im Hintergrund anzusprechen), wenn Sie im SECONDARY (Hintergrund)-Bereich arbeiten, setzen Sie "P." vor die Namen, um Daten im Vordergrund anzusprechen.

Ein Beispiel dafür finden Sie in der Programmdatei "Zweidat.CMD" (im Abschnitt 4). Einzelne Datensätze in einer Datei im Vordergrund oder PRIMARY-Bereich werden mit allen Datensätzen in einer anderen Datei verglichen, die sich im Hintergrund oder SECONDARY-Arbeitsbereich befindet. Als kleines Beispiel für die Verwendung von Vorder- und Hintergrund-Dateien habe ich ein Programm geschrieben, mit dem Sie die Dateien "Namen" und "Ausgaben" bequem gleichzeitig auf dem Bildschirm anschauen können. Sie können sich in jeder der Dateien unabhängig von der anderen durch einfache "Knopf-Druck-Funktion" vorwärts und rückwärts bewegen. Außerdem sehen Sie hier die Unterprogramm-Technik im praktischen Beispiel.

Hier ist das Haupt-Programm:

Beispiel 4.14: "zweidat.cmd"

```

* Datei: zweidat.cmd
* SCHALTE HERVORHEBEN DURCH INVERS/INTENSITÄT EIN
SET INTENSITY ON

ERASE
USE NAMEN
DO ZEIGNAM
SELECT SECONDARY
USE AUSGABEN
DO ZEIGAUS

STORE " " TO WAHL
DO WHILE WAHL <> "E" .AND. WAHL <> "e"

@ 0, 0 SAY " NAMEN-DATEI: V=VORWÄRTS, R=RÜCKWÄRTS"
@ 1, 0 SAY "AUSGABEN-DATEI: N=VORWÄRTS, Y=RÜCKWÄRTS"
@ 1, 40 SAY "E=ENDE"

WAIT TO WAHL

IF WAHL = "V" .OR. WAHL = "v" .OR. WAHL = "R" .OR. WAHL = "r"
  SELECT PRIMARY
  IF WAHL = "V" .OR. WAHL = "v"
    SKIP
  ELSE
    SKIP -1
  ENDIF VORWAERTS
  DO ZEIGNAM
ELSE
  IF WAHL = "N" .OR. WAHL = "n" .OR. WAHL = "Y" .OR. WAHL = "y"
    SELECT SECONDARY
    IF WAHL = "N" .OR. WAHL = "n"
      SKIP
    ELSE
      SKIP -1
    ENDIF VORWAERTS
    DO ZEIGAUS
  ENDIF SECONDARY
ENDIF BLAETTERN

ENDDO ENDE DES PROGRAMMS
ERASE

```

Natürlich brauchen Sie noch die beiden Unterprogramme "zeignam.cmd" (Beispiel 4.15) und "zeigaus.cmd" (Beispiel 4.16), um die Datensätze darzustellen. Hier sind sie:

Beispiel 4.15: "zeignam.cmd"

* Datei: zeignam.cmd

```
@ 3, 16 SAY "CODE"
@ 3, 26 SAY "NAME"
@ 3, 47 SAY "TELEFON"
```

SET INTENSITY OFF

```
@ 4, 15 SAY" +-----+-----+-----+"
@ 5, 15 SAY" !       !       !       !"
@ 6, 15 SAY" +-----+-----+-----+"
@ 5, 16 SAY KUNDCODE
@ 5, 26 SAY NAME
@ 5, 47 SAY TELEFON
```

SET INTENSITY ON

```
@ 7, 16 SAY "PLZ"
@ 7, 26 SAY "STADT"
@ 7, 47 SAY "ADRESSE"
```

SET INTENSITY OFF

```
@ 8, 15 SAY" +---+ +-----+-----+"
@ 9, 15 SAY" !  !  !       !       !"
@ 10, 15 SAY" +---+ +-----+-----+"
@ 9, 16 SAY POSTLTZAHL
@ 9, 26 SAY STADT
@ 9, 47 SAY ADRESSE
```

```
SET INTENSITY ON
RETURN
```

Ende Beispiel 4.15

Beispiel 4.16: "zeigaus.cmd"

* Datei: zeigaus.cmd

@ 12, 16 SAY "CODE"
 @ 12, 26 SAY "SCH.DAT"
 @ 12, 35 SAY "SCH.NR"
 @ 12, 47 SAY "AUFTR.NR"

SET INTENSITY OFF

@ 13, 15 SAY" +-----+-----+-----+-----+
 @ 14, 15 SAY" ! ! ! ! !
 @ 15, 15 SAY" +-----+-----+-----+-----+
 @ 14, 16 SAY KUNDE
 @ 14, 26 SAY SCHECK:DAT
 @ 14, 35 SAY SCHECK:NR
 @ 14, 47 SAY AUFTR:NR

SET INTENSITY ON

@ 16, 16 SAY "BESCHREIBUNG"
 @ 16, 37 SAY "HERSTELLER"

SET INTENSITY OFF

@ 17, 15 SAY" +-----+-----+-----+-----+
 @ 18, 15 SAY" ! ! ! ! !
 @ 19, 15 SAY" +-----+-----+-----+-----+
 @ 18, 16 SAY BESCHR
 @ 18, 37 SAY NAME

SET INTENSITY ON

@ 20, 16 SAY "BETRAG"
 @ 20, 28 SAY "RECH.DAT"
 @ 20, 37 SAY "RECHNR."
 @ 20, 45 SAY "STUNDEN"

SET INTENSITY OFF

@ 21, 15 SAY" +-----+-----+-----+-----+
 @ 22, 15 SAY" ! ! ! ! !
 @ 23, 15 SAY" +-----+-----+-----+-----+
 @ 22, 16 SAY BETRAG
 @ 22, 28 SAY RECH:DAT
 @ 22, 37 SAY RECH:NR
 @ 22, 45 SAY STUNDEN

SET INTENSITY ON
 RETURN

Ende Beispiel 4.16

Sicher werden Sie es auf die Dauer recht umständlich finden, derartige Unterprogramme zu schreiben, die mit SAY Linien ziehen.

Daher gibt es ein Hilfsprogramm, das automatisch solche Unterprogramme erzeugt. Es heißt ZIP und wird nach diesem Abschnitt besprochen. Weitere Hilfsmittel sind zum Beispiel Quickcode, Autocode I oder dBASE-Window, Programmgeneratoren, die Sie ebenfalls kaufen können und die Ihnen die Arbeit mit dBASE II erleichtern.

Allgemein nützliche Systembefehle und Funktionen

MODIFY COMMAND <dateiname> läßt Sie das angegebene Programm (command file) zu bearbeiten, wobei die üblichen bildschirmorientierten Editierbefehle benutzt werden können.

BROWSE (brause, rase durch...) zeigt bis zu 19 Datensätze und so viele Datenfelder, wie auf den Bildschirm passen. Um Felder jenseits der rechten Seite des Bildschirms zu sehen, geben Sie <CTL>-B, um mit dem Bildschirm auf die rechte Seite zu "rollen". Mit <CTL>-Z "rollen" Sie wieder nach links.

CLEAR bewirkt ein "Rücksetzen" von dBASE II, wobei alle Variablen gelöscht und alle Dateien geschlossen werden.

RESET wird nach dem Auswechseln einer Diskette benutzt, um im Betriebssystem das Inhaltsverzeichnis für die Laufwerke zu aktualisieren. Bitte lesen Sie die ausführliche Beschreibung der Befehle im 3. Kapitel, bevor Sie RESET benutzen!

* erlaubt Kommentare in einer Programm-Datei. Diese Kommentare werden aber nicht ausgegeben, wenn die Programm-Datei ausgeführt wird. Das erlaubt es dem Programmierer, sich Notizen zu machen, ohne dadurch den Operator (die Bedienungskraft) zu verwirren. Zwischen dem Kommentarzeichen und dem Text der Notiz muß mindestens ein Leerzeichen stehen, und Kommentare dürfen nicht zusammen mit einer vom Computer auszuführenden Anweisung in einer Zeile stehen. NOCHMALS: Befehle und Kommentare müssen in verschiedenen Zeilen stehen!

REMARK (ANMERKUNG) erlaubt es, Kommentare in einer Programm-Datei zu speichern, die als Nachrichten an den Benutzer ausgegeben werden, wenn das Programm ausgeführt wird. Zwischen dem Wort REMARK und der Anmerkung muß sich mindestens ein Leerzeichen befinden, und die Anmerkung darf sich nicht in der gleichen Zeile wie ein Befehl befinden.

RENAME <alteDatei> TO <neueDatei> ändert den Namen, unter dem eine Datei in der CP/M-Directory (dem Disk-Inhaltsverzeichnis) eingetragen ist. *Versuchen Sie NIE eine Datei umzubenennen, die gerade bearbeitet wird!*

QUIT TO '<betriebssystem.COM-file-liste>' gestattet es, dBASE II abubrechen und automatisch im CP/M die Ausführung anderer ".COM"-Dateien zu starten. Jede ".COM"-Datei muß in einfachen Anführungszeichen angegeben werden und von den anderen, jeweils in Anführungszeichen stehenden, Dateien durch Kommata getrennt werden.

? Sie können die Anweisung "?" auch zum Aufruf folgender Funktionen benutzen:

ist eine Funktion, welche die Nummer des laufenden Datensatzes (Record #) liefert.

* ist eine Funktion, die "wahr" (True) ergibt, wenn ein Datensatz als "gelöscht" markiert wurde, und "falsch" (False), wenn er nicht gelöscht ist.

EOF ist eine Funktion, die "wahr" ergibt, wenn das Ende der gerade bearbeiteten Datei (End Of File) erreicht wurde, "falsch", wenn das Ende noch nicht erreicht wurde.

Einige Worte über Programmierung und wie Sie Ihre Anwendung analysieren sollten

Das erste, was Sie tun sollten, wenn Sie ein Programm entwerfen wollen, ist: Schalten Sie Ihren Computer aus.

Ja, wirklich: Hier machen viele Programmierer einen entscheidenden Fehler. Sie fangen sofort an, eine Problemlösung zu "kodieren", bevor sie überhaupt eine klare Vorstellung davon haben, was sie eigentlich tun wollen.

Ein besserer Weg wird in etlichen Veröffentlichungen über "strukturierte Programmierung" beschrieben und in einigen strukturierten Sprachen. Eine Stelle, an der Sie nachlesen könnten, bildet Kapitel 2 von "An Introduction to Programming and Problem Solving in Pascal" von Schneider, Weingart und Perlman. Eine andere Empfehlung sind die Kapitel 1, 4 und die ersten paar Seiten von Kapitel 7 in "Pascal Programming Structures" von Cherry. Wenn Sie sich dann ernsthaft mit Programmierung beschäftigen wollen, da gibt es ein ausgezeichnetes Buch über PL/I-Programmierung: "PL/I Structured Programming" von Joan Hughes.

(Ein recht brauchbares deutsches Buch über strukturierte Programmierung, ist: "Strukturierte Programmierung" von W. Jordan und H. Urban, Springer-Verlag ISBN 3-540-08740-0.)

Kurz gefaßt sollten Sie folgendermaßen vorgehen:

Fangen Sie damit an, Ihr Problem in normalem Deutsch zu beschreiben. Notieren Sie allgemeine Feststellungen.

Nun beschreiben Sie es genauer. Welche Daten-Eingaben sind vorzusehen. In welcher Form sollen Ausgaben und Übersichten (REPORTS) erfolgen?

Als nächstes denken Sie über Ausnahmefälle nach. Welches sind die Ausgangs-Bedingungen? Was geschieht, wenn ein Datensatz nicht vorhanden ist, der bearbeitet werden soll?

Sobald Sie festgelegt haben, was Sie wollen, beschreiben Sie Ihr Problem in Pseudo-Englisch, d.h. entwerfen Sie (ob in Deutsch oder Englisch, ist zunächst egal) ein "Program" in einem Jargon, der mehr oder weniger der dBASE II-Programmiersprache entspricht, wie Sie sie inzwischen kennengelernt haben. Versuchen Sie, Ihr Problem in Begriffen darzustellen, die der Computer verstehen kann.

Sie könnten Ihren Programmentwurf z. B. so schreiben:

Benutze die Datenbank, welche die Kosten enthält.
 Drucke die unbezahlten Rechnungen aus.
 Schreibe einen Scheck für jede unbezahlte Rechnung aus.

Nach Anfügen einiger Einzelheiten könnte das Ganze im nächsten Schritt so aussehen:

USE KostenDatei

Drucke die unbezahlten Rechnungen des letzten Monats aus, verwende dabei die Form-
 Datei SUMMIER.FRM.

Fange mit dem ersten Datensatz der Datenbank an.

Und gehe sie bis zum Ende durch:

Wenn die Rechnung noch nicht bezahlt wurde

Zahle die Rechnung

Und füge sie in die Datei ein

Mache dies für jeden einzelnen Datensatz.

In etwa zwei weiteren Schritten könnte daraus folgendes Programm entstehen:

USE KostDat

* Drucke eine Zusammenfassung für Dezember 1980

REPORT FORM Summier FOR Rech:Dat >= '801201' .AND.

Rech:Dat <= '801231' TO PRINT

GOTO TOP

DO WHILE .NOT. EOF

IF Scheck:Nr = ' '

DO SchrbScheck

DO Berichtige

ENDIF

SKIP

ENDDO

* Gehe zum ersten Datensatz

* Wiederhole für ganze Datei

* Wenn Rechnung unbezahlt

* Schreibe einen Scheck

* Bringe Datensätze a. neuen Stand

* Gehe zum nächsten Datensatz

Dieses Vorgehen wird mit den Begriffen Top-Down, schrittweise Verfeinerung bezeichnet, aber das ist nur eine hochgestochene Ausdrucksweise für das Prinzip: 'Fange mit einer allgemeinen Übersicht an, unterteile das Ganze dann in Teilprobleme und versuche, diese zu bewältigen'.

Das ist weiter nichts als ein grundsätzlicher Ansatz, um mit fast jeder Art von Problem fertig zu werden. Stellen Sie erst einmal grob klar, worin das Problem besteht (und worin nicht). Wenden Sie sich dann schrittweise feineren und feineren Details zu, lösen Sie die leicht lösbaren Teilaufgaben, legen Sie die schwierigeren Fragen beiseite, um sie später durch weitere Zerlegung in Teil-Teilprobleme in Angriff zu nehmen.

An dieser Stelle haben wir in unserem Beispiel noch nicht geklärt, wie die Formatier-Datei "Summier.FRM" aussieht oder die Unterprogramme "SchrbsScheck.CMD" und "Berichtige.CMD", aber das macht nichts.

Tatsächlich tun wir gut daran, uns nicht mit der Lösung dieser Details zu belasten, sondern erst einmal die Lösung des Gesamtproblems zu entwerfen. Später, wenn wir die generelle Lösung erprobt haben, können wir auf diese Einzelheiten zurückkommen und die Unterprogramme ausarbeiten.

Dabei kann es manchmal nötig sein, den Entwurf für die Gesamtlösung zu überarbeiten, beispielsweise wenn man Details des Informations-Austausches zwischen Haupt- und Unterprogramm übersehen hat. Wichtig bleibt aber, daß man durch dieses Verfahren nicht nur eine klare Übersicht über das zu lösende Problem gewinnt, sondern auch zu einem Programm gelangt, das diese Klarheit widerspiegelt und so später gut verstanden und - wenn nötig - veränderten Anforderungen angepaßt werden kann.

Hinweis: Sie können ein erst teilweise fertiges Programm wie dieses bereits testen, indem Sie, wie die Programmierer sagen, "Dummy-Prozeduren" (Pseudo-Unterprogramme), verwenden. Erzeugen Sie eine Unterprogramm-Datei, die Sie in Ihrem Programm aufrufen, und schreiben Sie nur drei Dinge hinein: Die Ausgabe einer Nachricht, die uns wissen läßt, daß der Computer das Unterprogramm aufgerufen hat, WAIT und RETURN. dBASE II wird dann diese Prozeduren aufrufen, die Nachricht auf den Bildschirm schreiben, auf das Drücken einer Taste zur Bestätigung warten und zum Hauptprogramm zurückkehren, um dieses weiter zu bearbeiten.

Abschnitt 5: Weitere Befehle zur Datenmanipulation

Inhaltsverzeichnis

Weitere Befehle zur Datenmanipulation.....	157
Verändern von dBASE II-Standard-Werten (SET ...)	164
Mischen von Datensätzen aus zwei Dateien mit UPDATE.....	166
Zusammenfügen ganzer Datenbanken mit JOIN.....	167
Vollständige Bildschirm-Formatierung und Editiermöglichkeiten mit @ ... SAY ... GET ... PICTURE.....	168
Formatieren von Druckseiten mit SET FORMAT TO PRINT, @ ... SAY ... USING.....	170
Entwurf und Druck eines Formulars.....	171
Rückblick.....	173

Inzwischen sollten Sie bereits Programme schreiben, die für Sie nützliche Arbeit zu leisten vermögen. Um Ihnen dabei weiterzuhelfen, führen wir in diesem Abschnitt weitere Funktionen ein, ein paar neue Befehle, und sprechen etwas mehr im Detail darüber, wie Sie Ihre Daten genau in der Form drucken können, in der Sie sie haben wollen.

Weitere Befehle zur Datenmanipulation

Funktionen sind besondere Operationen, die in Ausdrücken gebraucht werden, um gewisse Informationen zu gewinnen, die mit gewöhnlicher Arithmetik, logischen oder String-Operatoren schwer oder gar nicht zu erreichen sind. Die Funktionen von dBASE II fallen in die gleichen drei Kategorien (numerisch, logisch und textuell), abhängig vom Typ der Ergebnisse, welche sie erzeugen.

Funktionen können direkt von der Tastatur aufgerufen werden, indem man die Anweisung "?", gefolgt von einem Leerzeichen und dem Funktionsnamen eingibt. Diese Anweisung besagt "Schreibe das Ergebnis von..." oder schlicht "wie lautet...".

Geben Sie nur die nackte Funktion ein, so erhalten Sie eine Syntax-Fehlermeldung, da dann das Ergebnis der Funktion kein Ziel hat. Geht aus anderem Zusammenhang hervor, was mit dem Funktionsergebnis zu geschehen hat, wie in

```
"STORE !('Groß-Kleinschreibung') TO Gross",  
(SPEICHERE DIE-GROSS-SCHREIBUNG von '...' IN Gross)
```

so entfällt natürlich die "?"-Anweisung. Beachten Sie übrigens, daß unser deutsches "ß", ebenso wie die Umlaute, von der unten beschriebenen Funktion "!" nicht in Großbuchstaben umgewandelt werden, da diese sie als "Sonderzeichen" betrachtet.

Funktionen können sowohl von der Tastatur als auch in einem Programm aufgerufen werden.

```
!( <Variable/Zeichenkette > )
```

Der Schrägstrich "/" bedeutet dabei ODER, d.h. Sie sollen entweder einen konkreten Variablen-Namen oder eine bestimmte String-Konstante ("Hallo") einsetzen.

(Denken Sie daran, daß "Strings" (Zeichenketten) einfach eine Aneinanderreihung von einzelnen Zeichen sind (einschließlich Leerzeichen, Ziffern und Sonderzeichen) die als Daten betrachtet und manipuliert werden.)

Plagen Sie sich nicht damit, sich jetzt alle Funktionen zu merken, aber überfliegen Sie die Beschreibungen, damit Sie wissen, wo Sie nachschauen können, wenn Sie irgendeine davon in einem Programm benötigen. Doch ist dies nur ein kurzgefaßter Überblick, der Ihnen zeigen soll, was für Funktionen es gibt und wozu sie zu gebrauchen sind. Eine erschöpfende Darstellung aller Funktionen und sämtlicher Anwendungsmöglichkeiten finden Sie im 3. Kapitel dieses Handbuches.

!(**<Variable/Zeichenkette>**)

ist eine Funktion, die Kleinschreibung in Großschreibung umwandelt. Sie verwandelt alle Zeichen von "a".."z" in einer Zeichenkette oder String-Variablen in Großbuchstaben. Alle sonstigen Zeichen in dem String werden nicht verändert. Man verwendet diese Funktion häufig zur Behandlung von Eingaben über die Tastatur, um sie in eine einheitliche Form umzuwandeln, in der sie in den Dateien abgelegt werden. Das vereinfacht es, die Daten später wiederzufinden, da Sie wissen, daß alle Daten in Großbuchstaben gespeichert wurden, gleichgültig, wie man sie eingegeben hat.

Das bedeutet, daß das System (sonst) beim Vergleich von Texten zwischen Groß- und Kleinschreibung unterscheidet, anders als ein Lexikon, das bei der alphabetischen Einordnung von Worten große und kleine Buchstaben im Alphabet gleich behandelt. Der Grund liegt darin, daß auf einem Computer für die Kleinbuchstaben größere Codezahlen benutzt werden, die den Codes für Großbuchstaben nach Einfügung verschiedener Sonderzeichen folgen.

Eine Konsequenz für den deutschen Benutzer daraus ist, daß kleine Umlaute nicht in große Umlaute verwandelt werden, da sie für diese Funktion Sonderzeichen sind. Auch werden nicht auf allen Systemen die gleichen Codes für Umlaute und "ü" verwendet, so daß die lexikalische Reihenfolge von Fall zu Fall zu klären ist. Sollte dies zu Problemen führen, schreibt man besser "MUENCHEN" statt "München".

\$ (<ausdruck/variable/string>, <start>, <länge>)

ist die Teilstring-Funktion. Sie sucht aus einer Zeichenkette oder String-Variablen die Zeichen heraus, die ab der start-Position stehen, und zwar so viele, wie länge bestimmt.

Zum Beispiel: Wenn wir eine Variable namens "Datum" haben, deren Inhalt '81023' ist, dann ergibt der Funktionsaufruf \$(Datum, 5, 2) den Wert '23'. Um diese Ziffern (die noch Zeichen sind, d.h. eine Zeichenkette bilden) in eine Zahl zu verwandeln können wir sagen:

VAL (\$(Datum, 5, 2)).

Ein Beispiel für die Anwendung dieser Funktion in einem Programm bestünde z. B. darin, Gruppen von zwei Zeichen aus einem 6-Zeichen umfassenden Datenfeld zu entnehmen die dann in ganze Zahlen verwandelt werden (unter Verwendung der VAL (...)-Funktion).

Verwechseln Sie dies nicht mit dem logischen-Teilstring Operator, der in Abschnitt 3 beschrieben wird.

Der Unterschied ergibt sich aus der Form des Aufrufes:

? 'BAY' \$ 'BAYERN'
.T.

(ergibt:)
(wahr - der String ist enthalten)

? \$ ('BAYERN', 1, 3)
'BAY'

(ergibt:)
(die ersten drei Zeichen)

&

ist die Funktion zum Einsetzen eines Makros. Ein Makro ist ein komplizierter Befehl, der durch ein Symbol abgekürzt vertreten wird. In dBASE II vertritt das Zeichen "&", gefolgt von einer Speicher-Variablen, den Befehl, dessen Text in dieser Variablen gespeichert ist. Wenn das Zeichen vor dem Namen einer Speicher-Variablen eingesetzt wird, so ersetzt dBASE II den Namen mit dem Inhalt der Speicher-Variablen (es muß sich dabei um eine Variable handeln, die eine Zeichenkette enthält). Das kann man zum Beispiel benutzen, wenn ein komplizierter Ausdruck mehrfach verwendet wird, um Parameter zwischen Programm-Dateien zu übertragen, oder in einem Programm, wenn der Wert des Parameters während des Programmablaufs eingesetzt werden soll.

Eine andere Möglichkeit bestünde darin, den Namen einer Datei in eine Variable einzulesen und diese dann folgendermaßen zu verwenden:

```
? 'Welche Datei wollen Sie ansehen?'
ACCEPT TO Datenbank
USE &Datenbank
```

@ (<variable1/string1>, <variable2/string2>)

ist eine Funktion, welche einen Teilstring sucht. Sie können sich Ihre Arbeitsweise anhand der Frage "Wo befindet sich string1 in string2?" vorstellen. Wenn Sie diese Funktion verwenden, so ergibt sie die Position des Zeichens, bei der die erste Zeichenkette (oder Zeichen-Variable) in der zweiten Zeichenkette (oder Variablen) anfängt. Wenn der zuerst genannte String nicht als Teil des zweiten vorkommt, so ergibt die Funktion den Wert "0".

Eine Anwendung dieser Funktion besteht darin, herauszufinden, wo eine bestimmte Folge von Zeichen in einer anderen Zeichenfolge anfängt, so daß Sie die oben genannte Funktion zum Herausfiltern der Zeichenfolge verwenden können. Eine andere Anwendung besteht darin, herauszufinden, ob eine Zeichenfolge überhaupt in einer anderen Zeichenfolge vorkommt.

```
? @ ('BAY', 'BAYERN')           (ergibt:)
1
```

ACHTUNG: Verwechseln Sie diese Funktion nicht mit der Ausgabe-Anweisung "@ <koordinaten> SAY...", die mit dem gleichen Zeichen eingeleitet wird.

Wenn Sie lediglich wissen wollen, ob ein String in einem anderen enthalten ist, können Sie den logischen String-Operator benutzen: String1 \$ String2, siehe Abschnitt 3.

Sie werden dies in einem Programm nützlich finden, wenn der Computer nach einer Zeichenfolge sucht, ohne daß eine Bedienkraft eingreift und nachsieht, wo sich die gesuchte Information befindet.

CHR (<zahl>)

ergibt das entsprechende ASCII-Zeichen zu einer Zahl. Je nachdem, wie Ihr Terminal den genormten ASCII-Code benutzt, könnte CHR (12) Ihren Bildschirm löschen, CHR (14) negative Schrift Darstellung aufrufen und CHR (15) wieder zur normalen (positiven) Darstellung zurückkehren. Andere Codes können dazu verwendet werden, um periphere Hardware zu steuern wie zum Beispiel einen Drucker. Sehen Sie in Ihrem Handbuch für den Drucker nach, mit welchen Steuerzeichen Sie Schriftzeichen und -typen einstellen können. Sicher werden Sie interessante Anwendungsmöglichkeiten entdecken.

Probieren Sie folgendes, um auf Ihrem Drucker das Unterstreichen von Texten zu erreichen: Verknüpfen Sie eine Zeichenkette, das Zeichen für "Wagenrücklauf" und das Unterstreich-Zeichen etwa so: ? 'string' + CHR (13) + '. Sie können auch ein Programm schreiben, das die LEN-Funktion dazu benutzt, um zu entscheiden, wie lang der Text ist, so daß die entsprechende Anzahl von Unterstreich-Zeichen bestimmt werden kann. (Anm. d. Üs.: Dies setzt voraus, daß Ihr Drucker die beiden Zeichen "Wagenrücklauf" und "Zeilenvorschub" (Carriage Return und Line Feed) benötigt, um normalerweise Texte zu drucken. Dann wird er, wenn man nur den "Wagenrücklauf" sendet, an den Anfang der bereits gedruckten Zeile gehen, ohne das Papier für eine neue Zeile vorzuschieben, so daß man mit dem Unterstreich-Zeichen das bereits Gedruckte nachträglich unterstreichen kann.)

Sie kann auch zur Abkürzung eines Befehls verwendet werden: STORE 'Delete Record' TO D. Der Befehl '&D 5' würde dann in einem laufenden Programm den Datensatz Nr.5 löschen.

Wenn auf den Makro-Befehl keine gültige Zeichenketten-Variable folgt, dann wird er übergangen. Das heißt, Sie können das Symbol selbst als Teil einer Zeichenkette verwenden, ohne eine Fehlermeldung zu erhalten.

FILE (<'filename'/variable/ausdruck>)

ergibt den Wert True (Wahr), wenn die genannte Datei (file) auf der Diskette (oder Platte) vorhanden ist, und den Wert False (Falsch), wenn sie nicht vorhanden ist. Wenn Sie direkt einen bestimmten Dateinamen angeben, dann schließen Sie ihn in Anführungszeichen ein. Wenn Sie eine Variable angeben, welche den Dateinamen enthält, dann werden keine Anführungszeichen verwendet. Sie können auch irgendeinen zulässigen Ausdruck einsetzen, der eine Zeichenkette liefert: 'FILE ("B:" + Datenbank)' teilt Ihnen mit, ob die Datei, deren Name in der Speicher-Variablen "Datenbank" gespeichert ist, sich auf der Diskette in Laufwerk B befindet. Damit kann ein Programm selbst feststellen, ob eine Datei vorhanden ist oder nicht und die Bedienungskraft anweisen, eine andere Diskette einzusetzen usw.

INT(<variable/ausdruck>)

ist eine Funktion, die aus einem numerischen Wert eine ganze Zahl ("integer") macht. Sie bildet von einer Zahl mit Stellen nach dem Komma (genauer: dem Dezimalpunkt, da die Amerikaner kein Komma, sondern einen Punkt schreiben!) die Abrundung, und zwar, indem sie einfach alles, was nach dem Dezimalpunkt steht, "wegwirft". Das, was zwischen den Klammern steht (Sie müssen die Klammern schreiben), der "Term", kann eine Zahl, der Name einer Variablen oder ein komplizierter numerischer Ausdruck sein. Im letzteren Fall wird dieser Ausdruck zuerst ausgewertet (d.h. berechnet), dann wird aus dem Ergebnis eine ganze Zahl gebildet.

Beachten Sie, daß INT (123.86) 123 zum Resultat hat, während INT (-123.86) -123 ergibt, der positive Wert ist zu klein, der negative zu groß gegenüber der kaufmännischen Rundung. Wenn eine Variable eingesetzt wurde, so erhalten Sie die entsprechende ganze Zahl nach "Abschneiden" (truncate) der Nachkommastellen von dem Wert, den die Variable gerade hat. Wenn wir uns gerade beim Datensatz 7 von "Ausgaben.DBF" befinden, dann ergibt ein Aufruf von INT (Betrag) den Wert 2333, den ganzzahligen Anteil von DM 2333.75.

Um zur nächstliegenden ganzen Zahl zu runden (anstatt einfach die Stellen nach dem Komma abzuschneiden), benutzen Sie die Form: 'INT (wert + 0.5)'. Zuerst wird der Wert in der Klammer berechnet, dann wird die INTeger-Funktion darauf angewendet.

Die Integer-Funktion kann auch dazu benutzt werden, um einen Betrag auf eine gewünschte Zahl von Stellen nach dem Komma aufzurunden. $\text{INT}(\text{wert} * 10 + 0.5) / 10$ rundet einen Wert auf die nächstgelegene Stelle nach dem Komma (Dezimalpunkt). Die entsprechende Rangfolge der Operationen ist dabei: Inhalt der Klammer, dann Bilden des INT-Wertes und schließlich Division. Um bis auf zwei Stellen zu runden verwenden Sie "100" statt "10", für drei Stellen "1000" usw.

LEN (<variable/string>)

ist die Funktion, welche die Länge einer Zeichenkette ergibt. Sie teilt Ihnen mit, wie viele Zeichen sich in einer Zeichenkette befinden, die Sie angeben. Das kann nützlich sein, wenn ein Programm entscheiden muß, wieviel Platz für eine Information benötigt wird, ohne daß dabei eine Bedienkraft eingeschaltet wird. Wenn aber eine Variable eingesetzt wird, die ein Zeichen-Feld enthält, dann ergibt diese Funktion die Größe des Feldes, nicht die aktuelle Länge des Inhaltes (da alle unbenutzten Positionen von dBASE II mit Leerzeichen aufgefüllt werden).

STR (<ausdruck/variable/zahl>, <laenge>,[<nachkommastellen>])

ist eine Funktion, die eine Zahl in eine Zeichenkette umwandelt. Sie verwandelt eine Zahl oder den Inhalt einer numerischen Variablen in eine Zeichenkette mit der angegebenen Länge und der angegebenen Zahl von Nachkomma-Stellen. Die angegebene Länge muß ausreichen, um alle Ziffern einschließlich des Dezimal-Punktes aufzunehmen. Wenn der numerische Wert kürzer ist als das definierte Feld, so wird der Rest mit Leerzeichen aufgefüllt. Wenn die Anzahl Stellen nach dem Komma nicht angegeben ist, so wird "0" dafür angenommen. Beispiel:

? STR (123.456, 6, 2)

Die Zahl 123.456 soll mit nur zwei Stellen nach dem Komma in einen String verwandelt werden. Die verbleibenden 5 Ziffern einschließlich des Dezimalpunktes ergeben 6 notwendige Stellen. Hätten wir drei Stellen nach dem Komma berücksichtigen wollen, so hätte der Aufruf lauten müssen:

? STR (123.456, 7, 3)

Diese Funktion wird oft verwendet, um Darstellungen zu vereinfachen. Zahlen werden in Strings verwandelt und dann mit anderen Strings verknüpft (zusammengehängt), um neue Zeichenketten zu bilden, die dargestellt werden.

TRIM

entfernt die nachfolgenden Leerzeichen aus dem Inhalt einer Zeichenketten-Variablen. Das kann durch Angabe von

STORE TRIM (<variable>) TO <neue-variable>

ausgeführt werden.

TYPE(<ausdruck>)

ist eine Funktion, die Ihnen den Daten-Typ einer Information liefert, sie ergibt 'C', 'N' oder 'L', je nachdem ob der "ausdruck" vom Datentyp Character (Zeichen), Numerisch oder Logisch ist. Beachten Sie, daß das Ergebnis der Funktion TYPE selbst den Typ "Character" hat. Sie sehen dies am Programm in Beispiel 4.1311 im Abschnitt 4, wo dBASE II den Datentyp einer Eingabe erkennt.

VAL (<variable/string/teilstring>)

ist eine Funktion, die eine Zeichenkette in eine ganze Zahl umwandelt. Sie verwandelt eine Zeichenkette oder den Teil einer Zeichenkette, die aus Ziffern besteht, einem Vorzeichen und ev. bis zu einem Dezimalpunkt in die entsprechende numerische Größe. "VAL ('123')" ergibt die Zahl 123, "VAL ('456.789')" ergibt 456. Um dies direkt auszuprobieren geben Sie ein: "? VAL ('...')".

VAL (Auftr:Nr) ergibt den numerischen Wert des Inhalts des Feldes, das die Auftrags-Nummer enthält, da wir alle Auftrags-Nummern als Zeichen gespeichert haben. Sie können die Funktion auch zusammen mit dem Substring-Operator benutzen: VAL (\$(<string>)). Das ist nützlich, wenn nur bestimmte Zeichen in dem String die gewünschte Zahl darstellen - siehe bei der Funktion \$.

Verändern von dBASE II-Standard-Werten

dBASE II besitzt eine Anzahl von Befehlen, die bestimmen, wie es mit Ihrem Computersystem zusammenarbeitet. Sie können die betreffenden Parameter entweder nach Belieben während der Arbeit verändern oder ein für alle Mal am Anfang eines Programmes einstellen und dann so belassen. In vielen Anwendungsfällen werden die Standardwerte (die sich ohne besonderes Zutun einstellen) Ihren Anforderungen entsprechen.

Diese Parameter werden in Programmen (oder interaktiv von der Tastatur aus) mit Hilfe des SET-Befehls verändert. In der nachstehenden Liste sind die üblichen Standardwerte, die sich ohne Modifikation ergeben, unterstrichen.

Auch diese Werte müssen Sie nicht auswendiglernen. Wenn Sie mit den normalen Standard-Werten arbeiten, können Sie von Fall zu Fall entscheiden, ob Sie den einen oder anderen Parameter in dieser Liste verändern wollen.

SET ALTERNATE TO <dateiname> erzeugt eine Datei mit der Namen-Erweiterung "T", auf die alles protokolliert wird, was auf Ihrem Bildschirm erscheint. Um dieses Mitschreiben zu starten, geben Sie ein:

'SET ALTERNATE ON'.

Sie können die Datei wechseln, auf die mitgeschrieben wird, indem Sie sagen: 'SET ALTERNATE TO <neue-datei>'.

WARNUNG: Bei der Version 2.3B, die dem Übersetzer zur Verfügung stand, funktioniert dies nicht so, wie im englischen Handbuch beschrieben. Es ist erforderlich, zuerst mit der alten Protokoll-Datei zu sagen SET ALTERNATE OFF, dann SET ALTERNATE TO <neue-datei> und dann erneut SET ALTERNATE ON. Andernfalls treten Fehler auf. Wir haben diese Funktion dazu benutzt, die meisten Beispiele im Handbuch direkt aus Protokollen von dBASE-Sitzungen zu gewinnen (um Schreibarbeit zu sparen und Fehler zu vermeiden).

Um die Protokollierung zu stoppen sagen Sie:

'SET ALTERNATE OFF'. Der Vorgang endet auch dann, wenn Sie das dBASE II-System mit QUIT verlassen.

SET BELL	<u>ON</u>	Schaltet Glocke (Pieps) ein, ertönt wenn Datenfeld voll ist (falls das Terminal eine Glocke oder einen Pieps hat).
	OFF	Schaltet Glocke (Pieps) aus.
SET CARRY	<u>ON</u>	Überträgt automatisch die Daten aus dem vorhergehenden Datensatz in den neuen, wenn der Befehl APPEND benutzt wird.
	OFF	APPEND erzeugt einen leeren Datensatz.

- SET COLON ON Verwendet Doppelpunkte, um Eingabefelder für Variable auf dem Bildschirm darzustellen.
 OFF Keine Anzeige durch Doppelpunkte.
- SET CONFIRM ON Wartet auf <RETURN> oder <ENTER>, bevor der Cursor in der bildschirmorientierte Editierung ein Eingabefeld verläßt.
 OFF Springt automatisch aus dem Feld heraus, wenn es voll ist.
- SET CONSOLE ON Echot alle Ausgaben auf Ihren Bildschirm.
 OFF Bildschirm aus.
- SET DEBUG ON Schickt Ausgabe der ECHO- und STEP-Anweisungen nur zum Drucker.
 OFF Schickt Ausgabe von ECHO und STEP zum Bildschirm.
- SET ECHO ON Alle Befehle in einem Programm werden auf den Bildschirm ausgegeben, während das Programm läuft.
 OFF Kein Echo der Befehle.
- SET EJECT ON Ermöglicht Blattvorschub im REPORT-Befehl.
 OFF Blattvorschub aus.
- SET ESCAPE ON Gestattet es, mit der <ESCAPE>-Taste (<ESC>) die Ausführung eines Programms abubrechen.
 OFF macht die <ESCAPE>-Taste unwirksam.
- SET EXACT ON Erfordert, daß alle Zeichen in einem Vergleich zweier Strings exakt übereinstimmen.
 OFF Erlaubt unterschiedlich lange Strings. Zum Beispiel wird 'ABCD' = 'AB' "wahr" (betrifft auch auch den FIND-Befehl).
- SET FORMAT TO PRINT Die "@"-Befehle wirken auf den Drucker.
- SET FORMAT TO SCREEN Die "@"-Befehle wirken auf den Bildschirm.
- SET HEADING TO <string> Verändert die Überschrift-Zeile (HEADING) im REPORT-Befehl.
- SET
INTENSITY ON Ermöglicht die Hervorhebung durch halbe Helligkeit oder negative Darstellung bei der Bildschirm-Editierung.
 OFF Keine Hervorhebung.

- SET LINKAGE ON Erlaubt die Kopplung von Vorder- und Hintergrund-Datenbanken bei der Darstellung auf dem Bildschirm mit bis zu 64 Feldern und bis zu 2000 Bytes (Zeichen) pro dargestelltem Datensatz. Wenn in beiden Dateien gleiche Feldnamen vorkommen, müssen P. oder S. als Präfix (Vorsatz) verwendet werden.
OFF Keine Koppelung.
- SET MARGIN TO <nnn> Stellt linken Druckrand auf Ihrem Drucker ein. (nnn <= 254, lies "kleiner oder gleich").
- SET PRINT ON Ausgabe auf das "LIST"-Device (z.B. Drucker).
OFF Kein Listing.
- SET RAW ON DISPLAY und LIST geben Datensätze ohne Leerzeichen zwischen den Feldern aus.
OFF DISPLAY und LIST schreiben Datensätze mit einem zusätzlichen Leerzeichen zwischen den Feldern.
- SET SCREEN ON Schaltet bildschirmorientierte Editierung für die Befehle APPEND, INSERT und CREATE ein.
 OFF Bildschirmorientierte Editierung aus.
- SET STEP ON Stoppt nach der Ausführung jedes Befehls in einem Programm, zur Fehlerbeseitigung in Programmen ("debugging").
OFF Normale ununterbrochene Programmausführung.
- SET TALK ON Stellt Antworten auf Befehle auf dem Bildschirm dar.
 OFF Keine Darstellung auf dem Bildschirm.

Mischen von Datensätzen aus zwei Datenbanken mit UPDATE

Daten können aus einer Datenbank in eine andere mit folgendem Befehl übertragen werden:

```
UPDATE FROM <datenbank> ON <schlüssel> [ADD <feldliste>]
[REPLACE <feldliste>]
```

Beachte: Beide Dateien müssen nach dem "schlüssel" vorsortiert sein, bevor dieser Befehl ausgeführt wird.

Beide Dateien werden auf den Anfang "zurückgespult", dann werden die Schlüssel-Felder verglichen. Wenn sie identisch sind, dann werden die Daten aus der nach FROM angegebenen Datei entweder zu der in USE befindlichen Datei addiert (ADD) oder sie ersetzen diese (REPLACE), und zwar für die in der "feldliste" angegebenen Datenfelder. Wenn die Felder nicht übereinstimmen, dann werden diese Datensätze übersprungen. Dieser Befehl kann zum Beispiel benutzt werden um eine Inventur-Datei auf den neuesten Stand zu bringen.

Zusammenfügen ganzer Datenbanken mit JOIN

JOIN ist einer der leistungsfähigsten Befehle von dBASE II. Er kann zwei Dateien kombinieren (die im Vordergrund (PRIMARY) und Hintergrund (SECONDARY) bearbeiteten Datenbanken) und daraus eine dritte Datenbank bilden. Die Form der Anweisung lautet:

```
JOIN TO <neuedatei> ON <ausdruck> [ FIELD <liste> ]
```

Der Befehl positioniert dBASE II auf den ersten Datensatz in der Vordergrund-Datei (PRIMARY) und wertet jeden Datensatz in der Hintergrund-Datei (SECONDARY) aus. Jedesmal, wenn der "ausdruck" wahr ist, wird ein Datensatz in die Datei "neuedatei" geschrieben. Falls Sie gerade im Vordergrund arbeiten, wenn Sie den Befehl JOIN ausführen lassen, dann stellen Sie allen Variablen-Namen der Hintergrund-Datei ein "S." voran. Wenn Sie im Hintergrund arbeiten, fügen Sie allen Variablennamen der Vordergrund-Datei den Präfix "P." an (siehe Beispiel 5.1 unten).

Sobald jeder Datensatz in der Hintergrunddatei für den ersten Datensatz in der Vordergrund-Datei ausgewertet worden ist, schreitet das System zum zweiten Datensatz der Vordergrunddatei und wertet erneut alle Datensätze der Hintergrunddatei aus. Das setzt sich so lange fort, bis alle Datensätze jeder Datei mit jedem Datensatz der anderen in Bezug gesetzt worden sind.

Beachte: Das kann eine sehr große Zeit in Anspruch nehmen, wenn beide Dateien groß sind. Es kann auch sein, daß die Operation nicht bis zum Ende ausgeführt werden kann, wenn die im Ausdruck angegebene Bedingung zu schwach ist. Wenn der Ausdruck stets wahr ist, so ergeben zwei Dateien mit je 1000 Datensätzen eine neue Datei mit 1 000 000 Einträgen, das geht aber nicht, weil dBASE II maximal 65 535 Datensätze in einer einzelnen Datei zuläßt.

Erproben Sie die Anweisung mit folgenden Befehlen:

Beispiel 5.1:

```
. select primary          (Vordergrund-Datei)
. use namen              (Datei aufrufen)
. select secondary       (Hintergrund-Datei)
. use ausgaben           (Datei aufrufen)
. select primary         (In Vordergrund schalten)
. join to kundausg for   s.kunde field name,
                        s. auftr nr., s.beschr, s.betrag
. use kundausg           (neue Datei aufrufen)
. list                  (anschauen...)
```

00001 Freitag, Johanna	103 Rundkopfschrauben	192.80
00002 Freitag, Johanna	103 Zylinderkopfschraub.	210.13

Wir benutzen die beiden Dateien "namen.DBF" und "ausgaben.DBF". Nun stellen wir uns die Frage: Welche Kunden, deren Adresse und Kundennummer in "namen" gespeichert sind, haben einen Eintrag in unserer "ausgaben"-Datei. Hier sind es nur zwei, weil nur bei einer Kundin der Kundencode übereinstimmt. Wir haben uns hier gezielt nur für den Namen des Kunden und den Namen und Preis des Artikels interessiert.

Dadurch wird die neue Datei "Kundcod.DBF" mit vier Feldern erzeugt: Kunde, Auftragsnummer, Artikel-Beschreibung, und Betrag. Die Struktur dieser Felder (Datentyp und Länge) ist die gleiche wie in den beiden kombinierten Dateien. (Beachten Sie, daß der Vorsatz "S." verwendet wurde, um eine Variable aus dem nicht angewählten Arbeitsbereich aufzurufen).

Vollständige Bildschirm-Formatierung und Editiermöglichkeiten mit (@..SAY..GET..PICTURE)

dBASE II besitzt eine Reihe leistungsfähiger Formatier-Befehle, die es Ihnen erlauben, Informationen exakt dahin zu setzen, wo Sie sie haben wollen. Sie haben dies bereits in unseren Programmen "Beispiel.CMD" und "Zweidat.CMD" in Aktion gesehen, wo wir gesagt haben:

```
@ <koordinaten> SAY ['nachricht'] GET <variable>
```

Diese Anweisung konnte Nachrichten und Variablen (und ihre Werte) an jede angewiesene Stelle auf dem Bildschirm positionieren. Wenn wir mehrere solche Befehle aufgelistet hatten und dann die Anweisung READ folgen ließen, konnten wir damit ein Formular auf dem ganzen Bildschirm bearbeiten. Vielleicht wollen Sie folgenden Programmausschnitt eingeben und laufen lassen, um Ihre Erinnerung aufzufrischen:

Beispiel 5.2:

```
set talk off (Ausgabe der Variablen-Initialisierung unterdrücken)
store " " to mdatum (Variable initialisieren, 8 Blanks)
store 0.0 to mbilanz (d.h. erzeugen)
store 0.0 to mzahlung
store T to mauswert (logische Variable erzeugen)
@ 5, 5 say "Gib das Datum in der Form TT/MM/JJ ein " get mdatum
@ 10, 5 say "Wie lautet die Bilanz? " get mbilanz
@ 15, 5 say "Wieviel ist zu zahlen? " get mzahlung
read
erase
@ 5, 5 say "Soll dies ausgewertet werden (Ya,Nein)?" get mauswert
read (*** Y oder N ! ***)
```

Der Befehl kann auch ohne die SAY-Ergänzung (to say = sagen) als

```
@ <koordinaten> GET <variable>
```

(zusammen mit einem später im Programm vorkommenden READ) benutzt werden. Dann werden nur die Doppelpunkte zur Anzeige der Feldlänge der Variablen (und der bisherige Inhalt) dargestellt.

Hinweis: Im Bildschirm-Modus (SCREEN) müssen die Zeilennummern nicht in der richtigen Reihenfolge stehen, aber es ist eine gute Praxis, sie in entsprechender Weise zu schreiben, denn im Druck-Modus (PRINT) müssen sie in der richtigen Reihenfolge stehen (da der Drucker ja nicht rückwärts fahren kann).

Diese Anweisung kann auch erweitert werden, um besondere Ein/Ausgabe-Formate zu erzielen:

@ <koordinaten> SAY [ausdruck] GET <variable> [PICTURE <format>]

Die wahlweise Angabe von PICTURE <form> wird mit den unten aufgeführten Format-Symbolen ausgefüllt. Der Befehl:

@ 5,1 SAY "Heute ist der" GET Datum PICTURE '99/99/99'

würde die folgende Anzeige ergeben:

Heute ist der: / / :

wenn wir annehmen, daß die Datums-Variable noch keinen Inhalt hatte. In diesem Beispiel können nur Ziffern eingegeben werden, was durch die "9"en nach PICTURE angezeigt wurde. Drückt man Tasten mit irgendwelchen anderen Zeichen, so hat dies keine Wirkung - sie erscheinen "gesperrt". Das ist ein sehr wirksamer Mechanismus zum Schutz vor Eingabefehlern.

Weitere Zeichen und ihre Bedeutung in dem "form"-Feld sehen Sie in der nachstehenden Tabelle:

Tabelle 5.1:

9 oder #	läßt nur Ziffern als Eingabe zu.
A	läßt nur alphabetische Zeichen zu.
!	konvertiert Zeichen zu Großbuchstaben.
X	erlaubt beliebige Zeichen.
\$	schreibt ein Dollar-Zeichen auf den Bildschirm.
*	schreibt einen Stern auf den Bildschirm.

Ändern Sie das Programm aus Beispiel 5.2 so, daß es Tabelle 5.1 entspricht und überzeugen Sie sich durch einen Versuch von der geänderten Arbeitsweise. Der Cursor springt selbsttätig über die Hilfszeichen wie "/" oder "." (Dezimalpunkt) hinweg. Um zum Beispiel die Zahl 9.99 als Geldbetrag einzugeben, füllt man die leeren Positionen am Anfang des Feldes mit Nullen oder Leerzeichen auf, man kann auch die Cursor-Befehle benutzen, um an die richtigen Eingabe-Positionen zu gelangen.

Mit <CTL>-E kann auch auf ein bereits eingegebenes Feld zurückgesprungen und die Eingabe erneut korrigiert werden.

Mit diesem Befehl können Sie ohne große Mühe Ihre Menüs und Eingabeformulare nach Belieben gestalten.

Hinweis: Die Buchreihe von Osborne über Finanzsysteme enthält, neben der Beschreibung einiger recht ausgeklügelter Systeme, Entwürfe von Bildschirm-Masken für Menues und Eingabe-Formulare, die übersichtlich mit Koordinaten beschriftet sind. Allein deswegen sind sie ihr Geld wert.

Eine sehr viel ausführlichere Darstellung dieser und der im folgenden beschriebenen Formular-Anweisungen finden Sie im 3. Kapitel des Handbuches.

Formatieren von Druckseiten mit SET FORMAT TO PRINT,

@ ... SAY ... USING

Nach der Anweisung SET FORMAT TO PRINT schickt der '@'-Befehl seine Anweisungen zum Drucker statt auf den Bildschirm.

Die GET und PICTURE-Anweisungen werden ignoriert, und der Befehl READ kann nicht benutzt werden, da Sie ja auf dem Drucker nichts eingeben können.

Daten, die auf Schecks, Kaufaufträge, Rechnungen oder andere genormte Formulare gedruckt werden sollen, können zuerst auf dem Bildschirm entworfen werden und dann exakt so ausgedruckt werden, wie sie dort zu sehen sind:

@ koordinaten SAY <variable/ausdruck/string> [USING format]

Für den Druck müssen die Koordinaten geordnet sein. Die Zeilen müssen in aufsteigender Reihenfolge angegeben werden (die Zeile 7 wird vor der Zeile 9 gedruckt usw.). Auf einer bestimmten Zeile müssen die Spalten in der entsprechenden Reihenfolge angegeben werden (Spalte 15 wird vor Spalte 63 gedruckt usw.).

Dieser Befehl kann den aktuellen Wert einer Variablen, die Sie benennen, das Ergebnis eines Ausdrucks oder einen wörtlich angegebenen String ausgeben.

Wenn der Zusatz USING <formatangabe> angefügt wird, dann definiert diese Anweisung, welche Zeichen gedruckt werden und wo sie auf dem Papier erscheinen. Die verwendeten Symbole sind:

Tabelle 5.2:

9 oder #	druckt nur Ziffern.
A	druckt nur alphabetische Zeichen.
X	druckt alle druckbaren Zeichen.
\$	druckt eine Ziffer oder ein "\$" an Stelle einer führenden NULL
*	druckt eine Ziffer oder einen "*" an Stelle einer führenden Null.

Der Befehl "@ 10,50 SAY Stunden * Satz USING '\$\$\$\$\$\$.99' " kann sowohl auf dem Bildschirm als auch dem Drucker benutzt werden, da er kein GET enthält. Für Stunden = 8 und Satz = 12.73 würde er die Ausgabe \$\$\$101.84 erzeugen, was beim Drucken von Schecks nützlich ist, die nicht so leicht verfälscht werden können.

Entwurf und Druck eines Formulars

Um ein Formular zu entwerfen, verwenden Sie Maße, die auf den Abständen beruhen, mit denen Ihr Drucker arbeitet (unserer druckt 10 Zeichen pro Zoll in der Zeile, 6 Zeilen pro Zoll vertikal. Manche Drucker lassen sich auch auf unterschiedliche Werte einstellen).

Das "Menü für Kontoführung" (in Beispiel 3.11), das wir in unserem früheren Programm benutzt haben, könnte leicht eine weitere Alternative erhalten mit der Bezeichnung "4 = Schreibe Scheck". Wir wollen einen Teil des Scheck-Schreibe-Programms entwerfen.

Zunächst müssen wir das Datum eingeben. Die folgenden Befehlszeilen akzeptieren ein Datum und speichern es in einer Variablen namens Mdatum, wobei sie prüfen, ob es (vermutlich) korrekt ist:

Beispiel 5.4:

```

erase
set talk off
store " " to mdatum      (Achten Sie darauf,
                          (daß es genau 8 Leerzeichen sind!))

store T to keinDatum
do while keinDatum
@ 5,5 say "Gib Datum ein TT.M/JJ" get mdatum picture "99/99/99"
read
if val ($ (mdatum,1,2)) < 1;
  .OR. val ($ (mdatum,1,2)) > 31;
  .OR. val ($ (mdatum,4,2)) < 1;
  .OR. val ($ (mdatum,4,2)) > 12;
  .OR. val ($ (mdatum,7,2)) < > 82
  store " " to mdatum
  @ 7, 5 say "**** Ungültiges Datum, bitte neu eingeben! ****"
  store T to keinDatum
else
  store F to keinDatum
endif
enddo ..da wir jetzt ein gültiges Datum haben
erase

```

Auf Deutsch beschrieben macht Beispiel 5.4 dies: Zuerst wird die Variable MDatum mit 8 Leerzeichen gefüllt, dann schreibt die Anweisung @.Y:

Gib Datum ein TT/MM/JJ: / / :

Sobald das Datum vom Benutzer eingetippt wurde, wird es in der IF-Anweisung darauf überprüft, ob sich die Angabe des Tages im Bereich 1-31 bewegt, die des Monats in 1-12 und die des Jahres gleich 82 ist. Das geschieht in drei Schritten:

- die Teilstring-Funktion \$ ermittelt die beiden Zeichen, die den Tag, den Monat oder das Jahr vertreten (beim Monat zum Beispiel fängt sie auf der 5. Stelle an und nimmt zwei Zeichen).
- die Funktion VAL verwandelt diese in eine (ganze) Zahl.
- diese Zahl wird mit den erlaubten Werten verglichen.

Wenn der Wert außerhalb des zulässigen Bereiches liegt, dann wird "MDatum" wieder mit Leerzeichen gefüllt und eine Fehlermeldung erscheint. Wenn ein zulässiges Datum eingegeben wird, fährt das Programm fort.

Das eigentliche Ausdrucken des Schecks könnte den nächsten Abschnitt des Programms bilden. Unter Verwendung der Maße amerikanischer Scheckformulare ergibt sich folgende Befehlsliste:

Beispiel 5.5:

```
@ 8,3 SAY Script * Eine Zeichen-Variable, welche den Betrag
                  * in Script druckt. Das wird durch eine andere Prozedur ausgefüllt. Wir könnten
                  * sie hier etwa so simulieren:
                  *   STORE 'Script Dummy' TO Script
                  *   RETURN
@ 11,38 SAY Verk:N
@ 11,50 SAY MDatum
@ 11,65 SAY Betrag
@ 13,10 SAY Verkäufer
@ 14,10 SAY Adresse
@ 15,10 SAY Pleit:Zahl
@ 15,15 SAY Stadt
@ 17,10 SAY Wer
```

Sie können das auf Ihrem Bildschirm ausprobieren. Bevor Sie es drucken, schalten Sie mit der Anweisung "SET FORMAT TO PRINT" vom SCREEN (Bildschirm)-Modus zum PRINT (Druck)-Modus um. Die Werte für die Variablen wurden an anderer Stelle in Ihrem Programm eingesetzt.

Auch große Formulare bieten keine Schwierigkeiten: Eine Druck-Seite kann bis zu 255 Zeilen lang sein. Um den Zeilen-Zähler zurückzusetzen fügen Sie einen EJECT-Befehl ein (wenn der Drucker angesteuert wird).

Rückblick

dBASE II ist ein sehr leistungsfähiges System, das eine große Anzahl von Befehlen und Techniken besitzt, um Ihre Datenbanken zu verwalten. Es erlaubt Ihnen so, mehr Informationen auf leichtere Weise zu gewinnen als kaum ein anderes gegenwärtig auf Mikrocomputern lauffähiges Datensystem.

Am einfachsten lernen Sie diese Techniken, indem Sie die Beispiele durcharbeiten, Sie in Ihren Computer eintippen, wobei Sie Namen usw. in dem Maße verändern, wie Sie sich auf Ihre speziellen Anforderungen besinnen.

Sie fangen mit dem Befehl CREATE (ERZEUGE) an, um Ihre Datenbank zu erzeugen. Neben Ausgaben.DBF haben wir eine Anzahl weiterer Datenbankstrukturen entworfen, die Ihnen vielleicht nützlich erscheinen. Sie finden sie im 4. Kapitel.

Beim Schreiben dieser Programme haben wir uns genau an das Verfahren gehalten, das wir an früherer Stelle vorschlugen: Beschreiben Sie zuerst das Problem in groben Zügen. Dringen Sie dann schrittweise zu immer feineren Details vor, wobei Sie zunächst gewöhnliches Deutsch zur Formulierung benutzen, dann Pseudocode, wobei Sie Begriffe einfließen lassen, die dBASE II tatsächlich verstehen kann, sobald das Programm entsprechend ausgearbeitet ist.

Wenn wir auf irgendetwas gestoßen sind, das vom Programm getan werden mußte, wovon wir aber noch nicht genau wußten, wie, haben wir einfach den Namen einer Prozedur (eines Unterprogrammes) daraus gemacht, um später zur Ausarbeitung dieses Problems zurückzukehren.

Die Einrückungen und das System der gemischten Groß- und Kleinschreibung wurden nicht eigens für dieses Handbuch entworfen, es ist die Art und Weise, wie wir stets arbeiten. Es macht das Schreiben von Programmen viel einfacher, weil zusammengehörige Strukturen sofort ins Auge fallen.

Die Bezeichner (Namen) wurden aus unserem halb-Englischen Pseudocode entwickelt, mit ein bißchen Verfeinerung, damit sie sich in die erlaubten 10 Zeichen pro Namen fügen, aber so, daß sie ihre Bedeutung noch erkennen lassen.

Kommentare wurden zur Dokumentation in den Programmen verstreut, obwohl sich die Programme in vielen Fällen selbst dokumentieren, da die meisten dBASE II Befehle englischem Klartext ähneln (für den deutschen Leser wurden bisweilen "Eindeutschungen" der Befehle als zusätzliche Kommentare eingefügt).

Abschnitt 6: ZIP - der Bildschirm- und Druck-Maskengenerator

Inhaltsverzeichnis

Welche Möglichkeiten bietet ZIP.....	175
Anpassen von ZIP an Ihren Rechner/Ihr Terminal.....	176
Wenn Ihr Terminal nicht aufgeführt ist.....	176
Einstellen einer benutzerspezifischen Terminalanpassung.....	178
Auswechseln von Befehlsabkürzungen und Markierungssymbolen.....	179
Kurzer Überblick, was ZIP kann.....	181
Arbeiten mit ZIP.....	183
Zeichnen von horizontalen/vertikalen Linien.....	185
Hilfstafel, Standard-Werte verändern.....	186
Abspeichern Ihres Entwurfs in einer Datei.....	187
Beginnen eines neuen Entwurfs, Quit.....	189
Nie wieder @ ... SAY ... GET programmieren.....	190
Einfügen von dBASE II-Anweisungen in die Formular-Tafeln.....	191
Probleme mit kurzen Feldern und langen Namen.....	192
Zusammenfassung der ZIP-Befehle.....	193
Dynamische Voreinstellungen (Standard-Werte).....	194
Übrige ZIP-Befehle und Symbole.....	194
Display-Unterprogramme für ZWEIDAT.CMD (Beispiel 4.14).....	196

Welche Möglichkeiten bietet ZIP?

ZIP vereinfacht das Erzeugen von Bildschirm- und Drucker-Formularen. Dieses Hilfsprogramm besitzt folgende Eigenschaften:

- Sie können Ausgabeformulare für Ihren Drucker sowie Bildschirmmasken für Ein/Ausgabe von Daten erzeugen. ZIP vermag bis zu 88 Zeilen zu verwalten.
- ZIP erzeugt automatisch lauffähigen Code: Es schreibt READ an das Ende jedes Programms, das GET-Anweisungen benutzt, es fügt in größeren Programmen nach jeweils 64 GET ein READ ein, überprüft, ob Variablen-Namen korrekt geschrieben sind, erzeugt ".FMT"-Dateien oder schreibt selbsttätig vollständige ".CMD"-Dateien mit den Anweisungen ERASE oder SET FORMAT TO PRINT/SCREEN, SET MARGIN TO xx und RETURN.
- Sie können beliebige dBASE II - Anweisungen auf den Bildschirm-Tafeln einfügen und so mit dem ZIP-Editor Unterprogramme sowohl für formatierte Ausgabe/Eingabe, als auch mit Verarbeitungsfunktionen der behandelten Daten erzeugen.
- Die Dialogzeile (eine Zeile, die am unteren Bildschirmrand dargestellt wird), sagt Ihnen jederzeit, in welcher Zeile und Spalte sich der Cursor befindet.
- Sie können die Zeichen, mit denen horizontale und vertikale Linien markiert werden, Tabulator-Abstände, Seitenlänge und Randeinstellung für den Drucker während des Erarbeitens eines Formulars dynamisch verändern.
- Per Tastendruck können Sie horizontale und vertikale Linien zeichnen und löschen und so leicht Kästen oder Spalten- und Zeileneinteilungen erzeugen sowie ganze Spalten oder Zeilen löschen.
- Die Bedienung ist rasch zu erlernen, Sie brauchen keine CONTROL-Tasten zu betätigen, sondern können die jeweiligen Tasten Ihres Rechners für <RETURN>, <ZEILEN-VORSCHUB>, <DELETE>, <BACKSPACE>, <TAB> und die Pfeiltasten für Cursorbewegungen mit genau den erwarteten Funktionen benutzen.
- Sie können die Voreinstellungen von ZIP so beeinflussen, daß Sie ZIP ganz Ihren persönlichen Bedürfnissen anpassen können

Für den englischen Begriff ZIP Talker (TM) wird in diesem Handbuch durchgehend das Wort Dialogzeile verwendet.

Anpassen von ZIP an Ihren Rechner

Die Vertriebsdiskette, auf der Sie dBASE II erhalten haben, enthält die Files ZIP.COM und ZIPIN.COM.

Setzen Sie eine Diskette mit einer Kopie von ZIP.COM in Ihr eingerastetes Laufwerk (logged in). Das Anpassungsprogramm ZIPIN.COM kann sich in einem beliebigen Laufwerk befinden. Geben Sie ein:

<Laufwerk>ZIPIN

Wenn Ihr Terminal/Rechner auf der ersten Tafel, die angezeigt wird, aufgezählt ist, geben Sie die entsprechende Zahl ein, danach die <RETURN> oder <ENTER>-Taste. Drücken Sie noch einmal <RETURN>, um Ihre Wahl zu bestätigen, schreiten Sie dann zum nächsten Abschnitt fort, in dem Sie die ABKÜRZUNGEN UND SYMBOLE VERÄNDERN können.

Hinweis: Das Hazeltine 1500-Terminal u.a. wird mit den verflixten CONTROL-Tasten-Funktionen eingerichtet, da es keinerlei Pfeiltasten zur Cursor-Bewegung besitzt. Die Funktionen sind aber die gleichen wie in dBASE II:

Aufwärts:	<CTL>-E
Abwärts:	<CTL>-X
Links:	<CTL>-S
Rechts:	<CTL>-D

Wenn Sie irgendeine andere Kombination verwenden wollen, lesen Sie bitte aufmerksam die folgenden Anleitungen, wie man ein benutzerspezifisches Terminal anpaßt.

Besonderer Hinweis: Wenn Ihr Terminal die entsprechenden Funktionen besitzt, so stellen Sie sicher, daß die Funktion AUTO SCROLL eingeschaltet (z. B. TRS-80 II mit P & T) und LOCAL ECHO ausgeschaltet ist (z. B. HP 125).

Wenn Ihr Terminal nicht in der Aufzählung vorkommt, dann müssen Sie die nötigen Folgen von Steuerzeichen für diese Funktionen in Erfahrung bringen:

- Löschen des Bildschirms, wobei der Cursor nach links oben springt.
- Direkte Cursor-Positionierung (und notwendiger Koordinaten-Versatz)
- Pfeiltasten

Beachten Sie das Wort "Folgen" (von Zeichen). Die meisten Terminals besitzen einige Tasten, bei deren Betätigung mehrere Zeichen ausgesendet werden. "CLEAR" (Löschen) gibt wahrscheinlich eine Folge von mindestens zwei Zeichen aus, eines, um den Bildschirm zu löschen, und das andere, um den Cursor auf die "HOME"-Position ("zu Hause", d.h. links oben) zu schicken. Auch die Pfeiltasten können mehrere Zeichen aussenden. Diese Folgen müssen als Zeichenfolgen eingegeben werden, sie können nicht durch einfaches Drücken der entsprechenden Taste angepaßt werden.

Lesen Sie im Handbuch Ihres Terminals (oder Computers) nach und geben Sie dann die Anzahl Zeichen in der Folge sowie die einzelnen Zeichen ein, sobald ZIPIN danach fragt.

