Markt&Technik

Schneider Joyce-Software

dBASE™ II

ASHTON·TATE

für den Schneider Joyce PCW 8256

3" Schneider-Format

dBASE II™

ASHTON-TATE

Das relationale Datenbanksystem für Ihren Schneider Joyce PCW 8256!

Allgemeine Daten

Datensätze pro Datenbank
maximal 65535
Zeichen pro Datensatz
maximal 1000
Felder pro Datensatz
maximal 32
Zeichen pro Feld
maximal 254
Numerische Genauigkeit
10 Stellen

Markt&Technik
Schneider Joyce-Software
Hans-Pinsel-Straße 2
8013 Haar

Spezielle Merkmale

- Eignet sich zur Lösung aller kaufmännischer Anwendungsprobleme wie Lagerverwaltung, Fakturierung, Betriebsabrechnung etc.
- Bietet alle Möglichkeiten der Datei- und Datenbehandlung wie Erfassen, Ändern, Einfügen, Mischen und Suchen.
- Sequentieller oder wahlfreier Zugriff nach frei wählbaren/kombinierbaren Kriterien.
- Reportgenerator zur Berichterstellung; Schema und Inhalt wird im Dialog definiert.
- Integrierte Programmiersprache zur Erstellung von Befehlsdateien bis hin zur Menüsteuerung.

dBASE II erhalten Sie mit

- Original-Handbuch von Ashton-Tate
- Beschreibung der Joyce-spezifischen Version

dBASE II ist lieferbar für den Schneider Joyce PCW 8256 auf 3-Zoll-Disketten.

dBASE II ist unter CP/M Plus für den Schneider Joyce bereits fertig installiert und unterstützt die Druckfunktionen des mitgelieferten Matrixdruckers.

Markt&Technik

Schneider Joyce-Software

dBASE™ II

ASHTON-TATE

für den Schneider Joyce PCW 8256

Markt & Technik Verlag Aktiengesellschaft · Hans-Pinsel-Straße 2 · 8013 Haar

dBASE II für den Schneider Joyce PCW 8256

Lieber Leser,

herzlichen Glückwunsch zum Kauf von dBASE II. Sie haben damit ein Datenbankprogramm erworben, das zur internationalen Spitzenklasse gehört.

Bitte lesen Sie die folgenden Seiten aufmerksam durch. Sie beschreiben, wie Sie sich eine Arbeitskopie von dBASE II erstellen können (arbeiten Sie bitte niemals mit Ihrer Original-Diskette) und welche Erweiterungen bei dieser Joyce-Version zur Erhöhung des Komforts vorgenommen wurden.

dBASE II ist für Ihren Schneider-Computer bereits **fertig** angepaßt. Alles was in Ihrem dBASE II Benutzer-Handbuch zur Installation gesagt wird, können Sie zunächst einmal ignorieren. Diese Informationen werden für Sie erst interessant, wenn Sie die vorgegebene Funktionstastenbelegung verändern wollen. Sie sollten sich daran jedoch erst wagen, wenn Sie mit dBASE II gut vertraut sind.

Erforderliche Hardware

dBASE II ist lauffähig auf dem Schneider Joyce Personal-Computer.

Der Einsatz mit einem Diskettenlaufwerk ist problemlos möglich. Am besten legen Sie sich das gesamte dBASE II-Programm mit allen Dateien auf die RAM-Floppy (Laufwerkskennung M) Ihres Schneider Joyce-Computers. Auf diese Weise können Sie den gesamten Speicherplatz nutzen und schnellere Programmlaufzeiten erzielen.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
89 88 87 86

Einbandgestaltung: Grafikdesign Heinz Rauner
Druck: Schoder, Gersthofen
Printed in Germany

dBASE II erhalten Sie auf einer beidseitig bespielten 3-Zoll-Diskette.

Seite A - Hauptprogramm mit den Dateien:

dbase.com, dbaseovr.com, dbasemsg.txt, db.sub, db.key

Seite B - Installationsprogramme mit den Dateien:

zipin.com, zscrn.ovl, zip.com, dgen.ovl, install.com, temp.dbf, dBASE II-Beispiele

Drucker

dBASE II unterstützt jeden zum Schneider Joyce kompatiblen Drucker.

Anlegen einer Arbeitsdiskette

Bevor Sie mit dBASE II arbeiten, fertigen Sie sich unbedingt erst eine Kopie Ihrer Original-dBASE II-Diskette an. Arbeiten Sie bitte niemals mit Ihrer Original-Diskette, damit Sie sich bei Beschädigung Ihrer Diskette eine neue Arbeitsdiskette erstellen können. Die folgenden Seiten beschreiben, wie Sie sich Schritt für Schritt eine Arbeitsdiskette anlegen können.

Achten Sie darauf, daß Ihre dBASE II-Original-Diskette vor Überschreiben geschützt ist!

1. Laden von CP/M

Nehmen Sie Ihre CP/M Plus-Original-Diskette (Seite 2) von Digital-Research und legen Sie sie in das Laufwerk. Schalten Sie Ihren Schneider Joyce-Computer wie gewohnt ein. CP/M wird nun automatisch geladen. Nach etwa 5 Sekunden erscheint die Systemmeldung:

```
CP/M Plus .....................
A>
```

Der Cursor steht hinter dem Prompt A>

2. Kopieren der dBASE II-Original-Diskette

Legen Sie eine leere 3-Zoll-Diskette bereit und geben Sie dann ein:

```
diskit <RETURN>
```

DISCKIT meldet sich mit:

```
f6/f5          Kopieren
f4/f3          Formatieren
f2/f1          Prüfen
EXIT           Programm verlassen
```

Wählen Sie mit der Funktionstaste f6/f5 das Kopier-Programm aus und es erscheint die Frage:

```
J              Kopieren CF2 Disketten

Menü verlassen mit beliebiger Taste
```

Entnehmen Sie nun die CP/M-Diskette und legen Sie die dBASE II-Original-Diskette (Seite A) in das Laufwerk ein. Den Kopiervorgang

```
f6/f5          Kopieren
f4/f3          Formatieren
f2/f1          Prüfen
EXIT           Programm verlassen
```

Mit der Funktionstaste "EXIT" kehren Sie in das CP/M Plus-Betriebssystem zurück. Es erscheint der Prompt A>.

3. Übertragung des Betriebssystems auf Ihre Leerdiskette

Wollen Sie dBASE II auch von der Arbeitsdiskette starten, müssen Sie von Ihrer Original-CP/M-Plus-Diskette die Datei J12DCPM3.EMS auf Ihre Arbeitsdiskette kopieren. Legen Sie dazu Ihre CP/M-Diskette (Seite 2) in das Laufwerk und laden Sie das Kopierprogramm PIP.

```
pip <RETURN>
```

PIP wird geladen und es erscheint die Meldung:

```
CP/M 3 PIP VERSION 3.0
*
```

Der Cursor steht hinter dem Sternchen. Geben Sie nun ein:

```
b:=a:j12dcpm3.ems <RETURN>
```

Die Datei J12DCPM3.EMS wird in den Arbeitsspeicher geladen. Nach einigen Sekunden erscheint folgende Meldung:

```
Bitte Diskette für B: einlegen, dann irgendeine Taste drücken
```

Da Ihr System mit nur einem Laufwerk ausgestattet ist, arbeitet das Betriebssystem mit einem "imaginären" Laufwerk, das die Bezeichnung B hat. Legen Sie Ihre Arbeitsdiskette in das Laufwerk A und geben Sie RETURN ein. Das Programm schreibt nun die Datei auf die Arbeitsdiskette. PIP meldet sich dann wieder mit einem Sternchen.

4. Übertragung weiterer Betriebssystem-Dateien

Für die Einstellung der Tastaturbelegung und der Funktionstasten werden weitere Dateien von der CP/M Plus-Diskette (Seite 2) benötigt. Legen Sie diese in das Laufwerk ein und schreiben Sie:

```
b:=a:setkeys.com <RETURN>
```

Am unteren Bildschirmrand erscheint:

```
Bitte Diskette für A: einlegen, dann irgendeine Taste drücken
```


starten Sie mit der Taste "J". Sie sehen wie das Programm die kopierten Spuren mitzählt und am Ende folgende Meldung ausgibt:

```
Diskette hat CF2 Format
Bitte Diskette zum Schreiben einlegen
Dann eine beliebige Taste drücken
```

Kommen Sie der Aufforderung nach und wechseln Sie die dBASE II-Diskette mit der bereitgestellten 3-Zoll-Leerdiskette. Sie sehen:

```
Diskette ist nicht formatiert (oder fehlerhaft)
Diskette wird beim Kopieren formatiert
Diskette erhält CF2-Format
```

Das Programm schreibt nun die im Arbeitsspeicher befindlichen Spuren auf Ihre Arbeitsdiskette. Nach kurzer Zeit erscheint die Meldung:

```
Diskette zum Lesen einschieben
Dann eine beliebige Taste drücken
```

Legen Sie die dBASE II-Original-Diskette (Seite A) abermals in das Laufwerk und drücken Sie z.B. RETURN.