Während der Anpassungs-Prozedur akzeptiert ZIPIN nur die Zeichen in einer Folge, nicht die Codes, durch welche sie dargestellt werden. Wenn Sie in Ihrem Handbuch die Codes in dezimaler oder hexadezimaler Form oder per Namen verzeichnet finden, so benutzen Sie bitte die folgende Tabelle, um sie zu übersetzen:

Tabelle 6.1:

Wenn angegebener Code:				Wenn angegebener Code:			
Dezimal	Hex	Name	Taste	Dezimal	Hex	Name	Taste
0	0	NUL	CTL-@ *	16	10	DLE	CTL-P
1	1	SOH	CTL-A	17	11	DC1	CTL-Q
2	2	STX	CTL-B	18	12	DC2	CTL-R
3	3	ETX	CTL-C	19	13	DC3	CTL-S
4	4	EOT	CTL-D	20	14	DC4	CTL-T
5	5	ENQ	CTL-E	21	15	NAK	CTL-U
6	6	ACK	CTL-F	22	16	SYN	CTL-V
7	7	BEL	CTL-G	23	17	ETB	CTL-W
8	8	BS	CTL-H/BS	24	18	CAN	CTL-X
9	9	HT	CTL-I/TAB	25	19	EM	CTL-Y
10	A	LF	CTL-J/LF	26	1A	SUB	CTL-Z
11	B	VT	CTL-K	27	1B	ESC	CTL-Ä *
12	C	FF	CTL-L	28	1C	FS	CTL-**
13	D	CR	<RETURN>	29	1D	GS	CTL-Ü*/ESC
14	E	SO	CTL-N	30	1E	RS	CTL-I
15	F	SI	CTL-O	31	1F	US	CTL-__

***Anmerkungen:** Auf deutschen Tastaturen finden Sie unter Umständen hier ein Paragraph-Zeichen (@), auf den amerikanischen ist es der sog. "Klammeraffe" (@). * steht für "BACKSLASH" (\), das ist ein rückwärts gerichteter Schrägstrich, umgekehrt wie "/", d.h. nach links geneigt statt nach rechts. "Ü" bzw. "Ä" steht für die sich schließende bzw. öffnende eckige Klammer ("]" bzw. "[").

Um ein Beispiel zu nennen: Um das Zeichen "ESCAPE" einzugeben (in der Tabelle unter dem Namen ESC) drücken Sie entweder die Taste <ESC> auf Ihrer Tastatur oder "<CTL>-Ä". Um ein CONTROL-Zeichen einzugeben, halten Sie die mit "CTRL" (oder ähnlich, manchmal auch als "ALT") bezeichnete Taste niedergedrückt, während Sie den bezeichneten Buchstaben oder die Graphik-Taste drücken.

Die Taste für Wagenrücklauf <CR> oder <ENTER> oder <RETURN> darf in einer Steuerzeichenfolge nicht verwendet werden.

Einstellen einer benutzerspezifischen Terminal-Anpassung

Um eine benutzerspezifische Terminal-Anpassung einzustellen, wählen Sie bitte "0" (Null) als Ihr Terminal aus dem Menue, dann drücken Sie <RETURN>, um Ihre Wahl zu bestätigen.

Geben Sie zur Antwort auf die folgende Meldung die Länge der Zeichenfolge ein, welche Ihr Terminal für CLEAR SCREEN AND HOME CURSOR (Löschen des Bildschirms und Cursor nach links oben) benötigt, drücken Sie danach <RETURN> oder <ENTER>.

Geben Sie nun die Zeichen der Zeichenfolge ein. Um die einzelnen Zeichen einzugeben drücken Sie einfach auf die in der Tabelle 1 angegebenen Tasten, wenn ZIPIN danach fragt. Sie können Buchstaben als Groß- oder Kleinbuchstaben eingeben - ZIPIN wandelt sie in jedem Fall in Großbuchstaben um.

Sobald die Zeichenfolge eingegeben ist, gibt Ihnen ZIPIN die Gelegenheit, sie zu korrigieren. Wenn sie korrekt ist, drücken Sie einfach <RETURN>.

Sobald Sie die Steuerfolge für CLEAR SCREEN AND HOME CURSOR eingegeben haben führt ZIPIN einen Test durch.

Falls der Bildschirm nicht gelöscht wird, wobei die nächste Meldung nahe dem oberen Bildschirmrand erscheint, so sehen Sie bitte im Handbuch Ihres Terminals nach. Es kann sein, daß ein vorausgehendes Zeichen (lead in character) erforderlich ist, welches Sie nicht eingegeben hatten.

Falls der Bildschirm gelöscht wird, geben Sie "Y" (Yes, für JA) ein. ZIPIN wird Sie dann nach der Zeichenfolge für die Positionierung des Cursors fragen, die Größe des Versatzes (OFFSET, manchmal BIAS genannt, die Zahl, die zur gewünschten Cursor-Position hinzuaddiert werden muß) und nach der Zeichenfolge, die von Ihren Pfeil-Tasten ausgegeben wird.

Wenn Ihre Tastatur keine Pfeil-Tasten besitzt, wollen Sie vielleicht die Codes eingeben, die auch in dBASE II benutzt werden (<CTL>-E für Aufwärts, <CTL>-X für Abwärts, <CTL>-S für Links und <CTL>-D für Rechts).

Sie werden auch gefragt, ob Ihre Tastatur eine besondere Taste für löschendes BACKSPACE (erasing Backspace) besitzt, d.h. ein Zurückfahren des Cursors, wobei das jeweils letzte Zeichen gelöscht wird. Einige Terminals geben das BACKSPACE-Signal mit der Links-Pfeil-Taste aus, wobei der Cursor zurückgefahren wird, ohne das Zeichen zu löschen, auf dem er steht. Wenn dies auf Ihr Terminal zutrifft, so können Sie entweder mit "N" antworten, dann arbeitet ihre Linkspfeil-Taste wie gewohnt, ohne Zeichen zu löschen, oder mit "Y" (Yes = Ja), dann löscht sie bei der Bewegung des Cursors. Dieses Problem stellt sich nicht, wenn Sie zwei Tasten haben, die verschiedene Codes ausgeben: z.B. "Links-Pfeil" und <RUBOUT>.

Einige Terminals benötigen nach der Aussendung der CLEAR und HOME-Signale eine Ruhepause, sonst werden die ersten paar Dutzend Zeichen am Anfang des Bildschirms nicht angezeigt. Die meisten Terminals kommen ohne diese Verzögerung aus (das Hazeltine 1500 benötigt sie) und ZIP läuft schneller, wenn keine Ruhezeit angegeben wird. Wenn Sie sich nicht sicher sind, passen Sie ZIP zuerst ohne die Verzögerung an. Sie können die Anpassung wiederholen, wenn eine Verzögerung erforderlich ist.

Auswechseln von Befehls-Abkürzungen und Markierungs-Symbolen

Sobald Sie die Cursor-Steuerung eingerichtet haben, löscht ZIPIN den Bildschirm und zeigt alle Befehle, die Sie während der Verwendung von ZIP aufrufen können, um Bildschirm-Masken und Formulare zu gestalten.

Um irgendwelche der Befehls-Abkürzungen oder Symbole zu verändern, geben Sie bitte "C" ein und drücken dann die Taste, die Sie tauschen wollen, gefolgt von der Taste, die Sie stattdessen benutzen wollen.

Um beispielsweise das Kennzeichen für Befehle zu ändern:

- Drücke C
- Drücke /
- Drücke \$ (Wenn Sie es zum Dollarzeichen ändern wollen)

Beachte: Die eckigen Klammern benutzt ZIP als String-Klammern in dBASE-Programmdateien (statt der Anführungszeichen), sie dürfen daher nicht als neue Symbole verwendet werden.

Sie werden das Zeichen, mit dem ein ZIP-Befehl eingeleitet wird, sicherlich zu einem solchen Zeichen verändern wollen, das Sie leicht eingeben können, aber sonst zur Gestaltung von Bildschirm- oder Druckformularen nicht benötigen (z.B. den rückwärts gerichteten Schrägstrich "/").

Beachte: *Setzen Sie nie* zwei Befehle auf das gleiche Symbol: Das würde ZIP völlig durcheinander bringen. Wenn Sie dies aus Versehen getan haben, so können Sie es wieder in Ordnung bringen, indem Sie die Anpassungsprozedur ZIPIN wiederholen.

Die Einträge am Fuß des Menues können während der Anpassungs-Prozedur verändert werden, sie können aber ebenso gut dynamisch während einer Sitzung mit ZIP anders eingestellt werden (unter der Überschrift "CHANGEABLE DYNAMIC VALUES").

Gleichgültig, welche Standard-Werte Sie jetzt einstellen, Sie können die Tabulator-Abstände, die Zeichen für horizontale und vertikale Linien, die Seitenlänge und die Randeinrückung auf dem Drucker so oft verändern wie Sie wollen, wenn Sie später mit ZIP Bildschirmmasken und Druckerformulare erzeugen.

Um die Voreinstellung zu beeinflussen, geben Sie während der Anpassungs-Prozedur folgendes ein:

		(alt)	(neu)		
Drücke:	C	5	9	verändert	Tabulator-Abstand
Drücke:	C	.	!	" "	senkrechten Markierer
Drücke:	C	-	*	" "	waagrechten Markierer
Drücke:	C	?	40	" "	Drucker-Randeinrückung

Drücken Sie <F>, wenn Sie alle gewünschten Veränderungen vorgenommen haben, um den Vorgang zu beenden.

Beispiel 6.1:

***** File BEISPIEL ***

```

+-----+
! Beispiel !
+-----+

```

Beispiel Auftrag

Auftrag Nr.: @auftr:nr Datum: #rech:dat

Kunde: #kunde

```

[use ausgaben]#[store 0 to sum]
[do while .not. eof]@[? auftr:nr, beschr, betrag]#[store sum + betrag to sum]
[skip]
[enddo]

```

* BEISPIEL.CMD

ERASE

@ 0,33 SAY "+-----+"

@ 1,33 SAY "! Beispiel !"

@ 2,33 SAY "+-----+"

@ 5,10 SAY "*****"

@ 6,12 SAY "Beispiel Auftrag"

@ 7,10 SAY "*****"

@ 11, 9 SAY "Auftrag Nr.:"

@ 11,22 SAY auftr:nr

@ 11,40 SAY "Datum:"

@ 11,47 GET rech:dat

@ 13,15 SAY "Kunde:"

@ 13,22 GET kunde

@ 16, 0 SAY "-----"

use ausgaben

store 0 to sum

do while .not. eof

? auftr:nr, beschr, betrag

store sum + betrag to sum

skip

enddo

@ 38, 0 SAY "-----"

READ

RETURN

Sie können das Programm ausführen, indem Sie dBASE starten, zunächst "USE ausgaben" eingeben (um die Variablen bekannt zu machen) und dann "DO beispiel". Aber es ist nicht sehr sinnvoll, was es tut. Am Ende dieses Abschnittes zeige ich Ihnen zwei sinnvollere Beispiele.

Kurzer Überblick, was ZIP kann

ZIP hat das auf der vorhergehenden Seite wiedergegebene Programm aus der darüber abgebildeten Bildschirmeingabe erzeugt.

Das Kästchen wurde mit Befehlen gezeichnet, die waagrechte und senkrechte Linien schreiben und löschen, wobei automatisch an jedem Schnittpunkt ein "+" eingefügt wird. Die Markierungssymbole für beide Linienarten können während der Sitzung ausgetauscht werden.

Der Text und die Variablennamen (mit vorangestelltem @ oder #) und die eingefügten Befehle (in eckige Klammern eingeschlossen) wurden einfach eingetippt, ohne daß es nötig gewesen wäre, irgendwelche <CTL>-Tasten zu betätigen. Sobald das Formular so aussieht, wie der Benutzer es sich wünscht, drückt er "/S" und ZIP macht sich an die Arbeit.

ZIP stellt mit "SET FORMAT TO PRINT" die Ausgabe auf den Drucker ein, da der Benutzer im Dialog die Frage danach mit "Yes" (Ja) beantwortet hat, und es hat den Rand auf den vom Operator gewünschten Wert eingestellt.

Danach tastet ZIP die Bildschirmmaske ab und schreibt automatisch alle nötigen @ <zeile, spalte> SAY <text> - Befehle, d.h. die Ausgabe von Textkonstanten.

Überall wo es auf ein Display-Symbol (@) stößt, schreibt es einen @ <zeile, spalte> SAY <variable> - Befehl, d.h. die Ausgabe einer Variable.

ZIP schreibt auch GET-Anweisungen, obwohl dies ein Drucker-Formular ist (und man vom Drucker nichts eingeben kann). Wenn es auf das GET-Symbol (#) stößt, fragt es, ob dies beabsichtigt war, und wenn der Benutzer mit "Yes" antwortet, arbeitet es weiter. Antwortet er mit "No", so bricht es das Erzeugen des Programmfiles ab und setzt den Cursor auf das unerwünschte Symbol. In diesem Beispiel hat der Operator auch eine Anzahl von in eckigen Klammern geschriebenen dBASE II - Befehlen eingefügt, so daß das sich ergebende Programm eine Kombination von Formular und funktionalem Unterprogramm ist.

Schließlich krönt ZIP sein Werk, indem es die Befehle "SET FORMAT TO SCREEN" und "RETURN" anfügt. Nun kann das Unterprogramm von jedem anderen dBASE II Programm aus aufgerufen werden, indem man einfach den Aufruf "DO SAMPLE" einsetzt.

Die Bildschirmvorlage hätte auch als *Format File (.FMT Namen-Erweiterung)* abgespeichert werden können, indem man den gewünschten Filetyp angibt.

In einen Format-File hätte ZIP lediglich die @ ... SAY-Anweisungen eingefügt sowie alle in eckige Klammern eingeschlossenen Einträge. Die SET-Anweisungen wären nicht erschienen, und der erste Kommentar hätte gelautet: * SAMPLE.FMT.

Beachten Sie aber daß in Format-Dateien nur Kommentare eingefügt werden dürfen. ZIP prüft nicht, ob die eingefügten Anweisungen korrekt sind.

Ab der nächsten Seite werden wir Ihnen zeigen, wie einfach dies alles erreicht werden kann.

Arbeiten mit ZIP

Geben Sie ein:

ZIP

ZIP beginnt damit, Ihnen die Hilfs-Tafel mit der Übersicht über alle Befehle und die eingestellten Standard-Werten auf den Bildschirm zu schreiben.

Es macht nichts, wenn Sie sich nicht alle Befehle merken können. Sie können die Hilfstafel jederzeit aufrufen, indem Sie zweimal die Befehlstaste drücken (// - oder welches Symbol Sie stattdessen eingestellt haben). Drücken Sie eine beliebige Taste, um fortzufahren.

Der Bildschirm sollte nun gelöscht werden und die Dialogzeile von ZIP am unteren Rand des Bildschirms sollte so aussehen:

<NEW> or <OLD> file (Q to Quit)?
(<NEUER> oder <ALTER> File (Q für Abbruch))

ZIP beginnt nicht zu arbeiten, ehe Sie ihm nicht gesagt haben, ob Sie eine neue Datei anlegen wollen (durch Drücken von N) oder eine alte nachbearbeiten (drücken Sie dann O für "old"). Sie können aber auch Ihre Absicht ändern und wieder ins Betriebssystem zurückkehren, indem Sie auf Q drücken. Da wir keinen früher erzeugten File haben, den wir laden könnten, geben Sie ein:

N

Die Dialogzeile fragt Sie nun nach dem Namen, den Sie der neuen Datei geben wollen:

FILE NAME (drive optional):
(DATEI-NAME (eventuell Laufwerk angeben))

Wenn Sie kein Laufwerk bezeichnen (durch einen Buchstaben, z.B. A oder B, gefolgt von einem Doppelpunkt), dann werden Dateien von dem gerade "eingerasteten" Laufwerk geladen und dort wieder abgespeichert. Dateinamen dürfen bis zu 8 Zeichen lang sein (zuzüglich der ev. vorangehenden beiden Zeichen für das Laufwerk) und dürfen die folgenden Zeichen enthalten:

A-Z (a-z), 0123456789, \$@ # / und den Doppelpunkt (:)

ZIP akzeptiert keinerlei andere Zeichen in einem Dateinamen. Das bedeutet, daß Sie selbst keine Namen-Ergänzung (Extension) angeben können. ZIP wird ein Abbild Ihrer Bildschirm-Gestaltung als <name>.ZIP (also mit der Namen-Ergänzung ".ZIP") abspeichern sowie Programme als <name>.CMD.

Alphanumerische Zeichen können als Groß- oder Kleinbuchstaben eingegeben werden - ZIP wandelt sie stets in Großbuchstaben um.

CP/M verlangt den Doppelpunkt als zweites Zeichen des Dateinamens (wenn er die Angabe des Laufwerks enthält). Wenn Sie an einer anderen Stelle einen Doppelpunkt eingegeben haben, wird ZIP den Dateinamen abfangen und Sie auffordern, ihn neu einzugeben.

Sobald Sie den Dateinamen eingegeben und <RETURN> gedrückt haben, springt der Cursor nach 0,0 (links oben) und die Dialogzeile gibt Ihnen seine exakte Position an. Probieren Sie nun die <TAB> (Tabulator-) Taste oder <RETURN> aus. Falls Ihre Tasten eine automatische Repeat-Funktion (Wiederholung des Zeichens durch dauerndes Niederdrücken der Taste) haben, halten Sie sie niedergedrückt. Der Cursor wird Zeile für Zeile nach unten springen, bis er den unteren Bildschirmrand erreicht, wobei die Dialogzeile jeden Schritt durch Angabe der Koordinaten verfolgt. (Auf manchen Tastaturen erreichen Sie das gleiche, indem Sie die Taste gedrückt halten und gleichzeitig eine besondere <REPEAT>-Taste betätigen).

Drücken Sie nun die Taste mit Pfeil nach oben, und der Cursor beginnt zeilenweise nach oben zu springen.

Es gibt aber eine viel schnellere Möglichkeit, um an den oberen Rand zu kommen. Geben Sie die drei folgenden Befehle ein (mit Groß- oder Kleinbuchstaben):

/T (Top = Oben)
/B (Bottom = Unten)
/M (Middle = Mitte)

Diese drei Tasten lassen Sie schnell auf dem ganzen Bildschirm herumfahren. Die Top- (Oben) und Bottom- (Unten) Anweisungen behalten die Spalte, in welcher sich der Cursor gerade befindet, bei. So haben Sie es einfacher, Ihre Einträge spaltenweise auszurichten. Der Mitten-Befehl bringt den Cursor in die Mitte einer Zeile und erleichtert es so, Überschriften zu zentrieren.

Beachten Sie, daß die Dialogzeile sie laufend begleitet. Sobald Sie das Befehls-Symbol (hier mit “/” angegeben) gedrückt haben, läßt Sie sie sogar wissen, daß sie auf die Eingabe eines Befehls wartet (indem Sie das Befehls-Symbol in der untersten Bildschirm-Zeile anzeigt).

Jedesmal, wenn Sie eine neue Sitzung mit ZIP beginnen, wird der Tabulatorabstand auf den Wert eingestellt, den Sie in der Anpassungs-Prozedur angegeben haben (ursprünglich 5), und die Seitenlänge wird auf die Maße Ihres Bildschirms eingestellt. Tabulatorabstand und Seitenlänge können jederzeit während einer Sitzung mit ZIP anders eingestellt werden. Geben Sie die folgenden drei Befehle ein (warten Sie nach der ersten Anweisung auf die Hilfstafel):

//
T
9

(Das Befehls-Symbol wird auf der Hilfstafel nicht gebraucht)

Drücken Sie eine beliebige Taste um zu Ihrem Arbeitsschirm zurückzukehren, dann probieren Sie die Tabulator-Taste <TAB>. Die Dialogzeile zeigt, daß Tabulator-Marken jetzt auf alle Positionen gesetzt sind, die Vielfache von 9 sind. Sie können diesen Wert so oft Sie wollen neu einstellen, indem Sie wieder zur Hilfstafel zurückkehren.

In der gleichen Weise kann die Seitenlänge modifiziert werden. Von der Hilfstafel aus können Sie jede beliebige Seitenlänge von der Größe eines Bildschirms (ZIP-Minimum) bis zu 88 Zeilen wählen. Dies erlaubt es, übliche Masken für den Bildschirm oder Druckformulare mit bis zu 8 Zeilen pro Zoll auf Standard-Papier vorzubereiten.

BEACHTEN: Wenn Sie die Seitenlänge während der Arbeit mit ZIP verkürzen, wird jegliche Information, die sich unterhalb der letzten Zeile des neuen Formates befindet, gelöscht. Sie kehrt auch nach Rückstellen des Formates nicht wieder!

Zeichnen von horizontalen/vertikalen Linien

Bringen Sie nun den Cursor in die linke obere Ecke des Bildschirms (drücken Sie <RETURN>, /T) und geben ein:

```
/H
/V
<zweimal TAB>
/V
<zweimal TAB>
/V
/H
```

Das Zeichnen und "Ausradieren" von waagrechten und senkrechten Linien geschieht mit den gleichen Befehlen, wobei automatisch ein "+" eingesetzt oder entfernt wird, wo sich zwei Linien kreuzen. Wenn sich der Cursor auf einem Symbol für die Markierung horizontaler Linien befindet und Sie geben den horizontalen Zeichenbefehl ein ("/H"), so wird der Rest der Linie nach rechts gelöscht. Wenn sich der Cursor auf einem "+"-Zeichen befindet, so reagiert ZIP in unterschiedlicher Weise.

Wenn sich der Cursor auf einem "+" befindet und links davon weitere Zeichen stehen, so bleibt das Pluszeichen erhalten, aber der Rest der Linie wird nach rechts gelöscht. Wenn sich unmittelbar links vom Cursor keine weiteren Zeichen befinden und das nächste Zeichen rechts ist kein "+", dann wird die horizontale Zeile gelöscht und das "+" wird zu dem Zeichen verändert, das gegenwärtig als vertikaler Markierer eingestellt ist.

Wenn das Zeichen von weiteren “+”-Zeichen umgeben ist, so bleiben sie alle stehen, nur die horizontale Linie jenseits des am weitesten rechts stehenden Pluszeichens wird gelöscht. In diesem Fall zeichnet ZIP jenen Teil der horizontalen Linie nicht neu bis Sie den Cursor auf das letzte “+”-Zeichen rechts bringen.

Das Zeichnen vertikaler Linien folgt derselben Logik.

Beide Befehle sind viel leichter im Gebrauch als es sich nach dieser Beschreibung anhört, üben Sie ein wenig damit. Wichtig ist, daß man von links nach rechts und von oben nach unten zeichnet, wobei vor allem rechts/unterhalb des Gebietes, in dem man bereits zeichnet, noch kein Text stehen sollte.

HINWEIS: Der zweimalige Gebrauch eines Zeichenbefehls (/H, /H oder /V, /V) ist ein schneller Weg, um eine ganze Zeile oder Spalte zu löschen.

Hilfstafel - Standardwerte verändern

Geben Sie nun die folgenden Befehle ein (warten Sie auf das Erscheinen der Hilfs-Tafel):

```
//
P
40
. (gegenwärtiger Wert für das Vertikal-Symbol)
*
```

Dadurch wird die Seitenlänge auf 40 Zeilen geändert und das Markierungs-Zeichen für senkrechte Linien in ein Sternchen. Kehren Sie durch Druck auf eine beliebige Taste zum Arbeitsschirm zurück. Betätigen Sie <RETURN>, um einige Zeilen nach unten zu gehen. Geben Sie dann /H und /V ein. Anstatt die vertikale Linie zu löschen hat der letzte Befehl alle Zeichen in Sternchen verwandelt. Das geschah, weil der Cursor sich nicht auf dem jetzt eingestellten Symbol für vertikale Markierungen befand.

Nun tippen Sie:

```
/N          (warten Sie auf die neue Bildschirmseite)
/B
```

Damit kommen Sie an den unteren Rand der nächsten Bildschirmseite, wobei die Dialogzeile sagt: “ROW (Reihe) 39, COL (Spalte) 0” (unsere Seitenlänge beträgt 40 Zeilen, nummeriert von 0 bis 39). Beachten Sie, daß die letzte vertikale Linie, die Sie gezeichnet haben, bis zum Ende der Seite geht, obwohl das zuvor auf dem Bildschirm nicht sichtbar war.

Wenn Sie die Seite länger als einen Bildschirm eingestellt haben können Sie folgende Befehle benutzen:

- /F um auf die erste Bildschirmseite (FIRST SCREEN) Ihres Formulars zu kommen, mit Zeile 0 am oberen Bildschirmrand; oder
- /L um auf die letzte (LAST) Bildschirmseite zu kommen, mit der letzten Zeile am unteren Bildschirmrand.

Abspeichern Ihres Entwurfs in einer Datei

Um abzuspeichern, was Sie eingegeben haben, schreiben Sie:

/S

Die Dialogzeile antwortet mit:

SAVE <name> as CMD or as FMT file (C, F or stop)?
(Speichere <name> als CMD oder als FMT Datei (C, F oder halt)?

Wenn Sie irgendeine andere Taste als <C> oder <F> drücken, so kehren Sie zur Arbeitstafel zurück.
Wenn Sie <C> drücken, so fragt ZIP:

Is this form to be printed (Y or N)?
(Soll dies Formular gedruckt werden (Ya, Nein)?)

(Verzeihen Sie den Witz mit "Ya" statt "Ja", aber Sie müssen ein englisches "Y" (YES) für JA eingeben!).

Jede andere Eingabe außer einem großen oder kleinen "Y" entspricht einem "NEIN".
Anschließend gibt Ihnen ZIP die Möglichkeit, den Filenamen zu verändern (sowohl die .CMD als auch die .FMT-Dateien betreffend), indem es fragt:

File <name>: do you want to change its name (Y or N)?
(Datei <name>: Wollen Sie den Namen ändern (Ya oder Nein)?)

Wenn Sie mit "Y" antworten, fragt ZIP Sie nach dem neuen Namen. Sie können auch ein anderes Laufwerk als das bisher benutzte als Teil des Namens angeben, um Ihre Arbeit auf einer anderen Diskette zu sichern. Oder Sie können irgendeine andere Taste drücken (als <Y>), dann benutzt ZIP den Dateinamen, unter dem Sie gearbeitet haben.

ZIP speichert zuerst ein Abbild des Bildschirm- oder Druckformulars, das Sie erzeugt haben. Die Dialogzeile sagt Ihnen, womit ZIP beschäftigt ist:

Writing screen image <name>.ZIP.

(Ich speichere gerade die Wiedergabe des Bildschirms)

Sobald es die direkte Wiedergabe des Bildschirms (die Sie später weitereditieren könnten) abgespeichert hat, speichert es eine druckbare Form davon, wobei die Dialogzeile sagt:

Writing printable file <name>.ZPR.

(Ich speichere gerade eine druckbare Datei des Formulars)

(Dieser File kann als "Schwarz-auf-Weiß"-Unterlage Ihres Entwurfs ausgedruckt werden, z. B. als Konferenzunterlage etc.)

Beim Abspeichern einer dBASE II-Programmdatei fügt ZIP zunächst den Filenamen als Kommentar ein (* <NAME>.CMD oder * <NAME>.FMT).

(Sie erinnern sich: In dBASE II-Programmen können Kommentare eingefügt werden, die für den menschlichen Leser das Programm näher erläutern. Ein Sternchen am Anfang der Zeile zeigt dem System an, daß dies ein Kommentar ist und nicht für es bestimmt, d.h. nicht "ausführbar").

Wenn der File eine ".CMD"-Datei ist, so fügt es dann

```
ERASE (wenn es eine Bildschirmmaske ist)
oder
SET FORMAT TO PRINT
SET MARGIN TO xx (wenn es ein Drucker-Formular ist)
ein.
```

Bei beiden Dateitypen (.CMD oder .FMT) fügt ZIP all die benötigten @ ... SAY und GET-Anweisungen ein sowie ein READ nach jedem 64ten GET. Die Dialogzeile sagt Ihnen, in welchen Zeilen dies geschieht.

ZIP fügt schließlich ein READ an das Ende jedes Bildschirm-Eingabe-Programms an, oder die Anweisung SET FORMAT TO SCREEN bei Druckerformularen, unmittelbar vor dem abschließenden RETURN, welches die letzte Anweisung in jedem Programm ist.

Nachdem ZIP Ihre Arbeitsdatei (.ZIP) gesichert und die Programme geschrieben hat, befindet sich das Original Ihrer Arbeit immer noch im Speicher, Sie können daran weiterarbeiten und es unter einem weiteren Filenamen abspeichern.

Wenn ZIP in einem Druck-Formular ein GET findet, sagt die Dialogzeile:

```
"GET" in PRINTOUT: Okay (Y or N)?
("GET" in Druckvorlage: In Ordnung (Ya oder Nein)?)
```

und wartet auf Ihre Antwort. Wenn Sie das GET absichtlich eingefügt haben, tippen Sie "Y", dann fährt ZIP mit dem Schreiben der Programmdatei fort.

Falls das GET versehentlich eingefügt wurde, drücken Sie irgend eine andere Taste. ZIP bricht den .CMD File ab, indem es ihn als INCOMPLETE COMMAND FILE (unvollständiges Programm) markiert, danach setzt es den Cursor auf das unerwünschte GET-Symbol, so daß Sie es entweder verändern oder löschen können.

BEACHTEN: Sie müssen anschließend Ihren File erneut abspeichern (/S), da er abgebrochen worden war, aber die Sekunden, die das unter ZIP in Anspruch nimmt, können Ihnen sehr viel mehr Zeit beim Entwurf Ihrer Anwendungsprogramme ersparen.

ZIP überprüft Ihre Variablen-Namen, um sicherzustellen, daß die Programme, die es schreibt, von dBASE II auch ausgeführt werden.

Wenn Sie versehentlich einen Variablen-Namen mit einem Zeichen angefangen haben, das dBASE II nicht akzeptiert (irgendeinem außer dem Alphabet, Ziffern und dem Doppelpunkt), so sagt ZIP:

NO VARIABLE: continue (Y or N)?
(KEINE VARIABLE: fortfahren (Ya oder Nein)?)

Das Drücken jeder anderen Taste als <Y> unterbricht das Abspeichern und stellt den Cursor auf die Fehlerstelle in Ihrer Arbeitsdatei ein, so daß Sie eine rasche Korrektur ausführen können.

Die Dialogzeile teilt Ihnen auch mit, wenn Sie einen Variablen-Namen mit einem Doppelpunkt anfangen oder beenden:

HANGING COLON: continue (Y or N)?
(ÜBERHÄNGENDER DOPPELPUNKT: fortfahren (Ya oder Nein)?)

Wiederum beendet jede andere Taste als "Y" die Programmabspeicherung und bringt den Cursor auf die Fehlerstelle im Bildschirm.

In den meisten Fällen werden Sie feststellen, daß Sie keinen Blick auf die von ZIP geschriebenen Programme zu werfen brauchen, außer Sie wollen Code für das Abfangen von Fehlern oder dergleichen einfügen. Gehen Sie einfach in Ihr Hauptprogramm und fügen den Aufruf

DO <name>
oder SET FORMAT TO <name>

ein und lassen das Programm ausführen. Wenn Ihr Programm Werte für die Variablen bereitstellt, die Sie unter ZIP eingesetzt haben, so werden Sie genau die gewünschte Bildschirmmaske oder das Druckformular erhalten, das Sie erzielen wollten.

Beginnen eines neuen Entwurfs, Quit

Um ein gänzlich neues Programm zu beginnen, löschen Sie erst die alte Arbeit durch:

/E

dann "Y", wenn ZIP fragt:

Erase everything (Yes or No)?
(Alles löschen (Ya oder Nein)?)

ZIP fragt Sie dann wieder, ob Sie eine alte oder neue Datei bearbeiten wollen. Wenn Sie die Arbeit jetzt abbrechen wollen so geben Sie "Q" ein, wenn die Dialogzeile sagt:

<NEW> or <OLD> file (Q to QUIT)?
(<NEUE> oder <ALTE> Datei (Q für Abbruch (quittiere den Dienst))

und antworten dann mit "Y", wenn ZIP fragt:

QUIT to system (Yes or No)?
(Rückkehr ins Betriebssystem (Ya oder Nein)?).

Sie können auch jederzeit die Arbeit abbrechen, wenn Sie sich im Arbeits-Schirm befinden, indem Sie eingeben:

/Q

Wiederum versichert sich ZIP, daß Sie wirklich abbrechen wollen, bevor es alles löscht und die Dateien schließt.

Wenn Sie quittieren und die Arbeitstafel länger ist als Ihr Bildschirm, so gibt Ihnen ZIP einen letzten, schnellen Blick auf die ganze Tafel, anschließend kehrt es ins Betriebssystem (CP/M oder welches Sie haben) zurück.

Nie wieder @ ... SAY ... GET programmieren

Nun wollen wir ZIP dazu benutzen, ein paar @ ... SAY's und GET's zu schreiben. Sie werden sehen, daß das viel einfacher ist, als alles von Hand auszutüfteln.

Vielleicht wollen Sie eines der in Ihrem Betrieb benutzten Formulare als Grundlage nehmen, denn dann erhalten Sie gleich ein Programm, das Sie später noch brauchen können.

ZIP stellt automatisch die Seitenlänge auf eine einzelne Bildschirmtafel ein, sobald Sie ein neues Programm beginnen. Falls Sie eine früher erzeugte Datei erneut zur Bearbeitung laden, so stellt ZIP die Seitenlänge natürlich auf die Länge dieser Datei ein.

Wenn Sie ein Druck-Formular erzeugen wollen, stellen Sie die Seitenlänge auf irgendeinen Wert bis maximal 88 Zeilen ein.

Die ständige Angabe von Zeilen- und Spaltennummer in der Dialogzeile wird es Ihnen erleichtern, Überschriften, Datum, Kommentare und Variable genau dorthin zu plazieren, wo Sie sie gerne haben wollen (beachten Sie auch das Bildschirm-Arbeitsblatt, das dem dBASE II-Handbuch beiliegt, machen Sie eventuell Fotokopien davon oder legen es unter Transparentpapier).

Um die Stelle festzulegen, an welcher der Inhalt einer Variablen ausgegeben werden soll, schreiben Sie dort ein "@"-Zeichen (Klammeraffe) (für SAY, d.h. nur Ausgabe des Wertes) oder das "#"-Zeichen (Doppelkreuz) (für GET, d.h. Eingabe eines Wertes) hin. Dies markiert den Anfang des Aus/Eingabe-Feldes.

Schreiben Sie nun den Namen der Variablen hin. Sie brauchen kein besonderes Zeichen, um das Ende des Namens zu kennzeichnen, da ZIP die Regeln für dBASE II-Variablen-Namen kennt. ZIP nimmt das Ende des Namens an, sobald es 10 zulässige Zeichen gelesen hat, oder wenn es auf ein Zeichen stößt, das nicht im Variablen-Namen vorkommen darf (z.B. ein Leerzeichen <BLANK>). Wenn Ihr Variablen-Name 10 Zeichen lang ist, so können Sie die allernächste Position, ohne Trennzeichen zwischen dem Variablennamen und dem Rest der Zeile, wieder benutzen.

Dies sind die Zeichen, die dBASE II in einem Variablennamen zuläßt:

A-Z, a-z, 0123456789 und einen *eingebetteten* Doppelpunkt (:)

ZIP läßt Sie an jeder Stelle eines Variablen-Namens einen Doppelpunkt hinschreiben, aber es wird einen ungültigen Eintrag abfangen, wenn es ein Programm erzeugt (<name>.CMD-File) und gibt Ihnen dann die Gelegenheit, den Fehler zu beseitigen (siehe den früheren Abschnitt über SAVE (Abspeichern)).

Wenn ein Variablen-Name eingegeben wird, der ein ungültiges Zeichen enthält, schneidet ZIP den Namen ab und nimmt an, daß Sie Ihren nächsten Kommentar oder Ihre nächste Überschrift auf dem Formular mit diesem (ungültigen) Zeichen beginnen wollen.

Sehen Sie sich nun wieder die Hilfstafel an, indem Sie zweimal das Befehls-Symbol eingeben (//). Um Ihr Formular zu bearbeiten werden Sie sicher die Befehle /D (oder Ihre <DELETE> oder <RUBOUT>-Taste), /I, /K, /A gut brauchen können.

Einfügen von dBASE II-Befehlen in die Formular-Tafeln

Damit Sie dafür nicht eigens einen Texteditor verwenden müssen, können Sie mit ZIP so viele dBASE II-Befehle einfügen, wie in Ihrem Formular Platz haben.

Sie müssen dabei aber folgende Regeln beachten:

- 1) Die Befehle müssen in eckige Klammern (Ä, Ü auf deutschen Tastaturen, hier mit “[” und “]” wiedergegeben) eingeschlossen sein.
- 2) Beide Klammern (öffnende und schließende) müssen sich in der selben Zeile befinden (verwenden Sie den Strichpunkt (;), mit dem in dBASE II Befehlszeilen erweitert werden können), wenn der Platz nicht ausreicht, um den Befehl in der nächsten Zeile fortzusetzen).
- 3) Anweisungen in eckigen Klammern müssen vom übrigen Text und anderen geklammerten Anweisungen getrennt sein.
Benutzen Sie ein einzelnes Befehls-Zeichen unmittelbar vor der öffnenden Klammer, oder zwei Befehls-Zeichen (@@, ##, @# oder #@). Eingeklammerte Anweisungen brauchen nicht von vorangehenden Variablen-Namen getrennt zu werden.
- 4) Beachten Sie, daß die eingebetteten Anweisungen räumlich an der Stelle im Programm stehen, die ihrer Position in der Zeichnung entspricht (wenn man sie zeilenweise liest). Das ist unter Umständen wichtig für das Initialisieren von Variablen etc.

PROBLEM: Wie können Sie das Ergebnis von Ausdrücken darstellen oder mehrere Felder eng nebeneinander darstellen, wenn die Feldnamen den Platz verdecken, an dem Sie sie ausgeben wollen?

LÖSUNG: Wenn die Möglichkeit, dBASE II-Anweisungen in Ihre Arbeitstafeln einzufügen nicht weiterhilft, so ist das ein Fall, in dem Sie den <name>.CMD-File, den ZIP geschrieben hat, nachträglich mit Ihrem Texteditor (oder MODIFY COMMAND <name> in dBASE II) bearbeiten müssen.

Trotzdem vereinfacht ZIP auch in diesem Fall das Schreiben eines Programms, da es bis zu 40 GET- und SAY-Symbole in einer 80-Zeichen langen Zeile verarbeiten kann.

Eine Möglichkeit besteht darin, nur die Symbole (@, #) auf den Positionen einzugeben, wo Sie die kurzen Ausgabefelder oder Eingabefelder haben wollen. Wenn dann beim Abspeichern die Dialogzeile von ZIP sagt:

NO VARIABLE -- continue (Y or N)?
(VARIABLE FEHLT -- fortfahren (Ya oder Nein)?)

Antworten Sie mit "Y".

Der Nutzen, den Sie davon haben, liegt darin, daß die Positionen, an welchen Sie die Ausdrücke oder Variablen ausgeben wollen, schon bestimmt sind, denn ZIP schreibt in das Programm:

@ xx, yy SAY (Rest der Zeile ist leer)
oder @ xx, yy GET (Rest der Zeile ist leer)

Nachteilig ist, daß Sie eine Menge von Anfragen der Dialogzeile erhalten, auf die Sie antworten müssen (Yes, Yes, Yes...). Um das abzustellen, mögen Sie es vorziehen, einfach einzelne Buchstaben oder Ziffern in ZIP einzugeben, die Sie anschließend mit dem Editor im Programm durch die eigentlichen Variablen-Namen oder rechnerischen Ausdrücke ersetzen.

Wenn Sie mit der Bearbeitung Ihrer Bildschirmmaske oder Ihres Druckformulars fertig sind, speichern Sie den File ab, indem Sie /S und dann /Q befehlen (Speichern und Quittieren).

WARNUNG: Rühren Sie den File, der das Abbild der Bildschirm-Tafeln enthält (<name>.ZIP) nicht an. Es ist zwar ein ASCII-File, so daß Sie ihn mit dem "TYPE"-Befehl in CP/M anschauen können (verwenden Sie <CTL>-S, um das Vorbeilaufen auf dem Bildschirm zu stoppen und fortzusetzen), aber wegen der besonderen Form, in welcher er gespeichert ist, wird er von einem Texteditor wahrscheinlich unbrauchbar gemacht!

Wenn Sie eine gedruckte Vorlage von Ihren Bildschirmtafeln haben wollen, so verwenden Sie Ihr Textsystem (z.B. WordStar oder ähnliche), um die Datei <name>.ZPR auszudrucken, oder drücken Sie <CTL>-P und verwenden dann den "TYPE"-Befehl von CP/M.

Wenn Sie wollen, können Sie auch Ihren Texteditor benutzen, um die dBASE II-Programme `<name>.CMD` oder `<name>.FMT` anzusehen, die ZIP geschrieben hat. Sie können auch für einen raschen Blick den "TYPE"-Befehl von CP/M benutzen, bevor Sie die Unterprogramme aufrufen.

Rufen Sie dBASE II auf und stellen Sie sicher, daß für die Variablen, die Sie in das Formular eingetragen haben, Werte bereitgestellt werden (erinnern Sie sich an die Bemerkung, daß die von SAY oder GET benutzten Variablen bereits existieren müssen, dies geschieht eben dadurch, daß Ihnen Werte zugewiesen wurden oder daß sie durch irgendwelche Vereinbarungen, z.B. in USE-nahmen einer Datenbank, verfügbar sind). Geben Sie dann ein:

```
DO <name>
oder SET FORMAT TO <name>
```

Zusammenfassung der ZIP-Befehle

@ (Klammeraffe)	zeigt Position für die Ausgabe einer Variablen mit SAY an
# (Sharp-Doppelkreuz)	zeigt Position für die Eingabe in Variable mit GET an.

Es wird kein Zeichen benötigt, um das Ende eines Feldes anzuzeigen, da ZIP automatisch annimmt, daß der Variablen-Name zu Ende ist, wenn es das erste Zeichen erreicht, das dBASE II nicht als Bestandteil eines Variablen-Namens akzeptieren würde (maximal 10 Zeichen). Die gültigen Zeichen für Variablen-Namen in dBASE II sind:

A-Z, a-z, 0123456789 und ein eingebetteter Doppelpunkt (:)

ZIP prüft Doppelpunkte, die nicht in andere Zeichen innerhalb des Variablennamens eingebettet sind, fehlende Variablen-Namen sowie GET-Anweisungen in einem Druckerformular ab und gibt eine Meldung in der Dialogzeile aus, verbunden mit der Frage, ob Sie dies korrigieren wollen oder ob ZIP dies ignorieren soll. ZIP fügt alle 64 GET's eine READ-Anweisung ein, verbunden mit einer Markierung in einem .CMD-File und einer Nachricht auf dem Bildschirm, um Sie wissen zu lassen, wo sich diese Anweisung befindet.

ZIP schreibt Programm-Dateien (`<name>.CMD` und `<name>.FMT`), die Sie ohne weitere Bearbeitung ausführen lassen können. Wenn es sich um eine Bildschirmmaske handelt, schreibt ZIP die Anweisung ERASE an den Anfang des Programms. Bei Druckformularen schreibt ZIP die Anweisungen SET FORMAT TO PRINT und SET MARGIN TO XX an den Anfang des erzeugten Programms sowie SET FORMAT TO SCREEN unmittelbar vor das abschließende RETURN.

ZIP speichert auch das Abbild der bearbeiteten Bildschirmtafeln als `<name>.ZIP` und eine druckbare Version davon als `<name>.ZPR`.

Die eckigen Klammern "[", "]" sind reserviert als Begrenzungszeichen für ZIP, sie dürfen auf den Arbeitstafeln nur für das Einschließen von dBASE II - Anweisungen verwendet werden oder für Kommentare, die Sie in Ihre Formulare einfügen wollen.

Dynamische Voreinstellungen

- wird als Markierungszeichen für vertikale Linien verwendet. Kann zu "!" oder irgendeinem anderen Symbol verändert werden, wenn Sie ZIP mit ZIPIN anpassen. Es kann auch während einer Arbeitssitzung vorübergehend für ein Formular verändert werden.
- ist das Markierungszeichen für horizontale Linien und kann bei der Anpassung oder für eine Sitzung verändert werden.

TABULATOR-MARKEN sind für jedes fünfte Zeichen gesetzt, können aber auf den von Ihnen bevorzugten Wert (im Bereich 1 bis 9) eingestellt werden, wenn Sie ZIP anpassen, auch bei jeder Sitzung.

DIE SEITENLÄNGE kann dynamisch auf jeden Wert von der Länge einer einzelnen Bildschirm-tafel bis zu 88 Zeilen eingestellt werden.

DIE RANDEINRÜCKUNG AUF DEM DRUCKER kann auf einen beliebigen Wert von 0 bis 127 eingestellt werden.

Übrige ZIP-Befehle und Symbole

- / ist das Symbol, das eine Befehlseingabe einleitet, sollte aber als rückwärtsgerichteter Schrägstrich (im Deutschen durch den Umlaut "Ö" belegt) oder ein anderes Zeichen, das im Text nicht gebraucht wird, eingestellt werden.
- // **zweimaliges** Betätigen der Taste mit dem Befehls-Zeichen während der Sitzung mit ZIP zeigt eine Zusammenfassung der Befehle und läßt Sie die Symbole für horizontale oder vertikale Linien sowie die Werte für TABulator-Abstand, Seitenlänge und Randeinrückung einstellen. Um die Seitenlänge auf 47 einzustellen würden Sie z.B. ein geben:

```
//
P
47
```

Um die anschließend aufgezählten Befehle zu benutzen, geben Sie zuerst das von Ihnen gewählte Zeichen zur Einleitung eines Befehls ein, und dann das Zeichen für den jeweiligen Befehl.

- H ist der Befehl, mit dem horizontale (waagrechte) Linien gezeichnet oder gelöscht werden können. Geben Sie ein: /H.
Wenn sich der Cursor auf irgendeinem anderen Zeichen befindet als dem, welches Sie als Markierungssymbol für horizontale Linien eingestellt haben, so wird von dieser Stelle aus eine Linie bis zum rechten Rand des Bildschirms gezeichnet.
Befindet sich der Cursor dagegen auf einem Horizontal-Marker selbst, so werden alle Zeichen von hier bis zum rechten Bildschirm-Rand gelöscht.

- V ist der Befehl, mit dem vertikale (senkrechte) Linien gezeichnet oder gelöscht werden können. Geben Sie ein: /V.

Wenn sich der Cursor auf irgendeinem anderen Zeichen befindet als dem, welches Sie als Markierungssymbol für vertikale Linien eingestellt haben, so wird von dieser Stelle aus bis zur untersten Zeile der Seite (also ev. auch über die sichtbare Bildschirmtafel hinaus) eine Linie gezeichnet. Wenn sich der Cursor auf dem als Vertikal-Marker eingestellten Zeichen befindet, so werden alle Zeichen in der entsprechenden Spalte von dieser Stelle bis zur untersten Zeile in der Seite gelöscht.

Wo immer sich vertikale und horizontale Linien kreuzen, wird automatisch der Schnittpunkt durch ein "+"-Zeichen ersetzt. Es wird auch wieder entfernt, wenn eine der beiden Linien gelöscht wird. Das funktioniert natürlich nur, wenn die Linien mit den Symbolen gezeichnet sind, die als aktuelle Markierungszeichen für vertikale und horizontale Linien eingestellt sind.

/H, /H und /V, /V können dazu benutzt werden, um ganze Zeilen bzw. Spalten zu löschen.

- T und B (mit vorangehendem Befehls-Symbol) bringen den Cursor an den oberen ("top") bzw. unteren ("bottom") Rand des Bildschirms, wobei er in der selben Spalte wie zuvor bleibt.

M bringt den Cursor in die Mitte der laufenden Zeile.

N zeigt die nächste Bildschirmtafel, Sie können damit Tafel für Tafel durch Ihren gesamten Entwurf wandern, bis Sie das Ende der eingestellten Seite (bis zu 88 Zeilen) erreicht haben.

P zeigt die vorhergehende Bildschirmtafel ("previous screen"), bringt Sie Tafel für Tafel nach oben, bis sich Zeile 0 am oberen Bildschirmrand befindet.

F zeigt die erste Bildschirmtafel (first screen), ein schneller Weg, um ganz an den Anfang der Seite zu kommen.

L zeigt die letzte Bildschirmtafel, ein schneller Weg, um ganz ans Ende der Seite zu kommen.

I fügt ein Leerzeichen an der Stelle ein, auf welcher der Cursor steht (insert). Falls dadurch irgendwelche Zeichen über den rechten Bildschirmrand hinausgestoßen werden, verschwinden sie - endgültig.

D löscht das Zeichen, auf dem sich der Cursor gerade befindet (wie Ihre "<DELETE>" oder <RUBOUT>-Taste).

- A "addiert" eine Zeile an der Stelle, wo sich der Cursor befindet, d.h. es wird eine Leerzeile eingefügt und der restliche Text nach unten gestoßen. Wird dabei Text über die letzte eingestellte Zeile in der Seite hinausgestoßen, so geht er verloren.
- K "killt" die Zeile, in welcher sich der Cursor befindet, d.h. löscht sie, der übrige Text rückt nach.
- E löscht die gesamte bearbeitete Seite ("erase") und erlaubt Ihnen, eine neue Format-Datei zu bearbeiten.
- S sichert die bearbeitete Datei als <name>.ZIP und <name>.ZPR, schreibt dann eine Programm-Datei für eine Bildschirm-Maske oder ein Drucker-Formular als <name>.CMD oder <name>.FMT (je nach Ihrer Wahl).
- Q "quittiert" den Dienst, d.h. bricht die Arbeit mit ZIP ab und bringt Sie ins Betriebssystem (z.B. CP/M) zurück.

Display-Unterprogramme für ZWEIDAT.CMD (Beispiel 4.14)

Im Folgenden sehen Sie zwei Beispiele, in denen zwei Unterprogramme für unser Programm "zweidat" erzeugt wurden. Sie erzeugen ein etwas anderes Display als in Beispiel 4.14. Natürlich müssen Sie sie entweder unter den Namen "zeignam.CMD" und "zeigaus.CMD" abspeichern, oder die Aufrufe in "zweidat.CMD" zu "shownam" und "showaus" ändern.

Es ist sehr leicht, mit ZIP bestimmte Eigenschaften eines solchen Displays zu ändern - z.B. alle Darstellungen ab einer bestimmten Zeile mit einem Tastendruck nach oben oder unten zu verschieben usw.

Probleme kann es geben, wenn etwas in die letzte Bildschirm-Zeile geschrieben wird, weil dies oft dazu führt, daß der gesamte übrige Bildschirm-Inhalt um eine Zeile nach oben geschoben wird.

Schwierigkeiten kann es auch mit MODIFY COMMAND geben, wenn die von ZIP erzeugten Programmzeilen länger als 80 Zeichen sind (was leicht passieren kann), da dann der Überhang über das 80ste Zeichen abgeschnitten wird. In diesem Fall sollte man einen guten Texteditor verwenden. Für unser Beispiel 4.14 war es z.B. nötig, die ERASE-Befehle aus den beiden Unterprogrammen zu entfernen, die ZIP an den Anfang jedes generierten Programms einfügt (da ja die andere Bildschirmhälfte nicht jedesmal gelöscht werden soll).

Beispiel 6.2:

***** File SHOWNAM ***

Code	+-----+	!	@kundcode	!	Name	!	@name	!	Telefon	!	@telefon	!
	+-----+											
PLZ	+-----+	!	@postltzahl	!	Stadt	!	@stadt	!	Adresse	!	@adresse	!
	+-----+											

* SHOWNAM.CMD

@ 4, 5 SAY	" +-----+ +-----+ +-----+ "
@ 5, 0 SAY	"Code !"
@ 5, 6 SAY	kundcode
@ 5,15 SAY	"! Name !"
@ 5,25 SAY	name
@ 5,45 SAY	"! Telefon !"
@ 5,56 SAY	telefon
@ 5,67 SAY	"!"
@ 6, 5 SAY	" +-----+ +-----+ +-----+ "
@ 7, 4 SAY	" +-----+ +-----+ +-----+ "
@ 8, 0 SAY	"PLZ !"
@ 8, 5 SAY	postltzahl
@ 8,16 SAY	"! Stadt !"
@ 8,25 SAY	stadt
@ 8,45 SAY	"! Adresse !"
@ 8,56 SAY	adresse
@ 8,77 SAY	"!"
@ 9, 4 SAY	" +-----+ +-----+ +-----+ "
RETURN	

Beispiel 6.3:

***** File SHOWAUS ***

Code	! @kunde!	Sch.Dat	! @scheck:dat!	Sch.Nr	! @scheck:nr!	Auftr.Nr	! @auftr:nr!
------	-----------	---------	----------------	--------	---------------	----------	--------------

Beschreibung	! @beschr	Hersteller	! @name
--------------	-----------	------------	---------

Betrag	! @betrag!	Rech.Dat	! @rech:dat!	Rech.Nr	! @rech:nr!	Stunden	! @stunden!
--------	------------	----------	--------------	---------	-------------	---------	-------------

* SHOWAUS.CMD

@ 10, 0 SAY ;

"=====

@ 11, 5 SAY "+-----+ +-----+ +-----+ +-----+"

@ 12, 0 SAY "Code !"

@ 12, 6 SAY kunde

@ 12,15 SAY "! Sch.Dat !"

@ 12,26 SAY scheck:dat

@ 12,37 SAY "! Sch.Nr !"

@ 12,47 SAY scheck:nr

@ 12,57 SAY "! Auftr.Nr !"

@ 12,69 SAY auftr:nr

@ 12,78 SAY "!"

@ 13, 5 SAY "+-----+ +-----+ +-----+ +-----+"

@ 15,13 SAY "+-----+ +-----+"

@ 16, 0 SAY "Beschreibung !"

@ 16,14 SAY beschr

@ 16,34 SAY "! Hersteller !"

@ 16,48 SAY name

@ 16,67 SAY "!"

@ 17,13 SAY "+-----+ +-----+ +-----+ +-----+"

@ 19, 7 SAY "+-----+ +-----+ +-----+ +-----+"

@ 20, 0 SAY "Betrag !"

@ 20, 8 SAY betrag

@ 20,15 SAY "! Rech.Dat !"

@ 20,27 SAY rech:dat

@ 20,36 SAY "! Rech.Nr !"

@ 20,47 SAY rech:nr

@ 20,55 SAY "! Stunden !"

@ 20,66 SAY stunden

@ 20,74 SAY "!"

@ 21, 7 SAY "+-----+ +-----+ +-----+ +-----+"

RETURN

3. Kapitel: Alle dBASE-Funktionen auf einem Blick

Inhaltsverzeichnis

Abschnitt 1: Einführung in die wichtigsten Funktionen.....	199
Abschnitt 2: dBASE-Befehle in alphabetischer Reihenfolge.....	235
Anhang A: dBASE-Befehle.....	367
Anhang B: Fehlermeldungen.....	371

1.0 EINFÜHRUNG IN dBASE II

Zum Starten des Datenbanksystems dBASE II legt man eine Kopie der Originaldiskette in ein verfügbares Diskettenlaufwerk ein und macht dieses Laufwerk zum CP/M-Hauptlaufwerk, indem man den Buchstaben, der das Laufwerk bezeichnet (z.B. "B"), gefolgt von einem Doppelpunkt (:) und der RETURN-Taste, eingibt. CP/M meldet sich hierauf mit dem Buchstaben des ausgewählten Laufwerks. Die nächste Eingabezeile lautet nun

dBASE

Hierdurch wird das Datenbanksystem in den Arbeitsspeicher geladen und gestartet. Auf dem Bildschirm erscheint dabei die Aufforderung

ENTER TODAY'S DATE AS MM/DD/YY OR RETURN FOR NONE :

Das daraufhin eingegebene Datum wird bei allen Datenbanken, deren Inhalte in der nachfolgenden Sitzung verändert werden, als Datum der letzten Änderung eingesetzt und bei allen durch den REPORT-Befehl erzeugten Ausgaben in deren Überschrift angegeben. dBASE II prüft das eingegebene Datum auf Plausibilität, wobei allerdings der 29. Februar bei allen ohne Rest durch vier teilbaren Jahreszahlen (d.h. auch bei vollen Jahrhunderten) akzeptiert wird. Zur Trennung der Monats- (MM), Tages- (DD) und Jahresziffern (YY) kann jedes beliebige Sonderzeichen außer dem Punkt (.) verwendet werden.

Einige Beispiele für zulässige Datumseingaben:

1,3,82	(3. Januar xx82)
02 16 83	(16. Februar xx83)
3/27/84	(27. März xx84)

Auf dem Bildschirm erscheinen nach der Datumseingabe die Zeilen

```
*** dBASE II      VER 2.xxx***
```

Der Punkt am Anfang der zweiten Zeile zeigt an, daß dBASE II nun zur Eingabe von Befehlen bereit ist. Befehle beginnen immer mit einem Steuerwort, das den Befehl identifiziert. In vielen Fällen können auf dieses Steuerwort Klauseln folgen, die die gewünschte Wirkung des Befehls näher spezifizieren. Vor der Ausführung eines Befehls überprüft dBASE II die gesamte eingegebene Zeile auf Fehler. Wird ein Fehler entdeckt, erscheint darüber eine Nachricht auf dem Bildschirm. In einem Korrekturdialog (sofern dieser bei der Installation von dBASE II vorgesehen wurde) kann der Benutzer dann die fehlerhaften Teile seiner Eingabe verbessern, ohne die gesamte Befehlszeile neu eingeben zu müssen. Kann dBASE II einen Fehler nicht näher beschreiben, so markiert es den Anfang des vermeintlich fehlerhaften Befehltextes mit einem Fragezeichen.

Beispiele für Fehlerkorrekturen:

. DISPRAY MEMORY	
*** UNKNOWN COMMAND	
DISPRAY MEMORY	Wiederholung des Fehlers
CORRECT AND RETRY (Y/N)? Y	Ja, verbessern
CHANGE FROM :PR	Buchstabenfolge PR
CHANGE TO :PL	in PL ändern
DISPLAY MEMORY	Befehl nach Korrektur
MORE CORRECTIONS (Y/N)? (RETURN)	keine weiteren Änderungen
. STORE (2 + 2 TO X	
*** SYNTAX ERROR ***	
?	Zeichenfolge (2 + 2 als
STORE (2 + 2 TO X	fehlerhaft markiert
CORRECT AND RETRY (Y/N)? Y	
CHANGE FROM :+2	
CHANGE TO :+2)	
STORE (2 + 2) TO X	
MORE CORRECTIONS (Y/N)? N	keine weiteren Änderungen
4	das Ergebnis

Neben der oben gezeigten Möglichkeit, dBASE II zu starten, gibt es noch eine andere Art des Aufrufs, nämlich

DBASE <Dateiname>

Dies bewirkt unmittelbar nach dem Laden von dBASE II die Ausführung der Befehlsdatei (siehe Abschnitt 3.3) mit dem angegebenen <Dateinamen>. Diese Möglichkeit des Programmstarts erweist sich als besonders praktisch beim Aufruf von dBASE II aus SUBMIT-Dateien oder bei Verwendung der Kett-Option im QUIT-Befehl.

STEUERZEICHEN

Im dBASE-Befehlsmodus sind folgende Tastenkombinationen mit SteuerCodes belegt:

ctl-P — Schaltet den Drucker ein bzw. aus (siehe SET PRINT-Befehl)

ctl-U — Löscht die aktuelle Befehlszeile

ctl-X — Löscht die aktuelle Befehlszeile

DELETE/RUBOUT — Löscht das zuletzt eingegebene Zeichen

ctl-H/BACKSPACE — wie DELETE/RUBOUT

ESCAPE- Bricht Befehle mit möglicherweise langen Laufzeiten ab; im einzelnen: COUNT, DELETE, DISPLAY, LIST, LOCATE, RECALL, REPLACE, SKIP und SUM. Daneben führt diese Taste auch bei Befehlen, die eine Eingabe vom Benutzer erwarten - wie ACCEPT, INPUT, REPORT (nur im Dialogteil) und WAIT - zum Abbruch. In all diesen Fällen geht dBASE II in den Befehlsmodus zurück und meldet sich mit dem Punkt.

Auch die Bearbeitung von Befehlsdateien läßt sich durch ESCAPE abbrechen, da dBASE vor der Ausführung jeder neuen Befehlszeile prüft, ob zwischenzeitlich an der Tastatur diese Taste gedrückt wurde.

Diese Funktion der ESCAPE-Taste kann durch den Befehl SET ESCAPE OFF ausgeschaltet werden.

2.0 FORDERUNGEN AN DAS SYSTEM

Um alle Eigenschaften von dBASE II voll nutzen zu können, sollte das verwendete Computersystem folgende Spezifikationen erfüllen:

I. 8-Bit-Systeme auf der Basis einer 8080- oder Z-80-CPU:

- a) Arbeitsspeicher mit einer Kapazität von mindestens 48 KB einschließlich CP/M (dBASE II verwendet den Arbeitsspeicher bis zur Hexadezimaladresse A400);
Hinweis: Bei Systemen mit übergroßem CP/M-Modul, wie z.B. Apple, Heath oder Northstar, wird mehr als 48 KB Arbeitsspeicher benötigt.
- b) Betriebssystem CP/M (Version 2.0 oder spätere)
- c) Mindestens eine von CP/M betriebene Massenspeicherstation (in der Regel Floppy-Disk- oder Hard-Disk-Laufwerke);
- d) Falls der Seitenmodus genutzt werden soll, ein Terminal mit adressierbarem Cursor (möglichst 24 Zeilen auf 80 Spalten),
- e) Drucker nach Bedarf (nur bei wenigen Befehlen erforderlich).

II. 16-Bit-Systeme auf der Basis einer 8086- oder 8088-CPU:

- a) Arbeitsspeicher mit einer Kapazität von mindestens 128 KB einschließlich Betriebssystem;
- b) Betriebssystem CP/M-86 oder MSDOS
- c) Mindestens eine Massenspeicherstation
- d) Falls der Seitenmodus genutzt werden soll, ein Terminal mit adressierbarem Cursor (möglichst 24 Zeilen auf 80 Spalten);
- e) Drucker nach Bedarf (nur bei wenigen Befehlen erforderlich).

3.0 dBASE-DATEIEN

Unter einer Datei versteht man eine auf einem Massenspeichermedium gelagerte Sammlung von Daten, die in eine Datei eingespeist oder aus ihr abgerufen werden können. dBASE II unterscheidet sieben Dateitypen, die jeweils mit bestimmten Anwendungen des Systems zusammenhängen.

Die Benennung von Dateien entspricht bei dBASE II dem CP/M-Standard, d.h. maximal acht Zeichen für den eigentlichen Namen, gefolgt von einem Punkt (.) und bis zu drei Zeichen zur Angabe des Dateityps. In untenstehender Tabelle sind die Voreinstellungen für die von dBASE II verwendeten Dateitypen aufgeführt. Bei jedem Befehl, der die Angabe eines Dateinamen erfordert, kann die Angabe des Typs unterbleiben, wenn dieser mit der Voreinstellung durch den Befehl übereinstimmt.

Datenbank-Dateien	— .DBF
Variablen-Dateien	— .MEM
Befehls-Dateien	— .CMD (oder .PRG bei 16-Bit-Version)
Format-Dateien	— .FRM
Text-(ASCII)-Dateien	— .TXT
Schlüssel-Dateien	— .NDX
Masken-Dateien	— .FMT

Für nähere Informationen über die Benennung von CP/M-Dateien sei an dieser Stelle auf das Buch "CP/M User's Guide" von Digital Research verwiesen.

3.1 DATENBANK-DATEIEN (.DBF)

Datenbankdateien stellen den wichtigsten Dateityp von dBASE dar. Sie bestehen aus einem die Struktur der Datenbank beschreibenden Satz und 0 bis 65535 Datensätzen. Der Struktursatz enthält Angaben über den Aufbau der nachfolgenden Datensätze. Er kann bis zu 32 Einträge umfassen, wobei jeder Eintrag einem Feld der Datenbank entspricht. Im einzelnen enthält der Struktursatz folgende Informationen:

- * die Datenfeld-Bezeichnungen
- * den Datentyp der Feldinhalte
- * die Länge der Datenfelder
- * die Position der Daten innerhalb des Satzes

DATENFELD-BEZEICHNUNGEN - Der Name eines Datenfeldes besteht aus bis zu zehn alphanumerischen Zeichen, wobei er jedoch mit einem Buchstaben beginnen muß. Auch Doppelpunkte (:) sind als Bestandteil einer Datenfeld-Bezeichnung zugelassen, doch müssen sie zwischen andere alphanumerische Zeichen eingebettet sein. Datenfelder werden immer über ihren Namen identifiziert.

Beispiele für Datenfeld-Bezeichnungen:

A
 A123456789
 ABC:DEF
 A:B:C:D:E
 ABCD:
 ABC,DEF

unzulässig, Doppelpunkt nicht eingebettet
 unzulässig, Komma nicht erlaubt

DATENTYPEN - dBASE II läßt drei verschiedene Datentypen für die Inhalte von Datenbankfeldern zu, und zwar: alphanumerische Zeichenreihen (z.B. "ABCD"), numerische Werte (z.B. 2 oder das Ergebnis eines Ausdruckes wie $5*18$) und Wahrheitswerte ("wahr"/"falsch").

LÄNGE EINES DATENFELDES - Die Länge eines Datenfeldes wird durch die Anzahl der Positionen bestimmt, die benötigt werden, um die gewünschten Daten in diesem Feld zu speichern. Bei Feldern des Typs Zeichenreihe kann die Feldlänge zwischen 1 und 254 variieren, bei numerischen Datenfeldern richtet sie sich nach der für die größte zu speichernde Zahl benötigten Stellenzahl, wobei ein Dezimalpunkt mitgezählt werden muß. Die Länge von Wahrheitswert-Feldern beträgt immer 1. Bei numerischen Datenfeldern enthält der Struktursatz auch Information über die Anzahl der Dezimalstellen.

Wurde die Struktur einer Datenbank einmal definiert, kann der Benutzer jederzeit beliebig viele Sätze mit Daten füllen. Normalerweise arbeitet man immer nur mit einer Datenbank, die dann als die "aktivierte Datenbank" bezeichnet wird. dBASE II bietet jedoch auch die Möglichkeit, gleichzeitig auf zwei Datenbanken zuzugreifen (siehe hierzu die Befehle SELECT und JOIN).

3.2 VARIABLEN-DATEIEN (.MEM)

In Variablendateien lassen sich die Werte von bis zu 64 temporären Variablen speichern.

Temporäre Variable sind unabhängig von der aktivierten Datenbank. Sie enthalten z.B. Konstanten, Rechenergebnisse oder Texte für Makro-Ersetzungen (siehe Kapitel 5). Für sie gelten in Bezug auf ihre Bezeichnung, die zulässigen Datentypen und die Variablenlänge dieselben Regeln wie für Datenbank-Felder (siehe oben).

Mit einem SAVE-Befehl werden die Inhalte aktueller temporärer Variablen in einer Variablendatei abgelegt. Mittels RESTORE lassen sie sich zu einem späteren Zeitpunkt wieder im Arbeitsspeicher restaurieren.

3.3 BEFEHLS-DATEIEN (.CMD) (.PRG in der 16-Bit-Version)

Eine Befehlsdatei enthält eine Folge von dBASE-Befehlen. In ihr kann der Benutzer häufig wiederkehrende Befehlsroutinen speichern, die bei Bedarf mit einem einzigen Befehl zur Ausführung gebracht werden können.

Befehlsdateien können mit Editoren oder Textverarbeitungssystemen erstellt werden. Auch dBASE II bietet mit dem MODIFY COMMAND-Befehl eine Möglichkeit zum Editieren von Befehlsdateien an. Durch den DO-Befehl werden Befehlsdateien gestartet. Sie können beliebige dBASE-Befehle enthalten, doch ist bei der Verwendung einiger Befehle (z.B. CREATE, INSERT, APPEND (ohne Zusatz)) zu beachten, daß diese Eingaben durch den Benutzer erwarten.