Es werden nun die restlichen Spuren in den Arbeitsspeicher gelesen. Sobald dies beendet ist, erscheint wieder die Aufforderung:

```
Diskette zum Schreiben einschieben
Dann eine beliebige Taste drücken
```

Nehmen Sie die Arbeitsdiskette zur Hand, legen Sie sie in das Laufwerk und drücken Sie RETURN. Es werden nun die restlichen dBase II-Dateien übertragen. Das Ende des Kopiervorgangs zeigt an:

```
Diskette ist kopiert
Nehmen Sie die Diskette aus dem Laufwerk
Dann eine beliebige Taste drücken
```

Kommen Sie der Aufforderung nach und drücken Sie irgendeine Taste. Es erscheint:

```
J       Kopieren weitere CF2 Diskette

Menü verlassen mit beliebiger Taste
```

Mit dem Drücken einer beliebigen Taste z.B. RETURN erscheint wieder das Eröffnungsmenü von DISCKIT:

Drücken Sie nun RETURN. Nach kurzer Zeit erscheint die Aufforderung die Arbeitsdiskette für die Übertragung der Datei einzulegen.

```
Bitte Diskette für B: einlegen, dann irgendeine Taste drücken
```

Das Ende des Kopiervorgangs wird mit dem Stern (*) angezeigt.

Die weitere Datei, die Sie mit dem PIP-Befehl noch kopieren müssen, heißt SUBMIT.COM. Verfahren Sie nach obigen Schema. Legen Sie dazu immer zuerst die CP/M-Diskette (Seite 2) in das Laufwerk und wechseln Sie bei der Aufforderung "Bitte Diskette für B einlegen, dann irgendeine Taste drücken" die CP/M-Diskette mit der Arbeitsdiskette. Die genaue Eingabe lautet:

```
b:=a:submit.com <RETURN>
```

Haben Sie alle Dateien kopiert, erhalten Sie wieder den Stern (*), der das Ende des Kopiervorgangs anzeigt. Entnehmen Sie die Arbeitsdiskette aus dem Laufwerk und drücken Sie die RETURN-Taste. Es erscheint der Prompt A>.

Nach erfolgreicher Beendigung dieser vier Schritte haben Sie eine **funktionsfähige Arbeitsdiskette** für die Arbeit mit **dBASE II** erstellt. Bitte arbeiten Sie immer nur mit dieser Arbeitsdiskette. Ihre Original-Diskette bewahren Sie bitte an einem sicheren Ort auf. Wenn aus irgendwelchen Gründen die Arbeitsdiskette defekt werden sollte, brauchen Sie lediglich die oben aufgeführten vier Schritte zu wiederholen, um erneut eine laufähige Version von dBASE II zu erhalten.

Die für die Installation notwendigen Dateien befinden sich aus Platzgründen auf der Rückseite Ihrer dBASE II-Original-Diskette (Seite B). Diese brauchen Sie mit dem Kopierprogramm von DISCKIT auf eine separate Arbeitsdiskette nur zu übertragen, wenn Sie die hier vorgegebene Anpassung verändern wollen. Dies ist in der Regel nicht notwendig und sollten Sie nur durchführen, wenn Sie über entsprechende Systemkenntnisse verfügen.

Sie haben damit die Erstellung der Arbeitsdiskette beendet. Fahren Sie nun fort mit dem Kapitel "dBASE II mit der Arbeitsdiskette starten" auf der Seite 10.

Feststellen des freien Speicherplatzes

Den Speicherplatz auf der Arbeitsdiskette können Sie mit dem CP/M-Befehl SHOW jederzeit nachprüfen. Legen Sie Ihre CP/M Plus-Diskette in das Laufwerk ein und schreiben Sie:

```
show b:<RETURN>
```

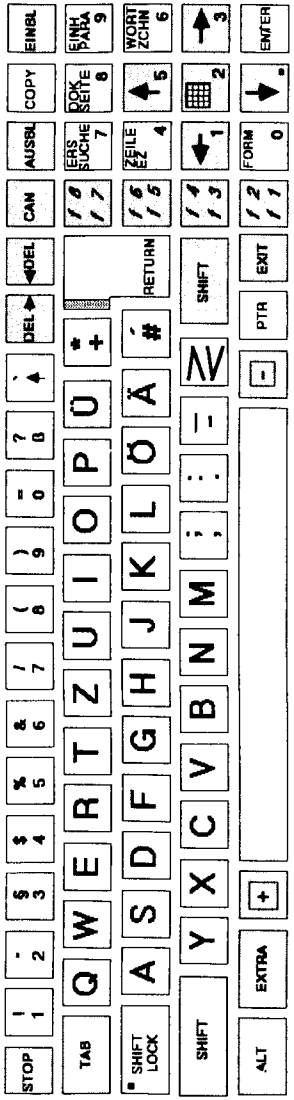
oder wenn die RAM-Disk installiert ist

```
show m:<RETURN>
```

Nun können Sie die Arbeitsdiskette einlegen. Mit dem Drücken irgendeiner Taste zeigt das Programm den freien Speicherplatz in KByte an. Sollte der Speicherplatz zu gering für Ihr nächstes Vorhaben sein, erstellen Sie sich eine neue Arbeitskopie. Dies können sie entweder erneut mit den oben erläuterten Schritten durchführen oder indem Sie ein Duplikat Ihrer Arbeitskopie erstellen und dann davon Ihre alten Texte löschen. Wie Sie ein Duplikat einer Diskette erstellen, können Sie dem Bedienungshandbuch Ihres Schneider Joyce entnehmen.

Tastatur-Belegung

Sie sehen eine Abbildung der Schneider Joyce-Tastaturbelegung. Die grauen Flächen zeigen, welche Tasten mit dBASE II-Funktionen belegt sind.



dBASE II mit der Arbeitsdiskette starten

Legen Sie Ihre vorher erstellte Arbeitsdiskette in das Laufwerk ein und schalten Sie den Joyce-Personalcomputer wie gewohnt ein. CP/M wird gestartet und meldet sich mit dem Prompt A>.

Man kann dBASE II auf zwei Arten starten.

Möglichkeit 1

Die korrekte Tastenbelegung von dBASE II laden Sie mit der Datei "DB.KEY". Dazu geben Sie ein:

```
setkeys db.key <RETURN>
```

Der Start von dBASE II erfolgt dann mit

```
dbase <RETURN>
```

Möglichkeit 2

Der Aufruf für das Laden der Funktionstastenbelegung kann mit Hilfe einer Submit-Datei automatisiert werden. Der Start erfolgt dann mit:

```
db.sub <RETURN>
```

Die gewünschte Tastatur- und Funktionstastenbelegung bleibt solange erhalten bis Sie Ihren Schneider Joyce-Computer ausschalten.

Der Maskengenerator ZIP (Seite B) war nicht an den größeren Joyce-Bildschirm anzupassen. ZIP starten Sie mit

```
zip <RETURN>
```

Wie man dBASE II bedient, lesen Sie am besten im Handbuch nach; wie die Tastatur- und Funktionstastenbelegung aussieht, können Sie folgenden Tabellen entnehmen.

Funktions- und Sondertastenbelegung

Der Schneider-Joyce-Funktionstasten- und Cursor-Block wurde mit den wichtigsten dBase II-Kommandos belegt. Auf diese Weise können Sie sich die Eingabe der entsprechenden Control-Befehle ersparen. Immer wenn im dBASE II-Handbuch von der Control-Taste gesprochen wird, heißt das für den Joyce, daß Sie die ALT-Taste betätigen müssen.

Wie die Belegung im einzelnen aussieht, zeigt folgende Übersicht:

Schneider Joyce	dBase II	Bedeutung
TAB	^I	Tabulator
STOP	Esc	Unterbrechen
SHIFT + STOP	^S	Ausgabe Stop/Weiter
CAN	^Q	Editieren beenden ohne speichern
SHIFT + CAN	^W	Editieren beenden mit Speichern
EXIT	^Q	Unterbrechen

◄DEL	^H	Zeichen vor Cursor löschen
DEL►	DEL	Zeichen hinter Cursor löschen
SHIFT + DEL►	^T	Zeile löschen
ALT + DEL►	^Y	Feld löschen
f2 f1		HELP <RETURN>
SHIFT + f2 f1		QUIT
f4 f3		USE
SHIFT + f4 f3		CREATE
f6 f5		EDIT
SHIFT + f6 f5		LIST <RETURN>

		APPEND <RETURN>
		LIST FILES ON A LIKE *.* <RETURN>
		Löschmerker ein/aus
		Drucker ein/aus
	^X	Zeile nach unten
	^E	Zeile nach oben
	^S	Zeichen nach links
	^D	Zeichen nach rechts
	^C	nächster Datensatz
	^R	vorheriger Datensatz

	^V	einfügen ein/aus
	^R	vorheriger Datensatz
	^C	nächster Datensatz

Die Zahlen auf der Zahlentastatur können durch gleichzeitiges Drücken von Alt, Shift-Alt oder Extra eingegeben werden.

dBASE II™

Relationales Datenbank-System

ANWENDER HANDBUCH

GESCHRIEBEN VON
WAYNE RATLIFF

Markt&Technik
Verlag Aktiengesellschaft
Hans-Pinsel-Straße 2
8013 Haar bei München

Markt&Technik
Verlag Aktiengesellschaft
Hans-Pinsel-Straße 2
8013 Haar bei München

dBASE II™

Relationales Datenbank-System

ANWENDER HANDBUCH

GESCHRIEBEN VON
WAYNE RATLIFF

Markt&Technik
Verlag Aktiengesellschaft
Hans-Pinsel-Straße 2
8013 Haar bei München

GESAMTINHALTSVERZEICHNIS

1. Kapitel: Anpassung und Installationen

Markt&Technik
Verlag Aktiengesellschaft
Hans-Pinsel-Straße 2
8013 Haar bei München

1. Kapitel Anpassung und Installationshinweise

(Diese Seite wurde
absichtlich nicht bedruckt)

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, das Handbuch 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 Ihnen mitteilt, 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 KLAMMERSYMBOLE (', d. h. das einfache Anführungszeichen) SOLLEN SIE NICHT EINGEBEN!