Befehlsdateien dürfen geschachtelt sein, d.h. in einer Befehlsdatei kann mittels DO eine weitere Befehlsdatei aufgerufen werden. Allerdings ist dabei zu beachten, daß dBASE II nur bis zu 16 offene Dateien gleichzeitig zuläßt. Wenn also bereits eine Datenbank aktiviert wurde, beträgt die zulässige Schachtelungstiefe für Befehlsdateien nur noch 15. Daneben verwenden einige Befehle zusätzliche Arbeitsdateien (SORT benötigt z.B. zwei zusätzliche Dateien, die Befehle REPORT, INDEX, COPY, SAVE, RESTORE und PACK jeweils eine). Dies muß ebenfalls bei der Schachtelung von Befehlsdateien berücksichtigt werden. Wenn z.B. bei Verwendung geschachtelter Befehlsdateien ein SORT-Befehl erteilt wird, darf die Schachtelungstiefe maximal 13 betragen, da andernfalls mehr als 16 Dateien benötigt würden (die aktivierte Datenbankdatei, 2 Arbeitsdateien für SORT und 13 Befehlsdateien ergeben zusammen 16 Dateien). Wenn das Ende einer Befehlsdatei oder der Befehl RETURN erreicht wird, wird die Ausführung der Befehlsdatei beendet. Der von ihr belegte Speicher steht dann anderen Befehlsdateien zur Verfügung.

3.4 FORMAT-DATEIEN (.FRM)

Formatdateien finden beim REPORT-Befehl Verwendung (siehe auch dort). Sie werden im Dialogteil dieses Befehls erzeugt und enthalten Informationen über Struktur und Inhalt der durch REPORT zu erstellenden Formblätter. Die Änderung von Formatdateien ist mit Hilfe von Editoren oder Textverarbeitungssystemen zwar möglich, doch wird es sich vielfach als einfacher erweisen, im REPORT-Dialog eine neue Formatdatei zu erstellen.

3.5 TEXT-(ASCII)-DATEIEN (.TXT)

Textdateien werden durch die Befehle SET ALTERNATE TO <Dateiname> und SET ALTERNATE ON angelegt. Nähere Informationen über die Wirkungsweise dieser Befehle finden sich bei der Beschreibung des SET-Befehls. Darüberhinaus legen die Befehle COPY und APPEND Textdateien an, wenn sie mindestens eine der Klauseln SDF oder DELIMITED enthalten.

3.6 SCHLÜSSEL-DATEIEN (.NDX)

Schlüsseldateien werden durch den INDEX-Befehl erzeugt und enthalten Schlüssel und Verweise auf die einzelnen Sätze einer Datenbank. Die Verwendung von Schlüsseldateien ermöglicht insbesondere bei sehr umfangreichen Datenbanken das schnelle Auffinden einzelner Sätze (siehe auch INDEX-Befehl).

3.7 MASKEN-DATEIEN (.FMT)

Eine Masken-Datei enthält nur "@"-Befehle und "*" -Kommentare. Mit dem Befehl SET FORMAT TO <Dateiname> wird sie bereitgestellt und durch nachfolgende READ-Befehle auf sie zugegriffen. Maskendateien lassen sich - ähnlich den Befehlsdateien - mit Hilfe von Textverarbeitungsprogrammen oder durch den MODIFY COMMAND-Befehl erstellen. Man kann auf ihre Verwendung verzichten, wenn man die "@"- und "*" -Befehle direkt in die entsprechende Befehlsdatei einbaut.

4.0 AUSDRÜCKE

Ein Ausdruck im Sinne von dBASE II ist die Zusammenstellung einfacher Werte und Operatoren, deren Auswertung einen neuen einfachen Wert ergibt. So läßt sich beispielsweise der Ausdruck "2 + 2" zum Wert "4" auswerten. Ausdrücke sind keineswegs nur numerischer Natur. So ergibt z.B. die Auswertung des Ausdrucks 'abc' + 'def' den Wert 'abcdef' (sogenannte Konkatenation von Zeichenreihen), und der Ausdruck 1 > 2 liefert den Wahrheitswert ".F." oder "falsch".

Folgende Komponenten können in dBASE-Ausdrücken enthalten sein:

- * Datenbank-Variablen (Datenbank-Feldbezeichnungen)
- * Temporäre Variablen
- * Konstanten
- * Funktionen
- * Operationen

VARIABLEN - dBASE II unterscheidet zwei Arten von Variablen: Datenbank-Variablen und temporäre Variablen. Datenbank-Variablen sind alle Felder des aktuellen Datensatzes, da diese unmittelbar geändert werden können. Ihr jeweils letzter Wert bleibt in der Datenbank gespeichert. Temporäre Variablen sind weder einem bestimmten Satz noch einer Datenbank zugeordnet und dienen lediglich der Zwischenspeicherung von Werten.

Nach ihrem Inhalt unterscheidet man Variablen in drei Typen:

- * Alphanumerische Variablen (Zeichenreihen-Variablen)
- * Numerische Variablen
- * Logische Variablen

KONSTANTEN - Eine Konstante ist ein Datengebilde mit unveränderlichem Wert. So stellen beispielsweise die Konstanten 1, 'abc' und .T. unabhängig vom aktuellen Datenbanksatz stets denselben Wert dar, nämlich eine numerische Eins, eine Zeichenreihe bestehend aus den Buchstaben "a", "b" und "c" bzw. den Wahrheitswert "wahr" (.T.). Demgegenüber repräsentieren Datenbankvariablen immer nur ihren jeweils aktuellen Wert.

Alphanumerische Konstanten müssen stets in Hochkomma ('), Anführungszeichen (") oder eckige Klammern ([,]) eingeschlossen sein. Enthält eine Zeichenreihe eines dieser Sonderzeichen, so muß sie durch ein Paar der anderen Sonderzeichen eingeschlossen werden. Beispielsweise stellen die Zeichenreihen 'abc[def]ghi' und [abc'def'ghi] korrekte alphanumerische Konstanten dar, 'abc'def'ghi' hingegen nicht.

Wahrheitswerte werden in dBASE II durch die Buchstaben "T", "t", "Y" und "y" für "wahr" (vom englischen TRUE bzw. YES) bzw. "F", "f", "N" und "n" für "falsch" (vom englischen FALSE bzw. NO) dargestellt.

4.1 FUNKTIONEN

Funktionen dienen der Erarbeitung von Ergebnissen, die mit den Möglichkeiten, die Ausdrücke bieten, nur schwer oder nicht zu erreichen sind. Funktionen werden durch Angabe ihrer Bezeichnung aufgerufen und führen dann vorprogrammierte Operationen aus. Diese Verarbeitung bezieht sich auf bestimmte Objekte (Argumente), die in der Regel beim Aufruf der Funktion anzugeben sind. Nach der Art des Ergebnisses, das von ihnen erarbeitet wird, unterscheidet dBASE II zwischen drei Arten von Funktionen: numerische, alphanumerische und logische. Numerische Funktionen liefern Zahlenwerte, alphanumerische Funktionen ergeben aus einem Zeichenvorrat zusammengesetzte Worte oder Texte (Zeichenreihen) und logische Funktionen haben Wahrheitswerte "wahr" oder "falsch" als Ergebnis. Nicht jedes Objekt ist als Argument bei Funktionen zugelassen. Aus der Art der Funktionen bzw. ihrer Bezeichnung lassen sich in der Regel die zugelassenen Objekt-Arten ableiten. Nachfolgend werden die in dBASE II vorhandenen Funktionen im einzelnen erläutert. Wird gegen die dort angegebenen Regeln für den Aufruf von Funktionen verstoßen, reagiert dBASE II mit einer Fehlermeldung (meist "**** SYNTAX ERROR ****").

CHR-Funktion (Umwandlung eines durch eine Zahl definierten Zeichens in das entsprechende ASCII-Zeichen)

Diese alphanumerische Funktion liefert als Ergebnis ein einzelnes Zeichen. Es bestimmt sich aus der Zahl, die seine Position im Alphabet des ASCII-Zeichensatzes angibt. Z. B. steht dort an 13. Stelle das Steuerzeichen für den Zeilenvorschub, das mit CHR(13) erzeugt werden kann. Die CHR-Funktion ist nützlich, um über vom Benutzer programmierte Ausgabebefehle aus dBASE II heraus Peripheriegeräte zu steuern (z.B. Anwendung bestimmter Darstellungsarten auf dem Bildschirm oder Wechsel des Zeichensatzes am Drucker).

Beispiel:

```
. ? 'abcd' + chr(13) + '_____'
abcd
```

Dateiende-Funktion (Abfrage, ob Dateiende erreicht wurde)

EOF

Die logische Funktion wird durch EOF (END OF FILE) aufgerufen und gibt an, ob das Ende der Datei (der gerade benutzte Satz ist dann der letzte in dieser Datei) erreicht ist. Trifft dies zu, wird als Ergebnis .T. (für "wahr"), andernfalls .F. (für "falsch") geliefert.

Beispiele:

```
. ? EOF
.T.
```

```
. GOTO TOP
```

```
. ? EOF
.F.
```

```
. SET DATE TO 10 2 82
```

```
. ? DATE( )
10/02/82
```

Datei-Funktion (Prüfung, ob eine Datei vorhanden ist)

FILE (<Zeichenreihe>)

Mit dieser logischen Funktion erfolgt die Prüfung, ob die in <Zeichenreihe> angegebene Datei vorhanden ist. In der Dateibezeichnung (die durch " oder ' einzuschließen ist) muß der Typ und das Laufwerk nicht enthalten sein. In diesem Fall sucht dBASE II auf dem aktuell eingestellten Laufwerk die Datei und ergänzt .DBF als Typ.

Ist die Datei vorhanden, wird .T. (für "wahr"), andernfalls .F. (für "falsch") als Ergebnis geliefert.

Beispiele:

```
. ? FILE('BESTAND')
.T.

. USE BESTAND

. ? FILE("A:BESTAND.NDX")
.F.
```

Datums-Funktion

DATE()

Diese alphanumerische Funktion erzeugt eine Zeichenreihe, die in der Schreibweise XX/XX/JJ das in der Sitzung mit dBASE II verwendete Datum enthält (Reihenfolge von Tag und Monat abhängig von der Installation bzw. dem zuletzt ausgeführten SET DATE-Befehl; siehe SET-Befehle). Die Zeichenreihe umfaßt 8 Stellen. Eine Eingabe zwischen den Klammern ist nicht erlaubt; sie dienen nur der Unterscheidung von einer Variablen mit dem Namen "DATE".

Beispiele:

```
. ? DATE( )
06/15/81
. STORE DATE( ) TO MEMVAR
06/15/81
. SET DATE TO 4 1 82
. ? DATE( )
04/01/82
```

Großbuchstaben-Funktion (Konvertierung von Klein- in Großbuchstaben)

!(<Zeichenreihen-Ausdruck>)

Die alphanumerische Funktion wird durch ein Ausrufezeichen aufgerufen. Sie wertet den <Zeichenreihen-Ausdruck> aus und liefert das Ergebnis unter Ersetzung von Klein- durch Großbuchstaben ab.

Beispiele:

```
. !('abc')  
ABC
```

```
. !('ABC' + 'def')  
ABCDEF
```

INT-Funktion (Berechnung eines ganzzahligen Wertes)

INT(<numerischer Ausdruck>)

Diese numerische Funktion ermittelt den ganzzahligen Anteil des <numerischen Ausdrucks> . Zunächst wird der Ausdruck selbst ausgewertet. Durch Abschneiden ab dem Dezimalpunkt (wenn vorhanden) ergibt sich der ganzzahlige Anteil. Man beachte, daß keine Aufrundung erfolgt; sie muß ggf. durch eine entsprechende Ergänzung im <numerischen Ausdruck> (z.B. mit + 0.5) explizit vorgesehen werden. INT liefert also einen abgerundeten Wert ohne den Dezimalpunkt und nachfolgende Stellen.

Beispiele:

```
. ? INT(123.657)  
123
```

```
. ? INT(123.657 + 0.5)  
124
```

```
. ? STORE 123.00 TO X  
123.00
```

```
. ? INT(X)  
123
```

Längen-Funktion (Ermittlung der Länge einer Zeichenreihe)

LEN (<Zeichenreihe>)

Die numerische Funktion liefert als Zahl die Länge der <Zeichenreihe>. Jedes Zeichen belegt dabei eine Stelle.

Beispiele:

. STORE 'abc' TO WORD

. ? LEN(WORD)

3

Abfrage der Löschmarkierung

*

In einer Datenbank lassen sich Sätze für eine spätere Löschung markieren (siehe Befehle DELETE, RECALL, PACK). Mit dieser logischen Funktion, aufgerufen durch den Stern, läßt sich prüfen, ob der im Zugriff befindliche Satz eine solche Markierung aufweist (die ebenfalls durch den Stern kenntlich gemacht ist). Ist dies der Fall, liefert die Funktion den Wahrheitswert .T. (für "wahr"), andernfalls .F. (für "falsch").

Beispiele:

. ? *

.T.

Satznummer-Funktion

#

Der Aufruf (mit dem Nummernzeichen) dieser numerischen Funktion liefert eine ganze Zahl. Sie gibt die Nummer des Satzes in der Datenbank an, auf den gerade zugegriffen wird. Man kann damit die aktuelle Position in der Datenbank bestimmen. Dabei ist auch der Wert "0" als Ergebnis möglich, wenn bei einem Suchvorgang (siehe FIND-Befehl) kein Satz gefunden wurde, der einer vorgegebenen Bedingung genügt.

Beispiele:

```
. ? #  
4
```

```
. SKIP
```

```
. ? #  
5
```

STRING-Funktion (Manipulation einer Zeichenreihe)

STR(<numerischer Ausdruck> , <Länge> , [<Dezimalstellen>]

STR ist eine alphanumerische Funktion und wertet zunächst den <numerischen Ausdruck> aus und formt dessen Zahlenwert dann in eine Zeichenreihe aus Ziffern um. Das Ergebnis ist eine Zeichenreihe der vorgegebenen <Länge> ; falls <Dezimalstellen> angegeben ist, wird damit die Anzahl der Stellen nach dem Dezimalpunkt festgelegt. Als Angabe von <Länge> und <Dezimalstellen> sind Konstanten, Variable oder Ausdrücke zugelassen.

STR wird häufig benutzt, um den Wert eines Schlüssels für eine Schlüsseldatei (siehe Befehle INDEX und FIND) zu ermitteln. In diesem Fall müssen <Länge> und <Dezimalstellen> als Konstanten, d.h. direkt als Zahl vorgegeben werden.

Beispiel:

```
. STORE STR(2 * 1.1111, 7, 3) TO  
ALPHA  
2.222  
  
. DISPLAY MEMORY  
ALPHA          (C) 01 2.222  
** TOTAL **      01 VARIABLES USED  00007 BYTES USED
```

Teil"STRING"-Funktion (Manipulation von Teilen einer Zeichenreihe)

\$(<alphanumerischer Ausdruck> , <Anfang> , <Länge>)

Mittels dieser alphanumerischen Funktion, die durch das Dollar-Zeichen angestoßen wird, wird ein Teil einer Zeichenreihe zu einer eigenen Zeichenreihe. \$ liefert als Ergebnis eine Zeichenreihe, die nach Erarbeitung der Zeichenreihe des <alphanumerischen Ausdrucks> dort bei <Anfang> beginnt und dann die mit <Länge> angegebene Zahl von Stellen umfaßt. Aus einer Zeichenreihe läßt sich damit ein beliebiger, zusammenhängender Teil ausschneiden. <Anfang> und <Länge> können als Konstante (d.h. direkt als Zahl), mit einer Variablen oder als noch auszuwertender Ausdruck angegeben werden.

Wenn die Funktion zur Erarbeitung von Schlüsselwerten für Schlüsseldateien benutzt wird (siehe Befehle INDEX und FIND), so müssen <Anfang> und <Länge> als Konstante, d.h. direkt als Zahl angegeben werden.

Beispiele:

```
. ? $('abcdefgi', 3, 3)
cde
```

```
. STORE 3 TO ANFANG
3
```

```
. STORE 3 TO LAENGE
3
```

```
. STORE 'defgh' TO ALPHA
defgh
```

```
. ? $('"abc"' + ALPHA, ANFANG, LAENGE)
cde
```

VAL-Funktion (Umwandlung Zeichenreihe in numerischen Wert)

VAL(<Zeichenreihe>)

Mit der numerischen Funktion VAL wird eine Zeichenreihe in den ihr entsprechenden ganzzahligen Wert umgewandelt. Die <Zeichenreihe> darf sich nur aus einem Vorzeichen und Ziffern zusammensetzen und höchstens einen Dezimalpunkt enthalten. Im Ergebnis werden der Dezimalpunkt und nachfolgende Stellen abgeschnitten; VAL liefert immer einen ganzzahligen, ggf. abgerundeten Wert. Wenn die Zeichenreihe nur am Anfang Ziffern aufweist, ergibt sich die Zahl aus diesen führenden Ziffern.

Eine andere Möglichkeit, Zeichenreihen aus Ziffern in Zahlen umzuwandeln, bietet die Verwendung von "&" (siehe bei Kapitel 5 "Makro-Ersetzungen"). Hierbei wird auch ein Dezimalpunkt berücksichtigt, wenn es in einer Operation zur Ersetzung der Zeichenreihe durch den ihr entsprechenden Zahlenwert kommt.

Beispiele:

```
. ? VAL('123')
123
```

```
. ? VAL('123abc')
123
```

```
. ? VAL('123.456')
123
```

```
. STORE '123.456' TO ZAHL
123.456
```

```
. ? 100 + &ZAHL
223.456
```

RANK-Funktion (Bestimmung der ASCII-Position eines Zeichens)

RANK(<Zeichenreihe>)

Diese Funktion liefert in Form einer ganzen Zahl die Position, die dem ersten Zeichen der <Zeichenreihe> im ASCII-Code zugeordnet ist. Sie entspricht damit der ASC-Funktion vieler BASIC-Dialekte.

Beispiel:

```
. ? RANK('A')
65
```

Zeichenreihe-Suchfunktion (Prüfung, ob eine Zeichenreihe in einer anderen vorkommt)

@(<Zeichenreihe 1> , <Zeichenreihe 2>).

Diese numerische Funktion, aufgerufen mit dem Sonderzeichen "@", liefert eine ganze Zahl. Sie gibt an, an welcher Stelle der <Zeichenreihe 2> eine Folge von Zeichen beginnt, die derjenigen von <Zeichenreihe 1> entspricht. Dabei wird nur die erste vorgefundene Übereinstimmung angezeigt; die Prüfung in <Zeichenreihe 2> beginnt am Anfang, also von links nach rechts. Falls die erste Zeichenreihe in der zweiten nicht enthalten ist, lautet das Ergebnis "0". Bei Gleichheit der beiden Zeichenreihen liefert die Funktion den Wert "1".

Diese Funktion ist der Funktion zur Manipulation von Teilen einer Zeichenreihe (siehe Teil "STRING"-Funktion) ähnlich. Jedoch gibt @ an, an welcher Stelle in einer Zeichenreihe eine vorgegebene Zeichenreihe beginnt.

Beispiel:

```
. ? @('def', 'abcdefghi')
4
```

Test-Funktion (Prüfen eines Ausdrucks)

TEST(<Ausdruck>)

Die Funktion Test dient dazu, einen vom Nutzer erzeugten <Ausdruck> auf Korrektheit zu prüfen. Wendet man diese Funktion beispielsweise auf eine Eingabe durch den Benutzer an, so lassen sich durch diese Eingabe eventuell hervorgerufene Fehler ('Syntax Errors') abfangen.

Diese Funktion liefert einen von Null verschiedenen Wert, wenn der <Ausdruck> korrekt ist, andernfalls ist das Resultat von Test Null.

Beispiel:

```
INPUT EINGABE
IF TEST(EINGABE) = 0
    DO KORRIG                      (EINGABE fehlerhaft, korrigieren)
ELSE
    DO WEITER                      (EINGABE o.k., weiter)
ENDIF
```

Typ-Funktion (Abfrage des Typs eines Ausdrucks)

TYPE (<Ausdruck>)

Hiermit kann die Art - numerisch, alphanumerisch oder logisch - eines Ausdruck oder einer Variable geprüft werden. Die alphanumerische Funktion liefert als Wert ein Zeichen. In der o.a. Reihenfolge sind dies "N", "C" bzw. "L". Kann der Typ eines Ausdrucks nicht eindeutig bestimmt werden (weil z.B. eine Variable noch nicht definiert ist), liefert die Funktion als Ergebnis den Wert "U" für "undefiniert".

Beispiel:

```
. STORE 1 TO EINS
1
. ? TYPE(EINS)
N
```

TRIM-Funktion (Abschneiden von Leerstellen am Ende von Zeichenreihen)

TRIM(<Variable> bzw. <Feldname>)

TRIM entfernt in einer Variablen oder einem Datenfeld die dort am Ende vorgefundenen Leerstellen. Der alphanumerische Wert dieser Funktion ist dann die so gekürzte Zeichenreihe.

Diese Funktion darf nicht benutzt werden, um Schlüsselwerte für den INDEX-Befehl festzulegen. dBASE II muß bei Schlüsselfeldern intern Längenberechnungen ausführen, die durch TRIM verfälscht werden können.

Beispiele:

```
. STORE "ABC  " TO LANG
ABC
```

```
. ? LEN(LANG)
6
```

```
. STORE TRIM(LANG) TO LANG
ABC
```

```
. ? LEN(LANG)
3
```

4.2 OPERATIONEN

dBASE II kennt vier Arten von Operationen: Berechnungen, Vergleiche, logische Auswertungen und Erarbeiten von Zeichenreihen. Die gewünschte Operation wird jeweils durch hierfür reservierte Zeichen angegeben. Damit werden auch die Operanden getrennt, auf denen die Operation ausgeführt werden soll. Ergebnisse von Operationen können selbst wieder Operanden sein. Sie sind ggf. in Klammern einzuschließen, wenn bei der Ausführung eine bestimmte Reihenfolge vorgeschrieben werden soll.

Wichtig ist, daß die Typen der Operanden mit der Operation vereinbar sind. So können z.B. nur Zeichenreihen aneinandergesetzt werden, nicht hingegen ein numerischer Wert an eine Zeichenreihe. Im letzten Fall ist durch eine entsprechende Funktion (hier STR) die Zahl in eine Zeichenreihe umzuwandeln:

```

. STORE '3' TO A
3
. STORE 3 TO B
3

. ? A + B

*** SYNTAX ERROR ***
?
? A + B
. ? VAL(A) + B
6

```

In diesem Beispiel tritt ein Fehler auf, weil die 3 zum einen ein numerischer Wert zum anderen eine Zeichenreihe ist. Im Rechner werden sie unterschiedlich dargestellt und dBASE II kann nicht mehr erkennen, ob eine arithmetische Addition oder das Aneinanderfügen von Zeichenreihen erfolgen soll.

ARITHMETISCHE OPERATIONEN

Sie liefern numerische Werte und sind mit folgenden Zeichen anzugeben:

- + für Addition
- für Subtraktion
- * für Multiplikation
- / für Division
- () zur Klammerung von Operationen in Operationen (geschachtelte Klammern werden in der Reihenfolge von innen nach außen berechnet)

Beispiele:

```

. ? (4 + 2) * 3
18

```

(Leerzeichen vor und nach Operationszeichen sind unerheblich, sie dienen hier nur zur besseren Lesbarkeit)

```

. ? 4 + (2 * 3)
10

```

VERGLEICHOPERATIONEN

Sie erzeugen logische Werte (.T. für "wahr" und .F. für "falsch") und sind mit folgenden Zeichen anzugeben:

- < für "kleiner als"
- > für "größer als"
- = für "gleich"
- <> oder # für "ungleich"
- < = für "kleiner gleich"
- > = für "größer gleich"
- \$ für "eine Zeichenreihe in anderer enthalten" (A\$B liefert .T., falls A mit B identisch oder A in B enthalten ist)

Beispiele:

. ? 'abc' \$ 'abcdefgh'
.T.

. ? 'abc' \$ 'ghijkl'
.F.

. DISPLAY FOR 'Anton' \$ TITEL

(liefert alle Sätze, in denen im Feld TITEL
'Anton' enthalten ist)

LOGISCHE OPERATIONEN

Sie liefern logische Werte und sind mit folgender Schreibweise anzugeben:

.OR. für logisches "oder"
.AND. für logisches "und"
.NOT. für logische Verneinung

Beispiele:

. STORE T TO A
.T.
. STORE F TO B
.F.
. ? A .OR. B
.T.
. ? STORE .NOT. B TO C
.T.
. ? A .AND. C
.T.

OPERATIONEN AUF ZEICHENREIHEN

Sie liefern als Wert eine Zeichenreihe und sind durch folgende Zeichen anzugeben:

- + für Aneinanderfügen von Zeichenreihen
- für Aneinanderfügen von Zeichenreihen, wobei Leerstellen am Ende zu verbindender Zeichenreihen an das Ende der Ergebniszeichenreihe verlagert werden. (Führende oder zwischen andere Zeichen eingebettete Leerstellen sind davon nicht betroffen).

Beispiele:

```
. STORE 'ABCD ' TO A
ABCD
```

```
. STORE 'EFGH' TO B
EFGH
```

```
. ? A + B
ABCD EFGH
```

(Man beachte, daß die Leerstellen in der Zeichenreihe unverändert übernommen wurden)

```
. ? A - B
ABCDEF GH
```

(Hier wurden die Leerstellen am Ende von A entfernt und zum Ende der Ergebniszeichenreihe verlagert; letzteres ist hier nicht sichtbar)

REIHENFOLGE VON OPERATIONEN

Operationen laufen in einer definierten Reihenfolge ab, wenn in einem Ausdruck mehrere Operationen aufeinandertreffen. Hierbei wirken Vorrangregeln, d.h. welche Operation vor einer anderen ausgeführt wird. Operationen können dabei im Rang gleich sein. Treten sie zusammen in einem Ausdruck auf, so ergibt sich aus der Abarbeitung eines Ausdruckes von links nach rechts, welche Operation vor einer anderen, gleichrangigen ausgeführt wird.

Aus nachfolgender Tabelle lassen sich die Vorrangregeln entnehmen.

Stufe	Reihenfolge bei		
	arithmetischen Operationen	Operationen auf Zeichenreihen	logischen Operationen
1	Klammern, Funktionen	Klammern, Funktionen	.NOT.
2	Vorzeichen (+, -)	Vergleiche \$ (Zeichenreihe in Zeichenreihe enthalten)	.AND.
3	*, / (Multipl., Division)	+, - (Aneinanderfügen von Zeichenreihen)	.OR.
4	+, - (Addition, Subtraktion)		
5	Vergleiche		

Die Reihenfolge der Stufen entspricht einer abnehmenden Wertigkeit im Vorrang von Operationen gegenüber anderen. Operationen derselben Stufe werden von links nach rechts abgearbeitet.

5.0 MAKRO-ERSETZUNGEN

Trifft dBASE II in Befehlen oder bei der Auswertung von Ausdrücken auf das Zeichen "&" am Anfang des Namens einer Variablen, der eine Zeichenreihe als Wert zugewiesen ist, so ersetzt dBASE II an derselben Stelle den Namen der Variablen (einschließlich &) durch diese Zeichenreihe.

Damit läßt sich z.B. die wiederholte Eingabe einer Zeichenreihen vermeiden. Sie wird einmal einer Variablen zugewiesen, die dann mit ihrem Namen stellvertretend (als "Makro") für diese Zeichenreihe auftritt. Erst wenn die Zeichenreihe benötigt wird, erfolgt die Ersetzung.

Auf diese Art lassen sich auch Parameterwerte zwischen Befehlsdateien austauschen, die sich untereinander aufrufen. Falls zu einer Zeichenreihe, die über Ersetzung einer Variablen verwendet wird, noch Zeichenreihen nach der Ersetzung hinzugefügt werden, muß der Name der Variablen mit einem Punkt beendet werden. dBASE II ersetzt alle Zeichen, beginnend bei "&" (einschließlich Leerstellen) bis zum nächsten Sonderzeichen durch die Zeichenreihe. Der Punkt im Namen der Variablen dient somit zur eindeutigen Anzeige des Endes des zu ersetzenden Teiles in einem Befehl oder Ausdruck.

Folgt dem "&" kein Name einer definierten Variablen, findet keine Ersetzung statt und dies Zeichen verbleibt in der Befehlszeile.

Beispiele:

```
. STORE 'DELETE RECORD ' TO T
DELETE RECORD
```

```
. &T 5
```

(&T wird ersetzt, so daß der Befehl DELETE RECORD 5 zur Ausführung kommt)

```
. ACCEPT "GIB LAUFWERK" TO L
```

```
. USE &L:DATEI
```

(&L wird ersetzt, so daß der Befehl USE B:DATEI ausgeführt wird, wenn bei ACCEPT B eingegeben wurde)

6.0 SCHNITTSTELLEN ZWISCHEN dBASE UND ANDEREN PROGRAMMEN

Mit dBASE II lassen sich Dateien lesen, die von anderen, nicht zu dBASE II gehörenden Programmen erzeugt wurden. Dies können z.B. Programme in BASIC, FORTRAN oder PASCAL aber auch andere Anwendungspakete sein. Ebenso kann dBASE II von seinen Datenbankdateien Kopien anlegen, die sich dann mit anderen Programmen weiterverarbeiten lassen.

Der Schlüssel zu dieser Schnittstelle liegt in der Verwendung von sogenannten Textdateien, deren Inhalt sich aus Zeichen des ASCII-Alphabetes zusammensetzt. Aufgrund der CP/M-Konventionen ist jeder Satz in einer Datei mit den Zeichen für RETURN und Zeilenvorschub abgeschlossen. Mit dem APPEND—Befehl lassen sich solche Dateien lesen, wenn die Option SDF in der Befehlszeile enthalten ist. Umgekehrt erzeugt COPY mit dieser Option solche Dateien. Falls nicht anders angegeben, erhalten dabei die mit DELIMITED und SDF angelegten Dateien eine um .TXT. ergänzte Bezeichnung.

Einige Programme lesen und schreiben Dateien darüberhinaus in einem speziellen Format: Zeichenreihen werden durch spezielle Zeichen (meist " oder ') eingeschlossen und Eintragungen sind durch Komma voneinander getrennt. Auch solche Dateien lassen sich mit dBASE II lesen bzw. erzeugen, wenn die Option DELIMITED und ggf. WITH genutzt wird. APPEND erkennt bei Angabe von DELIMITED selbständig, ob Hochkomma oder Anführungszeichen zur Begrenzung von Text-Eintragungen vorliegen. COPY führt bei DELIMITED die Begrenzung mit Hochkomma durch, läßt sich aber durch die Klausel WITH und Angabe eines frei wählbaren Zeichens auf die gewünschte Begrenzung einstellen. Es wird jedoch empfohlen, zur Begrenzung von Zeichenreihen nur Hochkomma oder Anführungszeichen zu verwenden.

Die Angabe "DELIMITED WITH ," bei COPY erzeugt Dateien, bei denen in Text-Eintragungen die Leerstellen am Ende und führende nicht belegte Stellen am Anfang von numerischen Feldern entfernt sind. Texte sind darüberhinaus nicht durch spezielle Zeichen am Anfang und Ende kenntlich gemacht.

Ausführliche Beispiele finden sich in der Beschreibung der Befehle APPEND und COPY.

Beispiele:

```
.USE <FILENAME> .DBF
.COPY TO <FILENAME> .TXT DELIMITED WITH "
```

```
.USE <FILENAME> .DBF
.APPEND FROM <FILENAME> .DAT SDF
```

7.0 GRUPPIERUNG DER BEFEHLE NACH IHRER WIRKUNG

In einer Sitzung mit dBASE, d.h. nach dem Start bis zur Eingabe des Befehls zur Beendigung, werden normalerweise viele Befehle in unterschiedlicher Kombination genutzt. Die in dBASE II möglichen Befehle lassen sich nach ihrer Wirkung gruppieren. In diesem Kapitel wird eine Übersicht der Befehle nach solchen Gruppen gegeben. Ausgehend von der beabsichtigten Wirkung soll hiermit die Auswahl des geeigneten Befehls erleichtert werden. Bei der Möglichkeit zum Aufbau von Befehlsdateien hat dBASE II das Konzept moderner Programmiersprachen weitgehend übernommen. Die in der Übersicht hierfür aufgeführten Befehle (siehe: Spezielle Befehle zur Ausführung und zum Aufbau von Befehlsdateien) müssen nach bestimmten Regeln verwendet werden, die in Kapitel 9 näher erläutert sind.

EINRICHTEN VON DATEIEN

Die folgenden Befehle legen eine Datenbank und ggf. weitere, zugehörige Dateien an:

- * CREATE — Einrichten einer Datenbank
- * COPY — erzeugt eine Datei als Kopie einer Datenbank
- * MODIFY — erlaubt die Änderung der Struktur einer bestehenden Datenbank
- * REPORT — legt eine Formatdatei (M) mit dem Schema für einen Bericht an
- * SAVE — speichert temporäre Variable in eine Datei ab
- * INDEX — erzeugt eine Schlüsseldatei
- * REINDEX — aktualisiert eine alte Schlüsseldatei
- * JOIN — extrahiert bestimmte Daten aus zwei Datenbanken und legt sie in einer dritten ab
- * TOTAL — summiert Werte in einer Datenbank und schreibt sie in eine neue Datenbank

ERFASSEN VON DATEN

Mit diesen Befehlen lassen sich Daten satzweise erfassen, d.h. die Datenfelder von Sätzen werden für Eintragungen angeboten.

- * APPEND — Anfügen von Datensätzen am Ende der Datenbank
- * CREATE — erlaubt Datenerfassung nach Einrichtung der Datenbank
- * INSERT — Einfügen von Sätzen in einer Datenbank

VERÄNDERN VON DATEN

Vorhandene Daten bzw. Sätze lassen sich mit folgenden Befehlen ändern.

- * CHANGE — Änderung von Feldern über mehrere Sätze hinweg
- * BROWSE — Anzeige mehrerer Sätze/Datenfelder unter Ausnutzung des ganzen Bildschirms (Seitenmodus)
- * DELETE — markiert Sätze für spätere Löschung
- * EDIT — zur Änderung bestimmter Sätze / Datenfelder
- * PACK — entfernt zur Löschung markierte Sätze aus der Datenbank (Komprimierung)
- * RECALL — entfernt die Markierung zur Löschung von Sätzen (Sätze sind wieder vollwertiger Bestandteil der Datenbank)
- * REPLACE — überschreibt die Eintragung von Datenfeldern
- * READ — liest Eintragungen in Felder aus einer Bildschirmmaske
- * UPDATE — ermöglicht die Änderung von Daten mit einem Befehl über die gesamte Datenbank (ohne Dialog)

BEFEHLE ZUR UNTERSTÜTZUNG DES ANWENDERS

Mit den folgenden Befehlen kann sich der Anwender aktuelle Übersichten anzeigen lassen.

- * DISPLAY — liefert Informationen über aktivierte Datenbankdateien und aktuelle
STATUS Systemparameter
- * DISPLAY — zeigt Informationen über alle Datenbankdateien auf dem aktuellen
FILES Laufwerk an
- * HELP — Kurzerläuterungen zu dBASE-Befehlen

BEFEHLE ZUR ANZEIGE VON DATEN

Zur Ausgabe von Daten auf Peripheriegeräten dienen diese Befehle:

- * @ — gibt Daten in vom Anwender festgelegtem Format am Bildschirm oder
auf dem Drucker aus
- * BROWSE — zeigt von bis zu 19 Sätzen so viele Felder an, wie die Bildschirmbreite
zuläßt
- * COUNT — zählt die Anzahl der Sätze, die eine bestimmte Bedingung erfüllen
- * DISPLAY — zeigt ganze Sätze, einzelne Felder und die Ergebnisse von Ausdrücken an
- * READ — zeigt im Seitenmodus Daten an und ermöglicht deren Änderung
- * REPORT — stellt Daten zu einem Bericht zusammen und gibt diesen aus
- * SUM — berechnet die Summe arithmetischer Ausdrücke über mehrere Datensätze
und zeigt sie an
- * ? — zeigt die Ergebnisse von Ausdrücken an
- * TEXT — bringt einen Textblock zur Anzeige

BEFEHLE ZUR POSITIONIERUNG IN DATENBANKDATEIEN

Der interne Satzzeiger wird durch folgende Befehle verändert:

- * CONTINUE — sucht den nächsten Satz, der der im letzten LOCATE-Befehl angegebenen
Bedingung entspricht
- * FIND — sucht in indizierten Dateien den dem angegebenen Schlüssel zugeordneten
Satz
- * GOTO — ausdrückliche Positionierung auf einen bestimmten Satz
- * LOCATE — sucht den nächsten Satz, der einer Bedingung entspricht
- * SKIP — positioniert vorwärts oder rückwärts relativ zum aktuellen Satz

BEFEHLE MIT AUSWIRKUNGEN AUF GANZE DATEIEN

Folgende Befehle beziehen sich jeweils auf eine komplette Datei:

- * APPEND — Anfügen von dBASE-Datenbankdateien oder umgewandelten ASCII-Dateien (SDF-Option) an die aktivierte Datenbankdatei
- * COPY — erzeugt Kopien einer Datenbankdatei in dBASE- oder ASCII-Format
- * DELETE — löscht Dateien
- * DO — Angabe einer Befehlsdatei, deren Befehle als nächstes ausgeführt werden sollen
- * RENAME — Umbenennen von Dateien
- * SELECT — Umschalten zwischen zwei aktivierten Datenbanken
- * SORT — erzeugt eine anhand eines Datenfeldes sortierte Kopie einer Datenbankdatei
- * USE — aktiviert eine Datenbankdatei und stellt sie zur weiteren Bearbeitung (bis zum nächsten USE-Befehl) zur Verfügung

BEFEHLE ZUM EINRICHTEN/ ÄNDERN VON TEMPORÄREN VARIABLEN

Folgende Befehle erzeugen bzw. verändern temporäre Variable, hier kurz Variable genannt.

- * ACCEPT — speichert eine Zeichenreihe in eine Variable
- * COUNT — weist den Wert eines Zählers einer Variablen zu
- * DISPLAY — zeigt den Wert von Variablen an
- * INPUT — weist den Wert eines eingegebenen Ausdrucks einer Variablen zu
- * RESTORE — restauriert in einer Datei abgelegte Variable im Arbeitsspeicher
- * SAVE — überträgt vorhandene Variable in eine Datei
- * STORE — weist den Wert eines Ausdrucks einer Variablen zu
- * SUM — übergibt aufsummierte Werte an Variable
- * WAIT — nimmt ein einzelnes Zeichen als Eingabe an und legt es in einer Variablen ab

SPEZIELLE BEFEHLE ZUR ANWENDUNG UND ZUM AUFBAU VON BEFEHLS DATEIEN

Diese Befehle dienen der Steuerung des Ablaufs bzw. dem Aufbau von Befehlsdateien.

- * ACCEPT — weist eine eingegebene Zeichenreihe einer Variablen zu
- * CANCEL — bricht die Ausführung einer Befehlsdatei ab
- * DO — ruft Befehlsdateien zur Ausführung auf bzw. ermöglicht das Programmieren von Schleifen
- * IF — stellt Bedingungen bei Ausführung von Befehlen
- * ELSE — Alternative bei Ablaufsteuerung mittels "IF"
- * ENDDO — Abschluß einer "DO WHILE" — Schleife
- * ENDIF — Abschluß eines "IF"-Befehles
- * INPUT — weist den Wert eines eingegeben Ausdrucks einer Variablen zu
- * LOOP — führt auf den Anfang einer "DO WHILE" Schleife zurück
- * MODIFY — zum Schreiben bzw. Ändern einer Befehlsdatei
- COMMAND
- * RETURN — beendet die Ausführung einer Befehlsdatei
- * SET — erlaubt die Einstellung von Parametern in dBASE
- * WAIT — unterbricht die Ausführung einer Befehlsdatei

BEFEHLE ZUR STEUERUNG VON GERÄTEN

Sie steuern die Ausgabe auf Peripheriegeräten (z.B. Drucker)

- * EJECT — bewirkt Übergang auf nächste Seite beim Ausgabegerät
- * ERASE — löscht die auf dem Bildschirm vorhandenen Ausgaben / Anzeigen

8.0 SEITENMODUS

Der Seitenmodus, der bei Verwendung der Befehle APPEND, BROWSE, CREATE, EDIT, INSERT und MODIFY genutzt werden kann, stellt eine bequeme Möglichkeit dar, die Feldinhalte eines Datensatzes schnell einzugeben und zu ändern, da im Seitenmodus der Cursor frei positionierbar ist. Dies bedeutet, daß die einzelnen Datenfelder in beliebiger Reihenfolge bearbeitet werden können.

Bei der folgenden Beschreibung der Tastenkombinationen für den Seitenmodus steht "ctl" für die CONTROL-Taste (auf den meisten Tastaturen mit CTRL, CNTR, CTL, ALT oder ALTERNATE bezeichnet). Sie muß gleichzeitig mit der Taste niedergehalten werden, die bei der jeweiligen Tastenkombination angegeben ist, wobei beim Niederdrücken der CONTROL-Taste ein geringfügiger Vorsprung einzuräumen ist, d.h. "ctl-X" bedeutet

1. CONTROL-Taste drücken und niederhalten
2. Taste X drücken, wobei die CONTROL-Taste gedrückt bleibt

8.1 Tastenkombinationen zur Cursor-Steuerung

- ctl-E,A — setzt den Cursor auf das vorangegangene Datenfeld zurück
- ctl-X,F — rückt den Cursor auf das nächste Datenfeld vor
- ctl-S — setzt den Cursor um ein Zeichen nach links
- ctl-D — setzt den Cursor um ein Zeichen nach rechts
- ctl-Y — füllt das aktuelle Datenfeld mit Leerzeichen
- ctl-V — schaltet um zwischen überschreibender und einfügender Eingabe
- ctl-G — löscht das Zeichen an der Cursorposition
- DELETE/
RUBOUT — löscht das Zeichen links vom Cursor
- ctl-Q — bricht den Seitenmodus ab und leitet zum dBASE-Befehlsmodus über. Die durchgeführten Änderungen bleiben dabei unwirksam, d.h. sie werden nicht gespeichert.

8.2 Zusätzliche Tastenkombinationen für den EDIT-Befehl

- ctl-U — setzt bzw. löscht die Löschmarkierung "*" des aktuellen Datensatzes
- ctl-R — speichert den aktuellen Datensatz auf der Diskette und zeigt den vorangegangenen Datensatz an
- ctl-C — speichert den aktuellen Datensatz auf der Diskette und zeigt den nächsten Datensatz an
- ctl-W — speichert den aktuellen Datensatz auf der Diskette und beendet den Seitenmodus (bei Superbrain: ctl-O)

8.3 Zusätzliche Tastenkombinationen für den MODIFY-Befehl

- ctl-N — fügt ein neues Datenfeld an der Cursorposition ein; alle folgenden Datenfelder werden um eine Position nach hinten verschoben
- ctl-T — löscht das Datenfeld an der Cursorposition; alle folgenden Datenfelder rücken um eine Position vor
- ctl-C — zeigt die folgenden Datenfelder an (nächste Bildschirmseite)
- ctl-R — zeigt die vorangegangenen Datenfelder an (vorangehende Bildschirmseite)
- ctl-W — speichert den aktuellen Datensatz auf der Diskette und beendet den Seitenmodus (bei Superbrain: ctl-O)
- ctl-Q — beendet den Seitenmodus, ohne die Änderungen zu speichern

8.4 Zusätzliche Tastenkombinationen für die Befehle APPEND, CREATE und INSERT

- ctl-C /
ctl-R — speichert den aktuellen Datensatz auf der Diskette und rückt vor zum nächsten Satz
- RETURN/ — nur wenn der Cursor am Anfang des ersten Daten-ENTER Feldes steht und am aktuellen Satz keine Änderungen vorgenommen wurden: beendet Seitenmodus und leitet in dBASE-Befehlsmodus über
- ctl-U — setzt bzw. löscht die Löschmarkierung "*" des aktuellen Datensatzes

8.5 Zusätzliche Tastenkombinationen für den BROWSE-Befehl

- ctl-U — setzt bzw. löscht die Löschmarkierung "*" des aktuellen Datensatzes
- ctl-R — speichert den aktuellen Datensatz auf der Diskette und zeigt den vorangegangenen Datensatz an
- ctl-C — speichert den aktuellen Datensatz auf der Diskette und zeigt den nächsten Datensatz an
- ctl-W — speichert den aktuellen Datensatz auf der Diskette und beendet den Seitenmodus (bei Superbrain: ctl-O)
- ctl-Z — verschiebt das Fenster um ein Datenfeld nach links
- ctl-B — verschiebt das Fenster um ein Datenfeld nach rechts

9.0 dBASE-BEFEHLE

In diesem Kapitel finden sich ausführliche Beschreibungen aller dBASE-Befehle. Vor dem Lesen der einzelnen Befehlsdefinitionen sollte sich der Benutzer jedoch zunächst mit den in den Abschnitten 9.1 und 9.2 beschriebenen Grundlagen der dBASE-Kommandosprache vertraut machen.

9.1 SYMBOL-DEFINITIONEN

Die Schreibweise der dBASE-Befehle ist im einzelnen in Abschnitt 2 unter Verwendung einer speziellen Symbolik dargestellt. Um die Befehlsdefinitionen in vollem Umfang verstehen und die Mächtigkeit der einzelnen Befehle erfassen zu können, wird dem Benutzer ein eingehendes Studium der folgenden Übersicht empfohlen.

Symbol	Bedeutung
< Befehlsfolge >	eine beliebige Folge zulässiger und vollständiger dBASE-Befehle
< Zeichenreihe >	eine beliebige Zeichenreihe. Unter einer Zeichenreihe versteht man die Zeichen, die zwischen einfachen (') oder doppelten Anführungszeichen (") oder eckigen Klammern ([]) eingeschlossen sind.
< Begrenzung >	ein beliebiges Sonderzeichen, Sonderzeichen sind bei dBASE II "()*=, @".
< Ausdruck >	Ein Ausdruck entsteht durch die sinnvolle Verknüpfung von Zahlen, Funktionen, Zeichenreihen oder Feldbezeichnungen. Beispiele: 1.) 4 + 8 2.) doc = '3' .or. doc = '4' 3.) \$('abc' + &somestr,n,3) = 'abcdefg'
< Ausdrucksfolge >	eine Folge von < Ausdrücken >, die jeweils durch Komma voneinander getrennt sind
< Feld >	die Bezeichnung eines beliebigen Datenfeldes
< Felderfolge >	eine Folge von Datenfeldbezeichnungen, die jeweils durch Komma voneinander getrennt sind
< Dateiname >	ein beliebiger Dateiname, der den Konventionen für Dateinamen entspricht (vgl. Abschnitt 3.0)

Symbol	Bedeutung
<Formatdatei>	der Name einer Formatdatei. Formatdateien haben die Typbezeichnung .FRM und werden durch den REPORT-Befehl erzeugt. Sie enthalten Informationen über die Titel, Überschriften, Zeilen- und Spaltenbezeichnungen, die in einem REPORT-Befehl benutzt werden sollen. Sie können mit Hilfe von Textverarbeitungsprogrammen oder Editoren geändert werden, doch erweist es sich in der Regel als weniger aufwendig, eine neue Formatdatei im REPORT-Dialog zu erzeugen. Weitere Informationen über Formatdateien finden sich bei der Beschreibung des REPORT-Befehls.
<Schlüsseldatei>	der Name einer Schlüsseldatei. Schlüsseldateien haben die Typbezeichnung .NDX und werden durch den INDEX-Befehl erzeugt. Sie enthalten Schlüssel und Verweise auf einzelne Sätze der Datenbank. Durch Indizierung lassen sich auch bei großen Datenbanken Suchvorgänge erheblich beschleunigen. Weitere Informationen über Schlüsseldateien finden sich bei der Beschreibung des INDEX-Befehls.
<Schlüssel>	eine Feldbezeichnung, die als Ordnungskriterium einer Datenbank dient. Schlüsseln kommt bei Datenbanksystemen eine hohe Bedeutung zu. Zu jeder Datenbank können mehrere Schlüsseldateien für jeweils verschiedene Schlüsselfelder bzw. ihre Kombinationen angelegt werden. Neben Feldbezeichnungen sind auch <Ausdrücke> als Schlüssel zulässig. Nähere Einzelheiten finden sich bei der Definition des INDEX-Befehls.
<Variable>	eine beliebige temporäre Variable. Temporäre Variablen dienen der Zwischenspeicherung von Werten zur späteren Wiederverwendung. Sie werden durch Befehle wie STORE, ACCEPT oder INPUT angelegt. dBASE II erlaubt die gleichzeitige Verwendung von bis zu 64 temporären Variablen.

Symbol	Bedeutung
<Variablenfolge>	eine Folge temporärer Variablen, die durch Komma getrennt sind
<n>	eine Ziffernfolge. Ziffernfolgen sind Zahlen, die sich nicht aus Berechnungen oder Variablen ergeben. So sind z.B. "4" oder "9876" Ziffernfolgen, "4 + 8" hingegen nicht.
<Geltungsbereich>	der Bereich der Datenbank, auf den der Befehl angewendet werden soll
ALL	Als <Geltungsbereich> sind folgende Angaben zulässig: — alle Sätze einer Datenbankdatei werden - unabhängig von der aktuellen Satznummer - von dem gegebenen Befehl betroffen. Bei einigen Befehlen ist ALL die Voreinstellung für den Geltungsbereich, bei anderen - insbesondere bei Befehlen, die die Löschung von Information ermöglichen, wie z.B. DELETE - umfaßt die Voreinstellung nur den aktuellen Datensatz.
NEXT <n>	— die nächsten n Sätze einschließlich des aktuellen Datensatzes (n muß eine explizit angegebene Ziffernfolge sein; s.o.)
RECORD <n>	— nur der Satz mit der Nummer n (n muß eine explizit angegebene Ziffernfolge sein; s.o.) Durch folgende Zusätze läßt sich der <Geltungsbereich> darüberhinaus logisch eingrenzen:
FOR <Ausdruck>	— alle Sätze, die die logische Bedingung des <Ausdrucks> erfüllen. Ist nichts Anderes angegeben, so impliziert das Vorhandensein einer FOR-Klausel den Geltungsbereich ALL.
WHILE <Ausdruck>	— alle Sätze, die auf den aktuellen Datensatz folgen, bis die im <Ausdruck> angegebene logische Bedingung nicht mehr erfüllt ist. Die Befehlsausführung endet beim ersten Satz, für den der <Ausdruck> den Wahrheitswert "falsch" ergibt. Ist nichts Anderes angegeben, so impliziert das Vorhandensein einer WHILE-Klausel den Geltungsbereich NEXT 65534.

Neben den hier definierten Symbolen können bei den Beschreibungen einzelner Befehle in Abschnitt 2 spezielle Symbole Verwendung finden, die nur auf den jeweiligen Befehl bezogen sind und deshalb in der zugehörigen Befehlsbeschreibung erläutert werden.

9.2 GRUNDREGELN DER dBASE-KOMMANDOSPRACHE

Um erfolgreich mit dBASE II arbeiten zu können, muß der Benutzer - wie z.B. bei Programmiersprachen auch - einige Regeln beachten. Die im folgenden aufgeführten Grundregeln sollen dabei helfen, die in Abschnitt 2 zu jedem Befehl angegebene Syntax richtig zu interpretieren.

1. In einer Befehlszeile muß das erste vom Leerzeichen verschiedene Zeichen zum Befehlswort selbst gehören, während die Zusatzklauseln in beliebiger Reihenfolge angehängt werden können. Befehlswörter sind die in Kapitel 10 erläuterten Befehle, wie z.B. CREATE, APPEND, REPORT, DISPLAY und SKIP. Zusatzklauseln spezifizieren die durch ein Befehlswort eingeleitete Aktion. Sie beginnen mit Wörtern wie FOR, NEXT oder WITH. Die Gesamtheit von Befehlswörtern und Einleitungswörtern von Zusatzklauseln stellen die sogenannten "Steuerwörter" von dBASE II dar. Sie sind immer in Großbuchstaben geschrieben.
2. Die maximal zulässige Länge einer Befehlszeile beträgt 254 Zeichen, worin Makro-Ersetzungen bereits enthalten sein müssen.
3. Einzelne Bestandteile einer Befehlszeile können durch beliebig viele Leerzeichen voneinander getrennt sein. Allerdings zählen diese Leerzeichen bei der Berechnung der Gesamtlänge einer Befehlszeile mit (siehe Regel 2).
4. Steuerwörter werden durch ihre ersten vier Buchstaben eindeutig identifiziert. Dies bedeutet, daß die Abkürzung von Steuerwörtern auf ihre ersten vier Buchstaben (oder mehr) zulässig ist. So könnte z.B. DISPLAY STRUCTURE abgekürzt werden zu DISP STRU oder DISPL STRUCT. Werden mehr als vier Buchstaben angegeben, so müssen auch die zusätzlichen Buchstaben der korrekten Schreibweise des Steuerwortes entsprechen.
5. Zur Eingabe von Befehlen, Steuerwörtern, Feldbezeichnungen, Namen von temporären Variablen oder Dateinamen können sowohl Groß- als auch Kleinbuchstaben verwendet werden.
6. Manche Befehle erlauben Zusätze, die bei Bedarf verwendet werden können. Diese sogenannten Optionen sind in den Befehlsbeschreibungen des Abschnitts 2 in eckige Klammern ([]) eingeschlossen. Sie beeinflussen die Wirkung einiger Befehle erheblich.
7. Die dBASE-Steuerwörter sind keine reservierten Wörter in dem Sinne, daß sie bei anderweitiger Verwendung Fehlermeldungen hervorrufen würden. Doch sei darauf hingewiesen, daß manche Steuerwörter, als Feldbezeichnung oder Dateiname gebraucht, zu unvorhergesehenen Schwierigkeiten führen können. Beispielsweise kann eine Befehlsdatei namens WHILE nicht ausgeführt werden, da der Aufruf "DO WHILE" lauten müßte, was einem Befehl für eine Schleife entspricht. Ähnlich führt die Feldbezeichnung ALL in einer Reihe von Befehlen zu Verwechslungen. Aus diesem Grunde sollte man möglichst auf die Verwendung von dBASE-Steuerwörtern als Feldbezeichnungen oder Dateinamen verzichten.

8. In Befehlsdateien ist auf die korrekte Schachtelung von zusammengesetzten Befehlen zu achten, d.h. der zuerst aufgerufene Befehl muß als letzter abgeschlossen werden, der zweite Befehl als vorletzter usw. Bei der Schachtelung von IF-ELSE-ENDIF- und DO WHILE-ENDDO-Konstruktionen kann es andernfalls zu einem unvorhergesehenen Programmablauf kommen, da dBASE II keine Prüfung auf korrekte Schachtelung durchführt. Bereits bei der Erstellung von Befehlsdateien empfiehlt es sich daher, die einzelnen Befehle entsprechend ihrer Schachtelungstiefe im Text einzurücken, wie auch untenstehendes Beispiel zeigt.

DO WHILE .NOT. EOF

 (Befehle)

 IF A .AND. B

 (Befehle)

 ELSE

 DO WHILE A < = 57

 (Befehle)

 ENDDO

 (Befehle)

 ENDIF

 (Befehle)

ENDDO

Dies ist ein Beispiel für die korrekte Schachtelung zusammengesetzter Befehle. Der IF-ELSE-ENDIF-Befehl liegt komplett innerhalb der äußeren DO-WHILE-ENDDO-Schleife, und auch die zweite DO-WHILE-ENDDO-Konstruktion ist voll in den ELSE-Teil des IF-Befehls eingebettet. Dieses Beispiel ließe sich beliebig fortsetzen, da dBASE II bis zu 16 Schachtelungsebenen zuläßt.

DO WHILE .NOT. EOF

 (Befehle)

 IF A .AND. B

 (Befehle)

 ENDDO

 (Befehle).

 ENDIF

Dieses Beispiel zeigt, wie Befehle nicht geschachtelt werden dürfen. Das Steuerwort ENDDO erscheint vor ENDIF und führt so zu unvorhersagbaren Effekten bei der Ausführung der Befehlsdatei, ohne daß dBASE II einen Fehler meldet.

10.0 MASCHINENSPRACHEN-SCHNITTSTELLE

10.1 SPEICHERBELEGUNGSÜBERSICHT

Der folgenden Tabelle ist zu entnehmen, wie der Arbeitsspeicher während einer dBASE-Sitzung genutzt wird.

CP/M-Arbeitsbereich (Puffer, Dateiverwaltungsblöcke)	0000h 00FFh 0100h
dBASE-Grundsegment (Ein-/Ausgabe, Bildschirmverwaltung, Verarbeitung von Ausdrücken, Arithmetische Berechnungen)	44FFh 4500h
dBASE-Overlay-Bereich (Befehlsverarbeitung)	5FFFh 6000h
dBASE-Arbeitsbereich (Puffer, Kellerspeicher)	A3FFh A400h
wird nur vom SORT-Befehl benötigt	Obergrenze der TPA

10.2 MASCHINENSPRACHEN-BEFEHLE

SET CALL TO <Adresse>

Durch diesen Befehl wird die (dezimale) Adresse festgelegt, zu der beim nächsten CALL-Befehl verzweigt wird.

CALL [<Variable>]

Dieser Befehl bewirkt den Aufruf eines Maschinensprachen Unterprogramms an der Adresse, die durch den letzten SET CALL TO-Befehl spezifiziert wurde. Wird als Parameter eine Zeichenreihen-<Variable> übergeben, so wird diese im Kellerspeicher abgelegt und das Registerpaar HL enthält die Adresse des ersten Bytes. Das Maschinenprogramm darf die Länge der Zeichenreihe auf keinen Fall verändern. Durch einen RET-Befehl erfolgt die Rückkehr zu dBASE.

LOAD [<Dateiname>]

Dieser Befehl lädt eine im INTEL-HEX-Format vorliegende Datei in den Arbeitsspeicher.

POKE <Adresse> <Bytefolge>

Poke erlaubt das Ändern einzelner Bytes im Arbeitsspeicher. Beginnend bei der angegebenen <Adresse> werden die in der <Bytefolge> aufgeführten Werte im Arbeitsspeicher abgelegt. Sowohl die Adresse als auch die einzelnen Bytes sind dabei als Dezimalzahlen anzugeben.

Beispiel:

```
. POKE 42072 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18
```

PEEK(<Adresse>)

PEEK ist eine numerische Funktion und liefert den Wert, der unter der angegebenen <Adresse> im Speicher steht. Dabei muß die Adresse als Dezimalzahl angegeben werden. Das Ergebnis ist ebenfalls eine Dezimalzahl.

Beispiele:

```
. STORE PEEK(128) TO X
    229
```

```
. STORE 41012 TO I
    41012
```

```
. ? PEEK(I), PEEK(I-10), PEEK(I+4)
    7      0      11
```

?

? [<Ausdrucksfolge>]

?? [<Ausdrucksfolge>]

Dieser Befehl stellt eine Sonderform des DISPLAY-Befehls dar und ist gleichbedeutend mit DISPLAY OFF <Ausdrucksfolge>. Mit ihm läßt sich der Wert eines Ausdrucks oder einer Ausdrucksfolge anzeigen. Die <Ausdrucksfolge> kann temporäre Variablen, Feldbezeichnungen, Konstanten und Funktionen enthalten. Ein "?" ohne nachfolgenden <Ausdruck> bewirkt auf dem Ausgabemedium (Bildschirm oder Drucker) einen Zeilenvorschub. Diese Eigenschaft ermöglicht das gezielte Einfügen von Leerzeilen in die Ausgabe.

Die zweite Form dieses Befehls, "??", unterscheidet sich von der oben beschriebenen lediglich dadurch, daß der Zeilenvorschub vor der Ausgabe der <Ausdrucksfolge> unterbleibt.

In der Ausgabe werden die einzelnen Werte jeweils durch ein Leerzeichen voneinander getrennt, es sein denn, ein SET RAW ON Befehl ist wirksam. In diesem Fall unterbleibt die Ausgabe trennender Leerzeichen.

Beispiele:

. USE MITGLIED

. DISPLAY STRUCTURE
STRUCTURE FOR FILE: MITGLIEDF
NUMBER OF RECORDS: 00015
DATE OF LAST UPDATE: 10/17/82
PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	MITGL:NR	N	005	
003	EINTR:JAHR	C	004	
** TOTAL **			00030	

. 4

. ? #

4

. ? NAME

Lehmann, Paul

. ? 5+9

14

(Im folgenden Beispiel ist eine Befehlsdatei angegeben, die den ?-Befehl zur Auflockerung der Bildschirmanzeige durch Einfügen einer Leerzeile verwendet. Ist diese Datei unter dem Namen START.CMD auf Laufwerk B vorhanden, kann sie durch den (CP/M)-Befehl "DBASE B:START" zur Ausführung gebracht werden. Man beachte bei der Ausgabe die eingefügte Leerzeile.)

```
SET DEFAULT TO A
USE BESTAND INDEX BESTAND
DISP STRU
?
ACCEPT "Gib heutiges Datum an" TO DAT
SET DATE TO &DAT
RELEASE DAT
RETURN
```

```
STRUCTURE FOR FILE: BESTAND.DBF
NUMBER OF RECORDS: 00011
DATE OF LAST UPDATE: 10/15/82
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

Gib heutiges Datum an:10/26/82

@

@ <Koordinaten> [SAY <Ausdruck> [USING <Maske>]]
 [GET <Variable> [PICTURE <Maske>]]

Hinweis: Auf einer deutschen Tastatur entspricht dieses Zeichen dem Paragraph-Zeichen "§".

Dieser Befehl bietet im Zusammenwirken mit den Befehlen SET FORMAT TO, ERASE, EJECT, CLEAR GETS und READ dem Benutzer die Möglichkeit, die Darstellung von Ausgaben auf Bildschirm oder Drucker selbst festzulegen. Sowohl die Wirkung des @-Befehls als auch die der anderen oben aufgeführten Befehle hängt entscheidend davon ab, welcher Parameter zuvor im SET FORMAT-Befehl angegeben wurde. Eine ausführliche Erläuterung dieser Abhängigkeiten folgt unten.

Die <Koordinaten> des @-Befehls beziehen sich entweder auf den Bildschirm oder auf den Drucker und werden immer als ein Paar "x,y" angegeben, wobei "x" die Nummer der Zeile und "y" die Nummer der Spalte bezeichnet. Bei den meisten Bildschirmen (24 Zeilen auf 80 Spalten) umfaßt der zulässige Wertebereich für "x" die Zahlen 0-23, der für "y" die Zahlen 0-79. Dabei sollte die Zeile 0 nicht genutzt werden, da sie für dBASE-Nachrichten reserviert ist. Bei Verwendung des @-Befehls zur Druckersteuerung läßt dBASE für beide Koordinaten Werte zwischen 0 und 254 zu. In allen Fällen können die Koordinaten explizit als Konstanten, durch temporäre numerische Variablen oder in Form numerischer Ausdrücke angegeben werden. Ob dBASE sie als Bildschirm- oder Druckerkoordinaten interpretiert, bestimmt der Benutzer zuvor durch einen SET FORMAT-Befehl.

Möglich ist auch die Angabe relativer <Koordinaten> für Bildschirm bzw. Drucker in der Form "\$ + <Ausdruck>". Bezugspunkt für relative <Koordinaten> ist diejenige Bildschirm- oder Druckerposition, die durch den vorangegangenen @-Befehl und der mit ihm verbundenen Ausgabe erreicht wurde. So erzeugen z.B. die beiden Befehle @ 12,10 SAY "FRÜH" und @ \$,\$+1 SAY "STÜCK" das Wort "FRÜHSTÜCK" in Zeile 12, beginnend in Spalte 10. In einem @-Befehl können sowohl die Zeile als auch die Spalte relativ adressiert werden. Der Abstand zum Bezugspunkt muß allerdings immer positiv sein, d.h. "\$ - <Ausdruck>" ist als relative Koordinate nicht zulässig.

Ging dem @-Befehl ein SET FORMAT TO SCREEN voran, erfolgt die Ausgabe auf dem Bildschirm. Dies ist auch die Voreinstellung, falls kein SET FORMAT-Befehl erteilt wurde. Beispielsweise entspricht beim Bildschirm dessen obere linke Ecke (sogenannte "Home-Position") dem Koordinatenpaar 0,0; das Paar 10,15 bezeichnet die Position am Schnittpunkt der 11. Zeile und der 16. Spalte. Die Reihenfolge, in der einzelne Zeilen und Spalten durch den @-Befehl angesprochen werden, ist für Bildschirmausgaben unerheblich, d.h. eine Ausgabe in Zeile 15 kann z.B. durchaus einer Ausgabe in Zeile 10 vorangehen. Wie bereits oben erwähnt, sollte die Zeile 0 nicht in die Ausgaben einbezogen werden.

Ging dem @-Befehl ein SET FORMAT TO PRINT voran, erfolgt die Ausgabe auf dem Drucker. Auch hier entspricht das Koordinatenpaar 0,0 der oberen linken Papierecke. Im Gegensatz zu Bildschirmausgaben müssen @-Befehle an einen Drucker in der Reihenfolge erteilt werden, in der die Ausgabezeilen auf dem Papier erscheinen sollen, also z.B. Zeile 5 vor Zeile 6, Zeile 10 vor Zeile 20 usw. Andernfalls verschwendet man eine Menge Papier, da ein Seitenvorschub erfolgt, wenn ein @-Befehl eine niedrigere Zeilenkoordinate enthält als der vorangegangene @-Befehl.

Nach einem SET FORMAT TO SCREEN bewirkt der Befehl ERASE die Löschung des gesamten Bildschirms, die Deaktivierung aller GET-Klauseln (siehe unten) und die Reinitialisierung der Koordinaten mit 0,0. Eine ähnliche Funktion erfüllt im Falle eines SET FORMAT TO PRINT der Befehl EJECT. Er bewirkt einen Seitenvorschub, wobei ebenfalls die Koordinaten auf 0,0 gesetzt werden.

Mit der SAY-Klausel läßt sich ein <Ausdruck> anzeigen, der nicht durch den Benutzer verändert werden soll. Es empfiehlt sich, für das gewünschte Anzeige- bzw. Druckbild des <Ausdrucks> mittels der USING-Klausel eine <Maske> anzugeben (siehe unten), da andernfalls unerwünschte Seiteneffekte auftreten können.

Während SAY-Klauseln sowohl bei Bildschirm- als auch bei Druckerausgaben Verwendung finden können, werden GET-Klauseln nur nach einem SET FORMAT TO SCREEN ausgeführt.

Eine GET-Klausel bewirkt die Anzeige des aktuellen Inhalts einer bereits existierenden <Variablen> (Datenbank- oder temporäre Variable). Im Gegensatz zur SAY-Klausel kann der durch GET angezeigte Wert vom Benutzer überschrieben und mit einem nachfolgenden READ-Befehl eingelesen werden. Eine in der PICTURE-Klausel angegebene <Maske> dient dabei neben der Formatierung der Anzeige auch der Plausibilitätsprüfung der Eingabe (siehe READ-Befehl). Fehlt die PICTURE-Klausel, wird die Plausibilitätsprüfung von dBASE anhand des Typs der <Variablen> durchgeführt.

Handelt es sich bei der angegebenen <Variablen> um einen Wahrheitswert, so akzeptiert dBASE nur die Buchstaben 'T', 'F', 'Y' und 'N' (in Groß- oder Kleinschreibung) als Eingabe.

Bis zu 64 GET-Klauseln können gleichzeitig aktiviert sein. Mittels der Befehle ERASE oder CLEAR GETS lassen sich alle aktiven GET's deaktivieren.

Enthält ein @-Befehl nach einem SET FORMAT TO SCREEN weder eine SAY- noch eine GET-Klausel, so wird die Ausgabezeile ab der Koordinatenposition mit Leerzeichen gefüllt. Der Befehl @ 10,0 bewirkt also beispielsweise die Löschung der gesamten elften Zeile.

Enthält ein @-Befehl zur Bildschirmausgabe (SET FORMAT TO SCREEN) eine GET-Klausel, so wird er erst durch einen nachfolgenden READ-Befehl abgeschlossen, der die neuen Werte der <Variablen> einliest. Da ein @-Befehl zur Druckerausgabe (SET FORMAT TO PRINT) keine GET-Klausel enthalten darf, erübrigt sich in diesem Fall auch ein READ-Befehl.

Die Möglichkeit, <Koordinaten> durch temporäre Variablen anzugeben, erweist sich besonders bei der Erstellung von Tabellen und anderen Ausgaben, die einem starren Schema unterliegen, von Vorteil, wie der folgende Auszug aus einer Befehlsdatei zeigt.