„..." wird verwendet, um die allgemeine Form von Befehlen anzuzeigen.

(Diese Seite wurde
absichtlich nicht bedruckt)

[...] 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 läuft sowohl auf 8-Bit-, wie auch auf 16-Bit-Microcomputern. dBASE II wird mit der anschließend beschriebenen Installations-Prozedur auf Ihren Rechner angepaßt und benötigt dann je nach System folgende Hard- und Software-Umgebung:

Anforderungen an 8-Bit-Microcomputer

- 8080, 8085 oder Z80 Mikrocomputer (wie z. B. TRS-80 Model II, Northstar, Apple mit Z80-Karte).
- mindestens 48 KBytes Schreiblesespeicher
- CP/M 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).

Anforderungen an 16-Bit-Microcomputer

- 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.).
- 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 Sätze ("records") per Datei-File:	65535 max.
Zeichen pro Satz:	1000 max.
Felder pro Satz:	32 max.
Zeichen pro Feld:	254 max.
Temporäre Variablen:	64 max.
Größte darstellbare Zahl:	etwa +1.8 x 10 hoch 63
Kleinste darstellbare Zahl:	etwa +1 x 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
Felder im Report:	24 max.
Länge eines Index-Schlüssels:	max. 99 Zeichen
Ausdrücke in SUM-Befehl:	5 max.

Anfertigen einer Sicherheitskopie

Falls Sie mit dem Rechner noch nicht so recht umgehen können, dann lassen Sie die Anpassung von Ihrem Fachhändler oder einer anderen 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 II! 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 und in Laufwerk B eine Sicherheitskopie von dBASE II.

Sicherheitskopien in verschiedenen Betriebssystemen

CP/M-Betriebssysteme

Geben Sie folgendes ein:

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

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

MS-DOS/PC-DOSBetriebssysteme

Geben Sie folgendes ein:

'COPY B:*.*'

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.

Die fertig kopierte Diskette enthält nun Ihr Betriebssystem und (!) die dBASE II-Befehlsdateien, die Sie benötigen. Mit dieser Diskette werden Sie in Zukunft arbeiten.

Beachten Sie: 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!

Hinsichtlich der Bezeichnung von Dateien gibt es einen wichtigen Unterschied zwischen verschiedenen Betriebssystemen:

Das Betriebssystem MS-DOS, bzw. PC-DOS oder auch CP/M-80 verwendet zur Kennzeichnung von Programmdateien den Dateityp ".COM", der beim CP/M - 86 Betriebssystem „.CMD" heißt.

Da früher unter dBASE II die Dateityp-Kennzeichnung ".CMD" jedoch zur Identifizierung von dBASE II-Befehlsdateien diente, wird nun bei allen 16-Bit-Versionen von dBASE II die einheitliche Bezeichnung „.PRG" für die dBASE II- Befehlsdateien verwendet.

Die anderen Dateibezeichnungskonventionen sind gemeinsam in Kapitel 2, Abschnitt 1 beschrieben und bei allen dBASE II-Versionen identisch.

Die Unterschiede sind noch einmal in untenstehender Tabelle zusammengefaßt:

	Betriebssystem
MS-DOS / PC-DOS	CP/M
DBASE.COM	DBASE.CMD (CP/M-86) DBASE.COM (CP/M-80)
DBASEOVR.COM	DBASEOVR.COM
DBASEMSG.TXT	DBASEMSG.TXT
INSTALL.COM (nur MS-DOS)	INSTALL.CMD
(Befehls- oder Programmdateien haben die Endung:)	
.PRG	.PRG .CMD (bei CP/M-80!!)

Installation von dBASE II auf Ihrem System

In diesem Handbuch werden zwei Installationsprozeduren getrennt voneinander beschrieben. Zuerst wird die Installation von dBASE II auf 8-Bit-Microcomputern beschrieben und anschließend auf 16-Bit-Microcomputern.

Installation für 8-Bit-Microcomputer

Setzen Sie die Kopie (Sie haben doch eine Kopie angefertigt, nicht wahr?) des dBASE II-Systems in Ihr Arbeitslaufwerk ein (das Laufwerk, dessen Bereitschaftszeichen ("Prompt") von Ihrem Betriebssystem gerade auf dem Bildschirm ausgegeben wird. Bei Laufwerk 'B:' z. B. meldet sich das System mit „B>").

Führen Sie alle sonst nötigen Handgriffe zur Initialisierung des Betriebssystems aus (Eingabe von Control-C in CP/M, RESET oder Sonstiges).

Nun geben Sie 'install' <RETURN> ein, um das dBASE II-System an die Besonderheiten Ihres Rechners anzupassen (GEBEN SIE DIE (') NICHT EIN!).

Vergleichen Sie die folgenden Ausführungen mit Abb. 1.1.

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 'J'.

Die Alternative 'J' 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 Bildschirmes 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 schikken, 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
INSTALLATIONSPROGRAMM VER 3.5E

COPYRIGHT (C) 1982 RSP INC.
MASKENORIENTIERTE VERARBEITUNG (J/N)? J

                    TERMINAL AUSWAHL
                       MENUE #1
A - ADDS VIEWPOINT                  M - HEATH 89
B - ADM-31                          N - HP 125
C - ADM-3A                          O - HP 2621
D - ACCESS                          P - INTERCOLOR
E - APPLE ///                       Q - KAYPRO II
F - APPLE II 40 COL                 R - NEC PC-8000/1
G - CROMEMCO 3102                   S - NS ADVANTAGE
H - DIALOG 81                       T - OSBORNE I
I - EAGLE AVL                       U - PERKIN ELMER 11
J - EPSON QX-10                     V - SANYO MBC 3000
K - GNAT-SYSTEM 10                  W - SOROC
L - HAZELTINE 1500

X - MENUE #2
Y - ALTE INSTALLATION VERAENDERN
Z - SPEZIELLES TERMINAL ANPASSEN
WAEHLEN SIE DEN TERMINAL-TYP: E
```


Sollte Ihr Terminal-Typ nicht aufgeführt sein, so geben sie 'X' ein, und das zweite Terminal Auswahl-Menue erscheint.

Abb. 1.1b:

```
                    TERMINAL AUSWAHL
                       MENUE #2

A - SPERRY UTS 40                   M - XEROX 820
B - SUPERBRAIN
C - TELEVIDEO
D - TOSHIBA T100
E - TOSHIBA T250
F - TRS-80 (FMG)
G - TRS-80 II (P&T)
H - TRS-80 III
I - VECTOR GRAPHICS
J - VISUAL-100
K - VPE-80
L - VT-100

  X - MENUE #1
  Y - ALTE INSTALLATION VERAENDERN
  Z - SPEZIELLES TERMINAL ANPASSEN

WAEHLEN SIE DEN TERMINAL-TYP:
```

Wenn Sie eines der aufgeführten Terminals gewählt haben, fragt dBASE II anschließend, ob Sie MACRO-Zeichen, Datum oder Begrenzungszeichen ändern wollen. Falls Sie mit 'N' antworten, gelangen Sie sofort an das Ende des Installationsprogramms.

Falls Sie mit 'J' antworten, können Sie die gewünschten Änderungen durchführen (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.

(Bei entsprechend anderer Wahl ,'Z', werden Sie menügesteuert weiter durch das Installationsverfahren geführt (vgl. Abb. 1.3 usw.))

Abb. 1.2:

```
SOLL MACRO-ZEICHEN, DATUM ODER BEGRENZUNGS-
ZEICHEN GEAENDERT WERDEN (J/N)? J

GEBEN SIE EIN ZEICHEN FUER MAKRO EIN, ODER
DRUECKEN SIE RETURN, WENN DAS ZEICHEN „&“ GILT:
<RETURN>

FEHLER-DIALOG ERWUENSCHT: -- RETURN DRUECKEN
KEIN DIALOG ERWUENSCHT: -- JEDE ANDERE TASTE
<RETURN>

BETRIEBSSYSTEM AUSWAHL
   A - CP/M 2.2
   B - TURBODOS SYSTEM

WAEHLEN SIE EIN BETRIEBSSYSTEM AUS: A


GEBEN SIE DAS LAUFWERK AN, WO SICH OVERLAY UND
HELP-DATEIEN BEFINDEN, ODER RETURN, WENN DIE
DATEIEN AUF DEM GLEICHEN LAUFWERK WIE dBASE SIND: <RETURN>

DAS DATUMSFORMAT IST TT/MM/JJ.
WOLLEN SIE MM/TT/JJ - (J/N)?: N


AKTUELLES BEGRENZUNGSZEICHEN SIND FUER:
   LINKES ZEICHEN ":"
   RECHTES ZEICHEN ":"

WOLLEN SIE DIE BEGRENZUNGSZEICHEN AENDERN (J/N)? N

DRUECKEN SIE „J“ ZUM SICHERN DER WERTE,
ODER JEDE ANDERE TASTE ZUM ABBRUCH DES PROGRAMMS : J
INSTALLATIONSPARAMETER WERDEN GESICHERT
```


Am Ende der Anpassungs-Prozedur können Sie diese gültig machen, indem Sie 'J' 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 könnnen diese Anpassungs-Prozedur auch wählen, um die normalen Werte für das von Ihnen gewählte Terminal zu modifizieren (beispielsweise die Ausgabe invertierter Texte, d. h. dunkel auf hell, durch bestimmte Code-Folgen).

Abb. 1.3:

```
SPEZIELLES TERMINAL ANPASSEN

SIE BENOETIGEN DEN HEX- ODER DEZIMAL-
CODE, DEN IHR TERMINAL ZUR STEUERUNG
DER VERSCHIEDENEN FUNKTIONEN BENOETIGT.