```

SET FORMAT TO PRINT
GOTO TOP
STORE 7 TO ZEILE
DO WHILE .NOT. EOF
    IF ZEILE >= 50
        EJECT
        STORE 7 TO ZEILE
    ENDIF
    @ ZEILE,12 SAY P USING 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
    @ ZEILE,48 SAY D USING 'XXXXXXXXXX'
    @ ZEILE,64 SAY P1 USING 'XXXXXXXXXXXXXXXXXXXXX'
    @ ZEILE,88 SAY U USING 'XXXXXXXXXX'
    @ ZEILE,104 SAY P2 USING 'XXXXXXXXXXXXXXXXXXXXX'
    IF R <> 0
        @ ZEILE,130 SAY R USING '9999'
    ENDIF
    STORE ZEILE + 1 TO ZEILE
SKIP
ENDDO
RETURN

```

In diesem Beispiel besteht jede Druckerseite aus 57 Zeilen. Die ersten sieben Zeilen bilden den oberen Rand der Seite und bleiben leer. Die übrigen Zeilen enthalten jeweils die Daten von fünf bzw. sechs Datenbankfeldern. Bei der Verwendung von <Masken> ist darauf zu achten, daß diese für die aufzunehmenden Werte ausreichend dimensioniert sind, da andernfalls die hinteren Stellen des auszugebenden Wertes abgeschnitten werden. Die Bedeutung der in obigem Beispiel verwendeten Masken ist aus untenstehender Tabelle zu entnehmen.

Werden bei der Druckerausgabe (SET FORMAT TO PRINT) mehrere @-Befehle für dieselbe Zeile angegeben, so ist darauf zu achten, daß keiner dieser Befehle auf einer Spaltenposition beginnt, die bereits durch einen vorangegangenen @-Befehl abgedeckt wurde, da dBASE nur in diesem Fall Gewähr für eine korrekte Ausgabe bietet. Auf Bildschirmausgaben (SET FORMAT TO SCREEN) trifft diese Einschränkung nicht zu; hier werden die ausgegebenen Daten vorangegangener @-Befehle überschrieben.

Eine weitere Form des SET FORMAT-Befehls lautet SET FORMAT TO <Maskendatei>. Hat der Benutzer diesen Befehl aufgerufen, so werden bei jedem nachfolgenden READ-Befehl die @-Befehle der angegebenen <Maskendatei> wirksam. Auf diese Weise läßt sich der Bildschirm je nach Verwendungszweck "maßschneidern". Es sei jedoch an dieser Stelle ausdrücklich darauf hingewiesen, daß die Verwendung von Maskendateien für den Einsatz des @-Befehls keineswegs notwendig ist, da letzterer auch in Befehlsdateien enthalten sein kann. Nähere Informationen über den Einsatz von Maskendateien finden sich beim READ-Befehl.

Masken:

Sowohl die USING- als auch die PICTURE-Klausel ziehen eine <Maske> nach sich. Eine solche Maske besteht aus einer Folge von Zeichen, die angeben, in welcher Darstellung eine Ausgabe erfolgen soll bzw. welche Werte als Eingaben bei einer GET-Klausel zulässig sind. Die folgende Tabelle gibt die in Masken zulässigen Zeichen und deren Bedeutung an:

Maskenzeichen	Bedeutung bei USING	Bedeutung bei PICTURE
# oder 9	bewirkt die Ausgabe einer Ziffer	erlaubt nur die Eingabe einer der Ziffern (1,2, ... ,8,9,0) oder eines der Sonderzeichen ".", "+", "-", und " " (Leerzeichen)
X	bewirkt die Ausgabe eines beliebigen Zeichens (Buchstabe, Ziffer, Sonderzeichen)	erlaubt die Eingabe eines beliebigen Zeichens (Buchstabe, Ziffer, Sonderzeichen)
A	wie X	erlaubt nur die Eingabe eines Buchstaben
\$ oder *	bewirkt die Ausgabe einer Ziffer oder eines "\$" bzw. "*" anstelle einer führenden Null	Ausgabe des anstehenden Zeichens; keine Wirkung auf Eingaben
!	ohne Auswirkung	wandelt Kleinbuchstaben in Großbuchstaben um

Beispiele:

. @ 5,1 SAY 'Gib Telefon-Nr. ein' GET TEL:NR PICTURE '(99999)999999'

(Bei diesem Beispiel würde auf dem Bildschirm die Aufforderung 'Gib Telefon-Nr. ein' gefolgt von der Zeichenfolge '(bbbbbb)bbbbbb' erscheinen, wenn TEL:NR zuvor noch kein Wert zugewiesen worden war ("b" steht in diesem Falle für jeweils ein Leerzeichen). Beim nächsten READ-Befehl akzeptiert dBASE anstelle der Leerzeichen nur Ziffern, also z.B. die Eingabe '(01234)555555'. Alle in einer Maske enthaltenen Zeichen, denen nach der obigen Tabelle keine besondere Bedeutung zukommt, werden als Bestandteile des Variablenwertes übernommen. Im obigen Beispiel trifft dies auf die Klammern zu.)

. @ 10,50 SAY STUNDEN * LOHN USING '*****.99'

(Dieser @-Befehl könnte sowohl für den Bildschirm als auch für den Drucker bestimmt sein, da er keine GET-Klausel enthält. In der Ausgabe erscheinen Sterne nur anstelle führender Nullen, so daß beispielsweise bei 40 STUNDEN und einem LOHN von 12.50 die Ausgabe '****500.00' lauten würde. Diese Eigenschaft des @-Befehls läßt sich insbesondere beim maschinellen Ausfüllen von Schecks und Rechnungsformularen nutzen.)

(Sind im ganzzahligen Teil einer Zahlenmaske Kommata enthalten, so erscheinen diese nur in der Ausgabe, wenn ihnen signifikante Ziffern vorangehen. Andernfalls werden sie durch das links von ihnen stehende Maskenzeichen ersetzt.)

. @ 10,50 SAY STUNDEN * LOHN USING '***,***.99'

(Mit denselben Daten wie oben würde dieser Befehl die Ausgabe '****500.00' und nicht '***,500.00' erzeugen.)

Das nächste Beispiel zeigt, wie man zunächst durch mehrere @-Befehle den Bildschirm füllen und mit einem einzigen nachfolgenden READ-Befehl alle Eingaben des Benutzers einlesen kann. Hier werden an die aktivierte Datenbank Sätze angehängt, bis für das Feld NAME eine einzelne Null eingegeben wird. Da der letzte Satz keine Information enthält, wird er abschließend gelöscht.

SET FORMAT TO SCREEN

USE B:BEISPIEL

ERASE

DO WHILE NAME # '0'

APPEND BLANK

@ 5,0 SAY "Gib nächsten Namen ein" ;

GET NAME PICTURE 'XXXXXXXXXXXXXXXXXXXXX'

@ 6,0 SAY "Gib Straße und Hausnummer ein" ;

GET STRASSE PICTURE 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXX'

@ 7,0 SAY "Gib Postleitzahl ein" ;

GET PLZ PICTURE '9999'

@ 7,5 SAY "Gib Ortsnamen ein" ;

GET ORT PICTURE 'XXXXXXXXXXXXXXXXXXXXX'

READ

ENDDO

GOTO BOTTOM

DELETE

PACK

LIST

RETURN

Folgende Befehle beeinflussen die Wirkungsweise des @-Befehls:

- * SET INTENSITY ON/OFF (Voreinstellung: ON) schaltet von zwei auf eine Helligkeitsstufe und umgekehrt bei SAY- UND GET- Klauseln um.
- * SET BELL ON/OFF (Voreinstellung: ON) schaltet den akustischen Alarm für unzulässige Eingaben und das Erreichen von Feldgrenzen ein bzw. aus.
- * SET COLON ON/OFF (Voreinstellung: ON) aktiviert bzw. deaktiviert die Markierung der Feldgrenzen von GET-Variablen durch Doppelpunkte.
- * SET DEBUG ON/OFF (Voreinstellung: OFF) aktiviert bzw. deaktiviert die Ausgabe von ECHO- und STEP-Meldungen auf den Drucker.
- * SET SCREEN ON/OFF (Voreinstellung: ON) aktiviert und deaktiviert den Seitenmodus.
- * SET FORMAT TO SCREEN/PRINT/<Maskendatei> (Voreinstellung: SCREEN) spezifiziert das Ausgabegerät (SCREEN = Bildschirm, PRINT = Drucker). SET FORMAT TO <Maskendatei> liefert dem READ-Befehl den Namen einer Maskendatei, in der die zu verwendenden @-Befehle enthalten sind.
- * READ schaltet den Seitenmodus ein, so daß die durch GET angegebenen Variablen eingegeben bzw. geändert werden können.

ACCEPT

ACCEPT ["<Text>"] TO <Variable>

Dieser Befehl ermöglicht die Eingabe alphanumerischer Werte in eine temporäre Variable. Im Unterschied zu INPUT braucht der Benutzer bei ACCEPT seine Eingaben jedoch nicht in Anführungszeichen einzuschließen, da ACCEPT jede Eingabe als alphanumerischen Wert auffaßt, während INPUT den Variablentyp anhand der Schreibweise der Eingabe festlegt.

Existiert die angegebene <Variable> noch nicht, so wird sie angelegt und die Eingabe des Benutzers in ihr gespeichert. Enthält der ACCEPT-Befehl vor dem Steuerwort TO einen in Anführungszeichen, Hochkomma oder eckige Klammern eingeschlossenen <Text> (Beidseitig das gleiche Zeichen verwenden!), so erscheint dieser Text von einem Doppelpunkt gefolgt auf dem Bildschirm, um den Nutzer zur Eingabe aufzufordern. Bei einer "leeren Eingabe" (d.h. nur RETURN-Taste drücken) enthält die <Variable> ein Leerzeichen.

Beispiele:

. ACCEPT "Bitte Namen eingeben" TO NAME

Bitte Namen eingeben:Hans Schneider

. ACCEPT "Bitte weiteren Namen eingeben" TO NAME:WEIT

Bitte weiteren Namen eingeben:Müller, Paul

. DISPLAY MEMORY

NAME	(C)	Hans Schneider	
NAME:WEIT	(C)	Müller, Paul	
** TOTAL **		02 VARIABLES USED	00026 BYTES USED

. ACCEPT TO ALLES

:Alles Mögliche

. DISP MEMO

NAME	(C)	Hans Schneider	
NAME:WEIT	(C)	Müller, Paul	
ALLES	(C)	Alles Mögliche	
** TOTAL **		03 VARIABLES USED	00040 BYTES USED

APPEND

- a. APPEND FROM <Dateiname> [FOR <Ausdruck>] [SDF]
WHILE <Ausdruck>] [DELIMITED]
- b. APPEND BLANK
- c. APPEND

Bei allen drei Formen des APPEND-Befehls werden an die durch USE aktivierte Datenbank Sätze angehängt. Neben APPEND erlauben nur die Befehle CREATE und INSERT das Hinzufügen von Sätzen zu einer Datenbank. Während man mit APPEND und CREATE mehrere Sätze hinzufügen kann, ermöglicht INSERT nur das Einfügen eines weiteren Satzes.

In der ersten Form des Befehls (a) werden die anzuhängenden Sätze aus der Datei mit dem angegebenen <Dateinamen>, die im weiteren kurz Quelldatei genannt wird, ausgelesen. Sind im Befehl weder die Steuerwörter SDF noch DELIMITED angegeben, so wird vorausgesetzt, daß es sich bei der Quelldatei um eine Datenbankdatei im dBASE-Format handelt. Die Strukturen der aktivierten Datenbank und der Quelldatei werden miteinander verglichen und die Felder, die in beiden Datenbanken vorhanden sind, werden in den anzuhängenden Satz übernommen. Sind die neu eingelesenen Feldinhalte kürzer als die Felder der aktiven Datenbank, so werden sie rechts mit Leerzeichen aufgefüllt. Sind sie länger, so werden sie durch rechtsseitiges Abschneiden auf das Format der bereits vorhandenen Felder gebracht. An die aktivierte Datenbank werden so lange neue Sätze angehängt, bis die Quelldatei erschöpft ist.

Ist die Option SDF (System Data Format) angegeben, so geht dBASE davon aus, daß die Quelldatei in Standard-ASCII-Format vorliegt. Auch hier erfolgt rechtsseitiges Auffüllen mit Leerzeichen bzw. Abschneiden, wenn die Feldlängen nicht übereinstimmen.

Ist das Steuerwort DELIMITED im APPEND-Befehl enthalten, so erwartet dBASE, daß die Daten der Quelldatei durch Sonderzeichen voneinander abgegrenzt werden. So erzeugen beispielsweise viele Programmiersprachen Dateien, bei denen Felder durch Kommata voneinander getrennt und Zeichenreihen in Hochkomma (') oder Anführungszeichen (") eingeschlossen sind. In diesen Fällen löscht dBASE die jeweiligen Sonderzeichen und fügt die Daten im dBASE-Format an die aktivierte Datenbank an.

Ist eine FOR-Klausel im APPEND-Befehl vorhanden, so überprüft dBASE für jeden Satz der Quelldatei, ob er die im <Ausdruck> angegebene Bedingung erfüllt. Ist dies der Fall, wird er angehängt, andernfalls geht dBASE zum nächsten Satz der Quelldatei über, bis deren Ende erreicht ist. Bei einer WHILE-Klausel wird die Aufnahme weiterer Sätze abgebrochen, wenn die im Ausdruck angegebene Bedingung erstmals nicht mehr erfüllt ist. Sowohl bei WHILE- als auch bei FOR-Klauseln ist zu beachten, daß die im <Ausdruck> verwendeten Feldbezeichnungen denen der aktivierten Datenbank entsprechen müssen.

Bei der zweiten Form des APPEND-Befehls (b) wird lediglich ein leerer Satz an die aktivierte Datenbank angehängt. Seine Felder können anschließend mittels der Befehle EDIT oder REPLACE gefüllt werden.

Bei der dritten Form des APPEND-Befehls (c) wird der Benutzer für Feld aufgefördert, die Daten des neu aufgenommenen Satzes einzugeben. Auf diese Weise lassen sich beliebig viele Sätze an die aktivierte Datenbank anhängen, da APPEND in Form c so lange neue Sätze anlegt, bis anstelle der Dateneingabe für das erste Feld eines Satzes nur die RETURN-Taste gedrückt wird.

Wurde vor einem APPEND-Befehl der Form c der Befehl SET FORMAT TO <Maskendatei> erteilt, so werden die in der <Maskendatei> enthaltenen @-Befehle zur Gestaltung des Bildschirms herangezogen. Auf diese Weise läßt sich die von dBASE standardmäßig durchgeführte Abfrage der Feldinhalte in der Reihenfolge der Datenbankstruktur umgehen und den Bedürfnissen des Nutzers anpassen.

Im Seitenmodus bewirkt der Befehl SET CARRY ON (siehe SET-Befehl), daß alle Daten aus dem vorangegangenen Satz in den neuen Satz übernommen werden. Anschließend lassen sich Änderungen durchführen. Diese Eigenschaft erweist sich als besonders nützlich, wenn aufeinanderfolgende Sätze viele gleiche Daten enthalten.

Ist die aktivierte Datenbank indiziert, so werden die im USE-Befehl angegebenen Schlüssel automatisch aktualisiert, sobald durch APPEND (Form a oder c) ein neuer Satz hinzugefügt wird. Ein durch APPEND BLANK angefügter Satz läßt sich durch einen INDEX-Befehl (ohne Parameter) nachträglich in die aktivierten Schlüsseldateien aufnehmen (siehe INDEX-Befehl). Jede nicht im USE-Befehl aufgeführte Schlüsseldatei der aktivierten Datenbank muß nach einem APPEND-Befehl natürlich neu indiziert werden.

Der APPEND-Befehl erweist sich als besonders nutzbringend, wenn es darum geht, Felder einer bereits existierenden Datenbank in ihrer Ausdehnung zu verändern, sie zu löschen bzw. neue Felder hinzuzufügen. Mittels des Befehls CREATE erzeugt man eine neue Datenbank mit der gewünschten Struktur und hängt anschließend (an die noch leere Datenbank) mit APPEND die komplette alte Datenbank an. Felder, die nur in der neuen Datenbank enthalten sind, werden dabei mit Leerzeichen gefüllt.

Beispiele:

```
. USE MITGLIED
```

```
. DISPLAY STRUCTURE
```

```
STRUCTURE FOR FILE: MITGLIED.DBF
```

```
NUMBER OF RECORDS: 00005
```

```
DATE OF LAST UPDATE: 10/06/82
```

```
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	MITGL:NR	N	005	
003	EINTR:JAHR	C	004	
** TOTAL **			00030	

APPEND...246

. DISPLAY ALL

00001	Müller, Hans	103 76
00002	Schneider, Manfred	204 78
00003	Wetzel, Irene	304 79
00004	Lehmann, Paul	403 81
00005	Kern, Christa	406 81

. APPEND

RECORD 00006

NAME : Weber, Peter
MITGL:NR : 513
EINTR:JAHR : 82

RECORD 00007

NAME : Richter, M. "Hansi"
MITGL:NR : 515
EINTR:JAHR : 82

RECORD 00008

NAME : (RETURN)

. DISPLAY ALL OFF NAME, MITGL:NR

Müller, Hans	103
Schneider, Manfred	204
Wetzel, Irene	304
Lehmann, Paul	403
Kern, Christa	406
Weber, Peter	513
Richter, M. "Hansi"	515

. APPEND FROM NEUMITGL

00007 RECORDS ADDED

. DISPLAY ALL

00001	Müller, Hans	103 76
00002	Schneider, Manfred	204 78
00003	Wetzel, Irene	304 79
00004	Lehmann, Paul	403 81
00005	Kern, Christa	406 81
00006	Weber, Peter	513 82
00007	Richter, M. "Hansi"	515 82
00008	Baum, Kurt	420
00009	Eckart, Franz	456
00010	Fritzen, Iris	501
00011	Mathies, Anton	67
00012	Nolling, Gerhard	114
00013	Stender, Max	404
00014	Zander, Ulf	512

. APPEND BLANK

. DISPL

00015 0

. REPLACE NAME WITH "Ullrichs, Elfriede", MITGL:NR WITH 515
 00001 REPLACEMENT(S)

. DISPLAY

00015 Ullrichs, Elfriede 515

. DISPLAY ALL NAME, " Mitgl.Nr.:", MITGL:NR

00001	Müller, Hans	Mitgl.Nr.:	103
00002	Schneider, Manfred	Mitgl.Nr.:	204
00003	Wetzel, Irene	Mitgl.Nr.:	304
00004	Lehmann, Paul	Mitgl.Nr.:	403
00005	Kern, Christa	Mitgl.Nr.:	406
00006	Weber, Peter	Mitgl.Nr.:	513
00007	Richter, M. "Hansi"	Mitgl.Nr.:	515
00008	Baum, Kurt	Mitgl.Nr.:	420
00009	Eckart, Franz	Mitgl.Nr.:	456
00010	Fritzen, Iris	Mitgl.Nr.:	501
00011	Mathies, Anton	Mitgl.Nr.:	67
00012	Nolling, Gerhard	Mitgl.Nr.:	114
00013	Stender, Max	Mitgl.Nr.:	404
00014	Zander, Ulf	Mitgl.Nr.:	512
00015	Ullrichs, Elfriede	Mitgl.Nr.:	515

. USE KATALOG

. DISPL STRUC

STRUCTURE FOR FILE: KATALOG.DBF

NUMBER OF RECORDS: 00009

DATE OF LAST UPDATE: 10/17/08

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	PREIS	N	010	002
004	BESTAND	N	005	
** TOTAL **			00041	

. CREATE KATALOG2

ENTER RECORD STRUCTURE AS FOLLOWS:

FIELD NAME,TYPE,WIDTH,DECIMAL PLACES

001 ARTIKEL,C,25

002 ARTIKEL:NR,N,5

003 PREIS,N,10,2

004 DATUM,C,8

005 (RETURN)

INPUT DATA NOW? N

. USE KATALOG2

. APPEND FROM KATALOG

00009 RECORDS ADDED

. LIST

00001	Zwinge	68	12.00
00002	Hammer	49	13.50
00003	Bohrmaschine	112	360.00
00004	Hobel	83	20.50
00005	Schubkarre	140	150.50
00006	Leiter	103	80.00
00007	Eimer	33	14.90
00008	Schaufel	12	13.00
00009	Waage	66	1260.80

. REPLACE ALL PREIS WITH PREIS * 1.10, DATUM WITH '01.06.82'

00009 REPLACEMENT(S)

. LIST				
00001	Zwinge	68	13.20	01.06.82
00002	Hammer	49	14.85	01.06.82
00003	Bohrmaschine	112	396.00	01.06.82
00004	Hobel	83	22.55	01.06.82
00005	Schubkarre	140	165.55	01.06.82
00006	Leiter	103	88.00	01.06.82
00007	Eimer	33	16.39	01.06.82
00008	Schaufel	12	14.30	01.06.82
00009	Waage	66	1386.88	01.06.82

(Das folgende Beispiel zeigt die Anwendung von APPEND mit der Option DELIMITED. Dabei werden Daten, die außerhalb von dBASE II mit anderer Anwendersoftware - z.B. mit einem Programm in BASIC oder Standardsoftware zur Textverarbeitung - erzeugt wurden, von dBASE II übernommen. Es wird vorausgesetzt, daß folgende Daten in der Datei A:EXTERN.DAT vorhanden sind:

'Hobby-Bau',49,35
 'Tischlerei Schmitz',68,10
 'Wilken und Sohn',140,2
 'Karl Anders',33,12
 'Sanitär Reinhardts',12,30

Die Felder sind durch Komma getrennt und Text wird von Zahlen durch die Einschließung in Hochkomma (') unterschieden.)

```
. USE BESTELLG
. DISPLAY STRUCTURE
STRUCTURE FOR FILE: BESTELLG.DBF
NUMBER OF RECORDS: 00008
DATE OF LAST UPDATE: 05/17/82
PRIMARY USE DATABASE
FLD      NAME      TYPE      WIDTH      DEC
001      NAME      C         020
002      ARTIKEL:NR C         005
003      ANZAHL     N         005
** TOTAL **                00031
```

. LIST				
00001	Müller & Co	68		2
00002	Schneider GmbH	112		2
00003	Hansen KG	66		1
00004	Stahlbau	112		3
00005	Fortschritt e.V.	33		12
00006	Hoch und Tief AG	103		4
00007	Heimbedarf Knobel	83		3
00008	Gartenbau Wild	12		34

. APPEND FROM A:EXTERN.DAT DELIMITED
00005 RECORDS ADDED

. LIST

00001	Müller & Co	68	2
00002	Schneider GmbH	112	2
00003	Hansen KG	66	1
00004	Stahlbau	112	3
00005	Fortschritt e.V.	33	12
00006	Hoch und Tief AG	103	4
00007	Heimbedarf Knobel	83	3
00008	Gartenbau Wild	12	34
00009	Hobby-Bau	49	35
00010	Tischlerei Schmitz	68	10
00011	Wilken und Sohn	140	2
00012	Karl Anders	33	12
00013	Sanitär Reinhardts	12	30

(Mit APPEND lassen sich auch gezielte Ergänzungen mit Daten aus anderen Dateien vornehmen. Dazu ist nach APPEND FOR die Bedingung anzugeben, welche von den zu übernehmenden Daten erfüllt werden muß. Man beachte, daß die dort aufgeführten Felder für beide Dateien - sowohl in der zu übernehmenden als auch in der aufnehmenden - definiert sein müssen)

. USE AUSZAHLG

. DISP STRUC

STRUCTURE FOR FILE: AUSZAHLG.DBF

NUMBER OF RECORDS: 00013

DATE OF LAST UPDATE: 06/17/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	SCHECK:NR	N	005	
002	EMPF	C	020	
003	BETRAG	N	010	002
004	ANGEWIESEN	L	001	
005	ABGEBUCHT	L	001	
** TOTAL **			00038	

. LIST

00001	2 Autowerkstatt	560.20	.T.	.F.
00002	5 Mietbau	12300.00	.F.	.T.
00003	6 selbst	300.00	.T.	.F.
00004	7 Zahnarzt	450.12	.T.	.T.
00005	8 Krankenhaus	1370.76	.T.	.F.
00006	9 Supermarkt	120.70	.T.	.T.
00007	25 selbst	450.00	.T.	.F.
00008	10 Warenhaus	345.60	.T.	.F.
00009	11 Tierarzt	80.30	.T.	.F.
00010	12 selbst	600.00	.T.	.F.
00011	13 Tankstelle	90.50	.T.	.F.
00012	14 Versicherung	540.80	.F.	.T.
00013	15 selbst	800.00	.T.	.F.

. USE ABBUCHG

. DISP STRU

STRUCTURE FOR FILE: ABBUCHG.DBF

NUMBER OF RECORDS: 00003

DATE OF LAST UPDATE: 06/10/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	SCHECK:NR	N	005	
002	BETRAG	N	010	002
003	ABGEBUCHT	L	001	
** TOTAL **			00017	

. LIST

00001	1	45.60	.T.
00002	3	1230.56	.T.
00003	4	120.23	.T.

. APPEND FROM AUSZAHLG FOR ABGEBUCHT

00004 RECORDS ADDED

. LIST

00001	1	45.60	.T.
00002	3	1230.56	.T.
00003	4	120.23	.T.
00004	5	12300.00	.T.
00005	7	450.12	.T.
00006	9	120.70	.T.
00007	14	540.80	.T.

. APPEND FROM AUSZAHLG FOR ANGEWIESEN

*** SYNTAX ERROR ***

?

APPEND FROM AUSZAHLG FOR ANGEWIESEN

CORRECT AND RETRY (Y/N)? N

(Das letzte APPEND führte zu einem Fehler, da das Feld ANGEWIESEN in der Datenbank ABGEBUCHT nicht vorhanden ist).

BROWSE

BROWSE [FIELDS <Felderfolge>]

Der BROWSE-Befehl ist einer der wirkungsvollsten Befehle zur Datenanzeige und -änderung. Die Daten von bis zu 19 Sätzen werden durch BROWSE auf dem Bildschirm angezeigt. In jeder Zeile erscheinen so viele Felder, wie diese fassen kann. Sind einzelne Felder breiter als 80 Zeichen, so verringert sich die Anzahl der angezeigten Sätze entsprechend.

Der Bildschirm wirkt bei BROWSE wie ein Fenster, durch das man die gesamte Datenbank einsehen kann. Da dBASE bei diesem Befehl in den Seitenmodus übergeht, kann man mit den entsprechenden Tastenkombinationen den Cursor auf jeden beliebigen Satz und jedes beliebige Feld der gesamten Datenbank fahren und dort ggf. Änderungen vornehmen.

Wird keine <Felderfolge> angegeben, erscheinen alle Felder in der Reihenfolge der Datenbankstruktur auf dem Bildschirm.

In der untenstehenden Übersicht sind noch einmal die bei BROWSE wirksamen Tastenkombinationen für den Seitenmodus zusammengefaßt:

- ctl-E,A — setzt den Cursor auf das vorangegangene Datenfeld zurück
- ctl-X,F — rückt den Cursor auf das nächste Datenfeld vor

- ctl-S — setzt den Cursor um ein Zeichen nach links
- ctl-D — setzt den Cursor um ein Zeichen nach rechts

- ctl-G — löscht das Zeichen an der Cursorposition
- DELETE/ — löscht das Zeichen links vom Cursor
- RUBOUT

- ctl-Q — bricht den Seitenmodus ab und leitet zum dBASE Befehlsmodus über. Die durchgeführten Änderungen bleiben dabei unwirksam, d.h. sie werden nicht gespeichert.
- ctl-W — speichert den aktuellen Datensatz auf der Diskette und beendet den Seitenmodus (bei Superbrain: ctl-O)

- ctl-R — speichert den aktuellen Datensatz auf der Diskette und zeigt den vorangegangenen Datensatz an
- ctl-C — speichert den aktuellen Datensatz auf der Diskette und zeigt den nächsten Datensatz an

- ctl-Z — verschiebt das Fenster um ein Datenfeld nach links
- ctl-B — verschiebt das Fenster um ein Datenfeld nach rechts

- ctl-U — setzt bzw. löscht die Löschmarkierung "*" des aktuellen Datensatzes

Beispiel:

BROWSE ARTIKEL:NR, ARTIKEL, BESTAND

CANCEL

CANCEL bricht die Ausführung einer Befehlsdatei ab und reaktiviert den dBASE-Befehlsmodus.

Beispiel:

```
ACCEPT 'Arbeit beendet (J/N)' TO X
IF !(X) = 'J'
  CANCEL
ENDIF
```

(In diesem Auszug aus einer Befehlsdatei wird bei Eingabe von 'J' oder 'j' der CANCEL-Befehl ausgeführt. Auf diese Weise läßt sich eine elegante Überleitung zum dBASE-Befehlsmodus herbeiführen.)

CHANGE

CHANGE [<Geltungsbereich>] FIELD <Felderfolge> [FOR <Ausdruck>]

Der Befehl CHANGE ermöglicht dem Benutzer die Änderung einzelner Datenfelder mit geringem Aufwand. Alle in der <Felderfolge> aufgeführten Datenfelder werden in der angegebenen Reihenfolge angezeigt, und der Benutzer hat jeweils die Möglichkeit, neue Daten einzugeben, vorhandene Daten zu ändern oder ohne Veränderung des Feldinhalts zum nächsten Feld fortzuschreiten. Wurden alle in der <Felderfolge> angegebenen Felder eines Satzes angezeigt, setzt CHANGE die Anzeige mit dem ersten Feld des durch den <Geltungsbereich> spezifizierten nächsten Satzes fort. Die Voreinstellung für den Geltungsbereich ist nur der aktuelle Satz.

Der gesamte Inhalt eines Datenfeldes läßt sich löschen, indem man auf die Frage "CHANGE?" die Tastenfolge ctl-Y, RETURN eingibt. Die Ausführung des CHANGE-Befehls kann durch Drücken der ESCAPE-Taste abgebrochen werden.

Beispiele:

. USE BESTAND

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: BESTAND.DBF

NUMBER OF RECORDS: 00006

DATE OF LAST UPDATE: 10/23/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	80.60	230	03.09.82
00003	Reifen	987	70.50	210	28.08.82
00004	Achse	70023	350.90	40	15.09.82
00005	Kupplung	3567	260.90	42	10.10.82
00006	Rahmen	230	3460.00	10	03.05.82

. CHANGE FIELD BESTAND

RECORD: 00006

BESTAND: 10

TO: 12

. CHANGE ALL FIELD DATUM

RECORD: 00001

DATUM: 12.07.82

CHANGE? (RETURN)

RECORD: 00002

DATUM: 03.09.82

CHANGE? 03

TO 04

DATUM: 04.09.82

CHANGE? (RETURN)

RECORD: 00003

DATUM: 28.08.82

CHANGE? (ESCAPE)

. CHANGE ALL FIELD EINZ:PREIS FOR (EINZ:PREIS - INT(EINZ:PREIS)) > 0

RECORD: 00002

EINZ:PREIS: 80.60

TO: 81

RECORD: 00003

EINZ:PREIS: 70.50

TO: 70

RECORD: 00004

EINZ:PREIS: 350.90

TO: 350

RECORD: 00005

EINZ:PREIS: 260.90

TO: 261

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Kupplung	3567	261.00	42	10.10.82
00006	Rahmen	230	3460.00	12	03.05.82

CLEAR

CLEAR [GETS]

Enthält der Befehl das Steuerwort GETS, so werden alle durch @-Befehle aktivierten GET-Klauseln deaktiviert. Im Gegensatz zum ERASE-Befehl bleibt dabei jedoch der Bildschirm unverändert.

Fehlt das Steuerwort GETS, versetzt der CLEAR-Befehl dBASE in den Anfangszustand. Dies bedeutet, daß alle Datenbanken geschlossen und deaktiviert sowie alle temporären Variablen gelöscht werden. Außerdem stellt CLEAR als Arbeitsbereich PRIMARY ein (siehe SELECT-Befehl).

Beispiel:

. CLEAR

CONTINUE**CONTINUE**

Dieser Befehl stellt die Fortsetzung des LOCATE-Befehls dar, wobei zwischen LOCATE und CONTINUE auch andere Befehle zur Ausführung gebracht werden können. Weitere Erläuterungen zum Zusammenhang zwischen LOCATE und CONTINUE finden sich bei der Beschreibung des LOCATE-Befehls.

COPY

- a. COPY TO <Dateiname> [<Geltungsbereich>] [FIELD <Felderfolge>]
 [FOR <Ausdruck>] [WHILE <Ausdruck>]
 [SDF] [DELIMITED [WITH <Begrenzung>]]
- b. COPY TO <Dateiname> STRUCTURE [EXTENDED]
 [FIELD <Felderfolge>]

Form a dieses Befehls kopiert die durch USE aktivierte Datenbank in die durch den <Dateinamen> angegebene Datei, die im Folgenden kurz Zielfeile genannt wird. Die Zielfeile muß entweder im dBASE-Format oder - falls die Option SDF (System Data Format) angegeben ist - im Standard-ASCII-Format vorliegen. Existiert die Zielfeile noch nicht, so wird sie angelegt.

Gibt der Benutzer keine FIELD-Klausel an, so werden alle Felder der aktivierten Datenbank in die Zielfeile übernommen. Ist jedoch eine FIELD-Klausel vorhanden, so enthält die Zielfeile nach dem COPY-Befehl nur die in der <Felderfolge> angegebenen Felder.

Ist im COPY-Befehl die Option SDF angegeben, legt dBASE eine Datei im Standard-ASCII-Format an, so daß die so erzeugte Datenbank auch mit anderen Programmen als dBASE bearbeitet werden kann. Die (nur für die Arbeit mit dBASE benötigte) Struktur der aktivierten Datenbank wird dabei nicht in die Zielfeile übernommen.

Enthält der COPY-Befehl das Steuerwort DELIMITED, werden in der Zielfeile alle Felder durch Kommata voneinander getrennt und die Inhalte der Felder vom Typ Zeichenreihe in Hochkomma (') eingeschlossen. Sollen die Zeichenreihen durch ein anderes Sonderzeichen gekennzeichnet werden, so kann das gewünschte Zeichen mittels der WITH-Klausel angegeben werden. Handelt es sich dabei um ein Komma, werden nicht belegte Stellen am Ende von alphanumerischen Feldern bzw. am Anfang von numerischen Feldern entfernt. Außerdem wird der Wert alphanumerischer Felder nicht in Sonderzeichen eingeschlossen.

Will man die so erzeugte Zielfeile später wieder mit APPEND einlesen, ist bei der Wahl der Sonderzeichen zu beachten, daß APPEND nur Hochkomma und Anführungszeichen akzeptiert.

Umfaßt der <Dateiname> keine explizite Angabe eines Dateityps, ergänzt dBASE automatisch ".DBF" bzw. - soweit DELIMITED oder SDF im Befehl angegeben ist - ".TXT".

Form b des COPY-Befehls bewirkt, daß nur die Struktur der aktivierten Datenbank in die Zielfeile übernommen wird. Enthält der Befehl eine FIELD-Klausel, so werden nur die in der <Felderfolge> aufgeführten Felder berücksichtigt.

Die Option EXTENDED bei einem COPY STRUCTURE-Befehl bewirkt, daß die Struktur der aktivierten Datenbank in Datensätze der Zielfeile übertragen wird. Hierdurch räumt dBASE dem Benutzer die Möglichkeit ein, auf die Struktur einer Datenbank mittels einer Befehlsdatei zuzugreifen, um beispielsweise den weiteren Programmablauf von der Struktur der gerade aktivierten Datenbank abhängig zu machen, oder aber um die Struktur zur Erzeugung einer neuen Datenbank zu modifizieren (siehe CREATE-Befehl).

Beispiele:

. DISPLAY STRUCTURE
 STRUCTURE FOR FILE: A:MITGLIED.DBF
 NUMBER OF RECORDS: 00008
 DATE OF LAST UPDATE: 10/17/82
 PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	MITGL:NR	N	005	
003	EINTR:JAHR	C	004	
** TOTAL **			00030	

. DISPLAY ALL OFF NAME, MITGL:NR

Müller, Hans	103
Schneider, Manfred	204
Wetzel, Irene	304
Lehmann, Paul	403
Kern, Christa	406
Weber, Peter	513
Zander, Ulf	512
Ullrichs, Elfriede	515

. COPY TO KOPIEMIT
 00008 RECORDS COPIED

. COPY TO KLEIN500 FOR MITGL:NR < 500
 00005 RECORDS COPIED

. USE KLEIN500

. DISPLAY ALL

00001	Müller, Hans	103	76
00002	Schneider, Manfred	204	78
00003	Wetzel, Irene	304	79
00004	Lehmann, Paul	403	81
00005	Kern, Christa	406	81

. USE A:MITGLIED

. COPY TO SEIT79 FOR EINTR:JAHR > '78'
 00004 RECORDS COPIED

. USE SEIT79

. DISPLAY ALL

00001	Wetzel, Irene	304	79
00002	Lehmann, Paul	403	81
00003	Kern, Christa	406	81
00004	Weber, Peter	513	82

. COPY TO SEIT79 FIELD NAME, MITGL:NR FOR EINTR:JAHR > '78'
00004 RECORDS COPIED

. USE SEIT79

. DISPLAY ALL

00001	Wetzel, Irene	304
00002	Lehmann, Paul	403
00003	Kern, Christa	406
00004	Weber, Peter	513

. USE A:MITGLIED

. COPY NEXT 4 TO 4MITGL
00004 RECORDS COPIED

. USE 4MITGL

. DISPLAY ALL

00001	Müller, Hans	103	76
00002	Schneider, Manfred	204	78
00003	Wetzel, Irene	304	79
00004	Lehmann, Paul	403	81

. USE BESTELLG

. DISPLAY STRUCTURE
STRUCTURE FOR FILE: BESTELLG.DBF
NUMBER OF RECORDS: 00013
DATE OF LAST UPDATE: 10/29/82
PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ARTIKEL:NR	C	005	
003	ANZAHL	N	005	
** TOTAL **			00031	

. COPY STRUCTURE EXTENDED TO BESTSTRU
00003 RECORDS COPIED

. USE BESTSTRU

. DISPLAY STRUCTURE
STRUCTURE FOR FILE: BESTSTRU.DBF
NUMBER OF RECORDS: 00003
DATE OF LAST UPDATE: 10/29/82
PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	FIELD:NAME	C	010	
002	FIELD:TYPE	C	001	
003	FIELD:LEN	N	003	
004	FIELD:DEC	N	003	
** TOTAL **			00018	

FLD	NAME	TYPE	WIDTH	DEC
00001	NAME	C	20	0
00002	ARTIKEL:NR	C	5	0
00003	ANZAHL	N	5	0

(Im Folgenden könnten Änderungen an diesen die Struktur beschreibenden Daten vorgenommen werden, z.B. um eine neue Datenbank mit modifizierter Struktur zu erzeugen; siehe CREATE-Befehl.)

(Die nachstehenden Beispiele zeigen die Anwendung von COPY mit der Option DELIMITED. Mit ihr lassen sich dBASE II-Datenbanken in sogenannte Textdateien überführen. Der Inhalt alphanumerisch definierter Felder (vom Typ C) kann bei Bedarf am Anfang und Ende durch ein spezielles, vom Benutzer anzugebendes Zeichen (z.B. Anführungszeichen) abgegrenzt werden. Ferner lassen sich nicht genutzte Stellen am Ende alphanumerischer bzw. am Anfang numerischer Felder entfernen.)

. USE BESTELLG

. DISPLAY STRUCTURE
STRUCTURE FOR FILE: BESTELLG.DBF
NUMBER OF RECORDS: 00013
DATE OF LAST UPDATE: 10/17/82
PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ARTIKEL:NR	C	005	
003	ANZAHL	N	005	
** TOTAL **			00031	

. DISPLAY ALL

00001	Müller & Co	68	2
00002	Schneider GmbH	112	2
00003	Hansen KG	66	1
00004	Stahlbau	112	3
00005	Fortschritt e.V.	33	12
00006	Hoch und Tief AG	103	4
00007	Heimbedarf Knobel	83	3
00008	Gartenbau Wild	12	34
00009	Hobby-Bau	49	35
00010	Tischlerei Schmitz	68	10
00011	Wilken und Sohn	140	2
00012	Karl Anders	33	12
00013	Sanitär Reinhardts	12	30

. COPY TO KOMMA DELIMITED WITH ,
00013 RECORDS COPIED

Anmerkung: Zur Erstellung von Sicherungskopien (Backup) ist bei ein- oder mehrfach indizierten Dateien der Befehl 'QUIT TO 'PIP B: <BACKUP.DBF> = A: <Dateiname.DBF>' zu empfehlen, da COPY Datensätze mit Löschmarkierung nicht überträgt und daher alle INDEX-Dateien neu generiert werden müssen.

(Auflistung der erzeugten Textdatei KOMMA.TXT :

Müller & Co,68,2
Schneider GmbH,112,2
Hansen KG,66,1
Stahlbau,112,3
Fortschritt e.V.,33,12
Hoch und Tief AG,103,4
Heimbedarf Knobel,83,3
Gartenbau Wild,12,34
Hobby-Bau,49,35
Tischlerei Schmitz,68,10
Wilken und Sohn,140,2
Karl Anders,33,12
Sanitär Reinhardts,12,30

Man beachte die Kürzung der Felder um nicht belegte Stellen. Dies geschieht nur bei der o.a. Verwendung von WITH mit Angabe eines Kommas). Diese Veränderung ermöglicht die Weiterverarbeitung mit anderen Sprachen und Textprozessoren, insbesondere WordStar und MailMerge.

. COPY TO HOCHKOMM DELIMITED

00013 RECORDS COPIED

(Auflistung der erzeugten Textdatei HOCHKOMM.TXT :

'Müller & Co	68	2
'Schneider GmbH	112	2
'Hansen KG	66	1
'Stahlbau	112	3
'Fortschritt e.V.	33	12
'Hoch und Tief AG	103	4
'Heimbedarf Knobel	83	3
'Gartenbau Wild	12	34
'Hobby-Bau	49	35
'Tischlerei Schmitz	68	10
'Wilken und Sohn	140	2
'Karl Anders	33	12
'Sanitär Reinhardts	12	30

Man beachte die Einschließung der alphanumerischen Felder in Hochkomma, dem von dBASE verwendeten Zeichen, wenn WITH nicht genutzt wird.)

. COPY TO ANFZEICH.DAT DELIMITED WITH "

00013 RECORDS COPIED

(Auflistung der erzeugten Textdatei ANFZEICH.DAT :

"Müller & Co	"68	"	2
"Schneider GmbH	"112	"	2
"Hansen KG	"66	"	1
"Stahlbau	"112	"	3
"Fortschritt e.V.	"33	"	12
"Hoch und Tief AG	"103	"	4
"Heimbedarf Knobel	"83	"	3
"Gartenbau Wild	"12	"	34
"Hobby-Bau	"49	"	35
"Tischlerei Schmitz	"68	"	10
"Wilken und Sohn	"140	"	2
"Karl Anders	"33	"	12
"Sanitär Reinhardts	"12	"	30

Man beachte die Einschließung der alphanumerischen Felder in Anführungszeichen entsprechend der Angabe zu WITH).

COUNT

COUNT [<Geltungsbereich>] [FOR <Ausdruck>] [TO <Variable>]
[WHILE <Ausdruck>]

Der COUNT-Befehl zählt die Anzahl der Sätze des angegebenen <Geltungsbereichs>; Voreinstellung ist dabei ALL. Enthält der Befehl eine FOR-Klausel, werden nur die Sätze gezählt, auf die die im <Ausdruck> angegebene Bedingung zutrifft. dBASE präsentiert das Ergebnis in Form der Nachricht "COUNT = xxxxx". Bei Verwendung der TO-Klausel wird es außerdem als ganzzahliger Wert in der angegebenen temporären <Variablen> abgelegt. Sätze mit Löschmarkierung werden vom COUNT-Befehl berücksichtigt, wenn nicht zuvor der Befehl SET DELETE ON erteilt wurde (siehe SET-Befehle).

Beispiele:

. USE BESTAND

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: BESTAND.DBF

NUMBER OF RECORDS: 00011

DATE OF LAST UPDATE: 10/15/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

. DISPLAY ALL

00001	Getriebe	10023	1320.00	14	15.10.82
00002	Kolben	12221	90.00	233	15.10.82
00003	Reifen	987	77.00	210	15.10.82
00004	Achse	70023	350.00	40	15.09.82
00005	Haube	997	46.90	89	12.10.82
00006	Feder	19888	56.70	23	12.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00008	Auspufftopf	12112	80.50	230	06.10.82
00009	Welle	8978	320.00	34	15.10.82
00010	Generator	10015	340.00	25	12.10.82
00011	Kanister	1234	5.90	456	01.09.82

. COUNT
COUNT = 00011

. COUNT FOR ARTIKEL:NR > 12000
COUNT = 00004

. GOTO TOP

. COUNT FOR EINZ:PREIS < 100 NEXT 6
COUNT = 00004

. GOTO TOP

. COUNT NEXT 6 FOR EINZ:PREIS < 100
COUNT = 00004

. COUNT TO XX FOR BESTAND < 30
COUNT = 00003
. ? XX

3

. USE ARTIKEL

. DISPLAY STRUCTURE
STRUCTURE FOR FILE: ARTIKEL.DBF
NUMBER OF RECORDS: 00010
DATE OF LAST UPDATE: 10/10/82
PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	TITEL	C	020	
002	AUTOR	C	015	
003	REGISTER	C	008	
004	ERSCHIENEN	C	008	
** TOTAL **			00052	

. LIST

00001	Kaltblut	Cappellen, Mi.	IS0908	02.12.59
00002	Weltgeschichte	Trauter, I.N.	M09081	12.05.60
00003	100 Geschichten	Trauter, I.N.	IB123045	03.02.72
00004	Der Streiter	Baum, H.	K12390	10.09.77
00005	Das Dunkel	Anton, W.	L14509	12.09.78
00006	Das Pferd	Anton, K.	I2903	13.09.80
00007	Stadt im Schnee	Schend, K.	IS9809	13.06.79
00008	Nur Heute	Haltig, L.	IS09788	12.08.53
00009	Weg nach vorn?	Keil, B.	K0790122	13.07.81
00010	Das war's	Endig, L.	I78608	05.06.78

. COUNT FOR 'Anton' \$ AUTOR
COUNT = 00002

CREATE

- a. CREATE [<Dateiname>]
- b. CREATE <Dateiname> FROM <Quelldatei>

Durch Form a dieses Befehls wird eine neue Datenbankdatei (Dateityp .DBF) erzeugt, wobei der Benutzer den Dateinamen, die Feldbezeichnungen und die Struktur der Datenbank frei bestimmen kann.

Falls kein <Dateiname> mit dem Befehlswort angegeben wurde, fordert dBASE den Benutzer mittels der Nachricht "FILENAME:" zu dessen Eingabe auf. Dabei gilt für den Dateinamen neben den bekannten CP/M-Konventionen die zusätzliche Einschränkung, daß Sonderzeichen nur dort zugelassen sind, wo ihnen bei CP/M eine besondere Bedeutung zukommt (z.B. der Doppelpunkt in "B:" zur Bezeichnung des Laufwerks B).

Ist unter dem angegebenen Namen bereits eine Datei vorhanden, fragt dBASE den Benutzer "DESTROY EXISTING FILE?", worauf dieser entsprechend seinem Wunsch mit Y(es) oder N(o) antworten muß.

Wird eine neue Datei angelegt bzw. die obige Frage mit Y beantwortet, so fordert dBASE den Benutzer zur Eingabe der Struktur der neuen Datenbank auf. Dabei erscheint folgende Meldung:

```
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD          NAME,TYPE,WIDTH,DECIMAL PLACES
001
```

Für jedes Feld seiner neuen Datenbank muß der Benutzer nun dessen Namen und einige zusätzliche Informationen über dessen Beschaffenheit angeben. Als Name eines Feldes ist eine maximal zehn Zeichen lange Zeichenreihe zulässig, die mit einem Buchstaben beginnen muß und sich darüberhinaus aus weiteren Buchstaben, Ziffern und Doppelpunkten zusammensetzen kann.

dBASE unterscheidet bei Datenfeldern drei verschiedene Typen, denen jeweils ein Buchstabe fest zugeordnet ist, der in der Rubrik TYPE angegeben werden muß; im einzelnen:

- C - Zeichenreihen (alphanumerische Felder)
- N - numerische Werte (Zahlen)
- L - Wahrheitswerte (einstellig mit F für "falsch" und T für "wahr")

Unter der Rubrik "WIDTH" muß die Länge des Feldes eingegeben werden. Bei alphanumerischen Feldern wird sie durch den längsten in diesem Feld aufzunehmenden Text (Stellenzahl) bestimmt. Bei numerischen Werten muß zwischen ganzen Zahlen und Zahlen mit Dezimalpunkt unterschieden werden. Bei ganzen Zahlen entspricht die Länge des Feldes der Anzahl der Ziffern der größten Zahl, die in dem Feld gespeichert werden soll. Bei Zahlen mit Dezimalpunkt sind zwei Längenangaben erforderlich: die maximale Anzahl der Stellen insgesamt (einschließlich Dezimalpunkt), die für die Zahl vorgesehen werden soll (einzugeben unter der Rubrik "WIDTH"), und die Anzahl der Stellen, die hinter dem Dezimalpunkt erscheinen sollen (einzugeben unter der Rubrik "DECIMAL PLACES"). Wahrheitswerte haben immer die Länge 1.

Die Eingabe der Datenbankstruktur läßt sich abschließen, indem man, statt den von dBASE geforderten nächsten Feldnamen einzugeben, die RETURN-Taste drückt. Beantwortet man die anschließende Frage "INPUT NOW?" mit N(o), so ist der CREATE-Befehl beendet. Bei der Antwort Y(es) werden Satz für Satz die Felder in der soeben definierten Reihenfolge am Bildschirm aufgelistet, und der Benutzer kann sie nach Belieben mit Inhalten füllen. Diese Dateneingabe läßt sich ebenfalls durch Drücken der RETURN-Taste beenden. Voraussetzung hierfür ist, daß sich der Cursor auf der Anfangsposition des ersten Feldes eines neuen Satzes befindet.

Ging dem CREATE-Befehl der Befehl SET FORMAT TO <Maskendatei> voran, so werden bei der Dateneingabe die in der <Maskendatei> enthaltenen @-Befehle zur Gestaltung des Bildschirms herangezogen. Auf diese Weise läßt sich die durch die Struktur vorgegebene Reihenfolge der Felder bei der Dateneingabe entsprechend den Bedürfnissen des Nutzers ändern.

Bei Form b des CREATE-Befehls wird die Strukturinformation für die zu erzeugende Datenbank den Datensätzen der <Quelldatei> entnommen. Der Inhalt der <Quelldatei> kann zuvor mit den üblichen Dateneingabemethoden oder durch einen COPY STRUCTURE EXTENDED-Befehl (siehe COPY-Befehl) angelegt worden sein.

Beispiele:

```
. CREATE
FILENAME: A:MITGLIED
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD      NAME, TYPE, WIDTH, DECIMAL PLACES
001        NAME ,C, 20
002        MITGL:NR ,N, 5
003        EINTR:JAHR ,C, 4
004        (RETURN)
INPUT DATA NOW? Y
```

CREATE...268

RECORD 00001

NAME : Müller, Hans
MITGL:NR : 103
EINTR:JAHR : 76

RECORD 00002

NAME : Schneider, Manfred
MITGL:NR : 204
EINTR:JAHR : 78

RECORD 00003

NAME : Wetzel, Irene
MITGL:NR : 304
EINTR:JAHR : 79

RECORD 00004

NAME : Lehmann, Paul
MITGL:NR : 403
EINTR:JAHR : 81

RECORD 00005

NAME : Kern, Christa
MITGL:NR : 406
EINTR:JAHR : 81

RECORD 00006

NAME : (RETURN)

. USE A:MITGLIED

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: A:MITGLIED.DBF

NUMBER OF RECORDS: 00005

DATE OF LAST UPDATE: 10/10/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	MITGL:NR	N	005	
003	EINTR:JAHR	C	004	
** TOTAL **			00030	

. DISPLAY ALL

00001	Müller, Hans	103	76
00002	Schneider, Manfred	204	78
00003	Wetzel, Irene	304	79
00004	Lehmann, Paul	403	81
00005	Kern, Christa	406	81

. USE BESTSTRU

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: BESTSTRU.DBF

NUMBER OF RECORDS: 00004

DATE OF LAST UPDATE: 11/19/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	FIELD:NAME	C	010	
002	FIELD:TYPE	C	001	
003	FIELD:LEN	N	003	
004	FIELD:DEC	N	003	
** TOTAL **			00018	

. LIST

	NAME	TYPE	WIDTH	DEC
00001	NAME	C	20	0
00002	ARTIKEL:NR	C	5	0
00003	ANZAHL	N	5	0
00004	DATUM	C	6	0

. CREATE NEUBEST FROM BESTSTRU

. USE NEUBEST

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: NEUBEST.DBF

NUMBER OF RECORDS: 00000

DATE OF LAST UPDATE: 11/19/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ARTIKEL:NR	C	005	
003	ANZAHL	N	005	
004	DATUM	C	006	
** TOTAL **			00037	

DELETE

```
DELETE [ <Geltungsbereich> ] [ FOR <Ausdruck> ]
DELETE FILE <Dateiname>
```

Durch diesen Befehl werden alle Sätze des <Geltungsbereichs>, auf welche die ggf. vorhandene FOR-Klausel zutrifft, zur Löschung gekennzeichnet. Die Voreinstellung für den <Geltungsbereich> ist nur der aktuelle Satz. Die physikalische Löschung des Satzes vom Speichermedium erfolgt allerdings erst durch den PACK-Befehl. Dennoch werden Sätze, die durch DELETE mit einer Löschmarkierung versehen wurden, bei Ausführung der Befehle COPY, APPEND und SORT nicht mehr berücksichtigt. Durch den Befehl RECALL lassen sich zur Löschung gekennzeichnete Sätze reaktivieren. Das Anzeigen durch DELETE markierter Sätze ist möglich; dabei erscheint zwischen der Satznummer und dem ersten Feld des Satzes die Löschmarkierung "***". Die Behandlung zur Löschung gekennzeichnete Sätze ist bei vielen Befehlen davon abhängig, wie der Parameter DELETE eingestellt ist (siehe SET-Befehle).

Die zweite Form dieses Befehls erlaubt das Löschen ganzer Dateien von der Diskette. Handelt es sich bei dem angegebenen <Dateinamen> allerdings um eine derzeit aktivierte Datenbank, wird der Befehl nicht ausgeführt.

Beispiele:

```
. USE AUSZAHLG
```

```
. DISPLAY STRUCTURE
```

```
STRUCTURE FOR FILE: AUSZAHLG.DBF
```

```
NUMBER OF RECORDS: 00013
```

```
DATE OF LAST UPDATE: 10/17/08
```

```
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	SCHECK:NR	N	005	
002	EMPF	C	020	
003	BETRAG	N	010	002
004	ANGEWIESEN	L	001	
005	ABGEBUCHT	L	001	
** TOTAL **			00038	

```
. LIST
```

00001	2 Autowerkstatt	560.20	.T. .F.
00002	5 Mietbau	12300.00	.F. .T.
00003	6 selbst	300.00	.T. .F.
00004	7 Zahnarzt	450.12	.T. .T.
00005	8 Krankenhaus	1370.76	.T. .F.
00006	9 Supermarkt	120.70	.T. .T.
00007	25 selbst	450.00	.T. .F.
00008	10 Warenhaus	345.60	.T. .F.
00009	11 Tierarzt	80.30	.T. .F.
00010	12 selbst	600.00	.T. .F.
00011	13 Tankstelle	90.50	.T. .F.
00012	14 Versicherung	540.80	.F. .T.
00013	15 selbst	800.00	.T. .F.

. DELETE RECORD 3
00001 DELETION(S)

. 5
. DELETE NEXT 3
00003 DELETION(S)

. DELETE ALL FOR .NOT. ANGEWIESEN
00002 DELETION(S)

. LIST			
00001	2 Autowerkstatt	560.20	.T. .F.
00002 *	5 Mietbau	12300.00	.F. .T.
00003 *	6 selbst	300.00	.T. .F.
00004	7 Zahnarzt	450.12	.T. .T.
00005 *	8 Krankenhaus	1370.76	.T. .F.
00006 *	9 Supermarkt	120.70	.T. .T.
00007 *	25 selbst	450.00	.T. .F.
00008	10 Warenhaus	345.60	.T. .F.
00009	11 Tierarzt	80.30	.T. .F.
00010	12 selbst	600.00	.T. .F.
00011	13 Tankstelle	90.50	.T. .F.
00012 *	14 Versicherung	540.80	.F. .T.
00013	15 selbst	800.00	.T. .F.

. RECALL ALL
00006 RECALL(S)

. LIST			
00001	2 Autowerkstatt	560.20	.T. .F.
00002	5 Mietbau	12300.00	.F. .T.
00003	6 selbst	300.00	.T. .F.
00004	7 Zahnarzt	450.12	.T. .T.
00005	8 Krankenhaus	1370.76	.T. .F.
00006	9 Supermarkt	120.70	.T. .T.
00007	25 selbst	450.00	.T. .F.
00008	10 Warenhaus	345.60	.T. .F.
00009	11 Tierarzt	80.30	.T. .F.
00010	12 selbst	600.00	.T. .F.
00011	13 Tankstelle	90.50	.T. .F.
00012	14 Versicherung	540.80	.F. .T.
00013	15 selbst	800.00	.T. .F.

DISPLAY

- a. DISPLAY [<Geltungsbereich>] [<Ausdrucksfolge>]
 [FOR <Ausdruck>] [FIELDS <Felderfolge>]
 [WHILE <Ausdruck>] [OFF]
- b. DISPLAY STRUCTURE
- c. DISPLAY MEMORY
- d. DISPLAY FILES [ON <Laufwerk>] [LIKE <Dateimaske>]
- e. DISPLAY STATUS

Dieser Befehl stellt den eigentlichen Kern von dBASE dar, da es das Ziel aller Datenbank-Operationen ist, Daten in beliebigen Zusammenstellungen und Aggregationsformen zur Anzeige zu bringen. Diesem Ziel kommt dBASE durch die hohe Flexibilität des DISPLAY- Befehls entgegen.

Form a dient zur Anzeige des Inhalts der durch USE aktivierten Datenbank. Enthält der Befehl weder einen <Geltungsbereich> noch eine FOR- bzw. WHILE-Klausel, so bezieht er sich nur auf den aktuellen Satz. Ist eine FOR- oder WHILE-Klausel vorhanden, lautet die Voreinstellung für den <Geltungsbereich> ALL, und alle Sätze, die die Bedingung des <Ausdrucks> erfüllen, werden von DISPLAY erfaßt. Gibt der Benutzer keine <Ausdrucksfolge> an, werden alle Feldinhalte der ausgewählten Sätze zur Anzeige gebracht, andernfalls nur die Werte der aufgeführten Ausdrücke. Am Anfang jeder Ausgabezeile erscheint dabei die Nummer des betreffenden Satzes. Durch Angabe des Steuerwortes OFF läßt sie sich unterdrücken. Die FIELDS-Klausel wird nur benötigt, um Mehrdeutigkeiten bei der Verwendung von Feldbezeichnungen wie STATUS, FILE, MEMO usw. zu vermeiden.

Nach jeweils 15 Sätzen unterbricht DISPLAY die Ausgabe, um dem Benutzer Gelegenheit zu geben, die angezeigten Daten in Ruhe zu betrachten. Durch einen beliebigen Tastendruck läßt sich die Ausgabe fortsetzen. In dieser Eigenschaft liegt der wesentliche Unterschied zum LIST-Befehl, der alle spezifizierten Daten ohne Unterbrechung anzeigt. Bei beiden Befehlen läßt sich jedoch die Ausgabe jederzeit durch Betätigen der ESCAPE-Taste abbrechen.

Form b des Befehls bringt die Struktur der derzeit aktivierten Datenbank zur Anzeige.

Bei Form c dieses Befehls werden alle temporären Variablen und ihre derzeitigen Werte angezeigt.

Mit Form d lassen sich alle auf dem voreingestellten bzw. explizit angegebenen <Laufwerk> vorhandenen Dateien vom Typ F zusammen mit einigen statistischen Angaben über die jeweilige Datenbank anzeigen. Durch Angabe der LIKE-Klausel lassen sich auch andere Dateien zur Anzeige bringen, wobei in der <Dateimaske> mit "*" und "?" gemäß den CP/M-Konventionen das Spektrum der anzuzeigenden Dateien beschrieben werden muß. Das Format der so erzeugten Anzeige entspricht dem eines analogen DIR-Befehls in CP/M.

Form e des DISPLAY-Befehls erstellt eine Übersicht über die aktivierten Datenbanken und Schlüsseldateien sowie die zugehörigen Schlüsselausdrücke. Darüberhinaus werden alle durch SET-Befehle manipulierbaren dBASE-Parameter mit ihrer aktuellen Einstellung angezeigt.

Beispiele:

. USE BESTAND

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: BESTAND.DBF

NUMBER OF RECORDS: 00006

DATE OF LAST UPDATE: 10/23/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

. DISPLAY ALL

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Kupplung	3567	261.00	42	10.10.82
00006	Rahmen	230	3460.00	12	03.05.82

. DISPLAY ALL ARTIKEL, ARTIKEL:NR, EINZ:PREIS * BESTAND, \$(DATUM,4,2) ;
 FOR EINZ:PREIS > 100 .AND. BESTAND < 50

00001	Getriebe	10023	14400.00	07
00004	Achse	70023	14000.00	09
00005	Kupplung	3567	10962.00	10
00006	Rahmen	230	41520.00	05

. DISPLAY MEMORY

NAME (C) Waldschmidt & Söhne

UNBEZAHLT (N) 123456.77

DATEIENDE (L) .T.

** TOTAL ** 03 VARIABLES USED 00026 BYTES USED

DISPLAY...274

. DISPLAY FILES ON A: LIKE *.FRM

SOLLIST .FRM

. DISPLAY FILES

DATABASE FILES	# RCDS	LAST UPDATE
MITGLIED DBF	00015	10/17/82
NEUMITGL DBF	00007	10/17/08
KATALOG DBF	00009	10/17/08
BESTELLG DBF	00013	10/10/82
AUSZAHLG DBF	00013	10/10/82
LAGER DBF	00007	10/17/82
TEST DBF		

NOT A dBASE II DATABASE

(Die letzte o.a. aufgeführte .DBF-Datei ist keine Datenbank aus dBASE II)

. DISPLAY STATUS

DATABASE SELECTED - B:BEISTAND .DBF
PRIMARY USE DATABASE

INDEXES: KEY EXPRESSION:
B:BEISTAND .NDX ARTIKEL

UNSELECTED DATABASE - B:PREIS .DBF
SECONDARY USE DATABASE
WAITING

TODAYS DATE — 17/09/83
DEFAULT DISK DRIVE — B:
ALTERNATE FILE — B:DISPSTAT.TXT

ALTERNATE	— ON	BELL	— ON
CARRY	— OFF	COLON	— ON
CONFIRM	— OFF	CONSOLE	— ON
DEBUG	— OFF	DELETE	— OFF
ECHO	— OFF	EJECT	— ON
ESCAPE	— ON	EXACT	— OFF
INTENSITY	— ON	LINKAGE	— OFF
PRINT	— OFF	RAW	— OFF
STEP	— OFF	TALK	— ON

(Hier kann nur eine Auswahl der Möglichkeiten von DISPLAY gezeigt werden; man beachte hierzu auch die Beispiele bei anderen Kommandos)

DO

- a. DO <Befehlsdatei>
- b. DO WHILE <Ausdruck>
 <Befehle>
 ENDDO

Der DO-Befehl der Form a bewirkt die Ausführung der angegebenen <Befehlsdatei> (Dateityp .CMD bzw. .PRG). Die in der Datei enthaltenen Befehle werden so ausgeführt, als ob sie über die Tastatur eingegeben würden. Aus einer Befehlsdatei können mit DO-Befehlen der Form a weitere Befehlsdateien aufgerufen werden, wobei die maximal zulässige Schachtelungstiefe von DO-Befehlen 16 beträgt. Die Ausführung einer Befehlsdatei endet bei Erreichen des Dateiendes oder eines RETURN-Befehls. War die Befehlsdatei von einer anderen Befehlsdatei aufgerufen worden, so wird letztere fortgesetzt. Ein CANCEL-Befehl bricht - unabhängig von der Schachtelungstiefe, in der er auftritt - die Ausführung aller offenen Befehlsdateien ab und reaktiviert den dBASE-Befehlsmodus.

Beispiel:

DO RECHNUNG

Form b des DO-Befehls findet nur in Befehlsdateien Verwendung. Wenn der angegebene <Ausdruck> den Wahrheitswert "wahr" ergibt, gelangen die folgenden <Befehle> zur Ausführung, bis der <Ausdruck> den Wert "falsch" annimmt. Erst dann wird die Programmausführung hinter dem Steuerwort ENDDO fortgesetzt. Liefert der <Ausdruck> bereits bei der ersten Prüfung den Wert "falsch", werden die zwischen DO WHILE und ENDDO aufgeführten <Befehle> übergangen.

Hinweis: Die DO WHILE - ENDDO-Konstruktion stellt im Sinne von dBASE einen Befehl dar. Dies bedeutet, daß ähnliche Befehle, die aus Einleitungs- und Schlußteil bestehen (z.B. IF - EN-DIF), voll in die DO WHILE-Schleife eingebettet sein müssen (siehe auch Abschnitt 1, 9.2, Regel 8).

Beispiel:

```
DO WHILE .NOT. EOF  
DISPLAY NAME
```

```
SKIP  
ENDDO
```

DO CASE

```

DO CASE
  CASE <Ausdruck>
    <Befehle>
  CASE <Ausdruck>
    <Befehle>
  .
  [ OTHERWISE
    <Befehle> ]
ENDCASE

```

Diese Form des DO-Befehls findet sich nur in Befehlsdateien. Mit der DO CASE-Konstruktion stellt dBASE eine Möglichkeit zur Verfügung, die weitere Verarbeitung von Daten fallgerecht zu steuern. Dabei kann die OTHERWISE-Klausel als Voreinstellung oder Regel angesehen werden, von der die einzelnen CASE-Klauseln die Ausnahmen darstellen. Die Anzahl der CASE-Klauseln pro Befehl ist nicht begrenzt.

Die Ausführung eines DO CASE-Befehls entspricht einer Serie ineinandergeschachtelter IF's, wie das folgende Beispiel verdeutlicht:

<pre> DO CASE CASE ARTIKEL = 'Äpfel' (Befehle für Äpfel) CASE ARTIKEL = 'Birnen' (Befehle für Birnen) OTHERWISE (Befehle für sonstiges) ENDCASE </pre>	=	<pre> IF ARTIKEL = 'Äpfel' (Befehle für Äpfel) ELSE IF ARTIKEL = 'Birnen' (Befehle für Birnen) ELSE (Befehle für sonstiges) ENDIF ENDIF </pre>
--	---	--

Dies bedeutet, daß dBASE die <Ausdrücke> der einzelnen CASE-Klauseln nacheinander auswertet und beim ersten "wahren" Ausdruck die ihm zugeordneten Befehle ausführt. Beim Erreichen der nächsten CASE-Klausel erfolgt ein Sprung zum Steuerwort ENDCASE, so daß auch bei mehreren zutreffenden <Ausdrücken> immer nur die CASE-Klausel des ersten wahren <Ausdrucks> ausgeführt wird.

Die Befehle der OTHERWISE-Klausel werden ausgeführt, wenn alle vorangegangenen CASE-<Ausdrücke> den Wert "falsch" ergeben haben. Enthält der DO CASE-Befehl keine OTHERWISE-Klausel und trifft keine CASE-Klausel zu, bleibt der Befehl ohne Wirkung.

Der DO CASE-Befehl wird durch das Steuerwort ENDCASE abgeschlossen. Nach Ausführung der durch eine CASE-Klausel bzw. OTHERWISE ausgewählten Befehle erfolgt die Programmfortsetzung bei dem Befehl unmittelbar hinter ENDCASE.

Befehle zwischen den Steuerwörtern DO CASE und der ersten CASE-Klausel werden nicht ausgeführt.

Beispiele:

```
DO CASE
  CASE NAME = 'Meier'
    (Befehle für 'Meier')
  CASE NAME = 'Müller'
    (Befehle für 'Müller')
  CASE NAME = 'Schmidt'
    (Befehle für 'Schmidt')
  OTHERWISE
    (Befehle für alle anderen Namen)
ENDCASE
```

(In diesem Beispiel beziehen sich alle Ausdrücke auf dasselbe Feld, nämlich NAME. Es ist jedoch nicht notwendig, daß die einzelnen CASE-Klauseln miteinander in Zusammenhang stehen.)

```
DO CASE
  CASE TAG = 'Montag'
    (Befehle für 'Montag')
  CASE WETTER = 'Regen'
    (Befehle für 'Regen')
  CASE ORT = 'München'
    (Befehle für 'München')
ENDCASE
```

(Man beachte, daß bei diesem Beispiel die CASE-Klauseln in **keinem** äußerlichen Zusammenhang stehen. Im Falle eines regnerischen Montags in München würde allerdings nur die Befehlsfolge für 'Montag' ausgeführt werden.

Die <Ausdrücke> in den CASE-Klauseln müssen nicht wie in den obigen Beispielen aus alphanumerischen Vergleichen bestehen. Jeder beliebige Ausdruck, der einen Wahrheitswert liefert, ist zugelassen.)

```
DO CASE
  CASE VAR1 + VAR2 = 12
    (Befehle für numerischen Vergleich)
  CASE .NOT. A
    (Befehle für Wahrheitswerte)
  CASE "ABC" $ ALPHA
    (Befehle für alphanumerischen Vergleich)
ENDCASE
```

EDIT**EDIT [<n>]**

Der EDIT-Befehl erlaubt dem Benutzer, gezielt Änderungen am Inhalt einzelner Datenfelder vorzunehmen. Die Anwendung und Wirkungsweise dieses Befehls ist davon abhängig, ob dBASE im Seitenmodus arbeitet oder nicht (siehe auch Befehl SET SCREEN).

Im Seitenmodus wird der Befehl durch "EDIT" oder "EDIT <n> " aufgerufen, wobei <n> der gewünschten Satznummer entspricht. Ist keine Satznummer angegeben, so wird der Benutzer durch die Nachricht "ENTER RECORD # :" zur Eingabe derselben aufgefordert. Danach können alle Änderungen mit den in Abschnitt 1, Ziffer 8.1 und 8.2 aufgeführten Tastenkombinationen vorgenommen werden.

Ging dem EDIT-Befehl der Befehl SET FORMAT TO <Maskendatei> voraus, so erfolgt die Bildschirmgestaltung anhand der in der <Maskendatei> enthaltenen @-Befehle. Andernfalls werden die Felder in der Reihenfolge der Datenbankstruktur aufgelistet.

Arbeitet dBASE nicht im Seitenmodus, erscheint nach dem Aufruf des EDIT-Befehls die Nachricht "COORD:", da der Benutzer in diesem Fall die Koordinaten des zu ändernden Feldes eingeben muß. Diese setzen sich aus der Satznummer und der Nummer oder dem Namen des betreffenden Feldes zusammen. Der neue Wert, den dieses Feld erhalten soll, kann ebenfalls angegeben werden. Wird er nicht bei der Koordinateneingabe mitgeliefert, zeigt dBASE den bisherigen Wert des ausgewählten Feldes an und fragt "CHANGE?".

Soll der angezeigte Wert erhalten bleiben, genügt es, diese Frage durch Drücken der RETURN-Taste zu beantworten; andernfalls kann der neue Wert eingegeben werden. In beiden Fällen fragt dBASE daraufhin wieder mit "COORD:" nach den Koordinaten des nächsten zu ändernden Feldes.

Nach der ersten Koordinateneingabe kann der Benutzer dabei eine der beiden Koordinaten auslassen, sofern diese sich gegenüber dem vorangegangenen Satz bzw. Feld nicht geändert hat. Der EDIT-Befehl wird abgeschlossen durch Betätigen der RETURN-Taste als Antwort auf "COORD:".

Ein Datenfeld kann komplett gelöscht werden, indem man auf die Frage "CHANGE?" die Tastenfolge ctl-Y, RETURN eingibt. Anschließend kann der gesamte Feldinhalt neu eingegeben werden. Durch die Tastenkombination ctl-Q läßt sich der Editiervorgang jederzeit abbrechen, wobei der ursprüngliche Inhalt des bearbeiteten Feldes wiederhergestellt wird.

Wendet man den EDIT-Befehl bei einer indizierten Datenbank an, so wird jeder durch USE aktivierte Schlüssel aktualisiert, sobald Änderungen an den betreffenden Schlüsselfeldern vorgenommen werden. Nicht aktivierte Schlüsseldateien der benutzten Datenbank können selbstverständlich nicht aktualisiert werden.

Beispiele:

. USE KUNDEN

. DISPLAY STRUCTURE

STRUCTURE FOR FILE:

KUNDEN.DBF

NUMBER OF RECORDS: 00004

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	030	
002	ABTEILUNG	C	025	
003	STRASSE	C	025	
004	POSTLEITZ	N	004	
005	ORT	C	020	
006	TELEFON	C	015	
007	BRANCHE	C	020	
008	KUNDEN:NR	N	005	
** TOTAL **			00145	

. DISPLAY ALL NAME,KUNDEN:NR

00001 Meier & Co 2001

00002 Anlagenbau Schmidt 1004

00003 Schneider GmbH 8003

00004 Mikro GmbH 89

. EDIT

ENTER RECORD #,FIELD (# OR NAME),NEW VALUE
COORD: 1,1

NAME: Meier & Co
CHANGE? & Co
TO & Sohn

NAME: Meier & Sohn
CHANGE? (RETURN)

EDIT...280

COORD: 4,1,Softmikro GmbH

COORD: ,8

KUNDEN:NR: 89

TO : 95

COORD: (RETURN)

. DISPLAY ALL NAME,KUNDEN:NR

00001 Meier & Sohn 2001

00002 Anlagenbau Schmidt 1004

00003 Schneider GmbH 8003

00004 Softmikro GmbH 95

(Der folgende Auszug aus einer Befehlsdatei erlaubt die bequeme Änderung ausgewählter Sätze, wobei nur noch die gewünschte Satznummer angegeben werden muß. Durch die Makroersetzung wird die als Zeichenreihe eingegebene Ziffernfolge in die von EDIT benötigte Satznummer umgewandelt.)

STORE "1" TO X

DO WHILE X # "0"

ACCEPT "Gib Satznummer ein" TO X

EDIT &X

ENDDO

EJECT**EJECT**

Dieser Befehl bewirkt einen Seitenvorschub (d.h. Positionierung auf den Anfang der nächsten Seite). Voraussetzung hierfür ist die Belegung der SET-Parameter PRINT mit ON oder FORMAT mit PRINT (oder beides; siehe SET-Befehl). Bei Verwendung des @-Befehls zur Seitenformatierung werden durch EJECT die Zeilen- und Spaltenregister wieder auf Null gesetzt.

Beispiel:

. EJECT

ERASE

ERASE

Dieser Befehl löscht den Bildschirm und setzt den Cursor in dessen obere linke Ecke. Bei Verwendung des @-Befehls zur Bildschirmausgabe deaktiviert ERASE alle GET-Klauseln (siehe @-Befehl).

Beispiel:

. ERASE

FIND

FIND <Zeichenreihe> oder '<Zeichenreihe>' oder "<Zeichenreihe>"

Mit diesem Befehl läßt sich auf schnelle Weise in einer indizierten Datenbank der erste Satz mit dem Schlüssel <Zeichenreihe> auffinden. Bei einem Floppy-Disk-Computersystem beträgt die typische Zeit für die Ausführung eines FIND-Befehls zwei Sekunden.

Der FIND-Befehl läßt sich nur anwenden, wenn der Schlüssel, nach dem man suchen will, aktueller Hauptschlüssel der aktivierten Datenbank ist (siehe USE-Befehl). Ist der Schlüssel vom Typ Zeichenreihe, so genügt es, im FIND-Befehl als <Zeichenreihe> nur die Anfangsbuchstaben des Schlüsselwortes anzugeben, nach dem gesucht werden soll. FIND macht dann den ersten Satz der Datenbank ausfindig, dessen Schlüssel mit dieser Zeichenkombination anfängt. Beispielsweise ließe sich der Schlüssel 'HUBER, ANTON' mit dem Befehl 'FIND HUB' lokalisieren, vorausgesetzt, daß in der aktivierten Datenbank kein anderer Schlüssel vor 'HUBER, ANTON' mit der Buchstabenkombination 'HUB' beginnt. Der FIND-Befehl ermittelt immer den ersten Satz, dessen Schlüssel mit der angegebenen <Zeichenreihe> übereinstimmt, da er - unabhängig von der Nummer des aktuellen Satzes - immer am Anfang der Datenbank mit der Suche beginnt.

Wird bei der Sortierung der Schlüssel mit Hilfe der !-Funktion gebildet, muß der Suchbegriff ebenfalls aus Großbuchstaben bestehen, da nur die Index-Datei durchsucht wird.

Bei einem numerisch definierten Schlüssel ermittelt FIND den ersten Satz, dessen Schlüssel arithmetisch mit dem als <Zeichenreihe> angegebenen Wert übereinstimmt.

Man beachte, daß als Suchbegriffe sowohl für numerische als auch alphanumerische Schlüssel stets Werte anzugeben sind, die in Hochkomma (') oder Anführungszeichen (") - dann aber nicht gemischt - eingeschlossen sein können. Notwendig sind diese Sonderzeichen allerdings nur, wenn der Schlüssel, nach dem die Datenbank indiziert ist, führende Leerzeichen enthält. In diesem Fall muß die exakte Anzahl der führenden Leerzeichen als Bestandteil der in Hochkomma/Anführungszeichen eingeschlossenen <Zeichenreihe> enthalten sein.

Soll der Inhalt einer temporären Variable als Schlüsselwert verwendet werden, ist diese Variable in Form einer Makro-Ersetzung, d.h. ihr Name mit vorangestelltem "&", nach FIND anzugeben. Eine numerische temporäre Variable ist hierzu mit Hilfe der STR-Funktion in einen alphanumerischen Wert umzuwandeln.

Der Satz, der durch einen FIND-Befehl lokalisiert wurde, wird automatisch zum aktuellen Satz für dBASE, so daß seine Feldinhalte unmittelbar angezeigt, verändert oder zu Berechnungen herangezogen werden können.

Enthält die aktivierte Datenbank keinen Satz, dessen Schlüssel mit der angegebenen <Zeichenreihe> übereinstimmt, erscheint auf dem Bildschirm die Nachricht "NO FIND". Ferner liefert die Funktion zur Angabe der aktuellen Satznummer "#" den Wert 0.

Der Zugriff auf einen weiteren Satz mit demselben Schlüssel läßt sich durch die Befehle SKIP oder LOCATE FOR <Ausdruck> erreichen. SKIP macht allerdings nur den gemäß Schlüssel nächsten Satz zum aktuellen Satz, ohne dabei die Übereinstimmung des Schlüssels zu prüfen. Hingegen führt der LOCATE-Befehl - sofern man den <Ausdruck> unter Verwendung des Suchschlüssels formuliert - nur zum Erfolg, wenn mindestens ein weiterer Satz mit demselben Schlüssel in der Datenbank enthalten ist.

Mit SET EXACT ON (siehe Befehl SET) führt der FIND-Befehl nur zum Erfolg, wenn der Schlüsselwert mit dem Wert des Feldes im Satz exakt übereinstimmt (ausgenommen Leerzeichen am Ende des Feldes).

SET DELETE ON (siehe SET Befehle) bewirkt, daß der FIND-Befehl nur solche Sätze zugänglich macht, die nicht mit einer Löschmarkierung versehen sind.

Beispiele:

```
. USE BESTAND INDEX BESTNDX
```

```
. DISPLAY STRUCTURE
```

```
STRUCTURE FOR FILE: BESTAND.DBF
```

```
NUMBER OF RECORDS: 00009
```

```
DATE OF LAST UPDATE: 10/10/82
```

```
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

. LIST					
00004	Achse	70023	350.00	40	15.09.82
00007	Auspufftopf	12112	80.50	230	06.10.82
00001	Getriebe	10023	1200.00	12	12.07.82
00009	Kanister	1234	5.90	456	01.09.82
00002	Kolben	12221	81.00	230	04.09.82
00005	Kupplung	3567	261.00	42	10.10.82
00006	Rahmen	230	3460.00	12	03.05.82
00003	Reifen	987	70.00	210	28.08.82
00008	Welle	8978	290.80	34	11.10.82

. FIND Reifen

. DISPLAY					
00003	Reifen	987	70.00	210	28.08.82

. DISPLAY NEXT 3

00003	Reifen	987	70.00	210	28.08.82
00008	Welle	8978	290.80	34	11.10.82

(Man beachte, daß in der Schlüsseldatei nach "Reifen" nur noch ein Wert - nämlich "Welle" - enthalten ist.)

. FIND R

. DISPLAY					
00006	Rahmen	230	3460.00	12	03.05.82

. FIND Re

. DISPLAY					
00003	Reifen	987	70.00	210	28.08.82

. FIND R

. DISPLAY					
00006	Rahmen	230	3460.00	12	03.05.82

. DELETE

00001 DELETION(S)

. SET DELETE ON

. FIND R

FIND...286

. DISPLAY

00003	Reifen	987	70.00	210	15.10.82
-------	--------	-----	-------	-----	----------

. SET DELETE OFF

. FIND R

. DISPLAY

00012	*Rahmen	230	3460.00	12	03.05.82
-------	---------	-----	---------	----	----------

. USE ARTIKEL

. DISPLAY STRUCT

STRUCTURE FOR FILE: ARTIKEL.DBF

NUMBER OF RECORDS: 00010

DATE OF LAST UPDATE: 10/10/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	TITEL	C	020	
002	AUTOR	C	015	
003	REGISTER	C	008	
004	ERSCHIENEN	C	008	
** TOTAL **			00052	

. LIST

00001	Kaltblut	Cappellen, M.	IS0908	02.12.59
00002	Weltgeschichte	Trauter, I.N.	M09081	12.05.60
00003	100 Geschichten	Trauter, I.N.	IB123045	03.02.72
00004	Der Streiter	Baum, H.	K12390	10.09.77
00005	Das Dunkel	Anton, W.	L14509	12.09.78
00006	Das Pferd	Anton, K.	I2903	13.09.80
00007	Stadt im Schnee	Schend, K.	IS9809	13.06.79
00008	Nur Heute	Haltig, L.	IS09788	12.08.53
00009	Weg nach vorn?	Keil, B.	K0790122	13.07.81
00010	Das war's	Endig, L.	I78608	05.06.78

(Ein Schlüssel kann auch aus Feldern kombiniert werden, um eindeutige Ergebnisse bei FIND zu erhalten:)

. INDEX ON AUTOR + REGISTER TO AUTREG
00010 RECORDS INDEXED

. USE ARTIKEL INDEX AUTREG

. FIND Anton

. DISPLAY				
00006	Das Pferd	Anton, K.	I2903	13.09.80

. FIND Trauter, I.N.

. DISPLAY				
00003	100 Geschichten	Trauter, I.N.	IB123045	03.02.72

. FIND "Trauter, I.N. M09081"

(Die Anführungszeichen wurden nur zur Verdeutlichung des Schlüsselwertes angegeben; sie werden von dBASE II nicht verlangt)

. DISPLAY				
00002	Weltgeschichte	Trauter, I.N.	M09081	12.05.60

GO oder GOTO

- a. GOTO RECORD <n>
- b. GOTO TOP
- c. GOTO BOTTOM
- d. <n>
- e. GOTO <Variable>

Mit diesem Befehl lässt sich der (dBASE-interne) Zeiger auf den aktuellen Datensatz neu positionieren.

In den Fällen a und d wird dieser Zeiger auf den Satz mit der Nummer <n> gesetzt. Dabei stellt Form d die Kurzschreibweise von Form a dar.

Nach Ausführung des GOTO-Befehls in Form b oder c deutet der Zeiger auf den ersten (TOP) bzw. letzten (BOTTOM) Satz der aktivierten Datenbank. Bei indizierten Datenbanken müssen diese logischen ersten bzw. letzten Sätze nicht mit den physikalischen ersten/letzten Sätzen der Datei übereinstimmen, da sich die logische Reihenfolge nach den Werten im Hauptschlüssel richtet.

Fall e kann dazu benutzt werden, um mit Hilfe einer temporären Variablen auf einen Satz zu positionieren.

Beispiele:

. USE BESTAND

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Kupplung	3567	261.00	42	10.10.82
00006	Rahmen	230	3460.00	12	03.05.82
00007	Auspufftopf	12112	80.50	230	06.10.82
00008	Welle	8978	290.80	34	11.10.82
00009	Kanister	1234	5.90	456	01.09.82

. GOTO RECORD 6

GOTO...289

. DISPLAY				
00006	Rahmen	230	3460.00	12 03.05.82
. GOTO BOTTOM				
. DISPLAY				
00009	Kanister	1234	5.90	456 01.09.82
. GOTO TOP				
. DISPLAY				
00001	Getriebe	10023	1200.00	12 12.07.82
. STORE 4 TO SATZNR				
4				
. GOTO SATZNR				
. DISPLAY				
00004	Achse	70023	350.00	40 15.09.82

HELP

HELP [<Befehlswort>]

Mit dem HELP-Befehl lassen sich allgemeine Informationen über dBASE und kurze Übersichten über die einzelnen Befehle und ihre Verwendung anzeigen.

Der HELP-Befehl greift auf die ASCII-Datei DBASEMSG.TXT zu, die vom Anwender mit Hilfe eines Textverarbeitungsprogramms geändert werden kann.

Beispiel:

. HELP CREATE

IF

```

IF <Ausdruck>
  <Befehle>
[ELSE
  <Befehle>]
ENDIF

```

Der IF-Befehl ermöglicht die bedingte Ausführung von Befehlsfolgen und wird daher nur in Befehlsdateien verwendet. Wenn der angegebene <Ausdruck> den Wert "wahr" hat, kommen die unmittelbar hinter ihm stehenden Befehle zur Ausführung, andernfalls die Befehle, die durch das Steuerwort ELSE eingeleitet werden. Enthält der IF-Befehl keinen ELSE-Teil, so werden alle Befehle bis zum nächsten Steuerwort ENDIF übergangen. Zwischen den quasi ein Klammerpaar bildenden Steuerwörtern IF und ENDIF können weitere IF-Befehle angegeben werden, wobei die Schachtelungstiefe nicht begrenzt ist.

Hinweis: Unter <Befehlen> sind in diesem Zusammenhang alle zulässigen dBASE-Befehle zu verstehen, d.h. auch zusammengesetzte Befehle wie IF - ENDIF oder DO WHILE - ENDDO werden hier als jeweils ein Befehl betrachtet. Auf die korrekte Verschachtelung solcher Konstruktionen ist dabei zu achten. Wenn z.B. in einem der beiden IF-Äste ein DO WHILE-Befehl auftaucht, so muß das zugehörige Steuerwort ENDDO in demselben Ast, also vor dem nächsten ELSE bzw. ENDIF erscheinen (siehe auch Abschnitt 1, Ziffer 9.2 Regel 8 zu diesem Thema).

Beispiele:

```

IF STATUS = 'VERHEIRATET'
  DO DOPPKOST
ELSE
  DO EINZKOST
ENDIF

IF X = 1
  STORE PLZ + ORT TO BEST:ORT
ENDIF

```

INDEX

INDEX [ON <Ausdruck> TO <Schlüsseldatei>]

Durch den INDEX-Befehl wird die mittels USE aktivierte Datenbank nach dem im <Ausdruck> angegebenen Schlüssel indiziert. Dabei legt dBASE unter dem vom Benutzer festgelegten Namen eine Schlüsseldatei an, die Verweise auf alle Sätze dieser Datenbank enthält. Damit erscheint die Datenbank für den Benutzer nach dem Schlüssel sortiert. In der Datenbankdatei selbst werden hierbei jedoch keine Änderungen vorgenommen. Generell bringt der INDEX-Befehl die Sätze in eine aufsteigende Reihenfolge, bei numerischen Schlüsseln läßt sich jedoch - bei entsprechender Formulierung des <Ausdrucks> - auch absteigend sortieren.

Der Vorteil indizierter Datenbanken liegt in der Möglichkeit zum schnellen Zugriff auf einzelne Sätze über den Schlüssel (bei dBASE mittels FIND-Befehl; siehe dort). Nur wenn diese Art des Datenzugriffs häufig angewandt wird, lohnt es sich, eine Datenbank zu indizieren.

Um eine korrekte lexikalische Reihenfolge nach der Indizierung zu erreichen, sollte man bei der Eingabe ä, ö, ü und ß möglichst vermeiden. Diese Zeichen werden im ASCII-Code als Sonderzeichen interpretiert und deshalb nach allen übrigen Buchstaben eingeordnet. Eine korrekte Sortierung von Groß- und Kleinbuchstaben wird erreicht durch die Verwendung der Großbuchstaben-Funktion (Kapitel 3, Abschnitt 1, Ziffer 4.1).

Eine einmal indizierte Datenbank kann bei späteren Anwendungen je nach Belieben in ihrer ursprünglichen oder in indizierter Form bearbeitet werden, da die Schlüsseldatei nur dann aktiviert wird, wenn der Benutzer sie im USE-Befehl der Datenbank explizit zuordnet (siehe USE-Befehl).

In vielen Fällen genügt es, eine Datenbank ein einziges Mal pro Schlüssel zu indizieren, da alle dBASE-Befehle, die den Inhalt von Datenbanken verändern (z.B. APPEND, EDIT, REPLACE oder BROWSE), gleichzeitig bis zu 7 Schlüsseldateien aktualisieren, sofern diese zusammen mit der Datenbank aktiviert worden sind. Die Schlüsseldateien müssen einzeln durch INDEX ON <Schlüsselfeld> TO <Schlüsseldatei> angelegt worden sein.

Zu einer Datenbank lassen sich beliebig viele Schlüsseldateien anlegen. Allerdings werden immer nur die Indizes aktualisiert, deren Dateien zum aktuellen Zeitpunkt aktiviert sind (s.o.).

HINWEIS: Die TRIM-Funktion darf nicht Bestandteil des Schlüssel-<Ausdrucks> sein. Ferner ist bei der \$- bzw. STR-Funktion der Längenparameter als Zahlenwert anzugeben. Er darf nicht mit Hilfe von Variablen oder Ausdrücken dargestellt werden, also:

INDEX ON \$(NAME,N,5) + STR(ANZAHL,5) TO NDXDATEI anstelle von
INDEX ON \$(NAME,N,N + 5) + STR(ANZAHL,STELLEN) TO NDXDATEI

Der Befehl INDEX (ohne Zusätze) fügt nur den Schlüssel des aktuellen Satzes in die aktivierten Schlüsseldateien ein. So läßt sich für einzelne Datensätze festlegen, ob sie über Schlüssel angesprochen können oder nicht.

Beispiele:

. USE BESTAND

. DISPLAY STRUCTURE
 STRUCTURE FOR FILE: BESTAND.DBF
 NUMBER OF RECORDS: 00009
 DATE OF LAST UPDATE: 10/10/82
 PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Kupplung	3567	261.00	42	10.10.82
00006	Rahmen	230	3460.00	12	03.05.82
00007	Auspufftopf	12112	80.50	230	06.10.82
00008	Welle	8978	290.80	34	11.10.82
00009	Kanister	1234	5.90	456	01.09.82

. INDEX ON ARTIKEL TO ARTIKEL
 00009 RECORDS INDEXED

. FIND Auspufftopf

. DISPLAY

00007	Auspufftopf	12112	80.50	230	06.10.82
-------	-------------	-------	-------	-----	----------

. FIND R

. DISPLAY

00006	Rahmen	230	3460.00	12	03.05.82
-------	--------	-----	---------	----	----------

. SKIP

RECORD: 00003

. DISPLAY

00003	Reifen	987	70.00	210	28.08.82
-------	--------	-----	-------	-----	----------

. SKIP -1

RECORD: 00006

. DISPLAY

00006	Rahmen	230	3460.00	12	03.05.82
-------	--------	-----	---------	----	----------

. USE BESTAND

(Mit diesem Komando zur Aktivierung der Datenbank werden keine Schlüsseldateien angeschlossen.
Dies wird deutlich beim Auflisten der Sätze. Man beachte den Unterschied in der Reihenfolge, wenn
die Datenbank zusammen mit einer Schlüsseldatei aufgerufen wird.)

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Kupplung	3567	261.00	42	10.10.82
00006	Rahmen	230	3460.00	12	03.05.82
00007	Auspufftopf	12112	80.50	230	06.10.82
00008	Welle	8978	290.80	34	11.10.82
00009	Kanister	1234	5.90	456	01.09.82

. USE BESTAND INDEX BESTAND

. LIST

00004	Achse	70023	350.00	40	15.09.82
00007	Auspufftopf	12112	80.50	230	06.10.82
00001	Getriebe	10023	1200.00	12	12.07.82
00009	Kanister	1234	5.90	456	01.09.82
00002	Kolben	12221	81.00	230	04.09.82
00005	Kupplung	3567	261.00	42	10.10.82
00006	Rahmen	230	3460.00	12	03.05.82
00003	Reifen	987	70.00	210	28.08.82
00008	Welle	8978	290.80	34	11.10.82

. USE ARTIKEL

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: ARTIKEL.DBF

NUMBER OF RECORDS: 00010

DATE OF LAST UPDATE: 10/10/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	TITEL	C	020	
002	AUTOR	C	015	
003	REGISTER	C	008	
004	ERSCHIENEN	C	008	
** TOTAL **			00052	

. LIST

00001	Kaltblut	Cappellen, M.	IS0908	02.12.59
00002	Weltgeschichte	Trauter, I.N.	M09081	12.05.60
00003	100 Geschichten	Trauter, I.N.	IB123045	03.02.72
00004	Der Streiter	Baum, H.	K12390	10.09.77
00005	DAS Dunkel	Anton, W.	L14509	12.09.78
00006	Das Pferd	Anton, K.	I2903	13.09.80
00007	Stadt im Schnee	Schend, K.	IS9809	13.06.79
00008	Nur Heute	Haltig, L.	IS09788	12.08.53
00009	Weg nach vorn?	Keil, B.	K0790122	13.07.81
00010	Das war's	Endig, L.	I78608	05.06.78

(Schlüssel können auch aus Feldern kombiniert werden, um eindeutige Ergebnisse bei FIND zu erhalten:)

. INDEX ON AUTOR + REGISTER TO AUTOREG

00010 RECORDS INDEXED

. LIST

00006	Das Pferd	Anton, K.	I2903	13.09.80
00005	Das Dunkel	Anton, W.	L14509	12.09.78
00004	Der Streiter	Baum, H.	K12390	10.09.77
00001	Kaltblut	Cappellen, M.	IS0908	02.12.59
00010	Das war's	Endig, L.	I78608	05.06.78
00008	Nur Heute	Haltig, L.	IS09788	12.08.53
00009	Weg nach vorn?	Keil, B.	K0790122	13.07.81
00007	Stadt im Schnee	Schend, K.	IS9809	13.06.79
00003	100 Geschichten	Trauter, I.N.	IB123045	03.02.72
00002	Weltgeschichte	Trauter, I.N.	M09081	12.05.60

. USE ARTIKEL INDEX AUTOREG

. FIND Trauter, I.N.

. DISPLAY

00003	100 Geschichten	Trauter, I.N.	IB123045	03.02.72
-------	-----------------	---------------	----------	----------

. FIND "Trauter, I.N. M09081"

(Die Anführungszeichen wurden nur zur Verdeutlichung des Schlüsselwertes angegeben; sie werden von dBASE II nicht verlangt)

. DISPLAY

00002	Weltgeschichte	Trauter, I.N.	M09081	12.05.60
-------	----------------	---------------	--------	----------

INPUT

INPUT ['<Text>'] TO <Variable>

Dieser Befehl findet in Befehlsdateien Verwendung und dient dazu, vom Benutzer eingegebene Daten in temporären Variablen zu speichern. Existierte die angegebene <Variable> zum Zeitpunkt des INPUT-Befehls noch nicht, so wird sie erzeugt und der vom Benutzer eingegebene Ausdruck in ihr gespeichert. Enthält der INPUT-Befehl einen in Hochkomma oder Anführungszeichen eingeschlossenen <Text> (Beidseitig gleiches Zeichen verwenden!), so erscheint dieser bei Ausführung des Befehls auf dem Bildschirm und macht den Benutzer auf die geforderte Eingabe aufmerksam.

Den Typ der temporären <Variablen> bestimmt dBASE anhand der Eingabe. Numerische Werte werden an numerische Variablen zugewiesen, in Sonderzeichen eingeschlossene alphanumerische Zeichenreihen an alphanumerische Variablen. Besteht die Eingabe nur aus den Buchstaben T, Y, F oder N, so erfolgt die Zuweisung des Wahrheitswerts "wahr" (bei T oder Y) bzw. "falsch" (bei F oder N) an eine logische Variable. Zur Bestimmung des Datentyps der Eingabe bietet sich die Funktion TYPE an.

Der INPUT-Befehl sollte hauptsächlich zur Eingabe numerischer und logischer Werte eingesetzt werden. Für alphanumerische Werte stellt der ACCEPT-Befehl eine bequemere Eingabemöglichkeit dar, da der Benutzer bei ihm die Zeichenreihe nicht in Hochkomma bzw. Anführungszeichen einschließen muß.

Beispiele:

. INPUT TO X

:3

3

. INPUT TO Z

:23/17.000 + X

4.352

. INPUT 'NUTZER ZUR EINGABE AUFFORDERN' TO Q

NUTZER ZUR EINGABE AUFFORDERN:12345

12345

. INPUT 'GIB T EIN, WENN ALLES O.K. IST' TO LOG
GIB T EIN, WENN ALLES O.K. IST:T

.T.

. INPUT 'GIB ALPHANUMERISCHE ZEICHEN EIN' TO ALPHANUM
GIB ALPHANUMERISCHE ZEICHEN EIN:'Sonderzeichen nicht vergessen!!!'
Sonderzeichen nicht vergessen!!!

. DISP MEMO

X	(N)	3
Z	(N)	4.352
Q	(N)	12345
LOG	(L)	.T.
ALPHANUM	(C)	Sonderzeichen nicht vergessen!!!
** TOTAL **		05 VARIABLES USED 00050 BYTES USED

. INPUT 'GIB EINEN WAHRHEITSWERT EIN' TO LOG2
GIB EINEN WAHRHEITSWERT EIN:y

.T.

INSERT

INSERT [BEFORE] [BLANK]

Dieser Befehl ermöglicht das Einfügen eines Satzes an einer beliebigen Position der aktivierten Datenbank.

Ist die Option BEFORE angegeben, wird der neue Satz vor dem aktuellen Satz eingefügt, andernfalls unmittelbar dahinter. Das Steuerwort BLANK bewirkt das Einfügen eines leeren Satzes. Fehlt es, fordert dBASE den Benutzer wie beim APPEND-Befehl zur Eingabe der Feldinhalte auf. Wurde vor dem INSERT-Befehl der Befehl SET FORMAT TO <Maskendatei erteilt, werden - analog zum APPEND-Befehl - die in der <Maskendatei> enthaltenen @-Befehle zur Bildschirmgestaltung herangezogen.

Bei SET CARRY ON (siehe SET-Befehl) werden die Daten des vorhergehenden in den neuen Satz übernommen.

Bei großen, nicht indizierten Datenbanken erfordert der INSERT-Befehl sehr viel Zeit und sollte daher nur angewandt werden, wenn es unumgänglich ist. Bei indizierten Datenbanken ist INSERT äquivalent zu APPEND.

Beispiele:

. USE BESTAND

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: BESTAND.DBF

NUMBER OF RECORDS: 00009

DATE OF LAST UPDATE: 10/10/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Kupplung	3567	261.00	42	10.10.82
00006	Rahmen	230	3460.00	12	03.05.82
00007	Auspufftopf	12112	80.50	230	06.10.82
00008	Welle	8978	290.80	34	11.10.82
00009	Kanister	1234	5.90	456	01.09.82

. GOTO RECORD 4

. INSERT

RECORD 00005

ARTIKEL : Feder

ARTIKEL:NR: 19888

EINZ:PREIS: 56.70

BESTAND : 23

DATUM : 12.10.82

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Feder	19888	56.70	23	12.10.82
00006	Kupplung	3567	261.00	42	10.10.82
00007	Rahmen	230	3460.00	12	03.05.82
00008	Auspufftopf	12112	80.50	230	06.10.82
00009	Welle	8978	290.80	34	11.10.82
00010	Kanister	1234	5.90	456	01.09.82

. INSERT BEFORE
RECORD 00010

ARTIKEL : Generator

ARTIKEL:NR: 10015

EINZ:PREIS : 340.

BESTAND : 25

DATUM : 12.10.82

INSERT...300

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Feder	19888	56.70	23	12.10.82
00006	Kupplung	3567	261.00	42	10.10.82
00007	Rahmen	230	3460.00	12	03.05.82
00008	Auspufftopf	12112	80.50	230	06.10.82
00009	Welle	8978	290.80	34	11.10.82
00010	Generator	10015	340.00	25	12.10.82
000011	Kanister	1234	5.90	456	01.09.82

. 4

. DISPLAY

00004	Achse	70023	350.00	40	15.09.82
-------	-------	-------	--------	----	----------

. INSERT BLANK

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005		0	0.00	0	
00006	Feder	19888	56.70	23	12.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00008	Rahmen	230	3460.00	12	03.05.82
00009	Auspufftopf	12112	80.50	230	06.10.82
00010	Welle	8978	290.80	34	11.10.82
00011	Generator	10015	340.00	25	12.10.82
00012	Kanister	1234	5.90	456	01.09.82

. 5

. REPLACE ARTIKEL WITH 'Haube' AND ARTIKEL:NR WITH 997 AND ;
EINZ:PREIS WITH 46.90 AND BESTAND WITH 89 AND DATUM WITH '12.10.82'

00001 REPLACEMENT(S)

. LIST					
00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Haube	997	46.90	89	12.10.82
00006	Feder	19888	56.70	23	12.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00008	Rahmen	230	3460.00	12	03.05.82
00009	Auspufftopf	12112	80.50	230	06.10.82
00010	Welle	8978	290.80	34	11.10.82
00011	Generator	10015	340.00	25	12.10.82
00012	Kanister	1234	5.90	456	01.09.82

JOIN

JOIN TO <Dateiname> FOR <Ausdruck> [FIELDS <Felderfolge>]

JOIN ist einer der mächtigsten Befehle in dBASE. Mit ihm kann aus zwei Datenbanken eine dritte erzeugt werden.

Die beiden Datenbanken werden im Primär- bzw. Sekundärbereich von dBASE aktiviert (siehe auch SELECT-Befehl). Zunächst ist der Befehl SELECT PRIMARY zu geben, dem sich der JOIN-Befehl anschließt. JOIN positioniert auf den ersten Satz der Datenbank im Primärbereich und wertet die FOR-Bedingung für jeden Satz der im Sekundärbereich aktivierten Datenbank aus. Bei jedem Satz, der dieser Bedingung genügt, wird ein neuer Satz in die mit <Dateiname> vorgegebene Datenbank geschrieben. Wird das Ende der zweiten Datenbank erreicht, wird in der ersten Datenbank der nächste Satz eingestellt. Anschließend wird die zweite Datenbank erneut vom ersten Satz an durchlaufen. Dieser Prozess wiederholt sich, bis das Ende der ersten Datenbank erreicht ist.

Wird die FIELDS-Klausel (mit einer Auflistung von Feldern) nicht benutzt, so erhalten die Sätze der dritten Datenbank die Eintragungen der Sätze aus der ersten Datenbank und ggf. bis zur möglichen Anzahl von 32 Feldern Eintragungen (entsprechend ihrer Reihenfolge) aus den Sätzen der zweiten Datenbank.

Mit FIELDS werden nur die mit der <Felderfolge> bestimmten Eintragungen in die dritte Datenbank übernommen.

Bei großen Datenbanken benötigt dieser Befehl viel Zeit zur Ausführung. Ferner besteht die Gefahr, daß eine zu geringe Selektionswirkung der FOR-Bedingung die möglichen Grenzwerte für Datenbanken übersteigt. Wenn z. B. die beiden mit JOIN verbundenen Datenbanken je 1000 Sätze enthalten und die FOR-Bedingung immer erfüllt ist, müßte die dritte Datenbank 1000000 Sätze aufnehmen. Es sind aber nur 65535 Sätze in einer Datenbank zugelassen.

Beispiele:

```
. USE KATALOG
```

```
. DISPLAY STRUCTURE
```

```
STRUCTURE FOR FILE: KATALOG.DBF
```

```
NUMBER OF RECORDS: 00009
```

```
DATE OF LAST UPDATE: 10/12/82
```

```
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	PREIS	N	010	002
004	BESTAND	N	005	
** TOTAL **			00041	

. LIST				
00001	Zwinge	680	12.00	22
00002	Hammer	492	13.50	52
00003	Bohrmaschine	112	360.00	6
00004	Hobel	830	20.50	16
00005	Schubkarre	140	150.50	5
00006	Leiter	103	80.00	4
00007	Eimer	331	14.90	63
00008	Schaufel	120	13.00	75
00009	Waage	660	1260.80	2

. SELECT SECONDARY

. USE BESTELLG

. DISPLAY STRUCTURE
 STRUCTURE FOR FILE: BESTELLG.DBF
 NUMBER OF RECORDS: 00013
 DATE OF LAST UPDATE: 10/12/82
 SECONDARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ARTIKEL:NR	N	005	
003	ANZAHL	N	005	
** TOTAL **			00031	

. LIST

00001	Müller & Co	680	2
00002	Schneider GmbH	112	2
00003	Hansen KG	660	1
00004	Stahlbau	112	3
00005	Fortschritt e.V.	331	12
00006	Hoch und Tief AG	103	4
00007	Heimbedarf Knobel	830	3
00008	Gartenbau Wild	120	34
00009	Hobby-Bau	492	35
00010	Tischlerei Schmitz	680	10
00011	Wilken und Sohn	140	2
00012	Karl Anders	331	12
00013	Sanitär Reinhardts	120	30

. SELECT PRIMARY

. JOIN TO AUFWERT FOR P.ARTIKEL:NR = S.ARTIKEL:NR FIELDS NAME, S.ANZAH,
PREIS

. USE AUFWERT

. LIST

00001	Müller & Co	2	12.00
00002	Tischlerei Schmitz	10	12.00
00003	Hobby-Bau	35	13.50
00004	Schneider GmbH	2	360.00
00005	Stahlbau	3	360.00
00006	Heimbedarf Knobel	3	20.50
00007	Wilken und Sohn	2	150.50
00008	Hoch und Tief AG	4	80.00
00009	Fortschritt e.V.	12	14.90
00010	Karl Anders	12	14.90
00011	Gartenbau Wild	34	13.00
00012	Sanitär Reinhardt's	30	13.00
00013	Hansen GD	1	1260.80

LIST

- a. LIST [<Geltungsbereich>] [<Ausdrucksfolge>] [FOR <Ausdruck>] [FIELDS <Felderfolge>]
[WHILE <Ausdruck>] [OFF]
- b. LIST STRUCTURE
- c. LIST MEMORY
- d. LIST FILES [ON <Laufwerk>] [LIKE <Dateimaske>]
- e. LIST STATUS

Der LIST-Befehl entspricht in seiner Wirkungsweise dem DISPLAY-Befehl mit folgenden zwei Unterschieden:

- Die Voreinstellung für den <Geltungsbereich> ist in jedem Falle ALL.
- Die Ausgabe der spezifizierten Daten erfolgt ohne Unterbrechungen nach jeweils 15 Sätzen. Sie kann jederzeit mit der ESCAPE-Taste abgebrochen werden.

Die Formen b bis e dieses Befehls sind also vollkommen äquivalent zu ihren DISPLAY-Pendants.

LOCATE

LOCATE [<Geltungsbereich>] [FOR <Ausdruck>]
[CONTINUE]

Dieser Befehl sucht nach dem ersten Satz des spezifizierten <Geltungsbereichs>, auf den der angegebene <Ausdruck> zutrifft. Voreinstellung für den <Geltungsbereich> ist dabei ALL. Wird LOCATE fündig, so erscheint die Nachricht "RECORD n" auf dem Bildschirm. Mit dem Befehl CONTINUE kann der Benutzer die Suche fortsetzen, wobei er nach LOCATE zunächst andere Befehle ausführen lassen kann, bevor er CONTINUE auslöst. Hierdurch wird allerdings die Länge des <Ausdrucks> auf 128 anstelle der üblichen 254 Zeichen verringert (siehe auch CONTINUE).

Enthält die aktivierte Datenbank keinen Satz, auf den der <Ausdruck> zutrifft, so erscheint die Nachricht "END OF FILE ENCOUNTERED" und der interne Satzzeiger deutet auf den letzten Satz der Datenbank. Wurde als <Geltungsbereich> eine NEXT-Klausel (siehe 3. Kapitel, Abschnitt 1, Ziffer 9.1) verwendet und wird LOCATE in dem angegebenen Bereich nicht fündig, so erscheint die Meldung "END OF LOCATE SCOPE", und der Satzzeiger deutet auf den letzten durch NEXT spezifizierten Satz.

Hinweis: LOCATE arbeitet am schnellsten auf nicht indizierten bzw. ohne Schlüsseldatei aktivierten Datenbanken.

Beispiele:

. USE KATALOG

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: KATALOG.DBF

NUMBER OF RECORDS: 00009

DATE OF LAST UPDATE: 10/17/08

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	PREIS	N	010	002
004	BESTAND	N	005	

** TOTAL **

00041

. LIST				
00001	Zwinge	68	12.00	22
00002	Hammer	49	13.50	52
00003	Bohrmaschine	112	360.00	6
00004	Hobel	83	20.50	16
00005	Schubkarre	140	150.50	5
00006	Leiter	103	80.00	4
00007	Eimer	33	14.90	63
00008	Schaufel	12	13.00	75
00009	Waage	66	1260.80	2

. LOCATE FOR BESTAND < 10
RECORD: 00003

. CONTINUE
RECORD: 00005

. DISP ARTIKEL
00005 Schubkarre

. CONTINUE
RECORD: 00006

. CONTINUE
RECORD: 00009

. CONTINUE
END OF FILE ENCOUNTERED
. GO TOP

. LOCATE FOR PREIS > 100 NEXT 5
RECORD: 00003

. CONTINUE
RECORD: 00005

. CONTINUE
. END OF LOCATE SCOPE

LOOP

LOOP

Dieser Befehl wird im Rumpf von DO WHILE-Schleifen eingesetzt, um den Programmteil zwischen LOOP und dem nächsten ENDDO zu überspringen. Auf diese Weise lassen sich Befehlssequenzen, die nicht bei jedem Schleifendurchlauf ausgeführt werden müssen, bei Bedarf überspringen, wodurch insbesondere bei umfangreichen DO WHILE-Schleifen viel Zeit gespart werden kann. Die Wirkung von LOOP entspricht weitestgehend der von ENDDO, d.h. die Programmausführung wird beim unmittelbar vorausgehenden DO WHILE-Befehl fortgesetzt.

Die Verwendung des LOOP-Befehls in DO WHILE-Schleifen entspricht keinem guten Programmierstil, da durch diese Konstruktion die Übersichtlichkeit eines Programms stark leidet. Wie die untenstehenden Beispiele zeigen, lassen sich in den meisten Fällen andere Befehlsfolgen finden, die zum gleichen Effekt führen.

Beispiel 1:

```
STORE 1 TO INDEX
DO WHILE INDEX < 10
  STORE INDEX + 1 TO INDEX
  IF ARTIKEL:NR = ''
    SKIP
  LOOP
ENDIF
DO ÄNDERUNG
ENDDO
```

Im Falle einer leeren ARTIKEL:NR soll der betreffende Satz übergangen und die Programmausführung bei DO WHILE fortgesetzt werden.

Beispiel 2:

```
STORE 1 TO INDEX
DO WHILE INDEX < 10
  STORE INDEX + 1 TO INDEX
  IF ARTIKEL:NR = ''
    SKIP
  ELSE
    DO ÄNDERUNG
  ENDIF
ENDIF
ENDDO
```


MODIFY

- a. MODIFY STRUCTURE
- b. MODIFY COMMAND [< Befehlsdatei >]

Mit Form a dieses Befehls kann der Benutzer die Struktur der aktivierten Datenbank verändern. Alle Änderungen sind erlaubt, so daß neue Felder hinzugefügt, bisherige Felder aus der Datenbank eliminiert oder die Parameter einzelner Felder (Name, Typ, Länge und ggf. Anzahl der Dezimalstellen) neu definiert werden können.

Die Struktur der aktivierten Datenbank erscheint beim MODIFY STRUCTURE-Befehl auf dem Bildschirm, und der Benutzer kann mit den im Kapitel 3, Abschnitt 1, Ziffern 8.1 und 8.3 beschriebenen Tastenkombinationen die gewünschten Änderungen vornehmen.

Hinweis: Der MODIFY STRUCTURE-Befehl löscht alle Datensätze der aktivierten Datenbank. Um die Struktur einer Datenbank zu verändern und die Daten dennoch zu erhalten, empfiehlt sich folgendes Vorgehen: Zunächst kopiert man die Struktur der aktivierten Datenbank in eine Arbeitsdatei, die man anschließend als neue Datenbank aktiviert. An ihr kann man nun die gewünschten Änderungen durchführen und anschließend mittels APPEND die Sätze der alten Datenbank einlesen. Um die so modifizierte Datenbank wieder mit ihrem alten Namen versehen zu können, muß die ursprüngliche Datenbank gelöscht oder umbenannt werden (siehe hierzu auch untenstehendes Beispiel).

Mit dem MODIFY COMMAND-Befehl lassen sich in einer Art eingeschränktem Seitenmodus Änderungen an Befehls- oder anderen Standard-ASCII-Dateien vornehmen. Gibt der Benutzer im Befehl selbst keinen Namen einer < Befehlsdatei > an, fordert dBASE ihn gesondert dazu auf. Ein Dateityp muß dabei nur spezifiziert werden, wenn er von der Voreinstellung .CMD bzw. .PRG abweicht. Existiert unter dem angegebenen Namen noch keine Datei, so wird eine neue Datei angelegt. Nach Beendigung des MODIFY COMMAND-Befehls speichert dBASE die bearbeitete Datei unter dem vom Benutzer gewählten Namen, während die ggf. vorhandene alte Version den Dateityp .BAK erhält.

Wie bei Form a des MODIFY-Befehls gelten auch bei Form b die in den Abschnitten 8.1 und 8.3 beschriebenen Tastenkombinationen für die Cursor-Steuerung.

Der MODIFY COMMAND-Befehl unterliegt einigen wesentlichen Einschränkungen, die bei seiner Anwendung zu beachten sind. Im einzelnen:

1. Die Länge einer Zeile darf maximal 77 Stellen betragen (ein schließlich nicht sichtbarer Steuerzeichen, wie das durch die RETURN-Taste ausgelöste Paar < Wagenrücklauf > / < Zeilenvorschub >). Alle Zeichen jenseits der 77. Stelle werden abgeschnitten!
2. Der Steuercode zum Vorrücken an die nächste Tabulatorposition (ctl-I) wird in ein einzelnes Leerzeichen umgewandelt.
3. Die Größe der zu ändernden Datei sollte ca. 4000 Bytes nicht überschreiten, da dBASE nur innerhalb dieses Bereiches in der Lage ist, die Cursorposition zu kontrollieren.
4. Der MODIFY COMMAND-Befehl enthält keine Möglichkeit zur Textsuche oder zur Blockverschiebung wie die meisten Texteditoren.

MODIFY...310

Beispiel:

- . NOTE -- BEISPIEL ZUR ÄNDERUNG DER STRUKTUR EINER DATENBANK
- . NOTE OHNE ZERSTÖRUNG DER IN IHR ENTHALTENEN INFORMATION
- . NOTE
- . USE BESTAND
- . COPY STRUCTURE TO ARBDAT
- . USE ARBDAT
- . MODIFY STRUCTURE
- . APPEND FROM BESTAND
- . DELETE FILE BESTAND
- . USE
- . RENAME ARBDAT TO BESTAND

NOTE

- a. NOTE <Text>
- b. * <Text>

Dieser Befehl ermöglicht die Einbettung von Kommentaren in Befehlsdateien. Im Gegensatz zum REMARK-Befehl erscheint jedoch der angegebene <Text> bei Ausführung des NOTE-Befehls nicht auf dem Ausgabemedium.

Beispiele:

NOTE - Letzte Änderung: 30.10.82

```
*****  
*           Eingabeschleife          *  
*****
```

PACK**PACK**

Dieser Befehl bewirkt die physikalische Löschung aller durch DELETE mit der Löschmarkierung "*" gekennzeichneten Sätze der aktivierten Datenbank. Nach Ausführung des PACK-Befehls ist auf diese Sätze kein Zugriff mehr möglich.

Bei indizierten Datenbanken aktualisiert PACK auch alle aktivierten Schlüsseldateien.

Eine Alternative zu PACK stellt der COPY-Befehl dar, da er nur Sätze in eine neue Datei übernimmt, die keine Löschmarkierung "*" enthalten. Auf diese Weise läßt sich auch die alte Datenbankversion als Backup erhalten.

Der von einer Datenbankdatei belegte Diskettenspeicherplatz wird durch einen PACK-Befehl nicht reduziert, was allerdings auf eine Unzulänglichkeit des Betriebssystems CP/M zurückzuführen ist. Um nicht mehr benötigten Speicherplatz freizugeben, empfiehlt es sich ebenfalls, mittels COPY eine neue Version der Datenbankdatei anzulegen und anschließend die alte Datei zu löschen.

Da der PACK-Befehl - insbesondere bei großen Dateien - viel Zeit kostet, sollte man ihn nur so selten wie nötig verwenden. Eine Alternative, zur Löschung markierte Sätze von der weiteren Verarbeitung auszuschließen, stellt der Befehl SET DELETE ON dar (siehe SET-Befehle).

Beispiele:

```
. USE BESTAND
```

```
. DISPLAY STRUCTURE
```

```
STRUCTURE FOR FILE: BESTAND.DBF
```

```
NUMBER OF RECORDS: 00013
```

```
DATE OF LAST UPDATE: 00/00/00
```

```
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Haube	997	46.90	89	12.10.82
00006	Feder	19888	56.70	23	12.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00008	Rahmen	230	3460.00	12	03.05.82
00009	Auspufftopf	12112	80.50	230	06.10.82
00010	Welle	8978	290.80	34	11.10.82
00011	Generator	10015	340.00	25	12.10.82
00012	Kanister	1234	5.90	456	01.09.82

. DELETE RECORD 8

00001 DELETION(S)

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Haube	997	46.90	89	12.10.82
00006	Feder	19888	56.70	23	12.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00008	*Rahmen	230	3460.00	12	03.05.82
00009	Auspufftopf	12112	80.50	230	06.10.82
00010	Welle	8978	290.80	34	11.10.82
00011	Generator	10015	340.00	25	12.10.82
00012	Kanister	1234	5.90	456	01.09.82

. PACK

PACK COMPLETE, 00011 RECORDS COPIED

. LIST

00001	Getriebe	10023	1200.00	12	12.07.82
00002	Kolben	12221	81.00	230	04.09.82
00003	Reifen	987	70.00	210	28.08.82
00004	Achse	70023	350.00	40	15.09.82
00005	Haube	997	46.90	89	12.10.82
00006	Feder	19888	56.70	23	12.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00008	Auspufftopf	12112	80.50	230	06.10.82
00009	Welle	8978	290.80	34	11.10.82
00010	Generator	10015	340.00	25	12.10.82
00011	Kanister	1234	5.90	456	01.09.82

QUIT

QUIT [TO <Folge von .COM-Datei-Namen>]

Dieser Befehl schließt alle Datenbankdateien, Befehlsdateien und Ausgabedateien und gibt die Kontrolle an das Betriebssystem zurück. Auf dem Bildschirm erscheint ***** END RUN dBASE *****.

Enthält der QUIT-Befehl eine TO-Klausel, so werden anschließend die in der <Folge von .COM-Datei-Namen> aufgeführten Programme in der angegebenen Reihenfolge durch CP/M zur Ausführung gebracht. Mit dieser Konstruktion stellt dBASE eine bequeme Möglichkeit zur Verfügung, unmittelbar nach der Bearbeitung von Datenbanken zu anderer Anwendersoftware zu verzweigen, ohne den Umweg über das Betriebssystem gehen zu müssen.

Die Anzahl der angegebenen Programmnamen bzw. CP/M-Befehle ist nicht beschränkt, allerdings darf kein Element dieser Folge die Länge von 254 Zeichen überschreiten. Nach Ausführung des letzten Befehls/Programms wird die Kontrolle an CP/M übergeben. Will der Benutzer anschließend mit dBASE weiterarbeiten, so kann er als letztes Element der <Folge von .COM-Datei-Namen> wieder dBASE angeben.

Beispiel:

```
. QUIT TO 'DIR B:','PIP PRN: = ALTERNAT.TXT','DBASE VOREINST'
```

(In diesem Beispiel wird dBASE zunächst beendet, anschließend das Inhaltsverzeichnis (Directory) von Laufwerk B angezeigt, die Datei ALTERNAT.TXT mittels PIP.COM über den Drucker ausgegeben und schließlich dBASE erneut gestartet, wobei sofort die Befehlsdatei VOREINST.COM zur Ausführung gebracht wird.)

READ

READ [NOUPDATE]

Dieser Befehl aktiviert den Seitenmodus zur Eingabe und Änderung der Inhalte von Variablen, die durch GET-Klauseln vorangegangener @-Befehle angezeigt wurden. Der Cursor läßt sich dabei nach Belieben auf jede der angezeigten Variablen fahren. Die durchgeführten Änderungen/Eingaben werden in den entsprechenden Datenfeldern bzw. temporären Variablen gespeichert.

Erfolgte vor dem READ-Befehl ein SET FORMAT TO <Maskendatei>, so bringt READ die in der <Maskendatei> enthaltenen @-Befehle zur Ausführung. Man beachte, daß sich der Benutzer auf diese Weise selbst eine Art von Editierbefehlen erstellen kann, die sich ganz auf seinen Bedarf abstimmen lassen.

Nach einem SET FORMAT TO SCREEN-Befehl läßt sich der Bildschirm mittels ERASE löschen. Durch eine Folge von @-Befehlen wird anschließend eine neue Bildschirmmaske erstellt und ein nachfolgendes READ ermöglicht dann die Eingabe und Änderung von Daten.

Folgen auf diesen READ-Befehl weitere @-Befehle, ohne daß zwischenzeitlich der Bildschirm gelöscht wurde, so wird beim nächsten READ der Cursor auf die erste Variable gesetzt, die nach dem vorangegangenen READ zur Anzeige gebracht wurde. Auf diese Weise kann die weitere Gestaltung des Bildschirms in Abhängigkeit von den Eingaben erfolgen, die mit dem ersten READ eingelesen wurden.

Die in den GET-Klauseln des @-Befehls verwendeten Variablen müssen entweder Datenfelder der aktivierten Datenbank oder temporäre alphanumerische Variablen sein. Temporäre Variablen müssen bereits vor Ausführung des @-Befehls existieren. Notfalls füllt man sie zur Initialisierung mit der der gewünschten Stellenzahl entsprechenden Anzahl von Leerzeichen (z.B. STORE ' ' TO VAR).

Die für die Eingabe und Änderung von Daten im Seitenmodus benötigten Tastencodes sind im 3. Kapitel, Abschnitt 1, Ziffer 8 angegeben. Zur Aktivierung des Seitenmodus muß zuvor der Befehl SET SCREEN ON erfolgen (dies entspricht der Voreinstellung, wenn der Benutzer bei der Installation von dBASE die für den Seitenmodus benötigten Informationen angegeben hat).

Die Option NOUPDATE muß angegeben werden, wenn die automatische Aktualisierung der aktivierten Schlüsseldateien unterbleiben soll. Dies bietet sich z.B. an, wenn man genau weiß, daß durch den READ-Befehl keine Schlüsselfelder verändert werden. Fehlt der Zusatz NOUPDATE, werden alle eingelesenen Daten daraufhin untersucht, ob durch sie einer der aktivierten Schlüssel verändert wird. Insbesondere bei mehreren aktivierten Schlüsseldateien kann dieser Prozeß viel Zeit in Anspruch nehmen.

Beispiele:

```

STORE ' ' TO ZAHL:ART
STORE ' ' TO KONTO:NR
ERASE
@ 5,0 SAY 'Gib B für Barzahlung ein'
@ 6,0 SAY 'oder Ü für Überweisung'
@ 8,10 GET ZAHL:ART
READ
IF ZAHL:ART = 'Ü'
    @ 10,10 'Gib Konto-Nr. ein' GET KONTO:NR PICTURE '999-99-9999'
    READ
ENDIF

```

(In diesem Auszug aus einer Befehlsdatei erscheint nach dem Löschen des Bildschirms zunächst die Ausgabe der drei ersten @-Befehle, wobei der Cursor zwischen zwei Doppelpunkten im für die Variable ZAHL:ART vorgesehenen Feld steht. Da dieser Variablen durch den ersten STORE-Befehl die Länge 1 zugewiesen wurde, akzeptiert auch der READ-Befehl nur ein einziges Zeichen als Eingabe. Nur wenn der Benutzer an dieser Stelle ein 'Ü' eingibt, gelangt der nächste @-Befehl, der um Eingabe der Kontonummer bittet, zur Ausführung. Man beachte, daß bei der Definition von KONTO:NR im zweiten STORE-Befehl auch die in der PICTURE-Klausel enthaltenen Querstriche berücksichtigt wurden.)

```

USE SCHECKS
SET FORMAT TO SCREEN
ACCEPT "Wähle Option" TO AUSWAHL
IF AUSWAHL $ 'Aa'
NOTE -- NEUE SCHECKS AUFNEHMEN
ERASE
DO WHILE LFD:NUMMER # 0
APPEND BLANK
@ 5,0 SAY "Gib nächste laufende Nummer ein" ;
GET LFD:NUMMER PICTURE '99999'
@ 6,0 SAY "Gib Empfänger an" ;
GET EMPFÄNGER PICTURE 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
@ 7,0 SAY "Gib Betrag an" ;
GET BETRAG PICTURE '9999999.99'
@ 8,0 SAY "Scheck bereits eingelöst" ;
GET EINGELÖST
READ
ENDDO
ENDIF

```

(In diesem Beispiel wird eine Datenbank aktiviert und ihr Inhalt in Abhängigkeit von der ausgewählten Option unmittelbar verändert. Man beachte, daß für die Eingabe des Datenfelds EINGELÖST (Typ Wahrheitswert) keine PICTURE-Klausel angegeben werden muß. Für weitere Beispiele sei auf den @-Befehl verwiesen.)

RECALL

RECALL [<Geltungsbereich>] [FOR <Ausdruck>]
 [WHILE <Ausdruck>]

Dieser Befehl entfernt bei allen mittels DELETE zur Löschung gekennzeichneten Sätzen die Löschmarkierung "*", d.h. diese Sätze sind wieder für alle Befehle (z.B. APPEND, COPY, SORT usw.) zugänglich. Nähere Einzelheiten hierzu finden sich beim DELETE-Befehl.

Beispiele:

. USE NEUMITGL

. DISPLAY STRUCTURE
 STRUCTURE FOR FILE: NEUMITGL.DBF
 NUMBER OF RECORDS: 00007
 DATE OF LAST UPDATE: 10/17/08
 PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	MITGL:NR	N	005	
** TOTAL **			00025	

. LIST

00001	Baum, Kurt	420
00002	Eckart, Franz	456
00003	Fritzen, Iris	501
00004	Mathies, Anton	67
00005	Nolling, Gerhard	114
00006	Stender, Max	404
00007	Zander, Ulf	512

. 3

. DELETE NEXT 3
 00003 DELETION(S)

RECALL...318

. LIST

00001	Baum, Kurt	420
00002	Eckart, Franz	456
00003	*Fritzen, Iris	501
00004	*Mathies, Anton	67
00005	*Nolling, Gerhard	114
00006	Stender, Max	404
00007	Zander, Ulf	512

. RECALL RECORD 4

00001 RECALL(S)

. RECALL ALL

00002 RECALL(S)

. LIST

00001	Baum, Kurt	420
00002	Eckart, Franz	456
00003	Fritzen, Iris	501
00004	Mathies, Anton	67
00005	Nolling, Gerhard	114
00006	Stender, Max	404
00007	Zander, Ulf	512

. DELETE ALL

00007 DELETION(S)

. RECALL ALL FOR MITGL:NR > 500

00002 RECALL(S)

. LIST

00001	*Baum, Kurt	420
00002	*Eckart, Franz	456
00003	Fritzen, Iris	501
00004	*Mathies, Anton	67
00005	*Nolling, Gerhard	114
00006	*Stender, Max	404
00007	Zander, Ulf	512

REINDEX

REINDEX

Der REINDEX-Befehl entspricht in seiner Wirkungsweise dem Befehl INDEX ON <Ausdruck> TO <Schlüsseldatei>. Einziger Unterschied ist, daß er keine Parameter benötigt, da er alle augenblicklich aktivierten Schlüsseldateien aktualisiert.

Beispiel:

```
. USE BESTAND INDEX BESTNDX
```

```
. REINDEX
```

```
REINDEXING INDEX FILE - B:BESTNDX.NDX  
00012 RECORDS INDEXED
```

RELEASE

RELEASE [<Variablenfolge>] [ALL]
 [ALL LIKE <Dateimaske>] [ALL EXCEPT <Dateimaske>]

Dieser Befehl deaktiviert die in der <Variablenfolge> angegebenen temporären Variablen und stellt ihren Speicherplatz dem System wieder zur Verfügung. Das Steuerwort ALL bewirkt die Löschung aller temporären Variablen.

Die <Dateimaske> entspricht der <Dateimaske> + des DISPLAY-Befehls, d.h. "?" und "*" können entsprechend den CP/M-Konventionen verwendet werden.

Beispiele:

. DISPLAY MEMORY

VARABX	(C)	ABCDE
VARFFX	(C)	0FFH
VARRRX	(N)	123
VARAB	(C)	abcd
VARABY	(N)	3001.22
VORAB	(L)	.T.
VOR	(L)	.F.
VA	(N)	17
VARXXXXX	(C)	XYZ
** TOTAL **		09 VARIABLES USED 00045 BYTES USED

. RELEASE ALL LIKE VAR??X

. DISPLAY MEMORY

VARAB	(C)	abcd
VARABY	(N)	3001.22
VORAB	(L)	.T.
VOR	(L)	.F.
VA	(N)	17
VARXXXXX	(C)	XYZ
** TOTAL **		06 VARIABLES USED 00027 BYTES USED

. RELEASE ALL EXCEPT VAR*

. DISPLAY MEMORY

VARAB	(C)	abcd
VARABY	(N)	3001.22
VARXXXXX	(C)	XYZ
** TOTAL **		03 VARIABLES USED 00016 BYTES USED

REMARK

REMARK <Text>

Dieser Befehl ermöglicht - im Gegensatz zum NOTE-Befehl - die Ausgabe eines Kommentar-<Textes> auf dem Ausgabemedium. In der Regel findet der REMARK-Befehl nur in Befehlsdateien Verwendung.

Beispiel:

```
. REMARK ***** REMARK TEST *****  
***** REMARK TEST *****
```

RENAME ...322

RENAME

RENAME <bisheriger Dateiname> TO <neuer Dateiname>

Mit diesem Befehl lassen sich beliebige Dateien von dBASE aus umbenennen. Die Voreinstellung für den Dateityp ist ".DBF" bei beiden Dateinamen, doch kann der Benutzer auch jeden anderen Dateityp wählen.