FOLGENDE STEUERCODES WERDEN BENOETIGT:

   ZEICHEN FUER BACKSPACE
   DIREKTE CURSOR-ADRESSIERUNG
   SCHIRM LOESCHEN
   CURSOR AM ANFANG
   (AUCH MIT „SCHIRM LOESCHEN“ KOMBINIERT)
   OPTIONAL: DARSTELLUNG = HELL/DUNKEL
   ODER NORMAL/INVERTIERT

„J“ FUER WEITER - FALLS ERWUENSCHT
J
```

Wenn Sie die Codes kennen, die Ihr Terminal zu den oben genannten Operationen veranlassen, kann geben Sie 'J' ein.

IBASE 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.

IBASE 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. 1.4:

```
WERDEN DIE WERTE IN HEX ODER DEZIMAL EINGEGEBEN?
DRUECKEN SIE „D" FUER DEZIMAL-
ODER „H" FUER HEX-EINGABE:
H

DIE WERTE WERDEN ALS FOLGE VON ZAHLEN EINGEGEBEN. SOWIE RETURN
ALS ABSCHLUSS.

GEBEN SIE DIE WERTE FUER BACKSPACE EIN
DIE MEISTEN TERMINALS BENUTZEN DIE SEQUENZ „BACKSPACE, SPACE,
BACKSPACE". DRUECKEN SIE „J" WENN DIE SEQUENZ RICHTIG IST.
J
```


"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. 1.5:

```
--- DIREKTE CURSOR ADRESSIERUNG ---

DIESE SEQUENZ IST NORMALERWEISE 3 - 4 ZEICHEN
LANG. DIE ERSTEN EIN ODER ZWEI ZEICHEN SIND
OFT FEST DEFINIERT. DIE ANDEREN ZEICHEN
GEBEN DIE REIHEN- BZW. SPALTEN-ADRESSE AN.

ERFOLGT DIE ADRESSIERUNG MIT BINAEREN ODER
MIT ASCII ZEICHEN? DRUECKEN SIE „J" WENN BINAERE.
J
WELCHES ZEICHEN ENTHAELT DIE SPALTENADRESSIERUNG?
'4'<RETURN>
WELCHES ZEICHEN ENTHAELT DIE ZEILENADRESSIERUNG?
'3'<RETURN>

VIELE TERMINALS BENOETIGEN EINE AUFZUADDIERENDE
KONSTANTE (BIAS). GEBEN SIE DIESE KONSTANTE
FUER IHR TERMINAL EIN.
'20'<RETURN>

GEBEN SIE NUN DIE ZEICHENFOLGE FUER DIE CURSOR-
ADRESSIERUNG EIN. VERWENDEN SIE EINE „0" FUER
DAS ZEILEN- BZW. SPALTEN-ZEICHEN.
(11 ZEICHEN MAXIMAL)

GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 1: 1A '1B'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 2: 00 '59'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 3: 00 '0'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 4: 00 '0'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 5: 00 <RETURN>

IST ES RICHTIG (J/N)? J
```

Abb. 1.6:

```
- HELL/DUNKEL - NORMAL/INVERTIERTE DARSTELLUNG

GEBEN SIE DIE SEQUENZ EIN, DIE AUF VOLLE
HELLIGKEIT ODER NORMALE DARSTELLUNG SCHALTET.
(5 ZEICHEN MAXIMAL)

GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 1: 11 <RETURN>
IST ES RICHTIG (J/N)? J

GEBEN SIE DIE SEQUENZ EIN, DIE AUF HALBE
HELLIGKEIT ODER INVERTIERTE DARSTELLUNG SCHALTET.
(5 ZEICHEN MAXIMAL)

GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 1: 11 <RETURN>
IST ES RICHTIG (J/N)? J
```

Abb. 1.7:

```
--- SCHIRM LOESCHEN/CURSOR AN DEN ANFANG ---

GEBEN SIE DIE SEQUENZ EIN, DIE DEN BILDSCHIRM
LOESCHT UND DEN CURSOR AN DEN ANFANG POSITIONIERT.
(11 ZEICHEN MAXIMAL)

GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 1: 1C '1B'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 2: 00 '4C'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 3: 00 <RETURN>

IST ES RICHTIG (J/N)? J
```


Abb. 1.8:

```
GEBEN SIE DIE SEQUENZ EIN, DIE BEIM START DER
MASKENORIENTIERTEN VERARBEITUNG BENOETIGT WIRD.
(FALLS DIREKTE CURSOR-ADRESSIERUNG MOEGLICH IST)
(11 ZEICHEN MAXIMAL)

GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 1: 00 <RETURN>

IST ES RICHTIG (J/N)? J
```

Abb. 1.9:

```
GEBEN SIE DIE ZEICHENFOLGE AN, DIE NACH VERLASSEN
DER MASKENORIENT. VERARBEITUNG GEGEBEN WERDEN SOLLTE.

ZUM BEISPIEL: AM ENDE DIESER VERARBEITUNGSART
SOLL DER CURSOR AN DEN ANFANG DER LETZTEN ZEILE.
(11 ZEICHEN MAXIMAL)

GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 1: 11 '1B'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 2: 1A '59'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 3: 00 '31'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 4: 16 '20'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 5: 00 <RETURN>
IST ES RICHTIG (J/N)? J
```

Abb. 1.10:

```
GEBEN SIE DIE SEQUENZ AN, DIE ZURUECK AUF DIE
NORMALE HELLIGKEIT BZW. DARSTELLUNG AM ENDE
DER MASKENORIENTIERTEN VERARBEITUNG SCHALTET.
(5 ZEICHEN MAXIMAL)

GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 1: 11 '1D'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 2: 00 <RETURN>
IST ES RICHTIG (J/N)? J
```

Abb. 1.11:

```
WIEVIELE ZEICHEN PRO ZEILE?
80<RETURN>

WIEVIELE ZEILEN HAT IHR BILDSCHIRM?
24<RETURN>
```

Abb. 1.12:

```
SOLL MACRO-ZEICHEN, DATUM ODER BEGRENZUNGS-
ZEICHEN GEAENDERT WERDEN (J/N)? N

DRUECKEN SIE „J" ZUM SICHERN DER WERTE,
ODER ANDERE TASTE ZUM ABBRUCH DES PROGRAMMS :J
INSTALLATIONSPARAMETER WERDEN GESICHERT.
```


Abb. 1.13:

```
ALTE INSTALLATION VERAENDERN

SIE BENOETIGEN DEN HEX- ODER DEZIMAL-
CODE, DEN IHR TERMINAL ZUR STEUERUNG
DER VERSCHIEDENEN FUNKTIONEN BENOETIGT.
„J" FUER WEITER - FALLS ERWUENSCHT
J

WERDEN DIE WERTE IN HEX ODER DEZIMAL EINGEGEBEN?
DRUECKEN SIE „D" FUER DEZIMAL-
ODER „H" FUER HEX-EINGABE:
H

DIE WERTE WERDEN ALS EINE FOLGE VON ZAHLEN
EINGEGEBEN. SOWIE RETURN ALS ABSCHLUSS.

1 -  ZEICHENFOLGE FUER BACKSPACE
2 -  DIREKTE CURSOR-ADRESSIERUNG
3 -  SCHIRM LOESCHEN/CURSOR AM ANFANG
4 -  VOLLE HELLIGKEIT EINSCHALTEN
5 -  HALBE HELLIGKEIT EINSCHALTEN
6 -  SEITENVERARBEITUNG EINSCHALTEN
7 -  ZURUECK ZUR ZEILENVERARBEITUNG
8 -  ZURUECK ZUM STANDART-MODE
9 -  SCHIRMGROESSE ANPASSEN

WAEHLEN SIE BITTE:
(ALLE ZEICHEN AUSSER 1 BIS 9 = ENDE DES MENUES)
'7'
```

Wenn Sie ein bereits eingerichtetes dBASE II-System modifizieren wollen, geben Sie 'install' dann 'J' oder 'N' als Antwort auf die Frage nach der bildschirmorientierten Editierung (FULL SCREEN EDITING), dann wählen Sie die 'J' 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. 1.14:

```
GEBEN SIE DIE ZEICHENFOLGE AN, DIE NACH VERLASSEN
DER MASKENORIENT. VERARBEITUNG GEGEBEN WERDEN SOLLTE.