Beispiele:

- . RENAME BESTELLG TO BESTALT
- . RENAME B:REPORT.FRM TO REPORT.BAK
- . RENAME OHNETYP. TO MITTYP.TYP

REPLACE

```

REPLACE [ <Geltungsbereich> ] <Feld> WITH <Ausdruck>
[ , <Feld> WITH <Ausdruck> ] [ FOR <Ausdruck> ]
[ NOUPDATE ] [WHILE <Ausdruck>]

```

Mit diesem Befehl lassen sich bei der durch USE aktivierten Datenbank die Inhalte der aufgeführten Datenfelder durch die zugehörigen <Ausdrücke> ersetzen.

Ist kein <Geltungsbereich> angegeben, erfolgen Ersetzungen nur beim aktuellen Satz.

Wird ein REPLACE bei einem Schlüsselfeld, das durch die INDEX-Klausel im USE-Befehl aktiviert wurde, durchgeführt, so erfolgt die Aktualisierung der entsprechenden Schlüsseldatei, indem der alte Eintrag gelöscht und der neue Eintrag an der entsprechenden Stelle vorgenommen wird. Dies kann bewirken, daß der aktuelle Satz seine Position innerhalb der Datenbank ändert, so daß der neue "nächste Satz" möglicherweise nicht mehr mit dem "nächsten Satz" vor Ausführung des REPLACE-Befehls übereinstimmt. Aus diesem Grunde sollten Schlüsselfelder nicht durch REPLACE verändert werden, wenn als <Geltungsbereich> "NEXT <n>" angegeben wird. Auf Schlüsseldateien, die nicht durch USE aktiviert wurden, wirkt sich der REPLACE-Befehl nicht aus.

Soll die automatische Prüfung und Aktualisierung der aktivierten Schlüsseldateien unterbleiben, so muß die Option NOUPDATE angegeben werden. Dies bietet sich insbesondere an, wenn in einer Datenbank mit mehreren aktivierten Schlüsseldateien viele Änderungen durchgeführt werden müssen. In diesem Fall spart man Zeit, wenn man erst nach Beendigung aller REPLACE-Befehle die aktivierten Schlüsseldateien mit einem REINDEX-Befehl aktualisiert (siehe auch REINDEX).

Hinweis: Bei Verwendung von PRIMARY- und SECONDARY-Datenbankdateien kann der REPLACE-Befehl nur auf der Datei ausgeführt werden, die mittels SELECT zuletzt angesprochen wurde.

Beispiele:

```
. USE AUSZHLG
```

```
. DISPLAY STRUCTURE
```

```
STRUCTURE FOR FILE: AUSZHLG.DBF
```

```
NUMBER OF RECORDS: 00013
```

```
DATE OF LAST UPDATE: 10/10/82
```

```
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	SCHECK:NR	N	005	
002	EMPF	C	020	
003	BETRAG	N	010	002
004	ANGEWIESEN	L	001	
005	ABGEBUCHT	L	001	
** TOTAL **			00038	

REPLACE...324

. LIST

00001	2 Autowerkstatt	560.20	.T. .F.
00002	5 Mietbau	12300.00	.F. .T.
00003	6 selbst	300.00	.T. .F.
00004	7 Zahnarzt	450.12	.T. .T.
00005	8 Krankenhaus	1370.76	.T. .F.
00006	9 Supermarkt	120.70	.T. .T.
00007	25 selbst	450.00	.T. .F.
00008	10 Warenhaus	345.60	.T. .F.
00009	11 Tierarzt	80.30	.T. .F.
00010	12 selbst	600.00	.T. .F.
00011	13 Tankstelle	90.50	.T. .F.
00012	14 Versicherung	540.80	.F. .T.
00013	15 selbst	800.00	.T. .F.

. REPLACE ABGEBUCHT WITH T FOR EMPF = 'selbst'

00004 REPLACEMENT(S)

. LIST

00001	2 Autowerkstatt	560.20	.T. .F.
00002	5 Mietbau	12300.00	.F. .T.
00003	6 selbst	300.00	.T. .T.
00004	7 Zahnarzt	450.12	.T. .T.
00005	8 Krankenhaus	1370.76	.T. .F.
00006	9 Supermarkt	120.70	.T. .T.
00007	25 selbst	450.00	.T. .T.
00008	10 Warenhaus	345.60	.T. .F.
00009	11 Tierarzt	80.30	.T. .F.
00010	12 selbst	600.00	.T. .T.
00011	13 Tankstelle	90.50	.T. .F.
00012	14 Versicherung	540.80	.F. .T.
00013	15 selbst	800.00	.T. .T.

. 13

. REPLACE ABGEGEBUCHT WITH F

00001 REPLACEMENT(S)

. DISPLAY

00013	15 selbst	800.00	.T. .F.
-------	-----------	--------	---------

. USE BESTAND

. COPY ALL TO B:PREISNEU FIELD ARTIKEL, EINZ:PREIS
00011 RECORDS COPIED

. USE PREISNEU

. DISPLAY STRUCT
STRUCTURE FOR FILE: PREISNEU.DBF
NUMBER OF RECORDS: 00011
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	EINZ:PREIS	N	007	002
** TOTAL **			00028	

. LIST		
00001	Getriebe	1200.00
00002	Kolben	81.00
00003	Reifen	70.00
00004	Achse	350.00
00005	Haube	46.90
00006	Feder	56.70
00007	Kupplung	261.00
00008	Auspufftopf	80.50
00009	Welle	290.80
00010	Generator	340.00
00011	Kanister	5.90

(Das folgende Beispiel zeigt die Einarbeitung einer Preiserhöhung um 10 % mit anschließender Aufrundung auf volle DM)

. REPLACE EINZ:PREIS WITH EINZ:PREIS * 1.1
00001 REPLACEMENT(S)

. REPLACE ALL EINZ:PREIS WITH EINZ:PREIS * 1.1
00011 REPLACEMENT(S)

REPLACE...326

. LIST

00001	Getriebe	1320.00
00002	Kolben	89.10
00003	Reifen	77.00
00004	Achse	385.00
00005	Haube	51.59
00006	Feder	62.37
00007	Kupplung	287.10
00008	Auspufftopf	88.55
00009	Welle	319.88
00010	Generator	374.00
00011	Kanister	6.49

. REPLACE ALL EINZ:PREIS WITH INT(EINZ:PREIS) + 1 FOR ;
(EINZ:PREIS - INT(EINZ:PREIS)) > 0

00007 REPLACEMENT(S)

. LIST

00001	Getriebe	1320.00
00002	Kolben	90.00
00003	Reifen	77.00
00004	Achse	385.00
00005	Haube	52.00
00006	Feder	63.00
00007	Kupplung	288.00
00008	Auspufftopf	89.00
00009	Welle	320.00
00010	Generator	374.00
00011	Kanister	7.00

REPORT

```
REPORT [ FORM <Formatdatei> ] [ <<Geltungsbereich> ] [ TO PRINT ]
      [ FOR <Ausdruck> ] [ PLAIN ] [ WHILE <Ausdruck> ]
```

REPORT ermöglicht die Abfassung standardisierter Berichte, die auf dem Bildschirm oder auf dem Drucker ausgegeben werden können. Standardisiert sind die Berichte insofern, als dBASE mit REPORT im Dialog eine Reihe von vorprogrammierten Möglichkeiten anbietet, aus denen eine Auswahl zu treffen ist. Hierdurch erspart man sich die aufwendigere Programmierung individuell gestalteter Berichte mittels der Ausgabebefehle ? und @. Mit REPORT wird das Schema festgelegt, welche der in der mit dem USE-Befehl aktivierten Datenbank enthaltenen Daten wie aufgelistet werden sollen. Zugelassen sind darüberhinaus Spaltenüberschriften, Berechnung von Summen über numerische Felder sowie Ausdrücke, deren Wert sich aus Datenfeldern, temporären Variablen und Konstanten bestimmt.

Die FOR-Angabe bezieht in den Bericht nur die Daten ein, die der im <Ausdruck> genannten Bedingung genügen. Ist TO PRINT im REPORT Befehl enthalten, erfolgt die Ausgabe des Berichts sowohl auf dem Bildschirm als auch auf dem Drucker. Wenn vom Benutzer nichts anderes vorgegeben wird bzw. wenn keine Ausgabe hierzu erfolgt, ist der <Geltungsbereich> entsprechend der Voreinstellung mit ALL festgelegt.

Das mit REPORT erzeugte Schema wird in einer Datei des Typs M abgelegt. Fehlt die Angabe FORM <Formatdatei> bzw. ist die angegebene Datei nicht vorhanden, so geht dBASE davon aus, daß ein neuer Bericht definiert werden soll. Nachdem im Dialog das Schema festgelegt wurde, legt dBASE hierfür die Datei an (ggf. unter Erfragung des Namens für die <Formatdatei>). Nachfolgende REPORT-Befehle können diese Datei ohne erneute Spezifikation des Schemas verwenden, z. B. mit anderen <Geltungsbereichen> oder geänderten Angaben zu FOR <Ausdruck>.

Im folgenden Beispiel werden fast alle Möglichkeiten für die Festlegung eines Schemas benutzt. Das auf einer Berichtssseite genutzte Format wird dabei wie folgt bestimmt:

- M: Breite linker Rand (Voreinstellung: 8 Leerstellen),
- L: Zeilen pro Seite (Voreinstellung: 57 Zeilen),
- W: Breite der Seite (Voreinst.: 80 Zeichen inkl. linker Rand).

Die Angabe zur Seitenbreite dient zur Zentrierung der Seitenüberschrift (Kopfzeile).

```
. DISPLAY STRUCTURE
STRUCTURE FOR FILE: BESTAND.DBF
NUMBER OF RECORDS: 00011
DATE OF LAST UPDATE: 10/15/82
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

REPORT...328

. REPORT

ENTER REPORT FORM NAME: BESTBERI

ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH

M=5,L=72,W=80

PAGE HEADING? (Y/N) Y

ENTER PAGE HEADING: BESTANDS - BERICHT

DOUBLE SPACE REPORT? (Y/N) Y

ARE TOTALS REQUIRED? (Y/N) Y

SUBTOTALS IN REPORT? (Y/N) N

COL WIDTH,CONTENTS

001 23,ARTIKEL + '...'

002 10,EINZ:PREIS

ENTER HEADING: <Einzel-;Preis

ARE TOTALS REQUIRED? (Y/N) N

003 8,BESTAND

ENTER HEADING: >Bestand;-----

ARE TOTALS REQUIRED? (Y/N) N

004 12,EINZ:PREIS * BESTAND

ENTER HEADING: Preis * Bestand

ARE TOTALS REQUIRED? (Y/N) Y

005 10,DATUM

ENTER HEADING: <Datum

006 (RETURN)

(Ausgabe des Berichts)

PAGE NO. 00001
10/28/82

BESTANDS - BERICHT

Artikel		Einzel- Preis	Bestand	Preis * Bestand	Datum
Getriebe	...	1320.00	14	18480.00	15.10.82
Kolben	...	90.00	233	20970.00	15.10.82
Reifen	...	77.00	210	16170.00	15.10.82
Achse	...	350.00	40	14000.00	15.09.82
Haube	...	46.90	89	4174.10	12.10.82
Feder	...	56.70	23	1304.10	12.10.82
Kupplung	...	261.00	42	10962.00	10.10.82
Auspufftopf	...	80.50	230	18515.00	06.10.82
Welle	...	320.00	34	10880.00	15.10.82
Generator	...	340.00	25	8500.00	12.10.82
Kanister	...	5.90	456	2690.40	01.09.82
** TOTAL **				126645.60	

Wie das Beispiel zeigt, erfragt REPORT nacheinander für die Spalten (COL) des Berichts die Anzahl der auszugebenden Stellen (WIDTH) und den Namen, des dafür vorgesehen Datenbankfeldes (CONTENTS). Dabei ist die Spaltenbreite im Bericht unabhängig von der aktuellen Länge des Datenbankfeldes definierbar. Im Beispiel wurde die Spalte 1 (COL 001) auf eine Länge von 23 Stellen festgelegt, obwohl das Datenbankfeld "ARTIKEL" nur 20 Stellen lang ist. Man beachte, daß der Inhalt von "ARTIKEL" im Bericht um die Zeichenreihe "..." ergänzt wird, was die zusätzlichen 3 Stellen der Spalte im Bericht ausmacht. Ist hingegen die Spaltenbreite des Berichts geringer als die Länge des dort auszugebenden Feldes, so bricht dBASE das Feld in mehrere Zeilen in dieser Spalte auf. Ein 80-stelliges alphanumerisches Feld würde also in einer 50-stelligen Spalte zwei Zeilen benötigen.

Außer Datenfeldern können in einer Spalte auch temporäre Variable oder das Ergebnis von Ausdrücken ausgegeben werden. Im Beispiel sind solche Ausdrücke bei Spalte 001 - Ergänzung des Feldinhaltes um "..." sowie in Spalte 004 - Produkt aus EINZ:PREIS und ANZAHL - gegeben. Die Datenbank wird durch solche Ausdrücke, die nur im Bericht selbst ausgewertet werden, nicht verändert. REPORT greift ausschließlich lesend auf die Datenbank zu, wie die folgende Auflistung der Datensätze zeigt.

. LIST

00001	Getriebe	10023	1320.00	14	15.10.82
00002	Kolben	12221	90.00	233	15.10.82
00003	Reifen	987	77.00	210	15.10.82
00004	Achse	70023	350.00	40	15.09.82
00005	Haube	997	46.90	89	12.10.82
00006	Feder	19888	56.70	23	12.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00008	Auspufftopf	12112	80.50	230	06.10.82
00009	Welle	8978	320.00	34	15.10.82
00010	Generator	10015	340.00	25	12.10.82
00011	Kanister	1234	5.90	456	01.09.82

Mit der Antwort "Y" auf DOUBLE SPACE REPORT? (Y/N) wird ein zweizeiliger Abstand zwischen den Zeilen für die Ausgabe eingestellt.

Bei PAGE HEADING: bzw. ENTER HEADING: sind die gewünschten Überschriften für die Seite(n) bzw. die Spalten einzugeben. Man beachte, daß die Spaltenüberschrift erst nach den Angaben zur Breite der Spalte und ihrem Inhalt erfragt wird.

Bei diesen wie bei allen alphanumerischen Angaben (etwa in einem Ausdruck) gilt, daß mit einem Semikolon (;) ein Zeilenumbruch bewirkt wird: der nach dem Semikolon stehende Text erscheint also auf der nächsten Zeile (ggf. in derselben Spalte). Eine Anwendung ist z. B. das Unterstreichen von Überschriften. Ist ein Text für den vorgesehenen Raum zu lang, so wird er von dBASE - soweit möglich - bis zu einer passenden Leerstelle auf einer Zeile ausgegeben und dann in der nächsten Zeile fortgesetzt. Die Sonderzeichen "<" bzw. ">" steuern, ob die Spaltenüberschrift links- bzw. rechtsbündig erfolgen soll. Werden sie nicht verwendet, erscheint die Überschrift in der Spaltenbreite zentriert.

Weitere, vorprogrammierte Möglichkeiten für die Festlegung des Schemas in einem Bericht bieten die Berechnungen von Summenzügen über numerische Felder. Mit der Antwort "Y" auf ARE TOTALS REQUIRED? (Y/N) wird erreicht, daß bei jeder Spalte mit numerischen Werten die gleiche Frage nochmals gestellt wird. Damit lassen sich gezielt die Spalten auswählen, die jeweils für sich zu summieren sind. Das Ergebnis wird unter ** TOTAL ** am Ende des Berichts ausgewiesen.

SUBTOTALS IN REPORT? (Y/N) mit "Y" liefert Zwischensummen über das bei ENTER SUBTOTALS FIELD: eingegebene Datenfeld. Im Bericht werden damit alle Sätze, die in diesem Feld den gleichen Wert aufweisen, nacheinander ausgegeben. Soweit numerische Felder im Bericht vorgesehen sind, werden sie über alle zusammengehörenden Sätze summiert und das Ergebnis als Zwischensumme ausgegeben. Es erfolgt also im Bericht eine Gruppierung der Datensätze nach gleichem Inhalt im Zwischensummenfeld. Der Inhalt dieses Feldes wird zu Beginn jeder Gruppe angegeben und kann mit einem eigenen Text gemäß der Angabe zu ENTER SUBTOTAL HEADING: versehen werden. dBASE ergänzt diesen Text zur besonderen Kennzeichnung mit einem Stern (*). Nach jeder Zwischensumme (in der Ausgabe als ** SUBTOTAL ** gekennzeichnet) kann ein Seitenumbruch mit "Y" auf EJECT PAGE AFTER SUBTOTALS? (Y/N) verlangt werden.

Nach wie vor lassen sich neben den Zwischensummen auch die Gesamtsummen bilden (siehe ARE TOTALS REQUIRED? (Y/N)). Zusätzlich sind Berichte definierbar, die nur die Ergebnisse der Summenberechnungen ausweisen, also auf die Auflistung der Werte der einzelnen Datensätze verzichten. Hierzu ist "Y" bei SUMMARY REPORT ONLY? (Y/N) einzugeben.

Mit RETURN auf die Ausgabe einer neuen Spaltennummer - REPORT erwartet dann die Eingaben zur Spaltenbreite und zum Inhalt - wird der Dialog zur Festlegung des Berichtsschemas beendet. Es beginnt die Erstellung des Berichtes durch dBASE, der dann auf dem Bildschirm, und bei Verwendung von TO PRINT auch auf dem Drucker erscheint.

Die Gestaltung des mit REPORT definierten Berichtes kann zusätzlich mit anderen dBASE-Befehlen geändert werden:

- SET EJECT OFF zur Unterdrückung des Seitenvorschubes, der sonst zu Beginn eines Berichtes erfolgt,
- SET HEADING TO als Möglichkeit zur Angabe einer weiteren Seitenüberschrift,
- SET DATE TO für die Ausgabe des Datums unterhalb der Seitennumerierung.

Im Gegensatz zu den Angaben in der <Formatdatei> bleiben diese Angaben nur während der dBASE-Sitzung erhalten, müssen also mit jedem Start von dBASE ggf. neu definiert werden. Weitere Einzelheiten hierzu finden sich bei den SET-Befehlen.

Mit der Option PLAIN im REPORT-Befehl kann die Angabe der Seitennummer und des Datums im Bericht unterdrückt werden; die Seitenüberschrift erscheint dann nur auf der ersten Seite. Damit soll die weitere Verarbeitung der Berichte mit Textverarbeitungsprogrammen erleichtert werden (siehe auch SET-Befehle zur Ausgabe von Berichten in Dateien)

Hinweis: Die von dBASE erzeugte <Formatdatei> kann mit Texteditoren bearbeitet werden (in dBASE selbst mittels MODIFY COMMAND). Die Eintragungen in diese Datei entsprechen den Antworten des Dialogs, in dem das Schema bestimmt wurde. Durch Änderungen direkt in dieser Datei, die natürlich mit den in REPORT vorgesehenen Möglichkeiten übereinstimmen müssen, kann eine Änderung der Spezifikationen zum Bericht vorgenommen werden, ohne daß der Bericht mit REPORT neu definiert werden muß.

Weitere Beispiele (Datenbank BESTAND):

. SET DATE TO 11.10.82

. SET HEADING TO

Lager 25

(Im folgenden Beispiel wird die Formatdatei BESTBERI erneut verwendet)

. REPORT FORM BESTBERI FOR ARTIKEL:NR > 1000

(Folgender Bericht erscheint:)

PAGE NO. 00001
11/10/82

Lager 25

BESTANDS - BERICHT

Artikel	Einzel- Preis	Bestand	Preis * Bestand	Datum
Getriebe	1320.00	14	18480.00	15.10.82
Kolben	90.00	233	20970.00	15.10.82
Achse	350.00	40	14000.00	15.09.82
Feder	56.70	23	1304.10	12.10.82
Kupplung	261.00	42	10962.00	10.10.82
Auspufftopf	80.50	230	18515.00	06.10.82
Welle	320.00	34	10880.00	15.10.82
Generator	340.00	25	8500.00	12.10.82
Kanister	5.90	456	2690.40	01.09.82
** TOTAL **				

(Beispiel mit Datenbank BESTELLG für Zwischen- und Gesamtsummen)

. USE BESTELLG INDEX BESTELLG

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: BESTELLG.DBF

NUMBER OF RECORDS: 00013

DATE OF LAST UPDATE: 10/15/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ARTIKEL:NR	C	005	
003	ANZAHL	N	005	
** TOTAL **			00031	

. LIST

00006	Hoch und Tief AG	103	4
00002	Schneider GmbH	112	2
00004	Stahlbau	112	3
00008	Gartenbau Wild	120	34
00013	Sanitär Reinhardts	120	30
00011	Wilken und Sohn	140	2
00005	Fortschritt e.V.	331	12
00012	Karl Anders	331	12
00009	Hobby-Bau	492	35
00003	Hansen KG	660	1
00001	Müller & Co	680	2
00010	Tischlerei Schmitz	680	10
00007	Heimbedarf Knobel	830	3

. REPORT FORM ARTSUM PLAIN

ENTER OPTIONS, M = LEFT MARGIN, L = LINES/PAGE, W = PAGE WIDTH M = 5

PAGE HEADING? (Y/N) Y

ENTER PAGE HEADING: Summierung aus Anzahl der Bestellungen je Artikel-Nr

DOUBLE SPACE REPORT? (Y/N) N

ARE TOTALS REQUIRED? (Y/N) Y

SUBTOTALS IN REPORT? (Y/N) Y

ENTER SUBTOTALS FIELD: ARTIKEL:NR

SUMMARY REPORT ONLY? (Y/N) N

EJECT PAGE AFTER SUBTOTALS? (Y/N) N

ENTER SUBTOTAL HEADING: Bestellungen für Artikel-Nummer:

COL WIDTH, CONTENTS

001 23, NAME + ' :'

ENTER HEADING: < Firma

002 7, ANZAHL

ENTER HEADING > Anzahl

ARE TOTALS REQUIRED? (Y/N) Y

003 (RETURN)

(Nachfolgend Ausgabe des Berichts)

Summierung aus Anzahl der Bestellungen je Artikel-Nr

Firma	Anzahl
* Bestellungen für Artikel-Nummer: 103	
Hoch und Tief AG :	4
** SUBTOTAL **	4
* Bestellungen für Artikel-Nummer: 112	
Schneider GmbH :	2
Stahlbau :	3
** SUBTOTAL **	5
* Bestellungen für Artikel-Nummer: 120	
Gartenbau Wild :	34
Sanitär Reinhardts :	30
** SUBTOTAL **	64
* Bestellungen für Artikel-Nummer: 140	
Wilken und Sohn :	2
** SUBTOTAL **	2
* Bestellungen für Artikel-Nummer: 331	
Fortschritt e.V. :	12
Karl Anders :	12
** SUBTOTAL **	24

REPORT...334

Bestellungen für Artikel-Nummer: 492	
Hobby-Bau :	35
** SUBTOTAL **	
	35
* Bestellungen für Artikel-Nummer: 660	
Hansen KG :	1
** SUBTOTAL **	
	1
* Bestellungen für Artikel-Nummer: 680	
Müller & Co :	2
Tischlerei Schmitz :	10
** SUBTOTAL **	
	12
* Bestellungen für Artikel-Nummer: 830	
Heimbedarf Knobel :	3
** SUBTOTAL **	
	3
** TOTAL **	
	150

. REPORT FORM GESAMT

ENTER OPTIONS, M = LEFT MARGIN, L = LINES/PAGE, W = PAGE WIDTH M = 20

PAGE HEADING? (Y/N) Y

ENTER PAGE HEADING: Zwischensumme je Artikel (ohne Besteller-Angaben)

DOUBLE SPACE REPORT? (Y/N) N

ARE TOTALS REQUIRED? (Y/N) Y

SUBTOTALS IN REPORT? (Y/N) Y

ENTER SUBTOTALS FIELD: ARTIKEL:NR

SUMMARY REPORT ONLY? (Y/N) Y

EJECT PAGE AFTER SUBTOTALS? (Y/N) N

ENTER SUBTOTAL HEADING: Artikel-Nummer:

COL WIDTH, CONTENTS

001 20, NAME

ENTER HEADING: (RETURN)

002 8, ANZAHL

ENTER HEADING: (RETURN)

ARE TOTALS REQUIRED? (Y/N) Y

003 (RETURN)

(Nachfolgend Ausgabe des Berichts)

Zwischensumme je Artikel (ohne Besteller-Angaben)

*Artikel-Nummer: 103	
**SUBTOTAL **	4
*Artikel-Nummer: 112	
** SUBTOTAL **	5
* Artikel-Nummer: 120	
** SUBTOTAL **	64
* Artikel-Nummer: 140	
** SUBTOTAL **	2
* Artikel-Nummer: 331	
** SUBTOTAL **	24
* Artikel-Nummer: 492	
** SUBTOTAL **	35
* Artikel-Nummer: 660	
** SUBTOTAL **	1
* Artikel-Nummer: 680	
** SUBTOTAL **	12
* Artikel-Nummer: 830	
** SUBTOTAL **	3
** TOTAL **	150

RESET**RESET [<Laufwerk>]**

Mit diesem Befehl werden die von CP/M verwalteten Informationen über die in den Laufwerken vorhandenen Disketten aktualisiert. Wenn eine Diskette gewechselt wurde, erlaubt CP/M keinen schreibenden Zugriff auf die neu eingelegte Diskette. Dies ist erst dann möglich, wenn das Betriebssystem neu gestartet wird, entweder mittels Hardware (z.B. Ein-/Ausschalten des Rechners bzw. Drücken einer hierfür vorgesehenen Taste am Rechner) oder durch die Tastenkombination **ctl-C**. **RESET** versucht, alle Dateien wieder zu aktivieren, die vor dem Diskettenwechsel in Benutzung waren.

Enthält der Befehl keine Laufwerksangabe, wird das gesamte System re-initialisiert. Wurde eine Diskette gewechselt, kann dies zu Ein-/Ausgabefehlern führen. Bei Angabe eines <Laufwerks> prüft dBASE, ob Dateien dieses <Laufwerks> noch offen sind und beugt durch entsprechende Maßnahmen Ein-/Ausgabefehlern vor. Aus diesem Grunde sollte der **RESET**-Befehl nur unter gleichzeitiger Angabe eines <Laufwerks> erteilt werden.

Bei dem Laufwerk, das die dBASE-Systemdateien enthält, darf die Diskette auf keinen Fall gewechselt werden!

Die Eingabe von **RESET** zeigt keine Wirkung und ist überflüssig, wenn kein Diskettenwechsel vorgenommen wurde.

RESTORE

RESTORE FROM <Dateiname> [ADDITIVE]

Dieser Befehl liest eine Datei, in die mit SAVE MEMORY TO <Dateiname> vorher temporäre Variable abgespeichert worden sind. Alle temporären Variablen, die vor Ausführung dieses Befehles vorhanden sind, werden dabei gelöscht und die in der Datei vorhandenen Variablen mit dem dort konservierten Wert eingerichtet. Wird bei <Dateiname> nicht explizit ein Typ angegeben, so wird von dBASE M ergänzt.

Bei Verwendung der Option ADDITIVE bleiben die bereits vorhandenen temporären Variablen erhalten. Aus der angegebenen Datei werden zusätzlich so viele Variablen wie möglich aufgenommen. Variablen, die sowohl im alten Bestand als auch in der eingelesenen Datei enthalten sind, nehmen den in der Datei gespeicherten Wert an.

Beispiele:

```
. STORE 1.11111 TO EINS
1.11111
```

```
. STORE "AbCdEfGhIjKIMn..." TO ALPHA
AbCdEfGhIjKIMn...
```

```
. STORE "alte Variablenbelegung" TO ALT
alte Variablenbelegung
```

```
. SAVE TO VARALT.DAT
. STORE 2.22222 TO EINS
2.22222
```

```
. STORE "neue Variablenbelegung" TO NEU
neue Variablenbelegung
```

```
. SAVE TO VARNEU.DAT
. DISPLAY MEMORY
```

EINS	(N)	2.22222
ALPHA	(C)	AbCdEfGhIjKIMn...
ALT	(C)	alte Variablenbelegung
NEU	(C)	neue Variablenbelegung
** TOTAL **		04 VAR

RESTORE...338

. RESTORE FROM VARALT.DAT

. DISPLAY MEMORY

EINS	(N)	1.11111
------	-----	---------

ALPHA	(C)	AbCdEfGhIjKlMn...
-------	-----	-------------------

ALT	(C)	alte Variablenbelegung
-----	-----	------------------------

** TOTAL ** 03 VARIABLES USED 00048 BYTES USED

. RESTORE FROM VARNEU.DAT ADDITIVE

. DISPLAY MEMORY

EINS	(N)	2.22222
------	-----	---------

ALPHA	(C)	AbCdEfGhIjKlMn...
-------	-----	-------------------

ALT	(C)	alte Variablenbelegung
-----	-----	------------------------

NEU	(C)	neue Variablenbelegung
-----	-----	------------------------

** TOTAL ** 04 VARIABLES USED 00071 BYTES USED

RETURN**RETURN**

Dieser Befehl findet in Befehlsdateien Verwendung. Wird er in einer solchen Datei erreicht, werden die Befehle hinter dem DO-Befehl ausgeführt, der zum Aufruf der jetzt mit RETURN beendeten Befehlsdatei diente. Erfolgte der Aufruf nicht aus einer übergeordneten Befehlsdatei, antwortet dBASE mit dem üblichen Zeichen zur Befehlseingabe, dem Punkt (Prompt).

Wird das Ende einer Befehlsdatei ohne RETURN erreicht, wirkt dies wie ein RETURN-Befehl.

SAVE

SAVE TO <Dateiname> [ALL LIKE <Dateimaske>]

Die aktuell vorhandenen temporären Variablen werden mit Namen, Typ und Wert in der Datei <Dateiname> gespeichert und können später von dort wieder restauriert werden (siehe RESTORE-Befehl). Von dBASE wird - sofern bei <Dateiname> keine nähere Angabe erfolgt - eine Datei vom Typ .MEM eingerichtet.

Die <Dateimaske> entspricht der <Dateimaske> des Befehls DISPLAY FILES, d.h. "?" und "*" können entsprechend den CP/M-Konventionen verwendet werden.

Beispiele:

```
. DISPLAY MEMORY
** TOTAL **                00 VARIABLES USED 00000 BYTES USED
```

```
. STORE "ABcdEFgh..." TO ALPHA
ABcdEFgh...
```

```
. STORE 1.111 TO EINS
1.111
```

```
. STORE EINS * 2 TO EINS:MAL:2
2.222
```

```
. DISPLAY MEMORY
ALPHA      (C)  ABcdEFgh...
EINS       (N)  1.111
EINS:MAL:2 (N)  2.222
** TOTAL **      03 VARIABLES USED 00026 BYTES USED
```


. SAVE TO ALLE.DAT

. SAVE TO ZWEI.DAT ALL LIKE EINS*

. RELEASE ALL

. DISPLAY MEMORY

** TOTAL ** 00 VARIABLES USED 00000 BYTES USED

. RESTORE FROM ZWEI.DAT

. DISPLAY MEMORY

EINS (N) 1.111

EINS:MAL:2 (N) 2.222

** TOTAL ** 02 VARIABLES USED 00014 BYTES USED

. RESTORE FROM ALLE.DAT

. DISPLAY MEMORY

ALPHA (C) ABcdEFgh...

EINS (N) 1.111

EINS:MAL:2 (N) 2.222

** TOTAL ** 03 VARIABLES USED 00026 BYTES USED

SELECT

```
SELECT [ PRIMARY ]  
      [ SECONDARY ]
```

dBASE gestattet, bis zu zwei Datenbanken nahezu gleichzeitig zu aktivieren. Zugriffe, Änderungen usw. können zur selben Zeit in zwei Datenbanken erfolgen. Typische Beispiele sind das Vergleichen von Datenfeldern in zwei Datenbanken oder Übertragen von Daten.

Nach dem Start richtet dBASE den sogenannten Primärbereich ein, in dem eine Datenbank mit USE aktiviert werden kann. Dieser PRIMARY-Bereich bleibt aktiv, bis der Befehl SELECT SECONDARY gegeben wird. In dem damit eingerichteten Sekundärbereich kann eine weitere Datenbank aktiviert werden. Das Umschalten zwischen dem Primär- und Sekundärbereich geschieht mit SELECT PRIMARY bzw. SELECT SECONDARY. Die dort aktivierten Datenbanken bleiben im Zugriff einschließlich der Positionierung auf den jeweils aktuellen Satz. Die mehrmalige, aufeinanderfolgende Eingabe von SELECT PRIMARY bzw. SELECT SECONDARY ist unschädlich. Durch Hin- und Herschalten zwischen den Bereichen ergibt sich die fast gleichzeitige Zugriffsmöglichkeit auf zwei Datenbanken.

Wenn in den beiden Bereichen jeweils eine Datenbank aktiviert wurde (z.B. nach SELECT SECONDARY mit USE auch im Sekundärbereich), kann auf Daten in jedem Bereich zugegriffen werden. Jeder Ausdruck kann z. B. Datenfelder in beiden Bereichen ansprechen. Falls Feldbezeichnungen in beiden Datenbanken identisch sind, so müssen sie durch ein vorangestelltes "P." (für Primärbereich) bzw. "S." (für Sekundärbereich) eindeutig gemacht werden. Dies führt zu keiner Änderung der Feldbezeichnungen in den Datenbanken, sondern ist nur ein auf den Aktivierungszeitraum befristetes Unterscheidungsmerkmal für dBASE.

Befehle, die die Positionierung in einer Datenbank ändern (z. B. GOTO, SKIP, REPORT, SORT, COPY, LIST und DISPLAY und im Geltungsbereich mehr als einen Satz umfassen) betreffen nur den gerade eingestellten Bereich (entweder Primär- oder Sekundärbereich). Mit dem SET-Befehl LINKAGE ON kann jedoch bewirkt werden, daß alle Befehle, die eine Angabe für den Geltungsbereich vorsehen, in beiden Datenbanken gleichartig und synchron positionieren. Ein REPLACE-Befehl betrifft nur den jeweils eingestellten Bereich und auch DISPLAY STRUCTURE bezieht sich nur auf die Datenbankdatei dieses Bereichs.

Die Einrichtung von nur zwei Bereichen bedeutet natürlich nicht, daß nur auf zwei Datenbanken in einer Sitzung zugegriffen werden kann. Durch wiederholte USE-Befehle können nacheinander in den Bereichen beliebig viele Datenbanken aktiviert werden. Bei Übertragung von Daten oder für das Speichern von Zwischenergebnissen müssen ggf. mit den entsprechenden Befehlen Dateien bzw. temporäre Variable eingerichtet werden, wenn gleichzeitig mehr als zwei Datenbanken angesprochen werden sollen.

Beispiele:

. SELECT PRIMARY

. USE KATALOG

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: KATALOG.DBF

NUMBER OF RECORDS: 00009

DATE OF LAST UPDATE: 10/12/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	PREIS	N	010	002
004	BESTAND	N	005	
** TOTAL **			00041	

. LIST

00001	Zwinge	680	12.00	22
00002	Hammer	492	13.50	52
00003	Bohrmaschine	112	360.00	6
00004	Hobel	830	20.50	16
00005	Schubkarre	140	150.50	5
00006	Leiter	103	80.00	4
00007	Eimer	331	14.90	63
00008	Schaufel	120	13.00	75
00009	Waage	660	1260.80	2

. SELECT SECONDARY

. USE BESTELLG

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: BESTELLG.DBF

NUMBER OF RECORDS: 00013

DATE OF LAST UPDATE: 10/12/82

SECONDARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	020	
002	ARTIKEL:NR	N	005	
003	ANZAHL	N	005	
** TOTAL **			00031	

SELECT...344

. LIST			
00001	Müller & Co	680	2
00002	Schneider GmbH	112	2
00003	Hansen KG	660	1
00004	Stahlbau	112	3
00005	Fortschritt e.V.	331	12
00006	Hoch und Tief AG	103	4
00007	Heimbedarf Knobel	830	3
00008	Gartenbau Wild	120	34
00009	Hobby-Bau	492	35
00010	Tischlerei Schmitz	680	10
00011	Wilken und Sohn	140	2
00012	Karl Anders	331	12
00013	Sanitär Reinhardts	120	30

(nach diesem LIST ist auf den letzten Satz mit Nr. 13 positioniert)

. SELECT PRIMARY

. 2

. DISPLAY

00002 Hammer 492 13.50 52

. ? S.NAME

Sanitär Reinhardts

. 3

. DISPLAY

00003 Bohrmaschine 112 360.00 6

. ? S.NAME

Sanitär Reinhardts

(Die Positionierung im Sekundärbereich wird also nicht verändert)

. SELECT SECONDARY

. ? P.ARTIKEL:NR

112

. SUM S.ANZAH * PREIS TO AUFT:WERT FOR S.ARTIKEL:NR = P.ARTIKEL:NR

1800.00

(sind Felder in beiden Datenbanken identisch bezeichnet, müssen sie - wie oben bei ANZAHL und ARTIKEL:NR gezeigt - durch vorangestelltes "P." bzw. "S." unterschieden werden. Bei PREIS wurde zur Verdeutlichung "P." verwendet. Da dieser Name nur einmal vorkommt, ist diese zusätzlich Angabe nicht notwendig.)

SET

- a. SET <Parameter1> [ON]
[OFF]
- b. SET <Parameter2> TO <Option>

Mit dem SET-Befehl läßt sich die Konfiguration von dBASE während einer Sitzung verändern und damit u.a. das Verhalten des Systems individuell beeinflussen.

Form a gestattet, die in <Parameter1> genannten Wirkungen ein- oder auszuschalten (ON bzw. OFF). Form b erlaubt die Änderung von Standardtexten (wie z. B. Datum) oder die Auswahl einer gewünschten aus mehreren Möglichkeiten (im Unterschied zu Form a aus mehr als zwei).

Zu beachten ist, daß durch SET-Befehle verursachte Änderungen nur für die Dauer einer Sitzung mit dBASE wirksam sind. Nach dem Start von dBASE gelten zunächst Voreinstellungen, wie sie u.a. mit dem Installationsprogramm zu dBASE festgelegt wurden. Diese Voreinstellungen können dann bei Bedarf mittels SET beliebig oft geändert werden (es gilt jeweils die letzte Änderung). Mit dem Ende der Sitzung (siehe Befehl QUIT) sind diese Änderungen jedoch hinfällig und dBASE geht beim nächsten Start - soweit dies bei den einzelnen Möglichkeiten zutrifft - wieder von den Voreinstellungen aus.

Die nachfolgende Tabelle beschreibt die Möglichkeiten zu Form a. Man beachte, daß für <Parameter1> nur die dort aufgeführten Angaben zulässig sind. Die Voreinstellung - ob die Wirkung ein- oder ausgeschaltet ist - wird durch die Unterstreichung von ON oder OFF gekennzeichnet. Diese Unterstreichung ist nicht Teil der Eingabe, wenn mit einer Änderung wieder der voreingestellte Wert erreicht werden soll.

<Parameter1>	Einstellung	Wirkung
1. ECHO	ON	Bei Ausführung einer Befehlsdatei (siehe DO-Befehl) werden die dort durchlaufenen Befehle auf dem Bildschirm ausgegeben (für Testzwecke).
	<u>OFF</u>	Keine Wiederholung der Befehle auf dem Bildschirm.
2. STEP	ON	Bei Ausführung einer Befehlsdatei hält dBASE nach Beendigung eines jeden Befehls an und wartet auf die Eingabe des Nutzers. Dieser kann zusätzliche Befehle eingeben, die Ausführung der Befehlsdatei abbrechen (Eingabe von ESCAPE) oder die Ausführung des nächsten Befehls in der Datei veranlassen (zum Testen).
	<u>OFF</u>	Keine Unterbrechung bei Ausführung einer Befehlsdatei (soweit nicht dort durch Befehle erzwungen).

- | | | |
|--------------|------------|--|
| 3. TALK | <u>ON</u> | Das Ergebnis von Befehlen wird angezeigt (z.B. Nummer eines gefundenen Satzes). |
| | <u>OFF</u> | Mit Ausnahme programmierter Ausgaben keine Anzeige des Ergebnisses von Befehlen. |
| 4. PRINT | <u>ON</u> | Protokollierung von Anzeigen auf dem Drucker. |
| | <u>OFF</u> | Keine Protokollierung auf Drucker. |
| 5. CONSOLE | <u>ON</u> | Anzeigen erfolgen auf dem Bildschirm. |
| | <u>OFF</u> | Unterdrückung von Anzeigen auf dem Bildschirm. |
| 6. ALTERNATE | <u>ON</u> | Anzeigen werden zusätzlich in eine Datei protokolliert. |
| | <u>OFF</u> | Beenden/Unterbrechen der Protokollierung |
| 7. SCREEN | <u>ON</u> | Umschaltung auf Seitenmodus für die Befehle APPEND, INSERT, EDIT und CREATE. |
| | <u>OFF</u> | Umschalten von Seitenmodus auf zeilenweise Eingabe. |
| 8. LINKAGE | <u>ON</u> | Alle diesem SET nachfolgenden Befehle, die (wie z.B. LIST, REPORT und SUM) mit einem <Geltungsbereich> arbeiten, führen die Auswahl von Sätzen gleichartig sowohl in der Primär- als auch in der Sekundärdatenbank durch (s. SELECT PRIMARY bzw. SECONDARY). |
| | <u>OFF</u> | Führt zur voneinander unabhängigen Positionierung in der Primär- und Sekundärdatenbank. |
| 9. COLON | <u>ON</u> | Bei mit GET einzulesenden Datenfeldern werden die Feldgrenzen bei der Anzeige (@-Befehl) mit Doppelpunkten markiert. |
| | <u>OFF</u> | Keine Anzeige der Doppelpunkte zur Feldbegrenzung. |
| 10. BELL | <u>ON</u> | Soweit Datensichtgerät dafür eingerichtet, ertönt Signal bei Eingabe unzulässiger Daten bzw. Überschreiten der Feldgrenze. |
| | <u>OFF</u> | Unterdrückung der akustischen Warnung. |
| 11. ESCAPE | <u>ON</u> | Mit Eingabe von ESCAPE, d.h. Drücken der entsprechend. Taste, wird die Ausführung einer Befehlsdatei abgebrochen. |
| | <u>OFF</u> | Eingabe von ESCAPE unwirksam. |

- | | | |
|---------------|------------|--|
| 12. EXACT | ON | Alphanumerische Werte (Zeichenreihen, Texte) die in Ausdrücken oder mit dem FIND-Befehl verglichen werden, müssen für Gleichheit bis auf Leerstellen am Ende genau übereinstimmen. |
| | <u>OFF</u> | Für Vergleich ist Länge des zweiten Wertes im Ausdruck maßgebend (bzw. Schlüsselwert im FIND-Befehl). In diesem Fall wird z.B. "ABCDEF" = "ABC" als richtig angesehen, d.h. der zweite Wert ("ABC") ist gleich dem ersten. |
| 13. INTENSITY | <u>ON</u> | Im Seitenmodus werden auf dem Bildschirm Markierungen z.B. mit unterschiedlicher Helligkeit oder invertierten Darstellungen vorgenommen (abhängig von Eigenschaften des Sichtgerätes). |
| | OFF | Unterdrückung unterschiedlicher Helligkeitsstufen. |
| 14. DEBUG | ON | Mit SET ECHO bzw. SET STEP angeforderte Anzeigen werden auf dem Drucker ausgegeben, so daß die Ausgaben im Seitenmodus (z. B. Erfassungsmasken) nicht unterbrochen werden (zum Testen). |
| | <u>OFF</u> | Keine getrennte Ausgabe auf Drucker. |
| 15. CARRY | ON | Beim Füllen von Datensätzen im Seitenmodus (APPEND), werden Daten des vorhergehenden Satzes automatisch in den nächsten Satz übertragen. |
| | <u>OFF</u> | Keine Übertragung der Datenwerte von einem Satz zum nächsten. |
| 16. CONFIRM | ON | Bei der Datenerfassung im Seitenmodus wird nach der Eingabe zu einem Feld erst nach Eingabe eines Kontrollzeichens (z. B. Drücken von RETURN) zum nächsten Feld gesprungen. |
| | <u>OFF</u> | Sprung zum nächsten Feld, wenn bei Eingabe Feldgrenze erreicht wird. |
| 17. EJECT | <u>ON</u> | Mit REPORT-Befehl erzeugter Bericht beginnt auf neuer Seite. |
| | OFF | Unterdrückung Seitenvorschub am Anfang des Berichts. |
| 18. RAW | ON | Unterdrückung von Leerzeichen. |
| | <u>OFF</u> | Einfügen von Leerzeichen, wenn DISPLAY- und LIST-Befehle ohne Angabe von bestimmten Feldern benutzt werden. |

19. DELETE ON Zur Löschung gekennzeichnete Sätze ("*") können weder durch einen FIND-Befehl angesprochen, noch von einem Befehl verarbeitet werden, der eine NEXT-Klausel zuläßt (wie z.B. LIST, LOCATE, COUNT usw.).
- OFF Zur Löschung gekennzeichnete Sätze können angesprochen und angezeigt werden (bei COPY- und APPEND-Befehlen bleiben sie jedoch unberücksichtigt!).

Die Form b erlaubt folgende Einstellungen:

1. SET HEADING TO <Text>

Hiermit wird intern in dBASE der <Text> mit einer Länge von höchstens 60 Zeichen abgespeichert und bei mit REPORT veranlaßten Berichten als Kopfzeile ausgedruckt (siehe Beispiele bei REPORT-Befehle).

2. SET FORMAT TO [SCREEN]
 [PRINT]
 [<Maskendatei>]

Die beiden erstgenannten Möglichkeiten legen fest, auf welchem Gerät mit @ erzeugte Ausgaben erfolgen sollen. Im dritten Fall wird bei einem READ-Befehl die angegebene <Maskendatei> gelesen und die dort enthaltenen @-Befehle ausgeführt (normalerweise Aufbau einer Maske für die Datenerfassung).

3. SET DEFAULT TO <Laufwerk>

Mit diesem Befehl werden Zugriffe auf vom Benutzer anzugebende Dateien wie z.B. Datenbanken, Schlüsseldateien usw. - also nicht die Systemdateien von dBASE selbst - auf das <Laufwerk> dirigiert. Wenn bei Dateinamen nicht explizit das Laufwerk mit angegeben ist, fügt dBASE intern die dem <Laufwerk> entsprechende Angabe hinzu. Nach dem Start von dBASE ist dies zu nächst das Laufwerk mit den Systemdateien, das in der Regel der unter CP/M gewählten Voreinstellung entspricht.

Die Angabe zu <Laufwerk> kann mit oder ohne den (bei CP/M sonst obligatorischen) Doppelpunkt erfolgen. Auch die Verwendung einer temporären Variablen mit vorangestelltem & (Macro) ist zugelassen. Die Voreinstellungen von CP/M und dBASE können unterschiedlich sein, d.h. der SET-Befehl ändert nicht die Voreinstellung von CP/M, sondern wirkt nur in dBASE selbst.

Beispiel:

. SET DEFAULT TO B:

. USE TESTDAT (dBASE greift für die Datei TESTDAT auf das Laufwerk B: zu, d.h. die vollständige Dateiangabe lautet B:TESTDAT.DBF)

4. SET ALTERNATE TO <Dateiname>

Dieser Befehl ist im Zusammenhang mit SET ALTERNATE ON zu sehen. Mit <Dateiname> wird eine Datei bestimmt, in die alle Ausgaben und Eingaben protokolliert werden, die auch auf dem Bildschirm erscheinen. Mit dem Befehl wird die Datei zunächst nur eingerichtet. Fehlt in der Angabe der Dateityp, wird er von dBASE mit .TXT ergänzt. Ist bereits eine Datei gleichen Namens vorhanden, so wird sie überschrieben. Die Protokollierung beginnt erst mit SET ALTERNATE ON. Die nachfolgenden Ausgaben und Eingaben (soweit sie nicht im Seitenmodus erfolgen) erfolgen zusätzlich auch in der Datei und sind damit konserviert. Die Protokollierung kann mit ON bzw. OFF ein- und ausgeschaltet werden, d. h. Protokolle lassen sich stückweise aufbauen.

Beispiel:

. SET ALTERNATE TO B:PROTOK
(Folge von Befehlen)

. SET ALTERNATE ON (weitere Folge von Befehlen, die nun in B:PROTOK.TXT aufgezeichnet werden)

. SET ALTERNATE OFF
(Unterbrechung der Protokollierung)

(Fortsetzung Protokollierung in B:PROTOK.TXT)

. SET ALTERNATE TO B:WEITER
(Protokollierung läuft weiter in B:WEITER.TXT)

5. SET DATE TO mm/tt/jj

Dieser Befehl erlaubt die Veränderung des beim Start von dBASE erfragten Datums. Anstelle von mm/tt/jj ist auch jede andere Reihenfolge von Tag, Monat und Jahr zulässig, da dBASE bei diesem Befehl keine Plausibilitätsprüfung des Datums durchführt (im Gegensatz zur Datumseingabe beim Start). dBASE verwendet das Datum zur Protokollierung des jeweils letzten Änderungsstandes von Datenbankdateien und in von REPORT erzeugten Berichten.

**6. SET INDEX TO <Schlüsseldatei> [<Schlüsseldatei> ,
... , <Schlüsseldatei>]**

Bis zu sieben <Schlüsseldateien> (bei mehreren Dateien je weils durch Komma getrennt) können mit diesem Befehl aufgerufen und für weitere Operationen verwendet werden.

Wenn bereits eine oder mehrere Schlüsseldateien aktiviert sind, so werden sie mit diesem Befehl geschlossen, d.h. sie werden aktuell nicht mehr verwendet. Zu beachten ist, daß mit dem Aufruf einer neuen Schlüsseldatei der Positionszeiger in der Datenbank nicht verändert wird, während die Position in der Schlüsseldatei noch undefiniert ist. Bevor also Befehle verwendet werden, die mittels NEXT von der aktuellen Position ausgehend auf vorhergehende bzw. nachfolgende Sätze schalten, muß zunächst ein FIND- oder GOTO-Befehl gegeben werden, um Schlüsseldatei und Datenbank zu synchronisieren.

Sind mehrere Schlüsseldateien angegeben, so wird die in der Reihenfolge erste als Hauptschlüsseldatei angesehen. FIND- Befehle müssen sich auf das zugehörige Schlüsselfeld beziehen und die Datenbank erscheint nach diesem Schlüssel sortiert.

SET INDEX TO ohne weitere Angabe (also ohne Dateinamen) gibt alle aktivierten Schlüsseldateien frei und für die weitere Bearbeitung der Datenbank wird nicht mehr auf Schlüsseldateien zugegriffen. Insbesondere wird bei Auflistungen in der Reihenfolge der Numerierung der Sätze vorgegangen.

7. SET MARGIN TO n

Bei Berichten (siehe REPORT) kann hiermit die Breite des linken Randes neu festgelegt werden. Alle Zeilen werden um n Spalten eingerückt, wobei n einen Wert von 1 bis 254 erhalten darf. Der Wert muß direkt als Zahl eingegeben werden, Ausdrücke sind nicht zulässig.

SKIP

SKIP [[+] [-] <Ausdruck>]

In dBASE deutet ein Zeiger auf den Satz in der Datenbank, auf den aktuell zugegriffen wird. Mit SKIP kann relativ zu dieser Position der Zugriff und damit der Zeiger auf vorhergehende (Angabe von "-") bzw. nachfolgende (Angabe von "+") Sätze geschaltet werden. Die neue Position bestimmt sich durch Subtraktion oder Addition des Wertes von <Ausdruck> vom aktuellen Wert. <Ausdruck> kann direkt als Zahl angegeben werden. In jedem Falle muß die Auswertung von <Ausdruck> eine Zahl ergeben.

Man beachte, daß bei Datenbanken mit angeschlossener Schlüsseldatei die Position von der Reihenfolge in der Schlüsseldatei bestimmt wird. Je nachdem, ob die Datenbank mit oder ohne Schlüsseldatei verwendet wird, sind also unterschiedliche Ergebnisse möglich.

Beispiele:

```
. DISPLAY STRUCTURE
STRUCTURE FOR FILE: BESTAND.DBF
NUMBER OF RECORDS: 00011
DATE OF LAST UPDATE: 00/00/00
PRIMARY USE DATABASE
FLD      NAME      TYPE      WIDTH      DEC
001      ARTIKEL   C         020
002      ARTIKEL:NR N         005
003      EINZ:PREIS N         007      002
004      BESTAND   N         005
005 DATUM C         008
** TOTAL **
```

00046

```
. LIST
00001      Getriebe      10023      1320.00      14      15.10.82
00002      Kolben       12221      90.00      233      15.10.82
00003      Reifen       987        77.00      210      15.10.82
00004      Achse        70023      350.00      40      15.09.82
00005      Haube        997        46.90      89      12.10.82
00006      Feder        19888      56.70      23      12.10.82
00007      Kupplung     3567       261.00      42      10.10.82
00008      Auspufftopf  12112      80.50      230      06.10.82
00009      Welle        8978       320.00      34      15.10.82
00010      Generator    10015      340.00      25      12.10.82
00011      Kanister     1234        5.90      456      01.09.82
```

SKIP...352

SKIP -2
RECORD: 00003

```

. DISPLAY
00003      Reifen      987      77.00  210  15.10.82

```

. SKIP 2 * 2
RECORD: 00007

```

. DISPLAY
00007      Kupplung      3567      261.00  42  10.10.82
(Nachfolgend wird die Datenbank mit dem Schlüsselfeld ARTIKEL verwendet)

```

USE BESTAND INDEX BESTAND

. LIST					
00004	Achse	70023	350.00	40	15.09.82
00008	Auspufftopf	12112	80.50	230	06.10.82
00006	Feder	19888	56.70	23	12.10.82
00010	Generator	10015	340.00	25	12.10.82
00001	Getriebe	10023	1320.00	14	15.10.82
00005	Haube	997	46.90	89	12.10.82
00011	Kanister	1234	5.90	456	01.09.82
00002	Kolben	12221	90.00	233	15.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00003	Reifen	987	77.00	210	15.10.82
00009	Welle	8978	320.00	34	15.10.82

5

. DISPLAY					
00005	Haube	997	46.90	89	12.10.82

SKIP -2
RECORD: 00010

. DISPLAY					
00010	Generator	10015	340.00	25	12.10.82

SKIP 2*2
RECORD: 00002

. DISPLAY					
00002	Kolben	12221	90.00	233	15.10.82

. SKIP 3
RECORD: 00009

. DISPLAY					
00009	Welle	8978	320.00	34	15.10.82

SORT

SORT ON <Feld> TO <Dateiname> [ASCENDING]
[DESCENDING]

Der Befehl bewirkt die Sortierung der aktivierten Datenbank nach dem Inhalt des unter <Feld> angegebenen Datenfeldes. Ausgabedatei ist die mit <Dateiname> benannte Datei. Die aktivierte Datenbank wird dabei nicht geändert und bleibt weiterhin im Zugriff; nur mit USE kann die neue, sortierte Datenbank aktiviert werden.

In einem SORT-Befehl kann immer nur nach einem Datenfeld sortiert werden. Durch Wiederholung des Befehls mit jeweils anderen <Feldern> ist eine stufenweise Sortierung über mehrere Datenfelder möglich. Man beginnt zunächst mit der Sortierung nach dem letzten Sortierfeld und arbeitet sich schrittweise bis zum ersten Datenfeld, dem Hauptsortierkriterium vor. dBASE ändert die Reihenfolge der Sätze nur, soweit dies für die Einhaltung der Sortierfolge zwingend notwendig ist.

Der Vergleich der Werte von Datenfeldern, der über die Reihenfolge der Sätze entscheidet, erfolgt in der Darstellung als ASCII-Zeichensatz. Mit diesem Zeichensatz ist eine aufsteigende Reihenfolge (ASCENDING) gegeben, z.B. "A" vor "B", Ziffern vor Buchstaben usw., die auch von dBASE verwendet wird, wenn nicht explizit eine absteigende Reihenfolge mit der Angabe DESCENDING verlangt wird. Fehlt die Angabe zur Sortierung nach auf- bzw. absteigenden Werten, nimmt dBASE ASCENDING an.

Mit INDEX kann das gleiche Ergebnis wie mit SORT erreicht werden. Jedoch vermeidet der INDEX-Befehl das Einrichten einer zweiten, sortierten Datenbank und bietet darüber hinaus mehr Möglichkeiten (z.B. Anlegen mehrerer, wahlweise zu aktivierender Schlüsseldateien oder die Kombination von Datenfeldern zu Schlüsseln). Ferner erfolgt der Aufbau einer Schlüsseldatei schneller und erlaubt später rasche Zugriffe auf bestimmte Sätze mit Hilfe von FIND.

Hinweis: Da SORT quasi ein Duplikat der aktivierten Datenbank anlegt, muß auf der Diskette, die die sortierte Datei aufnehmen soll, mindestens so viel Platz sein, wie die aktuelle Datenbankdatei belegt. Sätze mit Löschmarkierung ("**") werden von SORT nicht berücksichtigt.

Beispiele:

. USE BESTAND

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: BESTAND.DBF

NUMBER OF RECORDS: 00011

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

SORT...354

. LIST

00001	Getriebe	10023	1320.00	14	15.10.82
00002	Kolben	12221	90.00	233	15.10.82
00003	Reifen	987	77.00	210	15.10.82
00004	Achse	70023	350.00	40	15.09.82
00005	Haube	997	46.90	89	12.10.82
00006	Feder	19888	56.70	23	12.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00008	Auspufftopf	12112	80.50	230	06.10.82
00009	Welle	8978	320.00	34	15.10.82
00010	Generator	10015	340.00	25	12.10.82
00011	Kanister	1234	5.90	456	01.09.82

. SORT ON ARTIKEL TO ARTSORT
SORT COMPLETE

. USE ARTSORT

. LIST

00001	Achse	70023	350.00	40	15.09.82
00002	Auspufftopf	12112	80.50	230	06.10.82
00003	Feder	19888	56.70	23	12.10.82
00004	Generator	10015	340.00	25	12.10.82
00005	Getriebe	10023	1320.00	14	15.10.82
00006	Haube	997	46.90	89	12.10.82
00007	Kanister	1234	5.90	456	01.09.82
00008	Kolben	12221	90.00	233	15.10.82
00009	Kupplung	3567	261.00	42	10.10.82
00010	Reifen	987	77.00	210	15.10.82
00011	Welle	8978	320.00	34	15.10.82

. USE BESTAND

. SORT ON ARTIKEL TO ARTSORT DESCENDING
SORT COMPLETE

. USE ARTSORT

. LIST

00001	Welle	8978	320.00	34	15.10.82
00002	Reifen	987	77.00	210	15.10.82
00003	Kupplung	3567	261.00	42	10.10.82
00004	Kolben	12221	90.00	233	15.10.82
00005	Kanister	1234	5.90	456	01.09.82
00006	Haube	997	46.90	89	12.10.82
00007	Getriebe	10023	1320.00	14	15.10.82
00008	Generator	10015	340.00	25	12.10.82
00009	Feder	19888	56.70	23	12.10.82
00010	Auspufftopf	12112	80.50	230	06.10.82
00011	Achse	70023	350.00	40	15.09.82

. USE BESTAND

. SORT ON ARTIKEL:NR TO ARTSORT DESCENDING
SORT COMPLETE

. USE ARTSORT

. LIST

00001	Achse	70023	350.00	40	15.09.82
00002	Feder	19888	56.70	23	12.10.82
00003	Kolben	12221	90.00	233	15.10.82
00004	Auspufftopf	12112	80.50	230	06.10.82
00005	Getriebe	10023	1320.00	14	15.10.82
00006	Generator	10015	340.00	25	12.10.82
00007	Welle	8978	320.00	34	15.10.82
00008	Kupplung	3567	261.00	42	10.10.82
00009	Kanister	1234	5.90	456	01.09.82
00010	Haube	997	46.90	89	12.10.82
00011	Reifen	987	77.00	210	15.10.82

STORE

STORE <Ausdruck> TO <Variable>

Der Wert von <Ausdruck> wird berechnet, wenn er sich nicht bereits direkt ergibt (z.B. Eingabe einer Zahl oder eines Textes) und in einer temporären Variablen mit Namen <Variable> abgespeichert. Wenn die Variable noch nicht existiert, wird sie von dBASE selbständig eingerichtet. Diese Variable bleibt höchstens für die Dauer der Sitzung mit dBASE erhalten, kann beliebig oft geändert und unter ihrem Namen aufgerufen werden. Mit RELEASE wird die Variable gelöscht und ist dann nicht mehr vorhanden. Mit dem SAVE-Befehl können temporäre Variable in Dateien abgespeichert werden und bleiben dann über den aktuellen Lauf von dBASE hinaus erhalten.

STORE dient nur der Einrichtung und Änderung von temporären Variablen. Datenfelder in Datenbanken lassen sich mit REPLACE verändern.

Beispiele:

. RELEASE ALL

. STORE 1.11111 TO EINS
1.11111

. STORE EINS * 2 TO ZWEI
2.22222

. STORE 'AbCdEfGh...' TO ALPHA
AbCdEfGh...

. STORE ALPHA + 'IjKIMn...' TO ALPHALANG
AbCdEfGh...IjKIMn...

. DISPLAY MEMORY

EINS	(N)	1.11111
ZWEI	(N)	2.22222
ALPHA	(C)	AbCdEfGh...
ALPHALANG	(C)	AbCdEfGh...IjKIMn...
** TOTAL ** 04 VARIABLES USED 043 BYTES USED		

SUM

```
SUM <Ausdruck> [ <Ausdruck> ] [ TO <Variablenfolge> ]
                [ <Geltungsbereich> ] [ FOR <Ausdruck> ]
                                   [ WHILE <Ausdruck> ]
```

Mit SUM lassen sich numerische Datenfelder von Datensätzen bzw. damit gebildete <Ausdrücke> über mehrere Sätze hinweg aufsummieren. Dabei werden nur die Sätze berücksichtigt, die im <Geltungsbereich> liegen und den <Ausdrücken> der eventuell vorhandenen FOR- bzw. WHILE-Klausel entsprechen. Das Ergebnis kann einer temporären Variablen zugewiesen werden und steht dann für weitere Arbeiten zur Verfügung.

Bis zu 5 Datenfelder bzw. Ausdrücke lassen sich gleichzeitig von Satz zu Satz addieren und das Ergebnis getrennt in jeweils einer temporären Variablen ablegen. Die Summe über den ersten <Ausdruck> wird in der ersten bei TO angegebenen <Variablen> abgelegt. Analog wird mit den übrigen <Ausdrücken> verfahren, die nach ihrer Position einer <Variablen> bei TO zugeordnet werden.

Wird TO nicht genutzt, sind also keine Variablen benannt, so wird das Ergebnis nur angezeigt. Im <Geltungsbereich> liegen alle nicht für eine Löschung vorgesehenen Sätze (siehe Löschmarkierung mittels DELETE), wenn keine abweichende Angabe erfolgt.

Beispiele:

```
. USE BESTAND
```

```
. DISPLAY STRUCTURE
```

```
STRUCTURE FOR FILE: BESTAND.DBF
```

```
NUMBER OF RECORDS: 00011
```

```
DATE OF LAST UPDATE: 00/00/00
```

```
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC	
001	ARTIKEL	C		020	
002	ARTIKEL:NR	N		005	
003	EINZ:PREIS	N		007	002
004	BESTAND	N		005	
005	DATUM	C		008	
				00046	
** TOTAL **					

SUM...358

. LIST

00001	Getriebe	10023	1320.00	14	15.10.82
00002	Kolben	12221	90.00	233	15.10.82
00003	Reifen	987	77.00	210	15.10.82
00004	Achse	70023	350.00	40	15.09.82
00005	Haube	997	46.90	89	12.10.82
00006	Feder	19888	56.70	23	12.10.82
00007	Kupplung	3567	261.00	42	10.10.82
00008	Auspufftopf	12112	80.50	230	06.10.82
00009	Welle	8978	320.00	34	15.10.82
00010	Generator	10015	340.00	25	12.10.82
00011	Kanister	1234	5.90	456	01.09.82

. SUM EINZ:PREIS

2948.00

. SUM EINZ:PREIS*BESTAND, BESTAND TO KAPITAL, MENGE FOR ARTIKEL:NR . 10000

81769.10 565

. DISPLAY MEMORY

KAPITAL (N) 81769.10

MENGE (N) 565

** TOTAL ** 02 VARIABLES USED 00011 BYTES USED

. SUM (EINZ:PREIS * BESTAND), BESTAND, EINZ:PREIS, EINZ:PREIS / BESTAND

126645.60 1396 2948.00 136.36

TEXT

TEXT
<Textzeilen>
ENDTEXT

Alle <Textzeilen> zwischen den Steuerwörtern TEXT und ENDTEXT werden - je nach Einstellung der SET-Parameter - unmittelbar auf dem Bildschirm oder dem Drucker ausgegeben. Dabei werden Makros nicht ersetzt, das Zeilenfortsetzungssymbol ";" wird jedoch erkannt und entsprechend verarbeitet.

Der TEXT-Befehl findet nur in Befehlsdateien Verwendung. Er bietet eine komfortable Möglichkeit, Textblöcke in Befehlsdateien einzubetten, da sich durch ihn ganze Gruppen von @ SAY bzw. ?-Befehlen ersetzen lassen.

Beispiel:

TEXT
Alle Zeichen, die zwischen den Steuerwörtern TEXT und ENDTEXT stehen, werden ohne weitere Verarbeitung an den Bildschirm bzw. Drucker gesandt. Einzige Ausnahme ist das Zeilenfortsetzungssymbol ";".
ENDTEXT

TOTAL...360

TOTAL

TOTAL ON <Schlüssel> TO <Datenbank> [FIELDS <Felderfolge>]
[FOR <Ausdruck>]

Der TOTAL-Befehl entspricht in seiner Wirkung weitgehend der Möglichkeit zur Zwischensummenbildung, wie sie im REPORT-Befehl gegeben ist. Der Unterschied liegt jedoch darin, daß die Zwischensummen nicht wie bei REPORT ausgedruckt, sondern in einer anderen Datenbank abgelegt werden. Der TOTAL-Befehl ermöglicht somit die Verdichtung von Daten. Voraussetzung für seine Anwendbarkeit ist, daß die aktivierte Datenbank nach dem verwendeten <Schlüssel> sortiert oder indiziert ist.

Existierte die zur Aufnahme der Zwischensummen angegebene <Datenbank> bereits vor dem TOTAL-Befehl, bleibt ihre Struktur erhalten. Die Summenbildung kann allerdings nur für solche Felder erfolgen, die auch in dieser <Datenbank> enthalten sind.

Muß die angegebene <Datenbank> neu angelegt werden, so enthält sie nur die Felder, die der Benutzer in der FIELD-Klausel aufgeführt hat. Fehlt diese Klausel, so wird die Struktur der aktivierten Datenbank übernommen.

Enthält der TOTAL-Befehl die FIELD-Klausel, so erfolgt die Summenbildung nur für die in der <Felderfolge> aufgeführten numerischen Felder, während im anderen Falle die Inhalte aller in beiden Datenbanken enthaltenen numerischen Felder addiert werden.

Der TOTAL-Befehl kann auch dazu verwendet werden, doppelte Sätze aus einer Datenbank zu entfernen, da die <Felderfolge> auch nicht-numerische Felder enthalten darf, über die selbstverständlich nicht summiert wird. Da zu jedem <Schlüssel>-Wert in der neuen <Datenbank> jedoch nur ein Satz angelegt wird, lassen sich auf diese Weise doppelte Sätze der aktivierten Datenbank eliminieren.

Beispiele:

. USE SUMBEST

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: SUMBEST.DBF

NUMBER OF RECORDS: 00009

DATE OF LAST UPDATE: 10/15/82

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL:NR	C	005	
002	ANZAHL	N	005	
** TOTAL **			00011	

. USE BESTELLG INDEX ARTIKEL

. DISPLAY STRUCTURE
 STRUCTURE FOR FILE: BESTELLG.DBF
 NUMBER OF RECORDS: 00013
 DATE OF LAST UPDATE: 10/15/82
 PRIMARY USE DATABASE

FLD	NAME	TYP	WIDTH	DEC
001	NAME	C	020	
002	ARTIKEL:NR	C	005	
003	ANZAHL	N	005	
** TOTAL **			00031	

. LIST

00006	Hoch und Tief AG	103	4
00002	Schneider GmbH	112	2
00004	Stahlbau	112	3
00008	Gartenbau Wild	120	34
00013	Sanitär Reinhardts	120	30
00011	Wilken und Sohn	140	2
00005	Fortschritt e.V.	331	12
00012	Karl Anders	331	12
00009	Hobby-Bau	492	35
00003	Hansen KG	660	1
00001	Müller & Co	680	2
00010	Tischlerei Schmitz	680	10
00007	Heimbedarf Knobel	830	3

. TOTAL ON ARTIKEL:NR TO SUMBEST

00009 RECORDS COPIED

. USE SUMBEST

. LIST

00001	103	4
00002	112	5
00003	120	64
00004	140	2
00005	331	24
00006	492	35
00007	660	1
00008	680	12
00009	830	3

UPDATE

```
UPDATE FROM <Datenbank> ON <Schlüssel> [ ADD <Felderfolge> ]
      [ RANDOM ] [ REPLACE [ <Felderfolge> ] [ <Zielfeld> WITH <Quellfeld> ] ]
```

Mit dem UPDATE-Befehl läßt sich der Inhalt der durch USE aktivierten Datenbank durch den Inhalt einer anderen Datenbank ergänzen bzw. ersetzen. Durch das <Zielfeld> oder die <Felderfolgen> gibt der Benutzer an, welche Felder durch den Inhalt der zweiten <Datenbank> ersetzt werden sollen (REPLACE-Klausel) bzw. zu welchen Feldinhalten der Inhalt entsprechender Felder der zweiten <Datenbank> hinzuaddiert werden soll (ADD-Klausel). Eine Ersetzung bzw. Ergänzung der Feldinhalte eines Satzes erfolgt, wenn die Inhalte des als <Schlüssel> definierten Feldes in beiden Datenbanken übereinstimmen.

Wird die Option RANDOM verwendet, muß der angegebene <Schlüssel> einem Feld der Quell-<Datenbank> entsprechen, dessen Inhalt wiederum einen Schlüssel für die aktivierte Datenbank darstellen muß. Für die aktivierte Datenbank bedeutet dies, daß bei Verwendung von RANDOM die entsprechende Schlüsseldatei ebenfalls aktiviert ist. Die Reihenfolge der Sätze in der Quell-<Datenbank> ist unerheblich, da diese Datei Satz für Satz gelesen wird und zu jedem Satz mit einem internen FIND-Befehl der entsprechende Satz der aktivierten Datenbank ermittelt wird.

Fehlt das Steuerwort RANDOM, muß die aktivierte Datenbank entweder nach dem angegebenen <Schlüssel> aufsteigend sortiert oder indiziert sein. Auch die zweite Datenbank muß nach dem verwendeten <Schlüssel> (aufsteigend) sortiert sein, da dBASE beim UPDATE-Befehl ohne RANDOM folgendermaßen vorgeht: Beide Datenbanken werden auf ihren Anfang positioniert und aus jeder wird ein Satz gelesen. Stimmen die Inhalte der Schlüsselfelder überein, erfolgen die spezifizierten Ergänzungen bzw. Ersetzungen. Sind die Schlüssel nicht gleich, bleibt der aktuelle Datensatz unverändert, und aus der Datenbank, deren Schlüssel sich bei dem Vergleich als "kleiner" erwiesen hatte, wird der nächste Satz gelesen. Auf diese Weise fährt dBASE fort, bis eine der beiden Datenbanken erschöpft ist. Da beide Datenbanken nach aufsteigenden Schlüsseln sortiert sein müssen, ist durch dieses Verfahren sichergestellt, daß alle Änderungen erfaßt und durchgeführt werden.

Beispiele:

```
. USE PREISBST
```

```
. DISPLAY STRUCTURE
```

```
STRUCTURE FOR FILE: PREISBST.DBF
```

```
NUMBER OF RECORDS: 00011
```

```
DATE OF LAST UPDATE: 10/28/82
```

```
PRIMARY USE DATABASE
```

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	EINZ:PREIS	N	007	002
003	BESTAND	N	005	
004	DATUM	C	008	
** TOTAL **			00041	

. LIST				
00001	Achse	385.00	0	15.10.82
00002	Auspufftopf	89.00	0	12.10.82
00003	Feder	63.00	0	12.10.82
00004	Generator	374.00	0	10.10.82
00005	Getriebe	1320.00	2	15.10.82
00006	Haube	52.00	2	12.10.82
00007	Kanister	7.00	0	13.10.82
00008	Kolben	90.00	3	15.10.82
00009	Kupplung	288.00	120	12.10.82
00010	Reifen	77.00	0	15.10.82
00011	Welle	320.00	0	15.10.82

. USE BESTAND INDEX BESTNDX
 . DISPLAY STRUCTURE
 STRUCTURE FOR FILE: BESTAND.DBF
 NUMBER OF RECORDS: 00011
 DATE OF LAST UPDATE: 10/28/82
 PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	ARTIKEL	C	020	
002	ARTIKEL:NR	N	005	
003	EINZ:PREIS	N	007	002
004	BESTAND	N	005	
005	DATUM	C	008	
** TOTAL **			00046	

. LIST					
00004	Achse	70023	385.00	40	15.10.82
00008	Auspufftopf	12112	89.00	230	12.10.82
00006	Feder	19888	63.00	23	12.10.82
00010	Generator	10015	374.00	25	10.10.82
00001	Getriebe	10023	1200.00	12	12.07.82
00005	Haube	997	46.90	89	12.10.82
00011	Kanister	1234	7.00	456	13.10.82
00002	Kolben	12221	81.00	230	04.09.82
00007	Kupplung	3567	261.00	42	10.10.82
00003	Reifen	987	77.00	210	15.10.82
00009	Welle	8978	320.00	34	15.10.82

. UPDATE ON ARTIKEL FROM PREISBST ADD BESTAND REPLACE EINZ:PREIS, DATUM

. LIST					
00004	Achse	70023	385.00	40	15.10.82
00008	Auspufftopf	12112	89.00	230	12.10.82
00006	Feder	19888	63.00	23	12.10.82
00010	Generator	10015	374.00	25	10.10.82
00001	Getriebe	10023	1320.00	14	15.10.82
00005	Haube	997	52.00	91	12.10.82
00011	Kanister	1234	7.00	456	13.10.82
00002	Kolben	12221	90.00	233	15.10.82
00007	Kupplung	3567	288.00	162	12.10.82
00003	Reifen	987	77.00	210	15.10.82
00009	Welle	8978	320.00	34	15.10.82

(Man beachte die Änderungen bei den Feldern EINZ:PREIS, BESTAND (erhöht um Angabe in der Datei PREISBST) sowie DATUM. Maßgebend für die Identifizierung zusammengehöriger Sätze ist die Belegung des Feldes ARTIKEL)

USE

USE [<Dateiname>]

USE <Dateiname> INDEX <Schlüsseldatei>[, <Schlüsseldatei> ...
<Schlüsseldatei>]

Mit dem USE-Befehl wird eine (bereits existierende) Datenbankdatei (Dateityp .DBF) aktiviert. War vor der Ausführung dieses Befehls schon eine andere Datenbank aktiviert, so wird diese geschlossen und befindet sich nicht mehr im aktuellen Zugriff von dBASE. Insbesondere werden erst zu diesem Zeitpunkt ggf. statistische Daten wie z.B. DATE OF LAST UPDATE (siehe DISPLAY STRUCTURE) geändert. Enthält der USE-Befehl keinen <Dateinamen>, so wird lediglich die zur Zeit aktivierte Datenbankdatei geschlossen.

Die zweite Form des USE-Befehls dient dazu, eine zuvor durch den INDEX-Befehl indizierte Datenbank zusammen mit der angegebenen Schlüsseldatei zu aktivieren. Dadurch ist der Zugriff auf Sätze in der Reihenfolge des Schlüssels ("indexsequentiell") sowie die Benutzung von Befehlen, die Schlüssel verwenden, wie z.B. FIND, möglich.

Bis zu sieben Schlüsseldateien einer Datenbank können in einem USE-Befehl spezifiziert werden. Der als erstes angegebene Schlüssel gilt als Hauptschlüssel. Er wird bei allen FIND-Befehlen zugrundegelegt, und auch der SKIP-Befehl verwendet ihn als Ordnungskriterium der aktivierten Datenbank. Alle übrigen angeführten Schlüsseldateien werden automatisch aktualisiert, sobald ihre Schlüsselfelder durch Befehle wie APPEND, EDIT, REPLACE, READ oder BROWSE geändert werden.

Beispiele:

- . USE BESTELLG
- . USE BESTELLG INDEX ARTIKEL, NAME
- . USE MITGLIED INDEX MITGLIED

(Im letzten Beispiel wird von dBASE bei der Schlüsseldatei der Dateityp .NDX angenommen; es handelt sich also bei MITGLIED um zwei verschiedene Dateien)

WAIT

WAIT [TO <Variable>]

Dieser Befehl veranlaßt dBASE, so lange in einen Wartezustand überzugehen, bis an der Tastatur ein Tastendruck erfolgt. Auf dem Bildschirm erscheint dabei die Nachricht WAITING. Enthält der Befehl eine TO-Klausel, so wird das durch den Tastendruck erzeugte Zeichen in der angegebenen temporären < Variablen > gespeichert.

Die TO-Option erweist sich als besonders nützlich, wenn ein einziges Zeichen zur Steuerung des weiteren Ablaufs einer Befehlsdatei genügt, wie dies beispielsweise bei sogenannten Menüs häufig der Fall ist. Darüber hinaus ist es bei WAIT im Gegensatz zu ACCEPT und INPUT nicht erforderlich, die Eingabe mit der RETURN-Taste abzuschließen.

Gibt der Benutzer bei einem WAIT-Befehl mit TO-Klausel ein nicht sichtbares Steuerzeichen (z.B. RETURN, LINE FEED usw.) ein, so wird in der < Variablen > ein Leerzeichen abgelegt.

Beispiele:

```
. RELEASE ALL
```

```
. WAIT TO WEITER  
WAITING 1
```

```
. DISP MEMO  
WEITER      (C) 1  
** TOTAL **    01 VARIABLES USED 00001 BYTES USED
```

ANHANG A dBASE BEFEHLE

? [<Ausdrucksfolge>]	235
?? [<Ausdrucksfolge>]	235
@ <Koordinaten> SAY <Ausdruck> [USING <Maske>]]	237
[GET <Variable> [PICTURE <Maske>]]	
ACCEPT ["<Text>"] TO <Variable>	243
APPEND [FROM <Dateiname> [FOR <Ausdruck>] [SDF]	244
[DELIMITED WITH <Begrenzung>]	
APPEND BLANK	244
BROWSE	252
CANCEL	253
CHANGE [<Geltungsbereich>] FIELD <Felderfolge>	254
[FOR <Ausdruck>]	
CLEAR [GETS]	256
CONTINUE	257
COPY TO <Dateiname> [<Geltungsbereich>] [FIELD <Felderfolge>]	258
[FOR <Ausdruck>] [SDF] [STRUCTURE [EXTENDED]]	
[DELIMITED [WITH <Begrenzung>]]	
COUNT [<Geltungsbereich>] [FOR <Ausdruck>] [TO <Variable>]	264
CREATE [<Dateiname>] [FROM <Quelldatei>]	266
DELETE [<Geltungsbereich>] [FOR <Ausdruck>]	270
DELETE FILE <Dateiname>	270
DISPLAY [<Geltungsbereich>] [FOR <Ausdruck>]	272
[<Ausdrucksfolge>] [OFF]	
DISPLAY FILES [ON <Laufwerk>] [LIKE <Dateimaske>]	272
DISPLAY MEMORY	272
DISPLAY STATUS	272
DISPLAY STRUCTURE	272
DO <Befehlsdatei>	275
DO CASE <Ausdruck>	276
DO WHILE <Ausdruck>	275
EDIT [<n>]	278
EJECT	281
ELSE	291
ENDCASE	276
ENDDO	275
ENDIF	291
ENDTEXT	359
ERASE	282

FIND <Zeichenreihe> oder '<Zeichenreihe>' oder " <Zeichenreihe> "	283
GO oder GOTO	288
GOTO BOTTOM	288
GOTO RECORD <n>	288
GOTO TOP	288
GOTO <Variable>	288
HELP [<Befehlswort>]	290
IF <Ausdruck>	291
INDEX ON <Ausdruck> TO <Schlüsseldatei>	292
INSERT [BEFORE] [BLANK]	298
JOIN TO <Dateiname> FOR <Ausdruck> [FIELDS <Felderfolge>]	302
LIST [<Geltungsbereich>] [FOR <Ausdruck>]	305
[<Ausdrucksfolge>] [OFF]	
LIST FILES [ON <Laufwerk>] [LIKE <Dateimaske>]	305
LIST MEMORY	305
LIST STATUS	305
LIST STRUCTURE	306
LOCATE [<Geltungsbereich>] [FOR <Ausdruck>]	306
LOOP	308
MODIFY COMMAND [<Befehlsdatei>]	309
MODIFY STRUCTURE	309
NOTE <Text>	311
* <Text>	311
PACK	312
QUIT [TO <Folge von .COM-Datei-Namen>]	314
READ [NOUPDATE]	315
RECALL [<Geltungsbereich>] [FOR <Ausdruck>]	317
REINDEX	319
RELEASE [<Variablenfolge>] [ALL] [ALL LIKE <Dateimaske>]	320
[ALL EXCEPT <Dateimaske>]	
REMARK <Text>	321
RENAME <bisheriger Dateiname> TO <neuer Dateiname>	322
REPLACE [<Geltungsbereich>] <Feld> WITH <Ausdruck> [, <Feld>]	323
WITH <Ausdruck>] usw. [FOR <Ausdruck>]	
REPORT [FORM <Formatdatei>] [<Geltungsbereich>]	327
[FOR <Ausdruck>] [TO PRINT] [PLAIN]	
RESET [<Laufwerk>]	336
RESTORE FROM <Dateiname> [ADDITIVE]	337
RETURN	339

SAVE TO <Dateiname> [ALL LIKE <Dateimaske>]	340
SELECT [PRIMARY] oder [SECONDARY]	342
SET <Parameter> [ON] oder [OFF]	345
SET ALTERNATE TO <Dateiname>	349
SET DATE TO mm/tt/jj	349
SET DEFAULT TO <Laufwerk>	348
SET FORMAT TO [SCREEN] , [PRINT] , [<Maskendatei>]	348
SET HEADING TO <Text>	348
SET INDEX TO <Schlüsseldatei> [<Schlüsseldatei> , ... , <Schlüsseldatei>]	350
SET MARGIN TO <n>	350
SKIP [[+ / -] <Ausdruck>]	351
SORT ON <Feld> TO <Dateiname> [ASCENDING] [DESCENDING]	353
STORE <Ausdruck> TO <Variable>	356
SUM <Ausdruck> [, <Ausdruck>] [TO <Variablenfolge>]	357
[Geltungsbereich>] [FOR <Ausdruck>] [WHILE <Ausdruck>]	
TEXT	359
TOTAL ON <Schlüssel> TO <Datenbank> [FIELDS <Felderfolge>]	360
[FOR <Ausdruck>]	
UPDATE FROM <Datenbank> ON <Schlüssel> [ADD <Felderfolge>]	362
[RANDOM] [REPLACE <Felderfolge>]	
USE [<Dateiname>]	365
USE <Dateiname> INDEX <Schlüsseldatei> [, <Schlüsseldatei> , ... , <Schlüsseldatei>]	365
WAIT [TO <Variable>]	366

ANHANG B FEHLERMELDUNGEN**BAD DECIMAL WIDTH FIELD**

unzulässige Angabe der Nachkommastellen

BAD FILE NAME

Syntaxfehler im Dateinamen (Fehlerhafter Dateiname, eventuell unzulässige Zeichen)

BAD NAME FIELD

kein Feld dieses Namens vorhanden

BAD TYPE FIELD

Unzulässige Bezeichnung des Feldtyps bei CREATE (muss C, N oder L sein)

BAD WIDTH FIELD

Unzulässige Angabe der Feldlänge bei CREATE

CANNOT INSERT - THERE ARE NO RECORDS IN DATABASE FILE

In der Datei sind noch keine Einträge, APPEND benutzen

CANNOT OPEN FILE

Interner Fehler, wenden Sie sich an Ihren Händler.

COMMAND FILE CANNOT BE FOUND

Überprüfen Sie die Schreibweise.

DATA ITEM NOT FOUND

Der gesuchte Begriff ist nicht vorhanden.

DATABASE IN USE IS NOT INDEXED

FIND ist nur in indizierten Datenbanken zulässig.

DIRECTORY IS FULL

Das CP/M Diskettenverzeichnis kann keine weiteren Dateien aufnehmen.

DISK IS FULL

Die Diskette ist voll.

END OF FILE FOUND UNEXPECTEDLY

Die Datenbank in Benutzung hat nicht das korrekte Format.

Wenn alle Datensätze korrekt da sind, dann PACK und neu indizieren. Vorher auf jeden Fall mit PIP eine Sicherungs-Kopie machen.

"FIELD" PHRASE NOT FOUND

Angabe des Feldnamens fehlt.

FILE ALREADY EXISTS

Die Datei ist bereits angelegt.

FILE DOES NOT EXIST

Die Datei ist nicht vorhanden.

FILE IS CURRENTLY OPEN

Benutzen Sie **USE** oder **CLEAR** um die Datei zu schliessen.

FORMAT FILE CANNOT BE OPENED

Format-Datei kann nicht eröffnet werden (ev. falsche Datei, keine Format-Datei)

FORMAT FILE HAS NOT BEEN SET

Eine Formatdatei ist nicht vorhanden.

ILLEGAL DATA TYPE

Unzulässiger Datentyp.

ILLEGAL GOTO VALUE

GOTO nach unzulässiger Datensatz-Nummer

ILLEGAL VARIABLE NAME

Nur alphanumerische Zeichen und Kommas sind in Variablen und Feldnamen zugelassen.

INDEX DOES NOT MATCH DATABASE

dBASE kann den Index nicht mit der Datenbank verknüpfen.

Versuchen Sie eine andere Indexdatei.

INDEX FILE CANNOT BE OPENED

Index-Datei kann nicht eröffnet werden. Ueberprüfen Sie die Schreibweise oder indizieren Sie die Datei neu.

JOIN ATTEMPTED TO GENERATE MORE THAN 65,534 RECORDS

Mit **JOIN** würden mehr als 65.534 Datensätze erzeugt. Die **FOR**-Bedingung erlaubt zu viele Einträge in der Zieldatei, fassen Sie die Bedingung etwas enger.

KEYS ARE NOT THE SAME LENGTH

Die Schlüssel haben nicht die gleiche Länge.

MACRO IS NOT A CHARACTER STRING

&Makros müssen Zeichenketten sein.

MORE THAN 5 FIELDS TO SUM

Mehr als 5 Felder im **SUM**-Befehl

NESTING LIMIT VIOLATION EXCEEDED

Zu viele verschachtelte Unterprogrammaufrufe, zu viele Files eröffnet.

NO EXPRESSION TO SUM

kein Ausdruck/Feld zur Summierung angegeben

NO "FOR" PHRASE

Der "FOR"-Ausdruck fehlt.

NO "FROM" PHRASE

Der "FROM"-Ausdruck fehlt.

NO FIND

Mehr eine zusätzliche Information als eine Fehlermeldung.
dBASE konnte den Suchbegriff nicht finden.

NON-NUMERIC EXPRESSION

nicht-numerischer Ausdruck

NONEXISTENT FILE

Datei ist noch nicht angelegt.

"ON" PHRASE NOT FOUND

"ON"-Ausdruck wurde nicht gefunden.

OUT OF MEMORY FOR MEMORY VARIABLES

Die Speichervariablen belegen zu viel Platz. Verringern Sie die Größe (max. 1000) oder Anzahl (max. 64) der Speichervariablen.

RECORD LENGTH EXCEEDS MAXIMUM SIZE (OF 1000)

Die Datensätze (Records) sind zu lang.

RECORD NOT IN INDEX

Die Indexdatei wurde nicht aktualisiert nachdem ein Datensatz hinzugefügt bzw. geändert wurde.
Bitte neu indizieren.

RECORD OUT OF RANGE

Die Nummer des Datensatzes ist größer als die Zahl der Datensätze in der Datei. Der Datensatz existiert nicht.

SORTER INTERNAL ERROR, NOTIFY SCDP

Interner Fehler, fragen Sie Ihren Händler um Unterstützung.

SOURCE AND DESTINATION DATA TYPES ARE DIFFERENT

Die Datentypen in Quell- und Zielfeld sind verschieden.

***** SYNTAX ERROR *****

Die Grundregeln der dBASE-Kommandosprache wurden nicht eingehalten. Bitte überprüfen Sie Ihre Eingabe.

SYNTAX ERROR IN FORMAT SPECIFICATION

Syntaxfehler in der Festlegung des Formats.

SYNTAX ERROR, RE-ENTER

Syntaxfehler, Eingabe korrigieren.

"TO" PHRASE NOT FOUND

"TO" Ausdruck wurde nicht gefunden.

TOO MANY CHARACTERS

Zu viele Zeichen

TOO MANY FILES ARE OPEN

Es dürfen gleichzeitig nur maximal 16 Dateien offen sein.

TOO MANY MEMORY VARIABLES

Es sind maximal 64 Speichervariablen zugelassen.

TOO MANY RETURNS ENCOUNTERED

Vermutlich ein Fehler im Aufbau der Programmdatei.

"WITH" PHRASE NOT FOUND

"WITH"-Ausdruck wurde nicht gefunden.

UNASSIGNED FILE NUMBER

Interner Fehler, fragen Sie Ihren Händler um Unterstützung.

***** UNKNOWN COMMAND**

Unbekanntes Kommando, überprüfen Sie Ihre Eingabe.

VARIABLE CANNOT BE FOUND

Überprüfen Sie die Schreibweise, bzw. richten Sie die Variable neu ein.

4. Kapitel Praktische Beispiele mit dBASE II

Inhalt

Abschnitt 1: Erläuterungen zu den Beispielen und der Testversion.....	375
Abschnitt 2: Das Beispiel Adreßverwaltung.....	382
Abschnitt 3: Das Beispiel Rechnungsverwaltung.....	399

Testversion und Beispiele

Erläuterungen zu den Beispielen und der Testversion

Die Testversion von dBASE II

Vor einer endgültigen Kaufentscheidung für das Datenbanksystem dBASE II können Sie mit Hilfe der Testversion, deren Kapazität auf die Erstellung und Benutzung von 15 Datensätzen pro Datenbank begrenzt ist, alle Funktionen von dBASE II ausprobieren.

Auf diese Art lernen Sie am besten die volle Leistungsfähigkeit des Systems kennen. In Verbindung mit den dBASE II-Beispielen auf der dBASE II-Diskette können Sie sofort mit zwei komplett ausgearbeiteten Lösungen arbeiten.

Handhabung der dBASE II-Beispiele

Um Ihnen den Einstieg in das Datenbanksystem dBASE II zu erleichtern, geht der nachfolgende Überblick zunächst auf die wesentlichen Unterschiede zwischen Datenbanksystemen und üblichen Dateiverwaltungen ein.

Sie können die wichtigsten Funktionen von dBASE II an diesen Beispielen ohne langwieriges Studium dieses Handbuches kennenlernen. Natürlich erschließt sich damit noch nicht das gesamte Leistungsspektrum von dBASE II. Zur Konstruktion individueller Anwendungen ist das Studium des Handbuches unerlässlich. Es gehört jedoch zu den besten Handbüchern, die für CP/M-Anwendungspakete auf dem Markt sind und ist hervorragend zum Selbststudium geeignet.

dBASE II im Überblick

Was ist ein Datenbanksystem?

Bei der üblichen Dateiverwaltung, wie sie etwa vom Betriebssystem oder von Programmiersprachen zur Verfügung gestellt wird, ist es weitgehend Aufgabe des Benutzers, mit eigenen Programmen die Erfassung, Verwaltung und Verarbeitung seiner Daten zu übernehmen. Je weniger Funktionen von der Standardsoftware hierfür bereitstehen, um so mehr verlagert sich der Aufwand zum Benutzer. Wollte man unter CP/M eine typische Anwendung, wie z.B. die Verwaltung einer Kundendatei mit Unterstützung zum Schreiben von Angeboten oder Überwachung von Rechnungen aufbauen, so wären hierfür zahlreiche separate Programme mit eigenen Dateien die Regel. Allein für das Heraussuchen bestimmter Kunden und zur Aktualisierung ihrer Daten wäre ein größeres Programm/Teilprogramm erforderlich.

Anders ist es bei einem Datenbanksystem. Hier sind bereits alle Funktionen eingebaut, mit denen Sie ständig den Überblick über alle Daten haben und die nach Ihren Wünschen bestimmte Daten heraussuchen und weiterverarbeiten. Bei einer Kundendatei müssen also von Ihnen keine Programme geschrieben werden, um Daten zu erfassen oder Kunden z.B. branchenbezogen oder nach offenen Rechnungen herauszufinden.

Weitere, wesentliche Vorteile von Datenbanksystemen sind die mit ihnen erreichbare Flexibilität und höhere Zuverlässigkeit. So bereitet es z.B. keine Mühe, etwa bei Kundenadressen die Datenstruktur nachträglich um die Telefonnummer zu erweitern. Ferner steht mit einer Datenbank für unterschiedliche Aufgaben ein gemeinsamer Datenbestand zur Verfügung. Die Aufspaltung auf mehrere Dateien mit den daraus folgenden Aktualisierungsproblemen - etwa Übernahme geänderter Kundenadressen in Rechnungen - entfällt. Letztlich wird mit einem Datenbanksystem also nicht nur Arbeitsaufwand sondern auch Speicherplatz gespart.

dBASE II: ein Datenbanksystem für Personal Computer und CP/M

Mit dBASE II steht auch für Personal Computer und CP/M ein leistungsfähiges Datenbanksystem zur Verfügung, das den Vergleich mit den Systemen an größeren Rechnern nicht zu scheuen braucht. Dank des günstigen Preis-/Leistungsverhältnisses, seiner überaus hohen Flexibilität und der Sicherheit auch bei komplizierten Anwendungen hält es in den USA seit Jahren den Spitzenplatz in der Konkurrenz.

Die Struktur einer Datenbank in dBASE II entspricht der einer Tabelle: bis zu 65.535 Zeilen oder Sätze (RECORDS) - im Demonstrationspaket auf 14 begrenzt - und bis zu 32 Spalten, Felder (FIELDS) genannt, sind möglich. Jede Zeile kann bis zu 1024 Zeichen fassen, die sich auf die Spalten aufteilen. An Eintragungen sind Texte und Zahlen zugelassen, wobei deren maximale Länge jeweils vom Anwender vorher festzulegen ist. Als weiterer Datentyp sind logische Werte (Ja, Nein) erlaubt.

Mit nur einem Kommando (CREATE) wird eine Datenbank eingerichtet. Nach Beantwortung von Fragen zur Struktur (Spaltenüberschriften, Festlegung Datentyp und max. Länge) kann sofort zur Erfassung übergegangen werden.

Wie mächtig dBASE II ist, zeigt sich aber erst mit den Funktionen, die Daten heraussuchen und verknüpfen. Sie selbst bestimmen mit nur wenigen Kommandos - ohne jeden Programmieraufwand das "Sieb", mit dem bestimmte Daten aus dem Bestand herausgefiltert werden und legen deren weitere Verarbeitung fest. Auch bei denen im kommerziellen Einsatz so wichtigen Berichten hilft Ihnen dBASE II. Mit einer speziellen Funktion (REPORT) können in kurzer Zeit Auswertungen am Drucker erstellt werden, wobei Ihre Formatwünsche weitgehend berücksichtigt werden. Damit ist etwa eine Auflistung aller offenen Rechnungen einer Kundendatei mit Zwischen- und Gesamtsummen nur eine Sache von Minuten.

Insgesamt verkehrt der Nutzer im Dialog über 50 Kommandos mit dem System. Auch wenn die Kommandos in Englisch einzugeben sind, werden die von Ihnen festzulegenden Bezeichnungen wie Datei- oder Feldnamen, Überschriften in Berichten usw. natürlich in Deutsch akzeptiert. Mit wenigen Kommandos sind dabei auch komplizierte Aufgaben zu beherrschen.

Wo dies nicht ausreicht und der Komfort noch gesteigert werden soll, können Kommandos zu neuen Funktionen zusammengestellt werden, die dann durch nur ein Kommando aufgerufen werden (siehe Beispiele). So vorbereitete Anwendungen laufen dann bis hin zu Auswertungen und Ausgaben von Berichten automatisch ab. Dem Benutzer bleibt nur noch die Eingabe von Daten. Sie werden vom System im Dialog angefordert und können vor der Weiterverarbeitung noch auf Syntax oder Plausibilität geprüft werden. Kenntnisse über dBASE II sind dann nicht mehr erforderlich!

dBASE II ist außerordentlich schnell: Datenbanken können beliebig indiziert werden, d.h. Zugriffe sind direkt über Schlüssel möglich (z.B. den Nachnamen). In diesem Fall erfolgt der Zugriff auf Daten im Durchschnitt in nur 2 Sekunden (!) bei Verwendung von Floppy-Disks. Änderung von Daten aktualisiert automatisch auch die Indizierung. Das gesamte System ist in Maschinensprache implementiert, läuft also "direkt" auf der Hardware (ohne Interpreter). dBASE II ist daher auch für Anwendungen mit Forderungen nach kurzer Reaktionszeit (z.B. telefonische Auskünfte oder Auftragsannahme) geeignet.

Einsatz von dBASE II

Hinweis zur Implementierung

Im Dialog nutzt dBASE II besondere Eigenschaften von Terminals aus, etwa für Schrift mit halber Intensität sowie direkte Adressierung von Positionen auf dem Bildschirm. Daher bedarf es einer Anpassung des Systems an das vorhandene Terminal, um diese Leistungen nutzen zu können. Nachdem zur Sicherheit eine Kopie der Demonstrationsdiskette angefertigt wurde, ist das Kommando **INSTALL** einzugeben (bei mehreren Laufwerken muß die Demo-Diskette im Primärlaufwerk vorhanden sein; also Umschaltung unter CP/M mit B: für Diskette auf B-Laufwerk). Die Frage

ARE FULL SCREEN OPERATIONS WANTED (Y/N)?

ist mit Y zu beantworten. Daraufhin erscheint ein Menü, das verschiedene Terminals, jeweils mit einem Kennbuchstaben versehen, aufführt. Durch Auswahl eines Buchstabens wird die Anpassung an das entsprechende Terminal automatisch vorgenommen.

Fehlt das vorhandene Terminal im Menü, kann zunächst ein beliebiges Terminal ausgewählt werden. Später bei den Beispielen zeigt sich sofort, ob zufällig ein dem vorhandenen äquivalentes Terminal gefunden wurde oder der Anpassungsversuch mit einer anderen Terminalauswahl zu wiederholen ist (außerhalb des noch erläuterten Kommandos dBASE II kann **INSTALL** jederzeit aufgerufen werden).

Ein sicherer Weg zur Anpassung an beliebiges, im Menü nicht angegebenes Terminal, ist die Auswahl "Z", d.h. die charakteristischen Terminaldaten sind vom Nutzer anzugeben. In diesem Fall müssen die Zeichenfolgen zur Steuerung der Ausgabe auf dem Bildschirm für das Terminal bekannt sein (normalerweise im Benutzerhandbuch aufgeführt). Sie werden einzeln vom INSTALL-Programm abgefragt (siehe Beispiel im Handbuch, Seite dBASE II...8.)

Die nach der Auswahl des Terminals erfolgenden Aufforderungen ENTER A CHARACTER TO BE USED FOR INDICATING MACROS ... sowie TYPE RETURN IF THE ERROR CORRECTION DIALOGUE IS ... sind jeweils mit RETURN zu beantworten (spätere Re-Installation mit anderen Angaben für spezielle Anwendungen möglich).

Mit der Eingabe von Y auf TYPE 'Y' TO SAVE, ANY OTHER CHARACTER TO ABORT INSTALL ist das Installationsprogramm beendet und dBASE II implementiert.

Arbeiten mit dBASE II

dBASE II wird unter CP/M mit DBASE aufgerufen (Laufwerk mit Diskette in Primärlaufwerk.) Es meldet sich mit der Frage nach dem Datum:

ENTER TODAYS DATE AS MM/DD/YY OR RETURN NONE:

Für die Beispiele genügt als Eingabe RETURN. (Eine Datumsangabe - in der Reihenfolge Monat, Tag und Jahr mit je zwei Ziffern durch Schrägstrich getrennt - wird bei allen Datenbanken gespeichert, die im Verlauf einer Sitzung verändert werden).

Nach der Meldung mit Programmnamen und Versionsnummer steht dBASE II für Aufträge bereit. Dieser Zustand ist durch die Angabe eines Punktes am linken Rand des Bildschirms gekennzeichnet (PROMPT).

Mit dem Kommando

DO ADRESSEN

wird das Beispiel zur Adressenverwaltung aufgerufen. Durch umfangreiche Erläuterungen erklärt es sich weitgehend selbst.

DO RECHNUNGEN ruft das Beispiel zur Verwaltung von Rechnungs- und Zahlungsdaten auf, das auf dem vorher bei DO ADRESSEN erzeugten Datenbestand aufbaut.

Falls Daten in Felder (erkennlich an der Begrenzung durch Doppelpunkte) einzugeben sind, können zur Positionierung des CURSORS oder Editierung folgende Zeichenfolgen verwendet werden (Angabe CTL-X bedeutet gleichzeitiges Drücken der CONTROL-Taste und des Zeichens X):

CTL-X : CURSOR zum nächsten Eingabefeld,

CTL-E : CURSOR zum vorhergehenden Eingabefeld,

CTL-D : CURSOR ein Zeichen weiter,

CTL-S : CURSOR ein Zeichen zurück,

CTL-G : Löschen des Zeichens unter CURSOR,

RUB/DEL: Löschen des Zeichens links vom CURSOR.

Zum Beenden von dBASE II und zur Rückkehr in das Betriebssystem ist das Kommando QUIT einzugeben.

Wichtige Hinweise:

- * Falls die Beispiele zu nicht lesbaren Ausgaben führen, ist die Anpassung des Terminals mit `INSTALL` zu wiederholen (siehe 1. Kapitel)
- * Wenn die Verwendung unterschiedlicher Helligkeit zur Unterscheidung von Formaten und Eingaben stört, kann mit `SET INTENSITY OFF` (wenn Punkt erscheint, also außerhalb der Beispiele) auf normale Helligkeit umgeschaltet werden.
- * Die Namen der beim Beispiel zur Adressenverwaltung verwendeten Dateien beginnen mit `AD`. Soweit sie mit `.FMT` oder `.CMD` enden, enthalten sie `dBASE II`-Kommandos und können wie Textdateien ausgegeben werden. Diese Dateien sind zur Erläuterung einzelner Kommandos ausführlich kommentiert.
- * Bei den Beispielen sind z.T. extreme Sicherheitsvorkehrungen getroffen worden (Anwendung von `dBASE II` ohne jede Einweisung), um Fehlbedienungen auszuschließen. Sie würden bei üblichen Anwendungen so nicht vorgesehen werden, so daß dann eine Reihe von Funktionen schneller ablaufen.

Handhabung der `dBASE`-Beispiele

Nachfolgend werden Beispiele für Befehlsdateien gegeben und gezeigt, wie damit komplette, benutzerfreundliche Anwenderlösungen aufgebaut werden können. Solche Befehlsdateien definieren neue Befehle in `dBASE` und bieten damit vielfältige Erweiterungsmöglichkeiten. Sie setzen sich aus `dBASE`-Befehlen (ggf. auch neuen, mit anderen Befehlsdateien definierten) zusammen und gestatten die Programmierung eines beliebigen Dialogs mit dem Benutzer. Der Ablauf läßt sich mittels "IF"-Bedingungen oder "CASE" dynamisch steuern.

Die Beispiele wurden einem Demonstrationspaket zu `dBASE` entnommen. Als Aufgabenstellung liegt im ersten Teil die Verwaltung von Anschriften/Adressen zugrunde. Im zweiten Teil werden die damit angelegten Datenbestände genutzt, um Rechnungen und Zahlungen zu überwachen. Die Dialoge mit dem Benutzer sind bewußt ausführlich gehalten, so daß für die Anwendung keine spezifischen Kenntnisse über `dBASE` verlangt werden.

Die Anwenderlösungen wurden modular aufgebaut, d. h. mit mehreren Befehlsdateien, die sich über verschiedene Stufen hinweg aufrufen. Dies erleichtert spätere Änderungen erheblich und gestattet, einzelne Funktionen separat auszutesten. Es wird empfohlen, sich diesem Konstruktionsprinzip anzuschließen. Erweiterungen und Änderungen lassen sich dann wesentlich einfacher und übersichtlicher realisieren.

Ein Hinweis zu den Dateien selbst: an zahlreichen Stellen wird von "IF"-Abfragen Gebrauch gemacht. `dBASE` bietet inzwischen hierfür mit "CASE" eine kürzere Schreibweise an (siehe CASE-Befehl).

Befehlsdateien können auch programmgesteuert, also "automatisch" von `dBASE` generiert werden. Mit dem zu `dBASE` gelieferten Paket "ZIP" lassen sich die Dateien mit anwenderspezifischen Formaten (Masken) zur Erfassung und Ausgabe von Daten direkt aus einem vom Benutzer auf dem Bildschirm angegebenen Muster erzeugen.

Das zusätzlich erhältliche Paket "QUICKCODE" erlaubt die Konstruktion umfassender Anwenderlösungen ohne eigene Programmierung und weitgehend ohne spezielle Kenntnisse über `dBASE`. Die Erstellung selbstprogrammierter Anwendungen wird mit dem Zusatzpaket "dUTIL" wesentlich erleichtert.

TESTVERSION...380

Folgende Datenbanken werden in den Beispielen verwendet:

. USE ADRESSEN

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: ADRESSEN.DBF

NUMBER OF RECORDS: 00004

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NAME	C	030	
002	ABTEIL	C	025	
003	STRASSE	C	025	
004	PLZ	C	005	
005	ORT	C	020	
006	TELEFON	C	015	
007	BRANCHE	C	020	
** TOTAL **			00141	

. USE RECHADR

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: RECHADR.DBF

NUMBER OF RECORDS: 00004

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	KUNDENNR	N	005	
002	NAME	C	030	
003	ABTEIL	C	025	
004	STRASSE	C	025	
005	PLZ	C	005	
006	ORT	C	020	
007	TELEFON	C	015	
008	BRANCHE	C	020	
** TOTAL **			00146	

. USE RECH

STRUCTURE FOR FILE: RECH.DBF

NUMBER OF RECORDS: 00002

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	KUNDENNR	N	005	
002	NAME	C	030	
003	RECHNR	N	006	
004	LIEFERNR	N	006	
005	RECHDATUM	C	008	
006	BETRAG	N	009	002
007	IST	N	009	002
008	FEHL	N	009	002
009	BEMERK	C	040	
** TOTAL **			00123	

. USE RECHNUNG

. DISPLAY STRUCTURE

STRUCTURE FOR FILE: RECHNUNG.DBF

NUMBER OF RECORDS: 00003

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

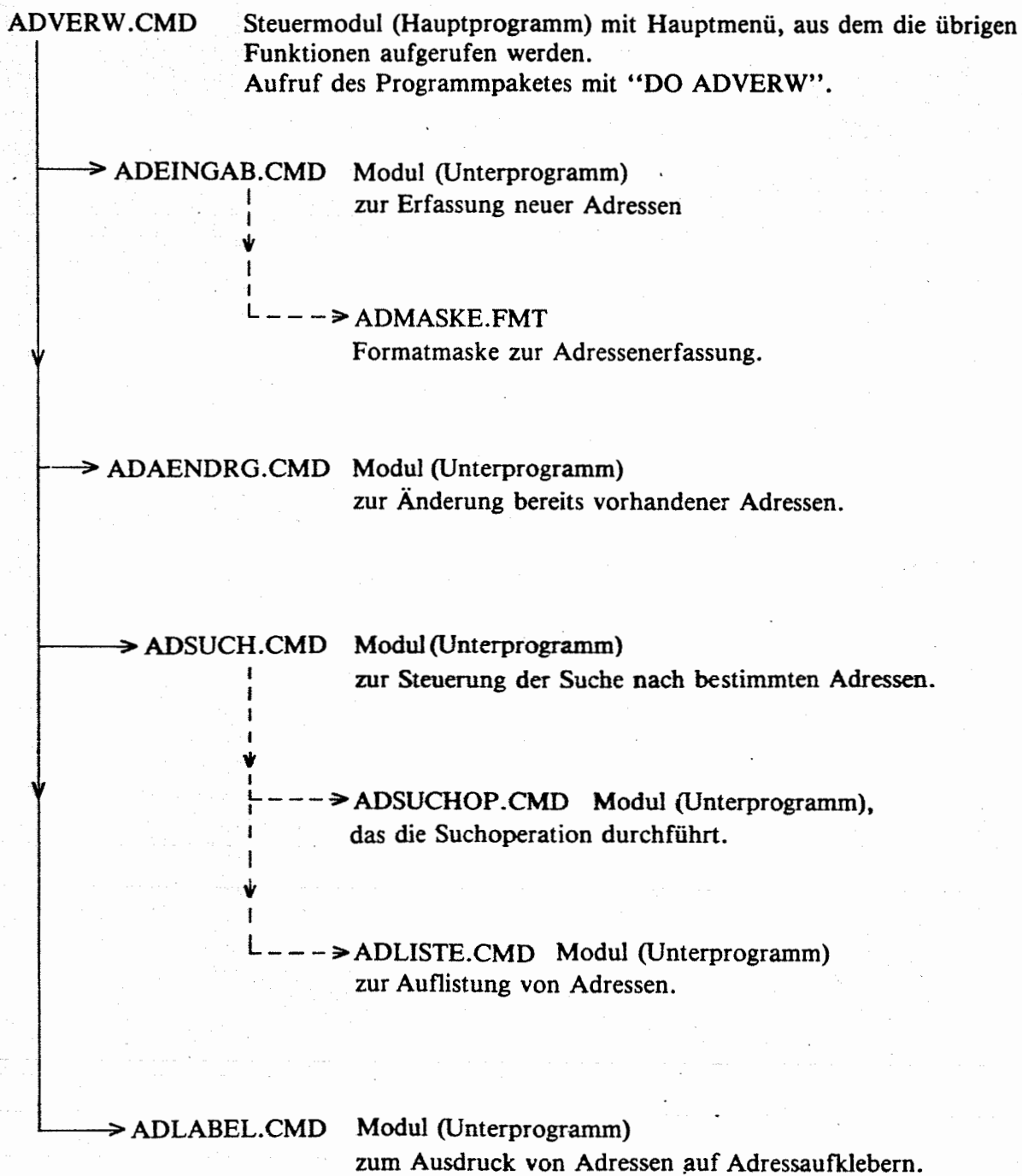
FLD	NAME	TYPE	WIDTH	DEC
001	KUNDENNR	N	005	
002	LIEFERNR	N	006	
003	RECHNR	N	006	
004	RECHDATUM	C	008	
005	BETRAG	N	009	002
006	IST	N	009	002
007	FEHL	N	009	002
008	BEMERK	C	040	
** TOTAL **			00093	

Programmbeispiel Adressverwaltung

Inhalt

Programmstruktur Adressverwaltung	383
Hauptprogramm zur Adressverwaltung (ADVERW.CMD).....	384
Eingabe von Adressen (ADEINGAB).....	387
Ändern/Löschen von Adressen (ADAENDRG).....	388
Suchen von Adressengruppen (ADSUCH.CMD).....	391
Suchprogramm zum Suchen von Adressengruppen (ADSUCHOP).....	394
Auflistung von Adressen (ADLISTE).....	395
Ausdruck von Adressaufklebern (ADLABEL).....	397
Maske zur Eingabe von Adressen (ADMASKE).....	398

Programmstruktur Adressenverwaltung



Programmbeispiel Adressenverwaltung

ADVERW

*****ADVERW.CMD*****

```

*
*           Demonstration dBASE II
*      Programm zur Adressenverwaltung
*      Markt & Technik, Softwareverlag
*

```

```

*
* Das Programm setzt sich aus dBASE II - Kommandos zusammen und übernimmt die
* Steuerung der Adressenverwaltung. Vom Bediener werden keine spezifischen Kenntnisse über
* dBASE II verlangt.
*

```

```

* dBASE II - Kommandos und Parameter sind zur Verdeutlichung in Grossbuchstaben
* angegeben.
*

```

```

* Das Programm beginnt hier:

```

```

SET TALK OFF

```

```

* = Unterdrueckung Kommandoausgabe auf Bildschirm

```

```

USE ADRESSEN

```

```

GOTO BOTTOM

```

```

IF .NOT. FILE('ADINDEX.NDX')

```

```

    INDEX (NAME-TELEFON) TO ADINDEX

```

```

ENDIF

```

```

* Prüfung, ob Schlüsseldatei "ADINDEX.NDX" vorhanden ist, ggf. wird sie neu aufgebaut.

```

```

USE ADRESSEN INDEX ADINDEX

```

```

* = Anschluss der Datenbank mit den Adressen (in diesem Beispiel vorgegeben; könnte auch
* beim Bediener erfragt werden, wenn mehrere, getrennte Datenbestände mit diesem Pro-
* gramm verwaltet werden sollen).

```

```

DO WHILE T

```

```

* = Schleife für Auswahlmenue, solange keine Eingabe zur Beendigung des Programmes erfolgt

```

```

    ERASE

```

```

    ?

```

```

    ?

```

```

    ?

```

```

    ?

```

```

    ?

```

? ***** M E N U E zur Adressenverwaltung ***** ,

?

? Folgende Funktionen stehen zu Ihrer Verfügung:

?

? 0 : Beenden des Programms'

? 1 : Neue Adressen eingeben'

? 2 : Loeschen / Aendern von Adressen'

? 3 : Heraussuchen von Adressen-Teilmengen '

? 4 : Auflistung der vorhandenen Adressen'

?

?

? Geben Sie die Nr. der gewuenschten Funktion ein,'

? nachdem WAITING erscheint.'

?

?

?

WAIT TO ANTW

* = Antwortzeichen wird in Variable ANTW übernommen

IF ANTW = '0' .

RETURN

ENDIF

IF ANTW = '1'

DO ADEINGAB

* = Aufruf Unterprogramm zur Adresseneingabe

ENDIF

IF ANTW = '2'

DO ADAENDRG

* = Aufruf Unterprogramm zum Aendern/Loeschen von Adressen

ENDIF

IF ANTW = '3'

DO ADSUCH

* = Aufruf Unterprogrammm zum Suchen von Adressen

ENDIF

```

IF ANTW = '4'
  ERASE
  GOTO TOP
IF # = 0
  * keine Adressen vorhanden
  ERASE
  @ 10,15 SAY '?????????????????????????????????????????'
  @ 12,15 SAY 'Keine Adressen im Bestand! Bitte im Ein-'
  @ 14,15 SAY 'gangsmenue die Funktion zur Eingabe neuer'
  @ 16,15 SAY 'Adressen waehlen und Adressen eingeben!'
  @ 18,15 SAY '?????????????????????????????????????????'
  ?
  ?
  ? 'Zur Fortsetzung druecken Sie bitte RETURN, nachdem'
  ? 'WAITING erscheint'
  WAIT
ELSE
  @ 5,10 SAY 'Ausgabe der Adressen erfolgt alphabetisch'
  @ 5,52 SAY 'sortiert.'
  @ 7,10 SAY 'Wo und in welcher Form soll Ausgabe erfolgen?'
  @ 7,10 SAY '*****'
  @ 9,10 SAY '          1 : Liste auf Bildschirm'
  @ 10,10 SAY '          2 : Liste auf Drucker'
  @ 11,10 SAY 'sonst. Eing.: Drucken von Adr.-Aufklebern'
  @ 14,10 SAY 'Geben Sie bitte Ihre Wahl ein, nachdem'
  @ 15,10 SAY 'WAITING erscheint.'
  ?
  IF ANTWAUS = '1'
    STORE '1' TO ANTW
    DO ADLISTE
  ELSE
    IF ANTWAUS = '2'
      DO ADLISTE
    ELSE
      DO ADLABEL
    ENDIF
  ENDIF
ENDIF
ENDIF
ENDIF
ENDDO
RETURN

```

Programmbeispiel Adressverwaltung

ADEINGAB

```

***** ADEINGAB.CMD *****
*                               *
*           Demonstration dBASE II                               *
*   Adressenverwaltung: Unterprogramm zur Eingabe von Adressen   *
*           Markt & Technik, Softwareverlag                     *
*****
*
* Das Unterprogramm dient der Eingabe neuer Adressen in die Datenbank.
*
* dBASE II - Kommandos und Parameter sind zur Verdeutlichung in Grossbuchstaben
* angegeben.
GOTO BOTTOM
* = Letzten belegten Satz in Datenbank aufsuchen
SET FORMAT TO ADMASKE.FMT
* = (in Datei ADMASKE.FMT vorprogrammierte) Eingabemaske übernehmen
DO WHILE T
* = Schleife für Eingabe
  STORE ' ' TO MNAME
  STORE ' ' TO MABTEIL
  STORE ' ' TO MSTRASSE
  STORE ' ' TO MPLZ
  STORE ' ' TO MORT, MBRANCHE
  STORE ' ' TO MTELEFON
  * = Vorbelegung der Eingabevariablen, wird ggf. durch Eingaben ueberschrieben
  READ
  * = Lesen der Eingaben
  IF MNAME = '
    RETURN
    * = bei 'leerer' Eingabe zum Namen wird Programm beendet
  ENDIF
  APPEND BLANK
  * = anfügen eines leeren Satzes der nachfolgend mit eingegebenen Werten belegt wird
  REPLACE NAME WITH MNAME, ABTEIL WITH MABTEIL
  REPLACE STRASSE WITH MSTRASSE, PLZ WITH MPLZ, ORT WITH MORT
  REPLACE BRANCHE WITH MBRANCHE, TELEFON WITH MTELEFON
ENDDO
SET FORMAT TO SCREEN
* weiterer Eingabeformate werden direkt auf Bildschirm angegeben
RETURN

```

Programmbeispiel Adressverwaltung

ADAENDRG

```

***** ADAENDRG.CMD *****
*                                     *
*      Demonstration dBASE II      *
*      Adressenverwaltung: Unterprogramm zum ÄNDERN/Löschen von Adressen  *
*      Markt & Technik, Softwareverlag                                *
*****
* Das Unterprogramm dient dem Ändern bzw. Löschen von Adressen in der Datenbank
ERASE
?
? '      *****Ändern / Löschen von Adressen *****'
?
? '      Mit dieser Funktion werden Adressen nach dem Namen, ggf.'
? '      erweitert um die TelefonNr, aus dem Datenbestand heraus-'
? '      gesucht. Die gefundene Adresse wird ausgegeben und kann'
? '      dann ganz oder teilweise geändert bzw. gelöscht werden'
?
? ' Zur Fortsetzung des Programms drücken Sie bitte RETURN, nachdem'
? ' WAITING erscheint'
WAIT
GOTO TOP
* = Ersten Satz in Datenbank aufsuchen
SET FORMAT TO SCREEN
SET TALK OFF
DO WHILE T
* = Schleife für Eingabe
ERASE
? '      Üblich ist bei Adressen eine eindeutige Identifizie-'
? '      rungsmöglichkeit (z.B. Kunden- oder Mitgliedsnummer)'
? '      Im Beispiel wurde davon kein Gebrauch gemacht. Adressen'
? '      sollen hier über den Namen - falls mehrere Adressen mit'
? '      gleichem Namen vorliegen: zusätzlich über TelefonNr -'
? '      identifiziert werden.'
? '      Sie werden nun zur Angabe des Namens und der TelefonNr der'
? '      gewünschten Adresse aufgefordert. Verkürzte Angaben sind'
? '      möglich, etwa Weglassen der TelefonNr und Verwendung nur'
? '      der ersten Buchstaben vom Namen. Falls die gewünschte'
? '      Adresse nicht gefunden wurde, sind weitere Versuche mit'
? '      ausführlicheren Angaben vorzunehmen.'

```



```

?' *****
STORE ' ' TO MNAME
STORE ' ' TO MTELEFON
* Vorbelegung der Suchvariablen
? ' Eingaben mit RETURN abschließen, wird kein Name ange-
? ' geben, so wird das Programm zum Ändern/Löschen beendet'
@ 18,2 SAY 'Angaben zum Namen ' GET MNAME
@ 19,2 SAY '(nur Name, nicht Abteilung/Ansprechpartner)'
@ 19,2 SAY 'ggf. zur TelefonNr (sonst nur RETURN) ' GET MTELEFON
@ 20,2 SAY '(bei Angab. hierzu muss Name vollst. sein )'
READ
* Lesen der Eingaben
IF MNAME = '
RETURN
ENDIF
STORE (MNAME - MTELEFON) TO MSUCH
STORE TRIM(MSUCH) TO MSUCH
* Zusammensetzen des Suchbegriffes aus MNAME und MTELEFON
FIND &MSUCH
* Suche in Datenbank (Index), ob MSUCH vorhanden

IF # = 0
* gesuchte Adresse nicht gefunden
ERASE
@ 5,10 SAY '?????????????????????????????????????'
@ 7,10 SAY ' Adresse NICHT gefunden !
@ 9,10 SAY 'Gesucht wurde nach:'
STORE 'Name : ' + MNAME TO MNAME
@ 11,25 SAY MNAME
IF MTELEFON = '
STORE 'Telefon : (keine Angabe)' TO MTELEFON
ELSE
STORE 'Telefon : ' + MTELEFON TO MTELEFON
ENDIF
@ 13,25 SAY MTELEFON
@ 15,10 SAY '?????????????????????????????????????'
?
?
? 'Zur Fortsetzung des Programms bitte RETURN drücken,'
? 'nachdem WAITING erscheint'
?
?
WAIT

```

ELSE

ERASE

@ 2,15 SAY '!!'

@ 3,15 SAY ' Folgende Adresse wurde gefunden:'

@ 4,15 SAY '!!'

@ 6,5 SAY 'Name' ' GET NAME

@ 7,5 SAY 'Abteilung/Ansprechpartner' ' GET ABTEIL

@ 9,5 SAY 'Strasse' ' GET STRASSE

@ 10,5 SAY 'PLZ/ORT' ' GET PLZ PICTURE 'X9999'

* man beachte die PICTURE-Angabe, die nur bestimmte Eingaben zuläßt

@ 10,38 GET ORT

@ 12,5 SAY 'Tel.' 'GET TELEFON

@ 12,31 SAY 'Branche' GET BRANCHE

?

?' Geben Sie nach WAITING Ihre Wahl zur weiteren Bearbeitung ein:'

?

?' 1 : Adresse ändern (Eingaben werden aus o.a. Feldern abge-'

?' gefragt. Wird nur RETURN eingegeben, bleibt alter Wert'

?' erhalten.)'

?' 9 : Adresse löschen'

?' sonst: neue Adresse suchen'

?

WAIT TO ANTW

IF ANTW = '1'

READ

ELSE

IF ANTW = '9'

DELETE

* = Adresse löschen

PACK

* = entfernen gelöschte Adresse aus Datei zu Demonstrationszwecken;

* normalerweise erst notwendig, wenn zahlreiche gelöschte Adressen

* vorliegen

ENDIF

ENDIF

ENDIF

ENDDO

RETURN

Programmbeispiel Adressverwaltung

ADSUCH

```

***** ADSUCH.CMD *****
*
*      Demonstration dBASE II
*
*      Adressenverwaltung: Unterprogramm zum Suchen von Adressengruppen
*      Markt & Technik, Softwareverlag
*
*****
*
* Das Unterprogramm dient zum Suchen von Adressen, die einem gemeinsamen Kriterium
* (im Beispiel: gleiche Postleitzahl und/oder gleiche Branche) genügen.
*
ERASE
? '      *****Suchen von Adressen nach gemeinsamen Kriterien*****'
?
? '      Mit dieser Funktion wird nach Adressen gesucht, bei denen'
? '      bestimmte Daten gleich sind. Hier sind es Adressen mit glei-'
? '      cher Postleitzahl und/oder Branche.'
?
? ' Zur Fortsetzung des Programms druecken Sie bitte RETURN, nachdem '
? ' WAITING erscheint'
WAIT
SET FORMAT TO SCREEN
DO WHILE T
* = Schleife für Eingabe
ERASE
? '      Geben Sie bitte die Postleitzahl bzw. die Branche an, nach'
? '      denen im Adressenbestand gesucht werden soll. (Sie haben'
? '      doch bereits Adressen eingegeben? Andernfalls waehlen Sie'
? '      bitte im Eingangsmaenue die Funktion EINGABE NEUER ADRESSEN).'
?
? '      Bei der Auswahl kann nur die Postleitzahl oder nur die '
? '      Branche angegeben werden. Sind beide Angaben vorhanden, so'
? '      werden diejenigen Adressen bestimmt, die beide Kriterien er-'
? '      füllen (UND-Verknüpfung). Eingaben sind mittels RETURN abzu-'
? '      schliessen. Nur RETURN fuer beide Angaben beendet Funktion.'
? '      *****'

```

```

STORE ' ' TO MPLZ
STORE ' ' TO MBRANCHE
* = Vorbelegung der Suchvariablen
@ 16,2 SAY 'Angabe der Postleitzahl ' GET MPLZ PICTURE 'X9999'
@ 17,2 SAY '(ab 2. Stelle nur Zifferneingaben moeglich !)'
@ 19,2 SAY 'Angabe der Branche ' GET MBRANCHE
READ
IF MPLZ = ' ' .AND. MBRANCHE = ' '
    SELECT PRIMARY
    SET FORMAT TO SCREEN
    RETURN
ENDIF
STORE TRIM(MPLZ) TO MPLZ
STORE TRIM(MBRANCHE) TO MBRANCHE
* TRIM entfernt ggf. vorhandene Leerzeichen am Ende der Suchangaben
DO ADSUCHOP
* = Aufruf der Suchoperation

ERASE
IF MADZAHL = 0
    @ 5,10 SAY '?????????????????????????????????????????'
    @ 7,10 SAY ' KEINE Adresse(n) gefunden !'
    IF MBRANCHE = ' '
        STORE 'Branche' : ' + '(keine Angabe)' TO MBRANCHE
    ELSE
        STORE 'Branche' : ' + MBRANCHE TO MBRANCHE
    ENDIF
    IF MPLZ = ' '
        STORE 'Postleitzahl : ' + '(keine Angabe)' TO MPLZ
    ELSE
        STORE 'Postleitzahl : ' + MPLZ TO MPLZ
    ENDIF
    @ 9,10 SAY 'Gesucht wurde nach:'
    @ 11,15 SAY MPLZ
    @ 13,15 SAY MBRANCHE
    @ 15,10 SAY '?????????????????????????????????????????'
    ?
    ?
    ? 'Zur Fortsetzung des Programms bitte RETURN druecken,'
    ? 'nachdem WAITING erscheint'
    ?
    ?
    ?
    WAIT

```

```
ELSE
  ERASE
  @ 2,10 SAY '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
  @ 3,10 SAY 'Es wurde(n)'
  @ 3,22 SAY MADZAHL USING '# #'
  @ 3,25 SAY 'Adresse(n) gefunden!'
  @ 4,10 SAY '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
  @ 7,5 SAY 'Bestimmen Sie die weitere Bearbeitung dieser Adresse(n):'
  @ 8,5 SAY '*****'
  @ 10,20 SAY '1 : Auflistung auf dem Bildschirm'
  @ 11,20 SAY '2 : Auflistung auf dem Drucker'
  @ 12,20 SAY '3 : Drucken als Adressaufkleber'
  @ 13,10 SAY 'andere Taste: keine weitere Bearbeitung'
  @ 15,10 SAY 'Geben Sie Ihre Wahl ein, nachdem WAITING erscheint.'
  WAIT TO ANTW
  IF ANTW = '1' .OR. ANTW = '2'
    DO ADLISTE
    * Auflistung
  ELSE
    IF ANTW = '3'
      DO ADLABEL
      * = Ausdruck von Aufklebern
    ENDIF
  ENDIF
ENDIF
ENDIF
SELECT PRIMARY
* Umschalten auf den Primärbereich mit der Datenbank ADRESSEN.DBF
ENDDO
RETURN
```

Programmbispiel Adressverwaltung

ADSUCHOP

```

***** ADSUCHOP.CMD *****
*                               Demonstration dBASE II                               *
*                               Adressenverwaltung: Suchprogramm zum Suchen von Adressengruppen *
*                               Markt & Technik, Softwareverlag                       *
*****
*
* Das Unterprogramm führt die Suche nach Adressen, die einem gemeinsamen Kriterium (im
* Beispiel: gleiche Postleitzahl und/oder gleiche Branche) genügen in der Datenbank durch.
* (Aufgerufen von ADSUCH.CMD).
*
GOTO TOP
* = Anfang der Datenbank aufsuchen
STORE F TO MADSUCH
IF FILE('ADSORT.DBF')
    DELETE FILE ADSORT.DBF
ENDIF
IF FILE('ADSORT.NDX')
    DELETE FILE ADSORT.NDX
ENDIF
* noch vorhandene (Hilfs-)Datenbanken aus früheren Suchoperationen löschen
* nachfolgend Übertragung der gefundenen Adressen in die (automatisch) neu
* eingerichtete Datenbank ADSORT.DBF
IF MPLZ = ''
    COPY TO ADSORT FOR BRANCHE = MBRANCHE
ELSE
    IF MBRANCHE = ''
        COPY TO ADSORT FOR PLZ = MPLZ
    ELSE
        COPY TO ADSORT FOR PLZ = MPLZ .AND. BRANCHE = MBRANCHE
    ENDIF
ENDIF
ENDIF
SELECT SECONDARY
USE ADSORT
* = zweite (Hilfs-)Datenbank zuschalten
GOTO BOTTOM
IF # > 0
    STORE # TO MADZAHL
    * = mindestens 1 Adresse gefunden
    INDEX ON NAME TO ADSORT
    * = alphabetisch nach Namen indizieren für spätere Ausgabe
    STORE T TO MADSUCH
ELSE
    STORE 0 TO MADZAHL
    * = keine Adresse gefunden
ENDIF
USE
SELECT PRIMARY
* = auf erste (Haupt-)Datenbank zurückschalten
RETURN

```

Programmbeispiel Adressverwaltung

ADLISTE

```

***** ADLISTE.FMT *****
*
*           Demonstration dBASE II
*
*           Adressenverwaltung: Auflistung von Adressen
*
*           Markt & Technik, Softwareverlag
*****
* Hiermit werden Adressen alphabetisch nach Namen sortiert zu je zwei Zeilen aufgelistet.
IF ANTW = '1'
    SET FORMAT TO SCREEN
    * = Ausgabe auf Bildschirm
ELSE
    SET FORMAT TO PRINT
    EJECT
    * = Ausgabe auf Drucker, Seitenvorschub
ENDIF
IF MADSUCH
    * es liegen gesuchte Adressen vor; Umschalten auf Sekundärbereich mit (Hilfs-)Daten-
    * bank, in die diese Adressen übertragen wurden
    SELECT SECONDARY
    USE ADSORT INDEX ADSORT
ENDIF
GOTO TOP
ERASE
@ 1,8 SAY 'NAME'
@ 1,30 SAY 'ABTEILUNG/ANSPRECHPARTN.'
@ 1,61 SAY 'TELEFON'
@ 2,8 SAY 'PLZ/ORT'
@ 2,30 SAY 'STRASSE'
@ 2,61 SAY 'BRANCHE'
STORE 3 TO ZEILE
DO WHILE .NOT. EOF
    STORE ZEILE + 1 TO ZEILE
    * = Zeilenzähler
    @ ZEILE,0 SAY NAME
    @ ZEILE,31 SAY ABTEIL
    @ ZEILE,59 SAY TELEFON
    STORE ZEILE + 1 TO ZEILE
    @ ZEILE,0 SAY PLZ
    @ ZEILE,6 SAY ORT
    @ ZEILE,31 SAY STRASSE
    @ ZEILE,59 SAY BRANCHE
    STORE ZEILE + 1 TO ZEILE
    IF ANTW = '1' .AND. ZEILE > 18
        ? 'Zur Fortsetzung bitte RETURN drücken, nachdem WAITING'
        ? 'erscheint'
        * = Bildschirm voll, neue Überschrift
        WAIT
    
```

```

ERASE
@ 1,8 SAY 'NAME'
@ 1,30 SAY 'ABTEILUNG/ANSPRECHPARTN.'
@ 1,61 SAY 'TELEFON'
@ 2,8 SAY 'PLZ/ORT'
@ 2,30 say 'STRASSE'
@ 2,61 SAY 'BRANCHE'
STORE 3 TO ZEILE
ENDIF
SKIP      (* = nächsten Satz in Datenbank speichern)
ENDDO
? '      '
IF ANTW = '1'
?
? 'Zur Fortsetzung bitte RETURN drücken, nachdem'
? 'WAITING erscheint'
?
WAIT
ENDIF
IF MADSUCH
STORE F TO MADSUCH
USE
SELECT PRIMARY
ENDIF
SET FORMAT TO SCREEN
RETURN
    
```


Programmbeispiel Adressverwaltung

ADLABEL

```

***** ADLABEL.FMT *****
*                               Demonstration dBASE II                               *
*                               Adressenverwaltung: Ausdruck von Adressaufklebern      *
*                               Markt & Technik, Softwareverlag                       *
*****
*
* Hiermit werden Adressen alphabetisch nach Namen sortiert auf Adressaufkleber gedruckt.
*
SET FORMAT TO PRINT
EJECT
* = Ausgabe auf Drucker, Seitenvorschub
IF MADSUCH
    * es liegen gesuchte Adressen vor: Umschalten auf Sekundärbereich mit Datenbank, in die
    * diese Adressen übertragen wurden
    SELECT SECONDARY
    USE ADSORT INDEX ADSORT
ENDIF
GOTO TOP
STORE 1 TO ZEILE
DO WHILE .NOT. EOF
    STORE ZEILE + 1 TO ZEILE
    @ ZEILE,10 SAY NAME
    STORE ZEILE + 1 TO ZEILE
    IF ABTEIL <> ' '
        STORE TRIM(ABTEIL) TO MABTEIL
        @ ZEILE,10 SAY '-' + MABTEIL + '-'
    ENDIF
    STORE ZEILE + 2 TO ZEILE
    @ ZEILE,10 SAY STRASSE
    STORE ZEILE + 1 TO ZEILE
    STORE TRIM(PLZ) TO MPLZ
    IF MPLZ = ' '
        @ ZEILE,10 SAY ORT
    ELSE
        @ ZEILE,10 SAY MPLZ + ' ' + ORT
    ENDIF
    STORE ZEILE + 3 TO ZEILE
    SKIP
ENDDO
? ' '
IF MADSUCH
    USE
    SELECT PRIMARY
    STORE F TO MADSUCH
ENDIF
SET FORMAT TO SCREEN
RETURN

```

Programmbeispiel Adressverwaltung

ADMASKE

```

***** ADMASKE.FMT *****
*                                     *
*      Demonstration dBASE II      *
*      Adressverwaltung: Maske zur Eingabe von Adressen      *
*      Markt & Technik, Softwareverlag      *
*****
*
* Hiermit wird eine Maske für die Eingabe von Adressen festgelegt und die Eingaben gelesen
*
* @ x,y bestimmt Position auf Bildschirm, wo Ausgabe beginnt SAY erzeugt Ausgabe des
* anschliessend angegebenen Wertes (hier Text) GET liest zugehörige Eingabe zwischen den
* Doppelpunkten, deren Abstand dem Format entspricht, das durch die Vorbelegung der
* Eingabevariablen (siehe ADEINGAB.CMD) definiert wurde
*
* Das Programm beginnt hier:
@ 1,10 SAY 'HINWEIS zur Adressenerfassung:'
@ 2,10 SAY '===== '
@ 4,10 SAY 'Eingabe zu einem Feld erfolgt zwischen den Doppelpunkten.'
@ 5,10 SAY 'Sie wird entweder mit Druecken von RETURN abgeschlossen oder'
@ 6,10 SAY 'wenn letzte Stelle im Format belegt wurde.'
@ 7,10 SAY 'Sollen keine weiteren Adressen erfasst werden, ist bei NAME'
@ 8,10 SAY 'und allen folgenden Feldern RETURN zu druecken'
@ 10,10 SAY '***** Geben Sie nun die Adresse ein: *****'
@ 12,2 SAY 'Name (Firmenbezeichnung etc.) ' GET MNAME
@ 14,2 SAY 'ggf. Abteilung/Ansprechpartner ' GET MABTEIL
@ 15,2 SAY 'Telefon ' GET MTELEFON
@ 17,2 SAY 'Strasse ' GET MSTRASSE
@ 19,2 SAY 'Postleitzahl ' GET MPLZ PICTURE 'X9999'
@ 19,29 SAY 'Ort ' GET MORT
@ 20,2 SAY '(ab 2. Stelle nur Ziffern!)'
@ 22,2 SAY 'Branche ' GET MBRANCHE

```

Programmbeispiel Rechnungsverwaltung

Inhalt

Programmstruktur zur Verwaltung von Rechnungs-/Zahlungsdaten	399
Hauptprogramm Rechnungsverwaltung (REVERW.CMD).....	400
Übernahme von Adressen aus der Adressenverwaltung (READUEB.CMD)	404
Ändern/Löschen von Adressen (READAEND.CMD).....	406
Eingabe von Rechnungs-/Zahlungsdaten (REEINGAB).....	409
Ändern/Löschen von Rechnungs-/Zahlungsdaten (REREAEND).....	410
Maske zur Eingabe von Rechnungen (REMASKE.FMT).....	412
Druck-Programm für Rechnungs-/Zahlungsübersicht (REREREPORT).....	413

Programmstruktur zur Verwaltung von Rechnungs-/Zahlungsdaten

Anmerkung: Das Programmpaket baut auf den Programmen zur Adressenverwaltung auf, d.h. die dort erfaßten Daten werden übernommen. Es wurde daher keine Funktion zur Erfassung neuer Adressen in diesem Paket vorgesehen. Mit geringfügigen Änderungen können jedoch die entsprechenden Funktionen aus der Adressenverwaltung übernommen werden.

REVERW.CMD Steuermodul

(Hauptprogramm) mit Hauptmenü, aus dem die übrigen Funktionen aufgerufen werden.

Ferner "zusammenstellen" der Adressen mit den: zugehörigen Rechnungs-/Zahlungsdaten und Ausgabe eines Berichtes.

Aufruf des Programmpaketes mit "DO REVERW".

→ REREPORT.FRM
Schema für die Ausgabe eines Berichtes mit Rechnungs-/Zahlungsdaten

→ READUEB.CMD

Modul (Unterprogramm) zur Übernahme der im Programmpaket zur Adressenverwaltung erfaßten Daten.

→ READAEND.CMD

Modul (Unterprogramm) zur Änderung von Adressen.

→ REEINGAB.CMD

Modul (Unterprogramm) zur Erfassung von Rechnungs-/ Zahlungsdaten.

→ REMASKE.FMT
Formatmaske zur Erfassung von Rechnungs-/ Zahlungsdaten.

→ REREAEND.CMD

Modul (Unterprogramm) zur Änderung bereits vorhandener Rechnungs-/ Zahlungsdaten.

Programmbeispiel Rechnungsverwaltung

REVERW

```

***** REVERW.CMD *****
*
*                               *
*           Demonstration dBASE II           *
*   Programm zur Rechnungsverwaltung         *
*   Markt & Technik, Softwareverlag         *
*                               *
*****
*
* Das Programm setzt sich aus dBASE II - Kommandos zusammen und übernimmt die
* Steuerung der Rechnungsverwaltung. Vom Bediener werden keine spezifischen Kenntnisse
* über dBASE II verlangt.
*
SET TALK OFF
* = Unterdrückung Kommandoausgabe auf Bildschirm
USE RECHADR
IF .NOT. FILE('READID.NDX')
    INDEX ON KUNDENNR TO READID
* Prüfung, ob Schlüsseldatei vorhanden, ggf. neu einrichten
ENDIF
USE
USE RECHNUNG
IF .NOT. FILE('REREID.NDX')
    INDEX ON RECHNR TO REREID
* Prüfung, ob Schlüsseldatei vorhanden, ggf. neu einrichten
ENDIF
USE
* = Anschluss der Datenbanken mit den Adressen und Rechnungsdaten (in diesem Beispiel
* vorgeben; könnte auch beim Bediener erfragt werden, wenn mehrere, getrennte
* Datenbestände mit diesem Programm verwaltet werden sollen).
DO WHILE T
* = Schleife für Auswahlmenü, solange keine Eingabe zur Beendigung des Programmes erfolgt
ERASE
?'  HINWEIS: Auf Möglichkeit zur Eingabe NEUER Adressen wurde in diesem Beispiel'
?'  verzichtet, da im Beispiel ADRESSEN bereits gezeigt. Natürlich können bei'
?'  Erweiterungen von dBASE II - Anwendungen vorhandene Verfahren übernommen'
?'  werden !'

```

?' ***** M E N U E zur Rechnungsverwaltung*****'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

?'

Folgende Funktionen stehen zu Ihrer Verfügung:'

0 : Beenden des Programms'

1 : Übernahme der Adressen aus Beispiel zur'
Adressenverwaltung und anschliessend Ein-'
gabe der (eindeutigen!) Kundennummer'

2 : Ändern von Adressen'

3 : Eingabe neuer Rechnungs/Zahlungsdaten'

4 : Ändern/Löschen von Rechnungs-/Zahlungsdaten'

5 : Auswertung Datenbestand Rechnungen/Zahlungen'

Geben Sie die Nr. der gewünschten Funktion ein,'
nachdem WAITING erscheint.'

WAIT TO ANTW

* = Antwortzeichen wird in Variable ANTW übernommen

IF ANTW = '0'

RETURN

ENDIF

IF ANTW = '1'

DO READUEB

* = Aufruf Unterprogramm zur Adressenübernahme

ENDIF

IF ANTW = '2'

DO READAEND

* = Aufruf Unterprogramm zum Eingeben/Ändern/Löschen von
* Adressen

ENDIF

IF ANTW = '3'

DO REEINGAB

* = Aufruf Unterprogramm zum Eingeben/Ändern/Löschen von
* Rechnungen

ENDIF

IF ANTW = '4'

DO REREAEND

ENDIF

IF ANTW = '5'

ERASE

SELECT SECONDARY

USE RECHADR INDEX READID

SELECT PRIMARY

USE RECHNUNG INDEX REREID

- * mit JOIN werden mit geringstem Aufwand die gewünschten
- * Daten aus den beiden Datenbanken mit den Adressen- und
- * Rechnungsdaten in eine dritte Datenbank (RECH.DBF)
- * übertragen

IF ANTWAUS = '1'

JOIN TO RECH FOR P.KUNDENNR = S.KUNDENNR
FIELD KUNDENNR, ; NAME, RECHNR, LIEFERNR,
RECHDATUM, BETRAG, IST, FEHL, ; BEMERK

ENDIF

IF ANTWAUS = '2'

JOIN TO RECH FOR P.KUNDENNR = S.KUNDENNR .AND.
FEHL > 0 ; FIELD KUNDENNR, NAME, RECHNR,
LIEFERNR, RECHDATUM,; BETRAG, IST, FEHL, BEMERK

ENDIF

USE RECH

REPORT FORM REREPORT TO PRINT PLAIN

- * REREPORT wurde im Dialog mittels REPORT erstellt

USE

SELECT SECONDARY

USE

SELECT PRIMARY

ENDIF

ENDIF

ENDDO
RETURN

Programmbeispiel Rechnungsverwaltung

READUEB

***** READUEB.CMD *****

*
 * Demonstration dBASE II *
 * Programm zur Übernahme von Adressen aus der Adressenverwaltung *
 * Markt & Technik, Softwareverlag *

* dBASE II - Kommandos und Parameter sind zur Verdeutlichung in Grossbuchstaben
 * angegeben.
 *

USE RECHNUNG INDEX REREID

GOTO BOTTOM

APPEND FROM ADRESSEN

* übernommene Adressen müssen in ADRESSEN.DBF gelöscht werden, sonst werden sie mit
 * jedem Aufruf dieser Funktion erneut übernommen und erscheinen dann mehrfach in
 * RECHNUNG.DBF (die Löschung übernommener Adressen wird hier nicht vorgesehen)

ERASE

?' Der im Beispiel zur Adressenverwaltung aufgebaute Adressenbestand wurde in die'

?' Datenbank für die Adressen zur Rechnungsverwaltung über? nommen.'

?

?' Es müssen noch die Kundennummern eingegeben werden (bei Übernahme "0" ,

?' gesetzt). Im Beispiel wird davon ausgegangen, dass zu jeder Adresse eine'

?' Kundennummer grösser "0" gehört. Daher wird aus diesem Programmteil erst in das

?' Eingangsmenü zurückgeschaltet, wenn ALLE Adressen diese Bedingung erfüllen (d.h.)

?' KEINE Möglichkeit vorgesehen, Programmteil vorher zu verlassen !).'

?

?' Nachfolgend werden nacheinander die Adressen mit Kundennr. = "0" zur Eingabe'

?' der Kundennummer bzw. Änderung der anderen Felder ausgegeben. Zur Fortsetzung'

?' geben Sie bitte RETURN ein nachdem'

?' WAITING erscheint.'

```

WAIT
GOTO TOP
SET FORMAT TO SCREEN
SET TALK OFF
DO WHILE # .NOT. 0
    FIND 0
    IF # > 0
        ERASE
        @ 3,5 SAY 'Kundennummer' ' GET KUNDENNR PICTURE '99999'
        @ 3,40 SAY '(nur Zifferneingabe erlaubt)'
        @ 6,5 SAY 'Name' ' GET NAME
        @ 7,5 SAY 'Abteilung/Ansprechpartner' ' GET ABTEIL
        @ 9,5 SAY 'Strasse' ' GET STRASSE
        @ 10,5 SAY 'PLZ/ORT' ' GET PLZ PICTURE 'X9999'
        @ 10,38 GET ORT
        @ 12,5 SAY 'Tel.' ' GET TELEFON
        @ 12,30 SAY 'Branche' ' GET BRANCHE
        ?
        ?
        ?
        ? 'Änderungen in der Adresse werden durch CTRL-W oder RETURN (dann'
        ? 'aber in jedem Feld) abgeschlossen'
    READ
    ENDIF
ENDDO
ERASE
?
?
?' Ergänzung fehlender Kundennummer abgeschlossen!'
?
?' Falls keine Aufforderung zur Angabe einer Kundennummer'
?' erfolgt ist, wurden KEINE Adressen aus Beispiel zu Adressen'
?' verwaltung übernommen, da Datenbank dort keine Daten'
?' enthält !'
?
?
?' Zur Fortsetzung bitte RETURN drücken, nachdem'
?' WAITING erscheint.'
WAIT
USE
RETURN

```

Programmbeispiel Rechnungsverwaltung

READAEND

```

***** READAEND.CMD *****
*                               *
*      Demonstration dBASE II   *
*      Rechnungsverwaltung: Programm zum Ändern/Löschen von Adressen   *
*      Markt & Technik, Softwareverlag                                *
*****
* Das Unterprogramm dient dem Ändern bzw. Löschen von Adressen in der Datenbank
ERASE
?
? '      *****Ändern / Löschen von Adressen *****'
?
? '      Mit dieser Funktion werden Adressen nach der Kundennummer'
? '      aus dem Datenbestand selektiert. Die gefundene Adresse '
? '      wird mit allen Feldern ausgegeben und kann dann ganz oder'
? '      teilweise geändert bzw. gelöscht werden'
?
?
? ' Zur Fortsetzung des Programms drücken Sie bitte RETURN, nachdem WAITING'
? ' erscheint'
?
?
WAIT
USE RECHADR INDEX READID
GOTO TOP
* = Ersten Satz in Datenbank aufsuchen
SET FORMAT TO SCREEN
SET TALK OFF
DO WHILE T
* = Schleife für Eingabe
STORE " " TO MKUNDENNR
ERASE
?
?
? ' Das Programm wird durch Eingabe einer "0" bei der Kundennummer beendet. '
?

```

```
ACCEPT'      Kundennummer der gesuchten Adresse ' TO MKUNDENNR
ERASE
IF MKUNDENNR = "0"
    RETURN
ENDIF
FIND &MKUNDENNR
IF # = 0
    * gesuchte Adresse nicht gefunden
    ERASE
    @ 5,10 SAY '?????????????????????????????????????????'
    @ 7,10 SAY ' Adresse NICHT gefunden !'
    @ 9,15 SAY 'Gesucht wurde nach Kundennummer:'
    @ 11,18 SAY MKUNDENNR
    @ 15,10 SAY '?????????????????????????????????????????'
    ?
    ?
    ?      'Zur Fortsetzung des Programms bitte RETURN drücken,'
    ?      'nachdem WAITING erscheint'
    ?
    ?
    ?
    WAIT
```

ELSE

ERASE

@ 1,15 SAY '!!'

@ 2,15 SAY ' Folgende Adresse wurde gefunden:'

@ 3,15 SAY '!!'

@ 5,5 SAY 'Kundennummer

' GET

KUNDENNR;

PICTURE '99999'

@ 6,5 SAY 'Name

' GET NAME

@ 7,5 SAY 'Abteilung/Ansprechpartner

' GET ABTEIL

@ 9,5 SAY 'Strasse

' GET STRASSE

@ 10,5 SAY 'PLZ/ORT

' GET PLZ PICTURE 'X9999'

@ 10,38 GET ORT

@ 12,5 SAY 'Tel. ' GET TELEFON

@ 12,31 SAY 'Branche' GET BRANCHE

?

? ' Geben Sie nach WAITING Ihre Wahl zur weiteren Bearbeitung ein:'

?

? ' 1 : Adresse ändern (Eingaben werden aus o.a. Feldern abge-'

? ' gefragt. Wird nur RETURN eingegeben, bleibt alter Wert'

? ' erhalten.)'

? ' 9 : Adresse löschen'

? ' sonst: neue Adresse suchen'

?

WAIT TO ANTW

IF ANTW = '1'

READ

ELSE

IF ANTW = '9'

DELETE

* = Adresse löschen

PACK

* = Entferne gelöschte Adresse aus Datei, nur zur Demonstration;

* normalerweise erst notwendig, wenn zahlreiche gelöschte Adressen

* vorliegen

ENDIF

ENDIF

ENDIF

ENDDO

RETURN

Programmbeispiel Rechnungsverwaltung

REEINGAB

```
***** REEINGAB.CMD *****
*
*           Demonstration dBASE II
*
*   Rechnungsverw.: Programm zur Eingabe von Rechnungs-/Zahlungsdaten
*               Markt & Technik, Softwareverlag
*
*****
*
* Das Unterprogramm dient der Eingabe von Rechnungs-/Zahlungsdaten
*
USE RECHNUNG INDEX REREID
DO WHILE T
  APPEND BLANK
  SET FORMAT TO REMASKE.FMT
  * Maske zur Eingabe von Rechnungs-/Zahlungsdaten
  READ
  IF KUNDENNR = 0
    DELETE
    PACK
    RETURN
  ENDIF
ENDDO
SET FORMAT TO SCREEN
* weitere Eingabeformate werden direkt auf Bildschirm angegeben
RETURN
```

Programmbeispiel Rechnungsverwaltung

REREAEND

```

***** REREAEND.CMD *****
*                               *
*      Demonstration dBASE II   *
*      Rechnungsverwaltung: Unterprogramm zum AENDERN/Löschen von Adressen *
*      Markt & Technik, Softwareverlag *
*****
*  Das Unterprogramm dient dem Ändern bzw. Löschen von Rechnungs-/Zahlungsdaten.
*
USE RECHNUNG INDEX REREID
ERASE
?
? '      *****Ändern / Löschen von Rechnungs-/Zahlungsdaten*****'
?
? '      Mit dieser Funktion werden Rechnungs-/Zahlungsdaten nach'
? '      der Rechnungsnummer herausgesucht. Alle Daten zu einer '
? '      Rechnungsnummer werden ausgegeben und können geändert'
? '      bzw. gelöscht werden'
?
? '  Zur Fortsetzung des Programms drücken Sie bitte RETURN, nachdem'
? '  WAITING erscheint'
?
?
WAIT
SET FORMAT TO SCREEN
SET TALK OFF
DO WHILE T
* = Schleife für Eingabe
ERASE
?
?
?
?
? '      Geben Sie die Rechnungsnummer ein (Eingabe mit RETURN absch.),'
? '      Eingabe "0" beendet das Programm.'
? '      *****'
?
STORE " " TO MRECHNR
ACCEPT '      gesuchte Rechnungsnummer      ' TO MRECHNR
IF MRECHNR = '0'
    USE
    RETURN
ENDIF
FIND &MRECHNR
IF # = 0
    * gesuchte Adresse nicht gefunden
    ERASE
    @ 5,10 SAY '?????????????????????????????????????????'
    @ 7,10 SAY 'Rechnungs-/Zahlungsdaten NICHT gefunden!'
    @ 9,15 SAY 'Gesucht wurde nach Rechnungsnummer:'
    @ 11,18 SAY MRECHNR
    @ 15,10 SAY '?????????????????????????????????????????'

```

```

?
?
?   'Zur Fortsetzung des Programms bitte RETURN drücken,'
?   'nachdem WAITING erscheint'
?
?
?
WAIT
ELSE
  ERASE
  SET FORMAT TO SCREEN
  @ 1,15 SAY '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
  @ 2,15 SAY ' Folgende Daten wurden gefunden:'
  @ 3,15 SAY '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!'
  @ 5,2 SAY 'Kunden-Nr.          ' GET KUNDENNR
  @ 5,40 SAY 'Liefer-Nr.          ' GET LIEFERNR
  @ 7,2 SAY 'Rechnungsdatum ' GET RECHDATUM
  @ 7,40 SAY 'Rechnungs-Nr.' GET RECHNR
  @ 9,2 SAY 'Rechnungsbetrag ' GET BETRAG
  @ 9,40 SAY 'bisher. Zahlung ' GET IST
  @ 10,2 SAY 'Bemerkungen          ' GET BEMERK
  ?
  ?' Geben Sie nach WAITING Ihre Wahl zur weiteren Bearbeitung ein:'
  ?
  ?'      1 : Daten ändern (Eingaben werden aus Feldern abgefragt.'
  ?'      Wird nur RETURN eingegeben, bleibt alter Wert erhalten.'
  ?'      9 : Datensatz löschen'
  ?'      sonst: neue Adresse suchen'
  ?
  WAIT TO ANTW
  IF ANTW = '1'
    READ
  ELSE
    IF ANTW = '9'
      DELETE
      * = Datensatz löschen
      PACK
      * = entfernen gelöschte Adresse aus Datei, nur zur
      *   Demonstration; normalerweise erst erforderlich, wenn
      *   zahlreiche gelöschte Sätze vorliegen
    ENDIF
  ENDIF
ENDIF
ENDDO
USE
RETURN

```


Programmbispiel Rechnungsverwaltung

REMASKE

```

***** REMASKE.FMT *****
*
*           Demonstration dBASE II
*
*   Rechnungsverwaltung: Maske zur Eingabe von Rechnungen
*
*           Markt & Technik, Softwareverlag
*****
*
* Hiermit wird das Format für die Eingabe von Rechnungs-/Zahlungswerten festgelegt.
*
* Das Programm beginnt hier:
*
@ 1,10 SAY 'HINWEISE zur Eingabe von Rechnungs-/Zahlungswerten'
@ 2,10 SAY ' = = = = = '
@ 3,10 SAY 'Eingabe zu einem Feld erfolgt zwischen den Doppelpunkten.'
@ 4,10 SAY 'Sie wird entweder mit Druecken von RETURN abgeschlossen'
@ 5,10 SAY 'oder wenn die letzte Stelle im Feld belegt wird. Sollen keine'
@ 6,10 SAY 'weiteren Daten erfasst werden, ist als Kundennummer "0"'
@ 7,10 SAY 'sowie anschliessend CTRL-W (gleichztg. CTRL und W) einzugeben.'
@ 8,10 SAY 'Die Formate sind wie folgt definiert und akzeptieren nur den'
@ 9,10 SAY 'angegebenen Typ bei der Eingabe:'
@ 10,12 SAY '* Kunden-Nr. : 5 Stellen, numerisch'
@ 11,12 SAY '* Liefer- bzw. Rechnungs-Nr. : 6 Stellen, numerisch'
@ 12,12 SAY '* Rechnungsdatum : 8 Stellen, alphanumerisch (fuer TT.MM.JJ)'
@ 13,12 SAY '* Rechnungsbetrag : 9 Stellen, numerisch (999999.99)'
@ 14,12 SAY '* bisher. Zahlung : 9 Stellen, numerisch (999999.99)'
@ 15,12 SAY '* Bemerkungen : 40 Stellen, alphanumerisch'
@ 16,10 SAY '***** Geben Sie nun die Daten ein : *****'
@ 18,2 SAY 'Kunden-Nr.      ' GET KUNDENNR
@ 18,40 SAY 'Liefer-Nr.     ' GET LIEFERNR
@ 20,2 SAY 'Rechnungsdatum   ' GET RECHDATUM
@ 20,40 SAY 'Rechnungs-Nr.   ' GET RECHNR
@ 21,2 SAY 'Rechnungsbetrag    ' GET BETRAG
@ 21,40 SAY 'bisher. Zahlung     ' GET IST
@ 22,2 SAY 'Bemerkungen         ' GET BEMERK

```

Programmbeispiel Adreßverwaltung

REREPORT

Rechnungs//Zahlungsübersicht

* Markt & Technik, Softwareverlag *

Die folgende Auflistung entspricht dem Inhalt der Datei REREPORT.FRM. Sie wird im Dialog von REPORT angelegt und bestimmt das Schema eines Berichts, der aus den Datenbanken für die Rechnungs-/Zahlungsübersicht erstellt wird. Die Angaben entsprechen den Antworten auf die Fragen in REPORT in dieser Reihenfolge.

M = 2, L = 66, W = 80

RECHNUNGS-/ZAHLUNGSUEBERSICHT

N

Y

Y

KUNDENNR

N

N

< Kundennr.

20, NAME

< N A M E

9, RECHNR

> Rechn.Nr.

N

8, RECHDATUM

> Datum

10, BETRAG

> BETRAG

Y

10, IST

> ZAHLUNG

Y

10, FEHL

> F E H L

Y

Stichwortverzeichnis

- ! (<variable/string>) - Konversion in Grossbuchstaben, 34, 158, 210
- ! (<Zeichenreihen-Ausdruck>) - Großbuchstabenfunktion, 34, 158, 210
- # (Datensatznummer), 34
- # (Nummernkreuz) - Datensatz-Nummer, 153
- \$ (Substring-Suchoperator) - Format: <teilsting> \$ <string>, 33, 90
- & (Makro-Aufruf), 34
- () Klammern, Zusammenfassung fuer Vorrangregeln, 33, 87
- * (gelöschter Datensatz), 34
- * (Joker-Zeichen), Verwendung, 66
- * (Multiplikation), 33, 85
- * - Abfrage der Löschmarkierung, 153, 211
- * <Text> - Kommentareinleitung in einem Programm ohne Anzeige, 38, 153, 311
- *** UNKNOWN COMMAND (unbekannter Befehl), 60
- + - (Addition), 33, 85
- + String-Verkettung durch einfaches Anhängen, 33
- (Subtraktion), 33, 85
- String-Verkettung, Leerzeichen werden nach rechts verschoben, 33
- . - "Prompt"-Punkt, dBASE wartet auf Eingabe von Befehlen, 20, 57
- .AND. (logisches UND), 33, 87
- .CMD (Befehls-Dateien), 127, 203, 205
- .COM (Programm-Dateien), 3
- .DBF (Datenbank-Dateien), 203
- .F. (logische Konstante = falsch), 92
- .FMT (Masken-Dateien), 203, 206
- .FRM (Format-Dateien), 121, 203, 205
- .MEM (Variablen-Dateien), 204, 205
- .NDX (Schlüssel-Dateien), 203, 206
- .NOT. (logische Verneinung) - einstelliger Operator, 33, 87, 88
- .OR. (logisches ODER), 33, 87
- .T. (logische Konstante = wahr), 92
- .TXT (Text-Dateien), 203, 205

- / (Division), 33, 85
- ; (Semikolon) bewirkt Verlängerung der Befehlszeile, 61
- < (kleiner als), 33, 61
- < > - spitze Klammern, umschließen symbolischen Begriff, 62
- <= (kleiner oder gleich), 33, 61
- <> (ungleich, nicht gleich), 33
- <aus> - Abkürzung für Ausdruck, 35
- <Ausdruck>, 228
- <Ausdrucksfolge>, 228
- <BACKSPACE>-Taste (oder <CTL>-H) - (ein Zeichen zurück), 9, 56
- <Befehlsfolge>, 228
- <Begrenzung>, 228
- <CAPS>-Taste - Feststellen nur von Großbuchstaben, 49
- <CONTROL>-Taste (CTRL oder ctl), 49
- <Dateiname>, 228
- <DELETE/RUBOUT>-Taste (<CTL>-"-") - löscht Zeichen vor Cursor, 56, 58
- <ESCAPE>-Taste (ESC), 49
- <Feld>, 228
- <Felderfolge>, 228
- <Formatdatei>, 229
- <Geltungsbereich>, 230
- <koord> - Abkürzung für Koordinaten, 35
- <n>, 23, 278
- <REPEAT>-Taste - Wiederholungs-Taste, 49
- <RETURN> im APPEND-Modus (Rückkehr zur dBASE II-Kommandoebene), 58
- <RETURN>-Taste, 49
- <RUBOUT>-Taste (oder <ctl>-"-") - löscht Zeichen links von Cursor, 56, 58
- <Schlüssel>, 229
- <Schlüsseldatei>, 229
- <SHIFT>-Taste - Groß-Buchstaben-Umschaltung, 49
- <str> - Abkürzung für String, 35
- <var> - Abkürzung für Variable, 35
- <Variable>, 229
- <Variablenfolge>, 230
- <zeichenreihe>, 228
- = (gleich), 33
- > (größer als) 33, 61
- >= (größer oder gleich) 33, 61
- ? (interaktiver Frage-Befehl), 35, 68, 142, 153
- ? <aus [liste]> - zeigt einen Ausdruck (oder eine Liste) an, 35, 68, 235
- ? [<Ausdrucksfolge>], 68, 235
- ?? [<Ausdrucksfolge>], 235

@ (<variable1/string1>, <variable2/string2>) - Teilstring-Suche, 34, 159
 @ (Paragraph-Zeichen), 138
 @ <Koordinaten> [SAY <Ausdruck> [USING <Maske>]] [GET <variable> ..., 35, 43, 44, 45, 142, 237
 @-Befehl - formatiert Ausgaben auf dem Bildschirm oder dem Drucker, 45, 141, 237
 @..SAY..GET, 141, 237
 @..SAY..GET..PICTURE..., 168, 237
 @..SAY..USING, 170, 237

Abbruch mit Speichern (ctl-W), 59
 Abbruch ohne Speichern (ctl-Q), 59
 Abfolge (von Befehlen), 127
 Abfrage, ob Datei existiert, 34
 ACCEPT [“<Text>”] to <Variable> - speichert Text in Variable, 35, 44, 137, 138
 ADAENDRG.CMD (Unterprogramm zum AENDERN/Löschen von Adressen), 387
 Addition (+), 33
 ADEINGAB.CMD (Unterprogramm zur Eingabe von Adressen), 386
 ADLABEL.FMT (Ausdruck von Adressaufklebern), 396
 ADLISTE.FMT - Unterprogramm zur Auflistung von Adressen, 394
 ADMASKE.FMT (Formatmaske zur Adressenerfassung), 397
 Adress-System, Dateistruktur, 50
 Adressaufkleber, ausdrucken, 396
 Adressen, ändern, 387
 Adressen, Aufkleber ausdrucken, 396
 Adressen, Auflistung (ADLISTE), 394
 Adressen, löschen, 387
 Adressen, Selektion nach bestimmten Kriterien, 390, 393
 Adressenänderung (Unterprogramm), 387
 Adressenerfassung (Unterprogramm ADEINGAB.CMD), 386
 Adressenerfassung, Formatmaske (ADMASKE.FMT), 397
 Adressengruppen suchen, 390, 393
 Adressenübernahme aus anderen Datenbanken (READUEB), 404
 Adressenverwaltung (ADVERW), 383
 Adressenverwaltung, Programmstruktur, 382
 ADSUCH.CMD (Suche nach bestimmten Adressen), 390
 ADSUCHOP.CMD (Suche nach bestimmten Adressen), 393
 Änderung bestimmter Sätze/Datenfelder (EDIT), 57, 278
 Änderungen, mehrfache in einer Datenbank - CHANGE, 35, 254

ALL, 230

Allgemeine Daten von dBASE II, 2

Alpha-Zeichen, 51

Anmerkung (REMARK), 152, 321

Anpassung von dBASE II auf Ihrem Rechner, 4

Anpassung, Einführung, 1

Anpassung, Fehler, 55

Anpassung, Modifizierung einer früheren, 5

Anwender-Programme, 48

Anwendung analysieren, 153

APPEND - Anfügen von Datensätzen am Ende, 35, 70, 114

APPEND BLANK, 35, 244

APPEND FROM <andere datei> - fügt Datensätze zu der Datei in USE, 41

APPEND FROM <dateiname> DELIMITED [WITH begrenzer]], 35, 104, 244

APPEND FROM <Dateiname> [FOR <Ausdruck>] [SDF], 35, 104, 244

append from neudat.txt sdf, 102

APPEND, Akzeptierung von Sonderzeichen als Feldgrenzen (' und '), 244

Arbeitskopie der Original-Systemdiskette, 3, 51

ASCENDING - aufsteigende Reihenfolge (SORT) (= Standard), 28, 39, 42, 112

Aufruf von Unterprogrammen (Prozeduren), 135

Aufsummieren mit COUNT und SUM, 122, 264, 357

auftrag, Struktur der Beispieldatei, 78

Ausblenden von (unwichtigen) Details (TOTAL), 123, 360

Ausdrücke, 207

Ausgabe von Daten mit DISPLAY, 64

AUSGABEN.DBF, Struktur der Beispiel-Datei, 77, 94

Austausch von dBASE-Dateien und CP/M-Dateien, 102

Auswahl und Entscheidung, 127

Auswahl zwischen Alternativen (IF ... ELSE), 130

Auswahlkriterien (Befehle), 61, 79

Auswahlkriterien, Gebrauch, 79

Autocode I - Programmgenerator, 152

Backup (Sicherheitskopie), 3

Bedien-Fehler, 146

Bedingungs-Ausdrücke bei Befehlen, 61, 77

Befehle, Aufbau von Befehlsdateien, 224

Befehle, Erfassen von Daten, 222

Befehle, Ergänzungen, 61

Befehle, Erweiterung mit Bedingungs-Ausdrücken, 61

Befehle, Gruppierung nach ihrer Wirkung, 222

Befehle, Steuerung von Geräten, 225

Befehle, ungültige, 60

Befehle, Verändern von Daten, 222

Befehlsdatei (- .CMD), 45

Befehlserweiterungen mit Auswahlkriterien (Beispiel LIST), 61

Befehlsübersicht, 235

Befehlszeile (max. Länge 254 Zeichen), 61, 136

- beispiel.cmd - Programm-Beispiel, 143
- Bereich (scope), 73, 103
- Berichts-Format (.FRM Dateien), 121
- Berichtsauswertung mit REPORT, 118
- Berichtsauswertung mit REPORT, Beispiel, 119
- Betriebssystem, 48
- Betriebssystem-Anpassung, 4
- Betriebssystem-Software, 49
- Bildschirm, vorbeistellen (LIST) unterbrechen/fortsetzen (<ctl>-S), 60
- Bildschirm-Formatierung, 168
- Bildschirmbreite, Einstellung, 14
- bildschirmorientierte Editierung, 4, 5, 54, 57
- BROWSE - Anzeige + Änderung mehrerer Sätze (1 Zeile pro Satz!), 43, 152, 252
- BROWSE, Fenster nach links (ctl-Z), 252
- BROWSE, Fenster nach rechts (ctl-B), 252

- CANCEL - Abbruch der Ausführung einer Programmdatei, 35, 253
- CASE-<Ausdrücke>, 46, 133, 277
- CHANGE - Änderung von Feldern, 35, 43, 254
- CHANGE FROM:/CHANGE TO: - (Beispiel für Eingabe), 61
- CHANGE [Geltungsbereich] FIELD <Felderfolge> [FOR <Ausdruck>], 35, 107, 254
- Character (Zeichen), 81
- character string (Zeichenkette), 81
- CHR (<zahl>) - Zahl in ASCII umwandeln, 34, 160, 208
- CLEAR [GETS] - Löschen der Speichervariablen, Dateien schliessen, 35, 42, 142, 152
- Code-Folgen, benutzerdefinierte, 7
- Codes, Eingabe in HEX oder dezimal, 8
- command line (Eingabezeile), 60
- Computer, Befehlssystem, 3
- Computer, Grundbegriffe, 3
- Computer-Typ, Einstellung bei der Installation, 4
- CONTINUE - setzt eine LOCATE-Anweisung fort, 35, 257
- Control-Funktionen auf der Tastatur im Überblick, 59
- Control-Zeichen (ctl-...), 57, 58, 59
- COPY TO <dateiname> - erzeugt (identische) Sicherheitskopie, 40, 258
- COPY TO <dateiname> STRUCTURE EXTENDED, 35, 100, 258
- COPY TO <Dateiname> [DELIMITED [WITH <Begrenzung>]], 35, 103, 258
- COPY TO <Dateiname> [FOR <Ausdruck>] [SDF] [STRUCTURE [EXTENDED]], 35, 258
- COPY TO <dateiname> [STRUCTURE] [FIELD Liste von Datenfeldern], 35, 99, 103, 258
- copy to sysdata sdf, 102
- COPY to temp, 97
- copy to temp delimited, 103

COPY [<Geltungsbereich>] TO <Dateiname> [FIELD <Felderfolge>], 35, 100, 258
 COUNT - weist Variable Wert eines Zählers zu, 35, 264
 COUNT [<Geltungsbereich>] [FOR <Ausdruck>] [TO <Variable>], 35, 122, 264
 CP/M-Dateien, 102
 CP/M-Dateien, Austausch mit dBASE-Dateien, 102
 CREATE - Erzeugt eine neue Datenbank, 35, 40, 50, 266
 CREATE <neue datei> FROM <Datenstrukturdatei>, 35, 40, 98, 266
 ctl (Control), 58, 226, 252
 ctl-A - zurück zum vorhergehenden Datenfeld, 58, 226, 252
 ctl-B (BROWSE)-verschiebt Fenster nach rechts, 226, 252
 ctl-C (MODIFY) zeigt folgende Datenfelder an, 59, 226
 ctl-C - speichert und geht zu nächstem Datensatz (APPEND und EDIT), 58, 59, 226, 252
 ctl-D - ein Zeichen vorwärts, 55, 58, 226, 252
 ctl-E - zum vorhergehenden Feld nach oben zurück, 55, 58, 226, 252
 ctl-F - Cursor nach unten zum nächsten Feld, 58, 226, 252
 ctl-G - löscht Zeichen unter dem Cursor, 58, 95, 226, 252
 ctl-H - Backspace, 59, 226, 252
 ctl-N (MODIFY) fügt ein neues Datenfeld ein, 58, 226
 ctl-O - Rückkehr zu Befehlsmodus ohne Änderung (Superbrain), 57, 72, 252
 ctl-P - schaltet Drucker ein oder aus, 58
 ctl-Q - Rückkehr zu Befehlsmodus ohne Änderung, 56, 57, 58, 59, 95, 226, 252
 ctl-R (APPEND,CREATE,INSERT) speichert + rückt vor, 58, 226, 252
 ctl-R - (EDIT/Browse) speichert aktuell und zeigt vorherigen an, 59, 226, 252
 ctl-S (LIST) - unterbricht/setzt fort Vorbeirollen auf Bildschirm, 60, 226, 252
 ctl-S - ein Zeichen zurück/nach links, 55, 57, 226, 252
 ctl-T - (MODIFY) Datenfeld löschen, 57, 226
 ctl-U - setzt/löscht Löschkmarkierung (*) - (DELETED), 57, 59, 226
 ctl-V - Umschalten Einfügen/Überschreiben (INSERT), 58, 226
 ctl-W - speichert aktuell und kehrt zu dBASE-Kommandoebene zurück, 57, 59, 72, 95, 226, 252
 ctl-X - zum nächsten Feld nach unten, 55, 58, 95, 226
 ctl-Y - löscht den Inhalt des aktuellen Datenfeldes, 58, 59, 95, 226
 ctl-Z (BROWSE)-verschiebt Fenster nach links, 226, 252
 Cursor, 49
 Cursor auf nächstes Datenfeld setzen (ctl-F), 58, 226
 Cursor auf nächstes Datenfeld setzen (ctl-X), 58, 226
 Cursor auf vorangeg. Datenfeld setzen (ctl-A), 58, 226
 Cursor auf vorangeg. Datenfeld setzen (ctl-E), 58, 226
 Cursor ein Zeichen nach links setzen (ctl-S), 58, 226
 Cursor ein Zeichen nach rechts setzen (ctl-D), 58, 226
 Cursor, X-Y-Adressierung, 4
 Cursor-Steuerung, Tastenkombinationen, 58, 226
 Cursorpositionierung, direkte, 9, 10

Darstellung auf dem Bildschirm, invertiert/normal, 10, 11
Darstellungsbereich (scope) von LIST, 62
DATE (Datums-Funktion), 209
Datei, 31, 47
Datei, Hintergrund-Datei, 148
Datei, primäre, 148
Datei, sekundäre, 148
Datei, Vordergrund-Datei, 148
Datei-Ende-Zeichen, 3
Datei-Funktion - FILE (<Zeichenreihe>), 209
Dateien umbenennen (RENAME), 39, 41, 152, 322
Dateiende-Funktion (EOF), 34, 208
Dateityp .CMD (CoMmAnD Files) - Befehls-Dateien, 32, 203, 205
Dateityp .DBF (DatenBank Files), dBASE-Datenbank-Dateien, 32, 203
Dateityp .FRM (Report FoRM Files) - Format-Dateien, 32, 203, 205
Dateityp .MEM (MEMory Files) - Variablen-Dateien, 32, 203, 204
Dateityp .NDX (INDeX Dateien) - Schlüssel-Dateien, 32, 203, 206
Dateityp .TXT (TeXT-(ASCII)-Dateien), 32, 203, 205
Dateitypen, 32, 203
Daten, ändern mit dem EDIT-Befehlen, 57
Daten, Eingabe, 54
Daten, erfassen, 222
Daten, Speicherung, 53
Daten, Veränderung, 222
Datenbank, 47
Datenbank erzeugen, 50
Datenbank Management System (DBMS), 25
Datenbank, Berichtsauswertung mit REPORT, 118
Datenbank, Grundlagen, 25
Datenbank, kopieren, 98
Datenbank, Organisation, 112
Datenbank-Managementsysteme, hierarchische, 27
Datenbank-Managementsysteme, relationale, 27
Datenbank-Programme, 48
Datenbank-Systeme, 27
Datenbankpflege mit DELETE, RECALL, PACK, 73
Datenbankstruktur, leere, Veränderung, 94
Datenfeld, Änderung mit REPLACE, 106
Datenfeld, Änderung mit CHANGE, 106
Datenfeld, Änderung mit CHANGE (Beispiele), 108
Datenfeld-Namen, 84

- Datenmanipulation, Erweiterung mit Funktionen, 157
- Datensätze, mischen aus zwei Dateien, 166, 362
- Datensatz (Record), 27, 28, 47, 51
- Datensatz aufsuchen mit GO, GOTO oder SKIP, 66
- Datensatz löschen (DELETE), 36, 41, 73, 270
- Datensatz, Darstellung auf dem Bildschirm, 54, 57
- Datensatz-Felder, Umbenennung, 104
- Datensatz-Nummer (#) - record number, 34, 54
- Datenstruktur, 47
- Datenstrukturen kopieren (COPY), 98
- Datentyp 'C' - (Zeichen, Character), 34
- Datentyp 'L' - (logisch), 34
- Datentyp 'N' - (numerisch), 34
- Datentypen, 29, 34
- Datums-Funktion, 209
- dBASE Einführung, 1
- dBASE II im Überblick, 375
- dBASE II, allgemeine Daten, 2
- dBASE II, Arbeiten mit dem Datenbanksystem, 378
- dBASE II, Einsatz, 377
- dBASE II, praktische Anwendung, 45
- dBASE II-Befehle, Abkürzung auf vier Buchstaben, 60
- dBASE II-Begriffe, Erläuterungen, 47
- dBASE II-Beispiele, Handhabung, 375
- dBASE II-Hauptbefehlsebene, 73
- dBASE II-Operatoren, Zusammenfassung, 33
- dBASE II-Testversion, 375
- dBASE-Befehle, Zusammenfassung, 222
- dBASE-Dateien, 203
- dBASE-Dateien, Austausch mit CP/M-Dateien, 102
- dBASE-Programme, 127
- dBASE-Programme schreiben, 127
- dBASE-Standard-Werte, 164, 345
- dBASE-Standard-Werte verändern (SET ...), 164, 345
- dBASE-Window - Programmgenerator, 152
- DBASE<return>, 50

DELETE - Löschen eines Datensatzes (Record) (#) 36 41 73 270

Dialog, fehlerkorrigierender, 5, 60
 DIR - (listet Inhaltsverzeichnis - Directory), 47
 Diskette, 47
 Disketten kopieren, 3
 DISPLAY - zeigt aktuellen Datensatz an, 60, 64
 DISPLAY <Ausdrucksfolge> [OFF], 64, 272
 DISPLAY FILES [ON <Laufwerk>] [LIKE <jokerzeichen>], 36, 42, 272
 DISPLAY MEMORY - Zeigt die Inhalte der Speicher-Variablen, 36, 272
 DISPLAY STRUCTURE - zeigt Datenstruktur der Datenbank in USE, 36, 41, 66, 101
 DISPLAY [<Geltungsbereich>] ... [<Ausdrucksfolge>] [OFF], 36, 272
 DISPLAY [<Geltungsbereich>] [FOR <Ausdruck>], 64, 272
 DISPLAY, Beispiele für die Ausgabe von Daten, 65
 Division (/), 33
 DO <dateiname> - Führt ein dBASE-Programm aus, 36, 45, 129, 275
 DO CASE <Bedingung>, 36, 45, 133, 276
 DO WHILE - ENDDO, Schachtelung, 133, 275
 DO WHILE <Ausdruck>, 36, 45, 275
 Dracken eines Formulars, 171
 Druckseiten, Formatierung, 170, 237
 Dummy-Prozeduren (Pseudo-Unterprogramme), 155

 EDIT - Bearbeiten von Daten in einer Datenbank, 36, 43, 57, 114
 EDIT [<n>], 36, 117, 278
 Editermöglichkeiten mit @ SAY ... GET ... PICTURE, 168
 Editierung, bildschirmorientierte, 4, 12, 13, 54
 Einführung in dBASE, 199
 Eingabe von Daten in ein laufendes Programm, 136
 Eingabe von Datensätzen mit APPEND und INSERT, 70
 Eingabe, Fehler, 56
 Eingabe, wahlweise [...], 35
 Eingabezeile (command line), 60
 Einträge, symbolische <.....>, 35
 EJECT - Führt einen Blattvorschub aus dem Drucker aus, 37, 281
 ELSE - Alternativer Ausführungspfad in einer IF-Anweisung, 37, 45, 291
 END OF FILE ENCOUNTERED (Ende der Datei erreicht), 117
 END OF LOCATE SCOPE, 117
 ENDCASE, 45, 133, 276
 ENDDO (Begrenzungszeichen in einer DO WHILE-Schleife), 37, 45, 134, 275
 Ende von Befehlsdateien (EOF), 34
 ENDIF (Begrenzungszeichen in einer IF-Anweisung), 37, 45, 291
 ENTER FILENAME:, 50
 Entscheidungen fällen mit IF ... ELSE, 130
 EOF (End of File) - Dateiende-Funktion, 34, 153, 208
 ERASE - löscht Bildschirm/Terminal-Bildschirm, 37, 61, 282
 ERROR MESSAGES - Fehlermeldungen, 371

- Fehler bei der Eingabe, 56
- fehlerkorrigierender Dialog, 5, 60
- Fehlermeldungen (ERROR MESSAGES), 371
- Feld in einem Datensatz (field), 26, 51
- Feld-Namen, 31
- Felder mit Daten hinzufügen, 99
- Felder mit Daten löschen, 99
- Feldinhalte verknüpfen, 33
- Feldlänge - Darstellung auf dem Bildschirm durch Doppelpunkte, 54, 55
- field - Feld in einem Datensatz, 26
- FILE (<"filename"/var/ausdruck>) - Abfrage, ob Datei existiert, 34, 160, 209
- File (Datei), 31, 47
- FIND <schlüssel> - sucht Datensatz in indizierter Datenbank, 37, 115
- FIND <Zeichenreihe> oder '<Zeichenreihe>' oder ..., 45, 283
- FOR <Ausdruck>, 230
- Formatdatei (- .FRM), 203, 205
- Formatierung von Druckseiten, 170, 237
- Formular drucken, 171
- Formularentwurf, 171
- Funktionen, 208
- Funktionen von dBASE II, Zusammenfassung, 34, 208
- Funktionen, INT-Funktion, 34, 210

- GET <Variable> [PICTURE <Maske>]], 139, 237
- gleich (=), 33
- GO, 36, 288
- GO [TO] BOTTOM - bringt sie zum letzten Datensatz, 37, 45, 66, 288
- GO [TO] TOP - bringt sie zum ersten Datensatz, 37, 45, 66, 288
- GO [TO] [RECORD] <n> - bringt sie zum Datensatz mit der Nr. <n>, 37, 45, 66, 288
- GOTO <Variable>, 289
- GOTO-Anweisung, 134
- größer als (>), 33
- größer gleich (>=), 33
- größer oder gleich (>=), 33
- Grossbuchstaben-/Kleinbuchstaben-Umwandlung, 210
- Grossbuchstaben-Funktion (!<Zeichenreihen-Ausdruck>), 210
- Grundregeln der dBASE-Kommandosprache, 230

- Handhabung der dBASE II-Beispiele, 375
- Handhabung der dBASE-Beispiele, 379
- Hardcopy (REPORT), 122
- Hardware-Anforderungen, 2
- hierarchische Datenbank-Management-Systeme, 26
- Hilfsdateien, 47

- IF <Ausdruck> (Bedingte Befehlsausführung), 37, 45, 291
- IF..ELSE..ENDIF, 130
- INDEX ON <schlüssel (variable/ausdruck)> TO <indexdatei-name>, 28, 37, 42, 114
- indexsequentieller Zugriff, 28
- indizieren (INDEX), 28
- Informationsaustausch zwischen Haupt- und Unterprogrammen, 155
- Inhaltsverzeichnis (Directory) anzeigen - (DIR), 47
- INPUT - weist Variable Wert des Ausdrucks zu, 37, 136, 137, 296
- INPUT ["<Text>"] TO <Variable> - speichert "Eingabe" in Variable, 37, 44, 136, 137
- INSERT [BEFORE] [BLANK], 38, 43, 298
- install, 4
- Installation, Einführung, 1
- INT(<variable/ausdruck>) - Integer-Funktion, 34, 161, 210
- Integer in String verwandeln (STR-Funktion), 34

- JOIN TO <Dateiname> FOR <ausdruck> [FIELD <liste>], 38, 167, 302
- Joker-Zeichen (*), 66
- key (Schlüssel), 112
- Klammern, eckige ([]), 2, 62
- Klammern, runde (auch Zusammenfassung für Vorrangregelung), 33
- Klammern, spitze (< >), 2, 62
- Klammerung von Operationen (), 33
- kleiner als (<), 33
- kleiner gleich (<=), 33
- kleiner oder gleich (<=), 33
- Kommentare in Befehlsdateien (REMARK), 38, 152, 321
- Konstante, 80, 207
- Koordinaten (@-Befehl), 138
- Koordinateneingabe, 138
- Koordinatenposition, 138

- Längen-Funktion (LEN (<variable/string>)) - Länge eines Strings, 34, 161, 210
- Laufwerk, eingerastetes, 4
- Leerstellen, nachfolgende am Ende entfernen (TRIM), 34
- Leerzeichen (space) - " " oder ' ', 9, 49
- LEN (<variable/string>) - Länge eines Strings, 34, 161, 210
- Linien zeichnen, 185
- LIST - zeigt alle Datensätze der bearbeiteten Datei an, 61
- LIST FILES [ON <Laufwerk>] [LIKE <Dateimaske>], 63, 305
- LIST MEMORY, 44, 305
- LIST STRUCTURE, 41, 305
- LIST [<Ausdrucksfolge>] [OFF], 38, 62, 305
- LIST [<Geltungsbereich>] [FOR <Ausdruck>], 62, 305
- LIST, Beispiele, 63

Literale - "selbsterklärende" Wörter, 80
 LOCATE NEXT <anzahl>, 117
 LOCATE [<Geltungsbereich>] [FOR <Ausdruck>] - sucht Datensatz, 38, 45, 117, 306
 LOCATE, Beispiele, 118
 Löschen ganzer Dateien - (DELETE FILE <dateiname>), 36, 41, 270
 Löschen von Feldern mit Daten (ctl-T oder ctl-Y), 58
 löschen, Zeichen links vom Cursor (<DELETE> oder ctl-"), 58
 löschen, Zeichen unter Cursor (ctl-G), 58
 Löschmarkierung "*", 34
 Löschmarkierung "*" entfernen (RECALL), 38, 43, 317
 Löschmarkierung "*" setzen/löschen (ctl-U), 58
 Löschmarkierung von Sätzen (DELETE), 58
 logische Verneinung (.NOT.), 33
 logisches "oder" (.OR.), 33
 logisches "und" (.AND.), 33
 LOOP - Abbrechen in einer "DO WHILE"-Schleife, 38, 134, 308

 Makro-Befehle, 5
 Makro-Ersetzungen (& <Name>), 220
 Makroaufruf (&), 34
 Makros, Anzeige, 6
 Master-Kopie, 3
 Mehrfachauswahlen mit CASE, 132
 Mehrfachauswahlen mit IF ... ELSE ... IF ... ELSE ..., 131
 MODIFY COMMAND [<dateiname>], 38, 43, 129, 152
 MODIFY COMMAND, Einschränkungen, 309
 MODIFY STRUCTURE - Änderung der Datenbankstruktur, 38, 41, 95, 309
 modify structure, Beispiel, 100
 Multiplikation (*), 33
 NEXT <n>, 230
 nicht gleich (<>), 33
 NOTE <Text> - Kommentar-Einleitung in einem Programm ohne Anzeige, 38, 311
 Nummern-Kreuz (#) - Zeichen für Datensatz-Nummer, 34
 Operationen, 33, 216
 Operationen auf Zeichenreihen, 33, 218
 Operationen, arithmetische Operationen, 33, 219
 Operationen, logische Operationen, 33, 220
 Operationen, Reihenfolge von Operationen, 219
 Operationen, Vorrangregeln, 33, 219
 Operatoren, 33, 216
 Operatoren, arithmetische, 33, 87, 217
 Operatoren, logische, 33, 87, 218
 Operatoren, logische Substring-Suche (\$), 33, 218
 Operatoren, relationale (vergleichende), 33, 217
 Operatoren, String-Operatoren (Zeichenketten als Ergebnis), 33, 218
 Operatoren, vergleichende (relationale), 33, 217
 Organisation von Datenbanken, 112
 Originaldiskette, 2

- PACK - Entfernt alle als gelöscht markierte Datensätze endgültig, 38, 43, 74, 312
- Personal Computer unter CP/M, Datenbanksystem, 376
- PIP (CP/M-Befehl), 2
- PIP, Optionen, 2
- Probleme mit kurzen Feldern, 192
- Probleme mit langen Namen, 192
- Programm, Ablauflogik, 134
- Programm, Strukturierung, 25, 153
- Programm-Beispiel "zeigaus-cmd", 151
- Programm-Beispiel "zeignam-cmd", 150
- Programm-Datei (command file .cmd), 135, 205
- Programmbeispiel "beispiel.cmd", 143
- Programmbeispiel "versuch.cmd", 141
- Programme schreiben, 127
- Programmeingabe, 139
- Programmwurf, 153
- Programmierung, strukturierte, 153
- Programmierung, übliche, 25
- Programmstruktur Adressenverwaltung, 382
- Programmstruktur Rechnungsverwaltung, 399
- Programmstrukturen - Übersicht, 127
- Prompt-Punkt (.) - dBASE wartet auf Eingabe, 20, 57
- Prozeduren (Unterprogramme), 127, 135
- Pseudo-Unterprogramme (Dummy-Prozeduren), 155
- QUICKCODE, Programmgenerator, 152
- QUIT - schließt alle Dateien und beendet Arbeit mit dBASE-System, 42, 50
- QUIT [TO <liste von CP/M-Befehlen oder .COM-Dateien>], 38

- READ - liest Eintragungen in Bildschirmmaske, 38, 57
- READAEND.CMD (Unterprogramm zur Änderung von Adressen), 406
- READUEB.CMD (Unterprogramm für Übernahme der Adressdaten), 404
- RECALL - macht die Löschmarkierung von Datensätzen rückgängig, 38, 43
- RECALL [<Geltungsbereich>] [FOR <Ausdruck>], 38, 73, 317
- Rechnungsdaten, Änderung (REREAEND.CMD), 410
- Rechnungsdaten, Erfassung (REEINGAB.CMD), 409
- Rechnungsübersicht (REREPORT), 413
- Rechnungsverwaltung (Hauptprogramm REVERW.CMD), 400
- RECORD (Datensatz), 27, 28, 47, 51
- RECORD <n> - Eintrag, 230
- record numer (Datensatz-Nummer), 54
- relationale Datenbank Managementsysteme, 27
- relationale Operatoren, 33

RELEASE [<Variable> [,liste]] oder [ALL]ü- deaktiviert Variablen, 38, 320
 REMARK <Text> (Ausgabe eines Kommentars auf dem Bildschirm), 38, 320
 REMASKE.FMT (Formatmaske zur Erfassung von Rechnungsdaten), 412
 RENAME <alter Dateiname> TO <neuer Dateiname>, 39, 41, 152, 322
 REPLACE [bereich] <feld> WITH <Ausdruck> [, <feld> WITH <aus>...], 39, 43, 323
 REPORT - Erzeugt Auswertungen der Datenbank-Informationen, 39, 118, 327
 REPORT [bereich] [FORM <Formatdatei>] [TO PRINT] [FOR <aus>], 39, 45, 118, 327
 REPORT, Berichtsauswertung - Beispiel, 120
 REREAEND.CMD (Unterprogramm zur Änderung von Rechnungsdaten), 410
 REREPORT.FRM (Schema für Ausgabe eines Rechnungsberichts), 413
 RESET - teilt CP/M mit, daß ev.eine Diskette ausgetauscht wurde, 4, 39, 152, 336
 RESTORE FROM <Dateiname> - Liest Variable in Arbeitsspeicher, 39, 44, 337
 RETURN - beendet Ausführung der Befehlsdatei, 2, 39, 136, 339
 REVERW.CMD (Hauptprogramm Rechnungsverwaltung), 400

Satznummer-Funktion (#), 211

SAVE TO <Dateiname> - schreibt Speichervariable in die Datei, 39, 340

Schlüssel (key), 28, 112

Schlüssel-Dateien (- .NDX), 32

Schnittstellen dBASE - andere Programme, 221

schrittweise Verfeinerung von Programmen (Top-Down), 154

scope (Bereich), 73, 106

SDF (Standard Daten Format), 105

Seitenmodus, 226

SELECT - Aktivierung von zwei Dateien gleichzeitig, 39, 342

SELECT PRIMARY - Bearbeitung der ersten (PRIMARY) Datei, 37, 148, 342

SELECT SECONDARY - Bearbeiten der zweiten (SECONDARY)-Datei, 39, 148, 342

SET - erlaubt Einstellung von Parametern, 164, 345

SET <Parameter2> TO <Option> - setzt Parameter auf Wert/Option <..>, 164, 345

SET <Parameter> [ON] oder [OFF] - dBASE-Parametern ein/aus, 39, 164, 345

SET ALTERNATE ON/OFF, 164, 345, 349

SET ALTERNATE TO <dateiname> - Zusatzprotokoll von Anzeigen, 164, 349

SET BELL ON/OFF - Signal bei Eingabefehlern, 164, 346

SET CARRY ON/OFF - Übernahme des vorherigen Datensatzes, 164, 346

SET COLON ON/OFF - Feldbegrenzung durch : ein/aus, 165, 346

SET CONFIRM ON/OFF, 165, 347

SET CONSOLE ON/OFF - Bildschirmanzeige ein/aus, 165, 346

SET DATE TO mm/tt/jj (angelsächsische Schreibw.), 349

SET DEBUG ON/OFF - Druckerausgabe von Anzeigen, 165, 347

SET DEFAULT TO <Laufwerk>, 348

SET ECHO - Bildschirmausgabe von Befehlen bei DO, 165, 345

SET EJECT OFF - Unterdrückung des Seitenvorschubs, 118, 165, 347

SET EJECT ON/OFF - Seitenvorschub bei REPORT ein/aus, 165

SET ESCAPE ON/OFF - ESCAPE-Taste ein-/ausschalten, 165, 346

SET EXACT ON/OFF, 165, 347

SET FORMAT TO <Maskendatei>, 348
 SET FORMAT TO PRINT, 165, 170, 348
 SET FORMAT TO SCREEN, 165, 348
 SET HEADING TO zeichenkette - Austausch der Überschrift/Kopfzeile, 122, 165, 348
 SET INDEX TO - gibt aktivierten Schlüssel frei, 350
 SET INTENSITY ON/OFF, 138, 165, 347
 SET LINKAGE ON/OFF, 166, 346
 SET MARGIN TO n, 166, 350
 SET PRINT ON/OFF - Protokollierung auf Drucker/aus, 118, 166, 346
 SET RAW ON/OFF - Einfüge/Unterdrücke Leerzeichen, 166, 347
 SET SCREEN ON/OFF - Seitenmodus ein-/ausschalten, 166, 346
 SET STEP ON (OFF) (keine) Unterechnung nach Befehl, 166, 345
 SET TALK ON/OFF - Ergebnisanzeige bei Befehlen, 166, 346
 Sicherheitskopie (BACKUP) anfertigen, 2
 SKIP [+/-] <Ausdruck/Nummer> - springt entsprechend vor-/rückwärts, 39, 45
 Software-Anforderungen, 10
 SORT ON <SchlüsselFeld> TO <neue datei> [DESCENDING], 28, 39, 67, 110
 sortieren, nach Schlüssel, 110, 112
 Sortierung von Daten, 28, 110, 112
 Speicherung von Daten, 53
 Speichervariable, 82
 Steuerzeichen, 201
 STORE <Ausdruck> TO <Variable> - legt Wert in eine Variable ab, 39, 82, 356
 store trim (<variable>) to <variable>, 93
 STR-Funktion - Integer in String verwandeln, 34, 162, 212
 String (Zeichenkette), Länge: max. 254 Zeichen, 81
 STRING-Funktion - STR(<numerischer Ausdruck>, <Länge, [<Dezimal>]), 212
 String-Konstanten (Texte), 81
 String-Operatoren, 33, 93
 String-Verkettung + (exakt), 33, 93
 String-Verkettung - (Leerzeichen ans Ende verschoben), 33, 93
 substring, 80
 Substring-Such-Operator, 33
 Subtraktion (-), 33
 SUM - Bildet Gesamtsummen der Felder in einer Datenbank, 39, 122, 357
 SUM [bereich] <feld [, liste]> [TO <var [, liste]> [FOR <Beding.>]], 39, 122, 357
 SUM-Befehl, Ausdrücke: max. 5, 122, 357
 Summen-Übersicht, 122, 265, 357
 Summieren von Daten (TOTAL), 123, 360
 Symbol-Definitionen, 228
 System-Informationen gewinnen mit LIST, 63
 Systemanforderungen, 202
 Systemdiskette, 2

Tastatur, 49

Tastatur-Befehle, bildschirmorientierte, 58

Tastenkombination, BROWSE-Befehl, 226, 252

Tastenkombination, MODIFY-Befehl, 226

Tastenkombinationen zur Cursor-Steuerung, 226

Tastenkombinationen, (APPEND, CREATE, INSERT), 226

Tastenkombinationen, EDIT-Befehl, 226

Teilaufgaben programmieren, 154

Teilstring herauskopieren - \$-Funktion, 34, 212

Teilstring-Suche - @ (<variable1/string1>, <variable2/string2>), 34, 159, 212

Terminal ohne direkte Cursoradressierung, 54

Terminal, Standardwerte modifizieren, 7

Terminalanpassung, individuelle, 7

Terminalroutinen modifizieren, 15, 16

Terminaltypen, Liste, 5

TEST-Funktion, 215

Testversion von dBASE II, 375

Text, invertierte Darstellung auf dem Bildschirm, 7

TOP-Down, schrittweise Verfeinerung, 154

TOTAL ON <Schlüssel> TO <Datenbank> [feld [, liste]> [FOR <aus>], 40, 123, 360

Totalübersichten, 53, 122, 360

TRIM-Funktion - entfernt nachfolgende Leerzeichen am Ende, 34, 93, 162, 216

Typ-Funktion, 34, 162, 215

TYPE (<Ausdruck>) - Datentyp, 34, 162, 215

typografische Regeln in diesem Handbuch, 1

Übertragen von Daten mit PIP, 3

Umwandlung in Integer (INT), 4

Umwandlung von String in Integer (VAL), 34

ungleich "<>", 33

Unterprogramm (Prozedur, Modul), 127

Unterprogramm, Aufruf (Prozeduren), 135

Unterprogramm-Dateien, 155

UPDATE FROM <andere datei> ON <schlüssel>, 40, 166, 362

UPDATE FROM <dateiname> ON <schlüssel> [REPLACE <feld [,liste]>], 40, 166, 362

UPDATE FROM <dateiname> ON <schlüssel> [ADD <feld [, liste]>], 40, 166, 362

USE - schließt alle zuvor eröffneten Datenbanken, 40, 365

USE <dateiname> [INDEX <dateiname>] - eröffnet [indizierte] Datei, 40, 41, 56, 57, 362

VAL (<variable/string/teilstring>) - Umwandlung String in Integer, 34, 163, 213
 Variable, 82, 207
 Variable an jeder beliebigen Bildschirmposition ausgeben, 138
 Variable, einrichten, 222
 Variable, temporäre ändern, 222
 Variablen-Namen, 85
 vergleichende Operatoren, 33
 Verneinung, logische (.NOT.), 33
 Verschachtelung von Unterprogrammen (Prozeduren), 135
 Video-Terminal, Anpassung, 7

Wagenrücklauf, 49
 Wahl zwischen zwei Alternativen (IF ... ELSE), 131
 WAIT - unterbricht Programmausführung, bis eine Taste gedrückt wird, 40, 136
 WAIT TO <speicher-variable> - legt Zeichen in Speichervariable ab, 40, 136
 Was ist ein Datenbanksystem, 375
 WHILE <Ausdruck>, 230
 Wiederholen eines Vorgangs (Schleifen), 127, 133
 Wiederholungs-Taste <REPEAT>, 49
 wild cards (Joker-Zeichen, *), 66
 X-Y-Adressierung, 4
 Y - CHANGE/MODIFY, 5, 54

Zählen, automatisches mit COUNT und SUM, 122, 265, 357
 Zahl (number), 51
 Zahl in ASCII umwandeln - CHR (<zahl>), 34
 Zahl, Umwandlung in Integer (INT), 34
 Zeichenkette (String), Länge: max. 254 Zeichen, 80, 86
 Zeichenreihe-Suchfunktion (@), 159, 214
 zeigaus.cmd - Programm-Beispiel, 151
 zeignam.cmd - Programmbeispiel, 150
 Zeilenvorschub, 49
 ZIP - Beginnen eines neuen Entwurfs, 189
 ZIP - der Bildschirm- und Druck-Maskengenerator, 175
 ZIP - Überblick, 181
 ZIP Entwurf in einer Datei abspeichern, 187
 ZIP Hilfstafel, 186
 ZIP Standardwerte verändern, 186
 ZIP Symbole, 194
 ZIP, Anpassung an den Rechner/Terminal, 176
 ZIP-Befehle, 193
 ZIP-Befehle, Zusammenfassung, 193
 Zusammenfassung der Funktionen von dBASE II, 34
 ZWEIDAT-CMD, Display-Unterprogramme, 196