ZUM BEISPIEL: AM ENDE DIESER VERARBEITUNGSART
SOLL DER CURSOR AN DEN ANFANG DER LETZTEN ZEILE.
(11 ZEICHEN MAXIMAL)

AKTUELLE SEQUENZ IST :
1B
59
31
20

IST ES RICHTIG (J/N)? N
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 1: 1B '1B'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 2: 59 '59'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 3: 31 '36'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 4: 20 '20'<RETURN>
GEBEN SIE DAS STEUERZEICHEN FUER BYTE - 5: 00 <RETURN>

IST ES RICHTIG (J/N)? J
1 -  ZEICHENFOLGE FUER BACKSPACE
2 -  DIREKTE CURSOR-ADRESSIERUNG
3 -  SCHIRM LOESCHEN/CURSOR AM ANFANG
4 -  VOLLE HELLIGKEIT EINSCHALTEN
5 -  HALBE HELLIGKEIT EINSCHALTEN
6 -  SEITENVERARBEITUNG EINSCHALTEN
7 -  ZURUECK ZUR ZEILENVERARBEITUNG
8 -  ZURUECK ZUM STANDART-MODE
9 -  BILDSCHIRMGROESSE ANPASSEN

WAEHLEN SIE BITTE:
(ALLE ZEICHEN AUSSER 1 BIS 9 = ENDE DES MENUES)

<RETURN>

SOLL MACRO-ZEICHEN, DATUM ODER BEGRENZUNGS-
ZEICHEN GEAENDERT WERDEN (J/N)? J
```



```
GEBEN SIE EIN ZEICHEN FUER MACRO EIN, ODER
DRUECKEN SIE RETURN, WENN DAS ZEICHEN „&“ GILT :
<RETURN>

FEHLER-DIALOG ERWUENSCHT: -- RETURN DRUECKEN
KEIN DIALOG ERWUENSCHT : -- JEDE ANDERE TASTE<RETURN>

BETRIEBSSYSTEM AUSWAHL
   A - CP/M 2.2
   B - TURBODOS SYSTEM

WAEHLEN SIE EIN BETRIEBSSYSTEM AUS: A

GEBEN SIE DAS LAUFWERK AN, WO SICH OVERLAY UND
HELP-DATEIEN BEFINDEN, ODER RETURN WENN DIE
DATEIEN AUF DEM GLEICHEN LAUFWERK WIE dBASE SIND: <RETURN>

DAS DATUMSFORMAT IST TT/MM/JJ.
WOLLEN SIE MM/TT/JJ - (J/N)?: J

AKTUELLES BEGRENZUNGSZEICHEN SIND FUER:
   LINKES ZEICHEN ":"
   RECHTES ZEICHEN „:“

WOLLEN SIE DIE BEGRENZUNGSZEICHEN AENDERN (J/N)? J
   AENDERN DES LINKEN ZEICHEN NACH !
   AENDERN DER RECHTEN ZEICHEN NACH !

DRUECHEN SIE „J“ ZUM SICHERN DER WERTE,
ODER ANDERE TASTE ZUM ABBRUCH DES PROGRAMMS : J
INSTALLATIONSPARAMETER WERDEN GESICHERT
```

Das dBASE II-System ist jetzt fertig auf Ihrem 8-Bit-Rechner angepaßt

Installation für 16-Bit-Mikrocomputer

Um dBASE II auf Ihrem System zu installieren, wechseln Sie bitte auf das Laufwerk über, das Ihre Sicherheitskopie von dBASE II enthält - Sie benutzen doch die Sicherheitskopie ? Geben Sie nun INSTALL <RETURN> ein:

Es erscheint danach folgendes auf Ihrem Bildschirm :

Abb. 1.15:

```
                COPYRIGHT 1984 ASHTON-TATE
                    VERSION 4.0 GER
                    MENUE NUMMER 1

A - ADDS VIEWPOINT          B - ADM 31
C - ADM 3A                  D - CANON AS100
E - COLUMBIA 1600           F - COMPAQ
G - CROMEMCO 3102           H - EAGLE 1600
I  - GRID COMPASS           J - HAZELTINE 1500
K - HEATH 89                L - HP 2621
M - IBM DISPLAY WRITER      N - IBM PC
O - NEC 5200                P - PERKIN ELMER
Q - SOROC                   R - TELEVIDEO
S - TI PROFESSIONAL         T - TOSHIBA T300
U - VISUAL 100              V - VT - 52

# - WAEHLEN SIE MENUE 1 BIS ? *
                    (X = ENDE):
```

* (Wird nur angezeigt, wenn mehr als 1 Menue existiert.)

Falls Sie Ihr Terminal nicht in diesem Menü finden, können Sie durch Eingabe einer Zahl in weiteren Menüs suchen. Andernfalls drücken Sie bitte den entsprechenden Buchstaben. Danach erscheint auf Ihrem Bildschirm Installation beendet. Sie befinden sich dann wieder in Ihrem Betriebssystem und können jetzt mit dBASE II beginnen. Wenn Sie dieses Menü mit 'X' verlassen, wird keine Änderung abgespeichert!

Falls Sie Ihr Terminal nicht in einem der Menüs gefunden haben, oder Sie dBASE II nach eigenen Wünschen anpassen möchten, gibt es hierzu entsprechende Möglichkeiten:

Achtung: Die Benutzung der folgenden Option erfordert tiefergehende Kenntnisse der technischen Grundlagen. Als Neuling auf diesem Gebiet sollten Sie die Option nicht benutzen.

Hinweis: Wenn die nachfolgende Operation eine Erstinstallation ist und Ihr Terminal mit einem der aufgelisteten vergleichbar ist, wählen Sie dasjenige für eine vorläufige Installation aus und starten danach erneut INSTALL. Sie benötigen für diese Arbeiten eine Liste der CONTROL-CODES Ihres Terminals. Diese befinden sich meist im Anhang des Terminal-Handbuches unter der Rubrik 'ESCAPE SEQUENCES'.

Drücken Sie nun gleichzeitig die Tasten CONTROL (auch mit CTRL oder CTL beschriftet) und den Buchstaben 'Y', worauf sich das folgende Menü meldet:

Abb.1.16:

```
AENDERUNG DER BESTEHENDEN INSTALLATION
 1. BEGRENZUNGSZEICHEN
 2. SEITENVERARBEITUNG EIN/AUS
 3. ZEICHENFOLGE FUER BACKSPACE
 4. DIREKTE CURSOR-ADRESSIERUNG
 5. VOLLE HELLIGKEIT EINSCHALTEN
 6. HALBE HELLIGKEIT EINSCHALTEN
 7. ZURUECK ZUR NORMALEN HELLIGKEIT
 8. SCHIRM LOESCHEN, CURSOR AN ANFANG
 9. ZURUECK ZUR ZEILENVERARBEITUNG
10. SCHIRMGROESSE (80 MAL 24 VORBELEGT)
11. VERZOEGERUNGSZEIT DES TERMINALS
12. FUNKTIONS- CURSOR-TASTENSTEUERUNG
13. FUNKTIONSTASTEN-BELEGUNG
14. SEITENVERARBEITUNG EINSCHALTEN
15. MACRO ZEICHEN
16. DATUMSFORMAT
17. ZEILE EINFUEGEN
18. ZEILE LOESCHEN
19. LOESCHEN BIS ZUM ENDE DER ZEILE
20. LOESCHEN BIS ZUM ENDE DES SCHIRMS
21. VORDERGRUND-FARBWAHL
22. HINTERGRUND-FARBWAHL
23. VORBELEGUNG DER SET-KOMANDOS
24. FARBSTEUERUNG AN
25. FARBSTEUERUNG AUS
26. FEHLER-DIALOG EIN/AUS

    X - ZURUECK ZUM HAUPT-MENUE
    W - AENDERUNGEN SPEICHERN
        WAEHLEN SIE BITTE:
```

Jede dieser Optionen wird in numerischer Reihenfolge beschrieben. Sie gelangen jedesmal wieder in dieses Menü zurück, nachdem eine der o. g. Optionen ausgewählt wurde, gleichgültig, ob nun eine Änderung erfolgte, oder nicht. Gleichzeitig behalten Sie durch eine Markierung der bereits bearbeiteten Optionen immer die Kontrolle darüber, welche Optionen schon bearbeitet wurden. Ein Stern vor der entsprechenden Option zeigt Ihnen dies an.

Wenn Sie an dieser Stelle keine weiteren Installationen durchführen wollen, können Sie das Installationsprogramm mit der Abspeicherung Ihrer bisherigen Änderungen verlassen.

Hinweis: Verlassen Sie das Installationsmenü dann aber nicht mit 'X', sondern suchen Sie in einem der wählbaren Terminalmenüs nach dem Namen, unter dem Sie Ihre Änderungen abgespeichert haben. Drücken Sie dann den zugeordneten Kennbuchstaben, um das Installationsprogramm wirklich mit der Abspeicherung Ihrer Daten zu verlassen!

Ein Abbruch des Programmes ist jederzeit durch Drücken von "CONTROL" (<CTRL>) und „C" möglich.

1. Feldbegrenzungszeichen

Diese Option erlaubt Ihnen eine freie Definition der Zeichen, die Sie als linke und rechte Markierung für die von dBASE II angezeigten Felder benutzen wollen.

Beispiel: Sie benutzen eine Datenbank mit einem Feld, das mit "NAME" bezeichnet ist. Die vorbelegten Feldbegrenzungen sind durch Doppelpunkte eingegrenzt. Das sieht folgendermaßen aus:

NAME : :

Wenn Sie nun diese Option wählen, erscheint folgendes:

LINKES BEGRENZUNGSZEICHEN IST:

VERAENDERN (J/N):

Wenn Sie nun mit „J" (oder „j") antworten, werden Sie nach „Neues Begrenzungszeichen?" gefragt. Danach geben Sie einfach das gewünschte neue Zeichen ein.

Die gleiche Sequence wird danach für das rechte Begrenzungszeichen durchgeführt.

2. Seitenverarbeitung ein/aus

Mit Hilfe dieser Option wird dem Programm mitgeteilt, ob das Terminal eine direkte CURSOR-Adressierung unterstützt. Wenn ja, kann eine maskenorientierte Verarbeitung erfolgen, andernfalls wird ein zeilenorientierter Dialog durchgeführt. Es erscheint dann folgende Information:

SEITENVERARBEITUNG IST EINGESCHALTET

VERAENDERN (J/N):

Nach der Eingabe von „J" (oder „j") erscheint das Menü wieder. Mit Hilfe von Zeilenschaltung und Leerschritten wird die Positionierung nun im Zeilenmodus durchgeführt.

3. Zeichenfolge für BACKSPACE

Mit Hilfe dieser Option können Sie die Zeichenfolge bestimmen, die einen Rückschritt mit Löschung definiert. Bei den meisten Systemen erfolgt dieses durch die Sequence: BACKSPACE, SPACE, BACKSPACE. Nachfolgender Dialog ermöglicht diese Anpassung:

AKTUELLE ZEICHENFOLGE FUER BACKSPACE IST -

DEZIMAL: 8 32 8

HEX: 8H 20H 8H

VERAENDERN (J/N):

Nach Eingabe von „J" können Sie diese nun verändern.
Es erscheint folgender Text:

TRENNUNG MIT KOMMA, ASCII-ZEICHEN IN " ", - NNH FOR HEX (...) -
- (z. B.:1BH,"2J"<CR>) -
VERAENDERN (J/N):

Nachfolgend werden wichtige Regeln, die bei der Eingabe von Zeichenfolgen bei allen nachfolgenden Eingabemöglichkeiten zu beachten sind, beschrieben.

Wenn Sie, wie oben aufgeführt, zu einer Antwort aufgefordert werden, können Sie jede Kombination von dezimaler, hexadezimaler oder ASCII-Texteingabe benutzen.

Beispiel: Wenn die Sequence ESC [000;000, (ESC ist dezimal 27, der Rest ist in ASCII-Zeichen) zur Steuerung der direkten CURSOR-Positionierung benötigt wird, können Sie dies in einer der nachfolgend dargestellten Arten eingeben:

```
1BH,'[000;000H'
1BH,'[000;000H'
1BH,'[000;000H'
1BH,"[000;000H"
27,'[000;000H'
1BH,5BH,30H,30H,30H,3BH,30H,30H,30H,48H
```

Sie können jedoch keine symbolische Bezeichnung wie ESC benutzen.

4. Direkte CURSOR-Adressierung

Mittels dieser Option geben Sie die Sequence ein, die dBASE II zur CURSOR-Positionierung benutzen soll.

Der Dialog sieht folgendermaßen aus:
X, Y-KOORDINATEN HABEN FOLGENDE WERTE: BINAERE

VERAENDERN (J/N):

Bemerkung: Es ist nur eine Art von Angaben erlaubt, entweder die BINAERE, oder die mit ASCII-Zeichen. Die Anzeige erfolgt abhängig von der vorhergehenden Installation.

Wählen Sie eine dieser Eingabearten nach Ihrem Bedarf. Falls Ihr Terminal die Sequence ESC '[rrr;ccc' benötigt, geben Sie die Parameter am besten in ASCII-Zeichenformat ein. Wenn jedoch die Sequence ESC ='< row,column >' angegeben ist, empfiehlt es sich, die Angabe in BINAER-Form einzugeben.

Als nächstes werden Sie nach der Position der X und Y Koordinaten innerhalb der Sequence gefragt:

DIE X-KOORDINATEN-POSITION IST BYTE-NR.: 2

VERAENDERN (J/N):

DIE Y-KOORDINATEN-POSITION IST BYTE-NR.: 3

VERAENDERN (J/N):

Wenn Sie mit „J" antworten, werden Sie nach der neuen Position gefragt.
Geben Sie nun einfach die neue Position an.

Sie werden dann nach der aufzuaddierenden Konstanten' gefragt. Viele Terminals benötigen diese Konstante für die CURSOR-Positionierung.

Beispiel: Oft wird als Konstante 32 (20H) benötigt, um die Positionierung durchführen zu können. Andere Terminals benötigen zum Beispiel eine Konstante von 1. Geben Sie nun Ihre terminal- spezifische Konstante ein. Zuletzt wird die neue Zeichen-Sequence angezeigt und Sie werden um eine Bestätigung gebeten. Bei der Eingabe von "N" beginnt der Dialog in der soeben beschriebenen Weise von vorne.

5. Volle Helligkeit einschalten

Diese Sequence wird von dBASE II benutzt, um auf volle Helligkeit oder invertierter Zeichendarstellung umzuschalten. Die Angaben erfolgen genau wie in Option 3. Falls halbe/volle Helligkeit erwünscht ist, setzen Sie diese Option gemäß der Zeichen für "Zurueck zur normalen Helligkeit" (RESET TO INTENSITY)

6. Halbe Helligkeit einschalten

Diese Sequence schaltet auf halbe Helligkeit. Falls die invertierte Darstellung in Option 5 eingeschaltet wurde, sollten Sie mit dieser Option die Sequence zur Rückschaltung auf den normalen Bildschirm geben. Der Dialog erfolgt genau wie bei Option 3.

7. Zurueck zur normalen Helligkeit

Diese Sequence sollte generell zum Rückschalten auf normalen Bildschirm verwendet werden, gleichgültig, ob vorher auf halbe Helligkeit, oder invertierte Darstellung geschaltet wurde. Der Dialog erfolgt wie bei Option 3.

8. Schirm loeschen, CURSOR an Anfang

Diese Funktion wird von dBASE II nur zum "Bildschirm loeschen/CURSOR an den Anfang" benutzt. Dialog siehe Option 3.

9. Zurueck zur Zeilenverarbeitung

Diese Sequence wird von dBASE II benutzt, wenn von einem der seitenorientierten Maskenverarbeitungsbefehle (z. B.:.- MODIFY COMMAND, APPEND, usw...) zur normalen Zeilenverarbeitung zurückgeschaltet wird. Diese Zeichenfolge sollte alle Video-Attribute zurücksetzen, sowie den CURSOR an den Anfang der letzten Zeile positionieren. Der Dialog erfolgt wie bei Option 3.

10. Schirmgroesse

Zur Seitenverarbeitung benötigt dBASE II die Angaben von Anzahl der Zeichen und Zeilen. Die meisten Terminals verfügen über 80 Zeichen x 24 Zeilen. Sollte dies auf Ihr Terminal nicht zutreffen, können Sie es durch Neueingabe wie nachfolgend beschrieben anpassen.

AKTUELLE SCHIRMGROESSE IST: - 80 MAL 24 -
VERAENDERN (J/N):

Wenn Sie mit "J" antworten, werden Sie nach weiteren Eingaben gefragt:

ANZAHL DER ZEILEN:

ANZAHL DER ZEICHEN:

11. Verzoegerungszeit des Terminals

Dieser Wert braucht normalerweise nie verändert zu werden. Es handelt sich dabei um den Verzögerungsfaktor, den dBASE II verwendet, um nach dem Löschen des Bildschirmes eine neue Ausgabe auf dem Schirm zu erzeugen. Bei einigen Terminals ist dieser Verzögerungsfaktor unbedingt notwendig, da es andernfalls zu einem Datenverlust kommen kann.

12. Funktions- /CURSOR/Tastensteuerung

Damit Sie auf den Funktionstasten vorbereitete Befehle abrufen können, müssen die Parameter im Programm installiert sein. Das gleiche gilt für die CURSOR-Steuertasten, wenn diese bei der Seitenverarbeitung benötigt werden.

Wenn Sie diese Option wählen, erscheint eine Auflistung der gerade aktuellen Tabelle. Das folgende Beispiel dient zur Veranschaulichung und kann von Ihrer Auflistung verschieden sein.

Beispiel:

Abb. 1.17:

```
CURSORTASTEN - ANPASSUNGEN:

ERSTES ZEICHEN IST: 0 (00H)

FUNKTIONSTASTEN SIND: EINGESCHALTET
<0=NICHT VORHANDEN, SONST=VORHANDEN> - 255 (FFH)

ZWEITES ZEICHEN FUER CURSOR-TASTEN IST: 0 (00H)
ZWEITES ZEICHEN FUER FUNKTIONSTASTEN IST: 91 (5BH)

STEUERZEICHEN:

        F01 - 59 (3BH)          F06 - 64 (40H)
        F02 - 60 (3CH)          F07 - 65 (41H)
        F03 - 61 (3DH)          F08 - 66 (42H)
        F04 - 62 (3EH)          F09 - 67 (43H)
        F05 - 64 (3FH)          F10 - 68 (44H)
CURSOR OBEN - 11 ( BH)   CURSOR UNTEN - 10 ( AH)
CURSOR LINKS - 8 ( 8H)   CURSOR RECHTS - 12 ( CH)
WUENSCHEN SIE DIESE ZU VERAENDERN? (J/N):
```


Falls die oben dargestellten Zeichen nicht mit denen Ihres Terminal identisch sind, antworten Sie bitte mit „J". Sie werden dann aufgefordert, einzeln die entsprechenden Tasten zu betätigen. Hierbei erkennt das Programm automatisch die Zeichenfolge, die durch Ihre Tastatur erzeugt wird. Sie erhalten den entsprechenden Code jeweils angezeigt, und das Programm schreibt ihn dann in die Tabelle für dBASE II ein. Der Dialog ist nachfolgend beschrieben:

```
DRUECKEN SIE DIE GEWUENSCHTE FUNKTIONSTASTE 1 -

JETZT:
```

Nachdem Sie die entsprechende Taste gedrückt haben, wird Ihnen die Zeichenfolge in dezimaler Form am Schirm angezeigt. Danach werden Sie zur Eingabe der nächsten Taste aufgefordert:

```
DRUECKEN SIE DIE GEWUENSCHTE FUNKTIONSTASTE 1 -

JETZT: 27 91 112

DRUECKEN SIE DIE GEWUENSCHTE FUNKTIONSTASTE 2 -

JETZT:
```

Diese Dialogform wird bis zur Funktionstaste 10 durchgeführt. Sollten Sie an Ihrem Terminal weniger als 10 Funktionstasten haben, so drücken Sie für alle Tasten, die Ihnen nicht zur Verfügung stehen, wieder die erste Funktionstaste.

Es ist wichtig, darauf zu achten, daß das erste Zeichen für beide, Funktionstaste , wie auch Cursortaste, identisch ist. Sollte das bei Ihrem Terminal nicht der Fall sein, müssen Sie sich für die Benutzung einer der beiden Tastengruppen entscheiden. Deshalb müssen Sie bei der Wahl der Funktionstasten im Dialog für die CURSOR-Tasten die erste Funktionstaste zur Zeicheneingabe benutzen. Denn das erste Zeichen in der Zeichenfolge wird immer von der zuletzt gedrückten Taste bestimmt. Bei der Eingabe für die CURSOR-Tasten sieht der Dialog folgendermaßen aus:

```
DRUECKEN SIE DIE CURSOR-TASTEN IN DER REIHENFOLGE
--> LINKS, RECHTS, OBEN, UNTEN -
```

13. Funktionstasten-Belegung

Wenn Funktionstasten vorhanden sind, können sie ohne vorgegebene Festlegung mit kompletten Befehlen belegt werden. Diese Befehle kommen dann bei einfacher Betätigung der Funktionstaste zur Ausführung. Hierbei ist noch folgendes zu beachten: Ein definierter Befehl mit einem Semicolon (';') als Abschluß wird von dBASE II als Befehl zur direkten Ausführung betrachtet, genauso, als würden Sie nach einer Befehlseingabe die <RETURN>-Taste drücken. Wenn dieses Semicolon fehlt, verbleibt dBASE II in der Eingabe-Zeile und wartet auf weitere Tasteneingaben. Der Dialog sieht folgendermaßen aus:

Abb. 1.18:

```
FUNKTIONSTASTEN-BELEGUNG:

A - F01: HELP;          F - F06: LIST STATUS;
B - F02: DISP;          G - F07: LIST MEMO;
C - F03: LIST;          H - F08: CREATE;
D - F04: LIST FILES;    I - APPEND;
E - F05: LIST STRU;     J - EDIT #;

        X - ZURUECK ZUM AUSGANGSMENUE

            WAS WOLLEN SIE VERAENDERN:
```

Um die Befehle zu verändern, drücken Sie den Buchstaben der entsprechenden Zeile und geben im nachfolgenden Dialog den neuen Befehl ein. Beachten Sie bitte, daß eine Befehlszeile nicht länger als 20 Zeichen sein darf.

GEBEN SIE DIE NEUE BELEGUNG EIN:

14. Seitenverarbeitung einschalten

Mit Hilfe dieser Sequence initialisiert dBASE II den Schirm für die Seitenverarbeitung (maskenorientiert). Der Dialog erfolgt wie bei Option 3.

15. MACRO Zeichen

Das Zeichen „&" wird normalerweise von dBASE II als „Macro-Zeichen" benutzt. Sollten Sie hierfür jedoch ein anderes Zeichen bevorzugen, können Sie dieses wie im nachfolgend geschilderten Dialog verändern, z. B. für '%':

DAS MACRO ZEICHEN IST: &

VERAENDERN (J/N): J

NEUES MACRO ZEICHEN: %

16. Datumsformat

Das aktuelle Format für die Datumseingabe bei dBASE II ist MM/TT/JJ. Sie können es in das deutsche Format TT/MM/JJ verändern. Nach Eingabe von „J" wird das Format umgesetzt und das Programm kehrt zum Menü zurück.

AKTUELLES DATUMSFORMAT IST: TT/MM/JJ

VERAENDERN (J/N):

17. Zeile einfuegen

Wenn Ihr Terminal diese Fähigkeit besitzt, sollten Sie die entsprechende Zeichenfolge eingeben. Der Dialog erfolgt wie bei Option 3.

18. Zeile loeschen

Wenn auch diese Funktion von Ihrem Terminal unterstützt wird, geben Sie bitte die entsprechende Zeichenfolge ein.

19. Loeschen bis zum Ende der Zeile

Auch hier handelt es sich um eine Funktion, die nicht von allen Terminals unterstützt wird. Wenn sie jedoch vorhanden ist, beschleunigt der Einsatz dieser Funktion, wie auch bei Option 17,18 und 20, den Bildaufbau. Verfahren Sie bitte bei der Eingabe wie bei Option 1.

20. Loeschen bis zum Ende des Schirms

Falls auch diese Funktion bei Ihrem Terminal zur Verfügung steht, verfahren Sie bitte wie bei den drei vorhergehenden Optionen.

21. Vordergrund-Farbwahl

Wenn Sie einen Farbbildschirm benutzen, und es für dieses Gerät bereits eine angepaßte Version von dBASE II gibt, können Sie mittels dieser Option die Vordergrundfarbe nach Ihrem Wunsch wählen. Bitte benutzen Sie Ihr Terminalhandbuch für die Zeichenfolge.

22. Hintergrund-Farbwahl

Diese Option ist identisch mit der Option 21, nur mit dem Unterschied, daß Sie hier die Hintergrundfarbe bestimmen können.

23 Vorbelegung der 'SET'-Kommandos

dBASE II besitzt eine Anzahl verschiedener Programmschalter, die mittels des 'SET'-Befehls verändert werden können.

Bitte beachten Sie hierbei die ausführlichen Erklärungen in Ihrem dBASE II-Handbuch. Mit Hilfe dieser Option können Sie eine Vorbelegung auf "ON" oder „OFF" für die entsprechenden Befehle durchführen. Wenn Sie diese Option wählen, erhalten Sie eine Liste der aktuellen Vorbelegungen:

Abb. 1.19:

```
        VORBELEGUNGEN DER 'SET' KOMMANDOS

A - SET BELL IS ON          B - SET CARRY IS OFF
C - SET COLON IS ON         D - SET CONFIRM IS OFF
E - SET DELETE IS OFF       F - SET ESCAPE IS ON
G - SET EXACT IS OFF        H - SET INTENSITY IS ON
I  - SET LINKAGE IS OFF     J - SET RAW IS OFF
K - SET EJECT IS ON         L - SET TALK IS ON

        X - ZURUECK ZUM AUSGANGSMENUE

DRUECKEN SIE DIE GEWUENSCHTE TASTE UM DIE VORBELEGUNG AUF -
ON/OFF- ZU SCHALTEN
```

Wenn Sie einen dieser Schalter verändern möchten, tippen Sie bitte den gewünschten Buchstaben. Danach erscheint dieses Menue erneut und zeigt die bereits durchgeführten Änderungen schon an. Nachdem Sie alle gewünschten Änderungen durchgeführt haben, gelangen Sie mit „X" wieder ins Hauptmenü zurück.

24. Farbsteuerung an

Diese Option erlaubt die Angabe der Zeichenfolge, die auf Ihrem Terminal eine Umschaltung auf Farbe bewirkt. Eingaben erfolgen wie bei Option 3.

25. Farbsteuerung aus

Diese Option wird zum Abschalten der Farbsteuerung benutzt. Verfahren Sie wie bei Option 24.

26. Fehler-Dialog ein/aus

Normalerweise antwortet dBASE II nach einem Fehler in einer Befehlszeile mit einer Fehlermeldung. Zusätzlich wird Ihnen jeweils die Möglichkeit gegeben, die Fehleingabe zu korrigieren.

Beispiel: Ein fiktives Menue 3

Abb. 1.20:

```
                MENUE 3

A - ORION 125              B - MANN-SPARKASSE
C - EIGENER PC

     # - WAEHLEN SIE MENUE 1 BIS 3

                (X = ENDE):
```

Damit Sie einmal ein praktisches Beispiel sehen können, gehen wir nun davon aus, daß Sie Ihr dBASE II-Programm auf einem PC installieren wollen.

Nach den vorher durchgeführten Installationsarbeiten geben Sie nun W <RETURN> ein. Danach werden Sie wie im nachfolgenden Dialog gebeten, Ihre Terminalbezeichnung einzugeben:

```
GEBEN SIE DEN TERMINALNAMEN EIN: Test-Baby

NEUER TERMINALTYP:Test-Baby
```

Nach Ihrer Eingabe kommt das Programm zurück zum Installations-Menü. Wenn Sie nun "X" eingeben, gelangen Sie ins Terminal-Menü. Das Menü, indem die neue Terminalbezeichnung angeführt wird, sieht im vorliegenden Fall dann z. B. als 'Menü 3' so aus:

Abb. 1.21:

```
                MENUE 3

A - ORION 125              B - MANN-SPARKASSE
C - EIGENER PC             D - TEST-BABY

     # - WAEHLEN SIE MENUE 1 BIS 3

                (X = ENDE):
```

Wenn Sie im Menü 3 nun „D" eingeben, wird diese Installation in die dBASE II-Auflistung eingetragen. Sollten Sie nach einer weiteren Veränderung die Taste „W" drücken und danach den Terminalnamen mit 'Test-Baby' eingeben, erscheint die Meldung: Die Konfiguration für das Terminal Test-Baby wird verändert'. Ihr Menü ändert sich jedoch nicht. Die neuen Parameter für das Terminal werden allerdings in die Datei INSTALL.DAT eingetragen, ebenso die neuen Änderungen in dBASE II, wenn die Taste „D" nochmals gedrückt wird.

Achtung: Geben Sie nicht mehr als 197 verschiedene Terminal-Namen ein.

Die CONTROL-Z Option - Neue Terminal-Art

Falls Ihnen keines der Terminals bekannt ist, die sich in der Terminalliste befinden, können Sie die Erstinstallation mit dieser Option durchführen. Sie werden in einem Dialog dann automatisch nach allen 26 Optionen gefragt, ohne in das Terminal-Installationsmenü zu gelangen. Sie können diese Option aufrufen, indem Sie "CTRL" und "Z" gleichzeitig drücken, wenn Sie sich in einem der Terminal-Menüs befinden.

Die CONTROL-W Option

Diese Option erlaubt Ihnen, Veränderungen ins dBASE II-Programm einzutragen, ohne die Daten in der INSTALL.DAT - Datei zu verändern. Das kann manchmal sehr hilfreich sein, weil Sie Änderungen so erst in dBASE II austesten können, bevor Sie die Änderungen in der INSTALL.-DAT - Datei als endgültige Version abspeichern.

Bemerkung: Dieses INSTALL-Programm läßt die im dBASE II enthaltenen Parameter jedesmal als die für Sie aktuellen zurück. Sie können diese Parameter dann mit der Modifikations-Option oder der Z-Option verändern. Sie brauchen sie also nicht in der INSTALL.-DAT - Datei zu sichern. Diese Vorgehensweise kann sehr praktisch sein, da Sie in die INSTALL.DAT-Datei eben höchstens 197 verschiedene Terminals eintragen können.

Ergebnis der Installationsprozedur

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

dBASE II lädt sich in den Speicher Ihres Rechners, meldet sich mit seinem Namen und einem kurzen Hinweis und gibt danach den „Prompt"-Punkt (.) aus, um anzuzeigen, daß es bereit für Ihre Kommandos ist.

Um Ihnen nun zu zeigen, wie leistungsfähig und leicht zu bedienen dBASE II tatsächlich ist, werden wir im nächsten Kapitel als allererstes eine Datenbank erzeugen und Daten in ihr abspeichern. Das wird nur ein paar Minuten beanspruchen.

Beispiel:

```
. MODDIFY COMMAND <RETURN>
*** UNBEKANNTER BEFEHL
MODDIFY COMMAND
KORRIGIEREN UND WIEDERHOLEN (J/N)? J
AENDERN VON :MODD
AENDERN NACH :MOD
MODIFY COMMAND
WEITERE KORREKTUREN (J/N)? N
BITTE DATEINAMEN EINGEBEN: (Jetzt ist der Befehl richtig)
```

Wenn Sie diese Option ausschalten, wird Ihnen dBASE II nur den Fehler anzeigen, Sie erhalten aber keine Möglichkeit zur Korrektur. dBASE II meldet sich dann mit dem Promptpunkt an einer neuen Eingabezeile, z.B.:

```
. MODDIFY COMMAND <RETURN>
*** UNBEKANNTER BEFEHL
MODDIFY COMMAND
```

Sie müssen nun in jedem Falle den fehlerhaften Befehl neu eintippen. Nach Ihrer Eingabe geht das Programm zum Hauptmenü zurück. Es bleibt Ihrer persönlichen Vorliebe überlassen, ob Sie mit oder ohne Fehler-Dialog arbeiten wollen.

Die W Option schreibt die Änderungen auf die Datei INSTALL.DAT

Anders als bei normalen INSTALL-Routinen werden hier die veränderten Daten in eine eigene Datei geschrieben. Dadurch ist eine dynamische Verwaltung und Erweiterung der Terminal-Liste gewährleistet. Nach einer Neuanlage wird das neue Terminal mit seinem eigenen Namen in die Datei und in das Terminal-Menü eingetragen. Sollte durch die Neueintragung das augenblicklich vorhandene Terminal-Menue vollgeschrieben werden, wird automatisch eine weitere Seite erzeugt. Eine Einschränkung besteht nur in der Begrenzung der maximalen Anzahl auf 197 verschiedene Terminals. Sie können jedoch auch ein bereits installiertes Terminal verändern, indem Sie am Ende der Installation nach „W" den Namen des entsprechenden Terminals angeben.

Bemerkung: Es ist nicht notwendig, bestehende Terminalversionen zu verändern. Sie können eins der Terminals für die Erstinstallation auswählen, danach Ihre gewünschten Änderungen durchführen und diese unter einem neuen Namen abspeichern.

Großes C-Cedille	:
Kleines U-Umlaut	: 7D
Kleines E-Acute	:
Kleines A-Circonflex	:
Kleines A-Umlaut	: 7B
Kleines A-Grave	:
Kleines A-Ring	:
Kleines C-Cedille	:
Kleines E-Circonflex	:
Kleines E-Umlaut	:
Kleines E-Grave	:
Kleines I-Umlaut	:
Kleines I-Circonflex	:
Kleines I-Grave	:
Großes A-Umlaut	: 5B
Großes A-Ring	:
Großes E-Acute	:
Kleines AE-Diphtong	:
Großes AE-Diphtong	:
Kleines O-Circonflex	:
Kleines O-Umlaut	: 7C
Kleines O-Grave	:
Kleines U-Circonflex	:
Kleines U-Grave	:
Kleines Y-Umlaut	:
Großes O-Umlaut	: 5C
Großes U-Umlaut	: 5D
Amerik. Cent-Zeichen	:
Englisches Pfund-Zeichen	:
Japanisches Yen-Zeichen	:
Pt charakter	:
Curly f	:
Kleines A-Acute	:
Kleines I-Acute	:
Kleines O-Acute	:
Kleines U-Acute	:
Kleines N-Tilde	:
Großes N-Tilde	:
Kleines unterstrichenes A	:
Kleines unterstrichenes O	:
Kopfstehendes Fragezeichen	:
Deutsches scharfes-S	: 7E

Drucker-Installationsprogramm
Einst V1.1

dBASE / Friday! verwendet intern einen andern Code für die deutschen Sonderzeichen, als Ihr Drucker.

Dieses zusätzliche Installationsprogramm ist Ihnen bei der Anpassung Ihres Druckers behilflich.

Um Ihren Drucker richtig zu installieren, starten Sie das Programm „EINST".

In der Rubrik Druckertyp ist ein Druckertyp schon installiert. Wenn Sie derzeit einen anderen Druckertyp an Ihrem Computer installiert haben, dann wählen Sie bitte mit „p" die Druckertypentabelle an.

Sollte Ihr Drucker vorhanden sein, so geben Sie zur Auswahl nur den danebenstehenden Buchstaben ein. Das Installationsprogramm wird dann automatisch aktualisiert.

Durch Eingeben des Buchstaben „X" gelangen Sie wieder ins Betriebssystem, nachdem Sie die Frage „Installation abspeichern" mit „J" beantwortet haben. Sollten Sie mit einer anderen Taste geantwortet haben, wird die gerade durchgeführte Änderung nicht abgespeichert.

Wenn Ihr Drucker nicht in der Druckertypentabelle enthalten ist, drücken Sie Bitte die Taste <Return>oder <ENTER>, um ins Hauptmenue zu gelangen. Wählen Sie dann mit „S" das Druckerdefinitionsmenue auf. Sie benötigen für diese Installation die HEX-Werte Ihres Drukkers. Als Beispiel für den nationalen deutschen 7-Bit Code ist dies:

Am Ende der Eingabe geben Sie bitte einen Namen für Ihre Druckerinstallation an. Wenn Sie dann die Frage „Soll diese Installation in die Datendatei zugefügt werden (J / N)" mit „J" beantworten, wird unter dem zuvor angegebenen Namen Ihre Installation abgespeichert.
Danach kehren Sie automatisch ins Hauptmenue zurück. Ihr gerade eingegebener Druckertyp erscheint nun auch im Hauptmenue. Wenn Sie nun das Installationsprogramm beenden wollen, geben Sie bitte „X" ein.
Auf die nun folgende Frage „Installation abspeichern" antworten Sie entsprechend Ihrer Wahl.

Während der gesamten Arbeit mit „EINST" muß auf der gleichen Diskette oder Festplatte die Overlay-Datei (DBASEOVR oder DBRUNOVR) vorhanden sein.
Dieses Installationsprogramm arbeitet unabhängig von dem Terminal-Installierungsprogramm „INSTALL".

2. Kapitel: Die Anwendung von dBASE II

Inhaltsverzeichnis

Abschnitt 1: Grundsätzliches über dBASE II

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

2. Kapitel: Die Anwendung von d BASE II

Markt&Technik
Verlag Aktiengesellschaft
Hans-Pinsel-Straße 2
8013 Haar bei München

Abschnitt 5: Zusatzfunktionen zu Programmierung und Manipulation der Datenbank

Abschnitt 3: Arithetische Funktionen und Manipulationen der Datenbank

Abschnitt 4: Die Programmierung der Datenbank

Abschnitt 1: Grundsätzliches über dBASE II

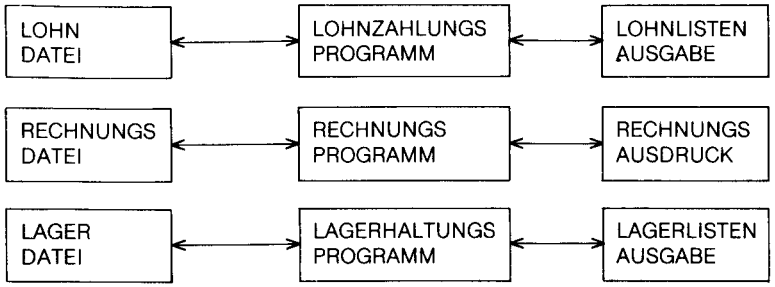
(Diese Seite wurde
absichtlich nicht bedruckt)

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, von dem nicht sicher ist, daß es 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:

(Diese Seite wurde absichtlich nicht bedruckt)

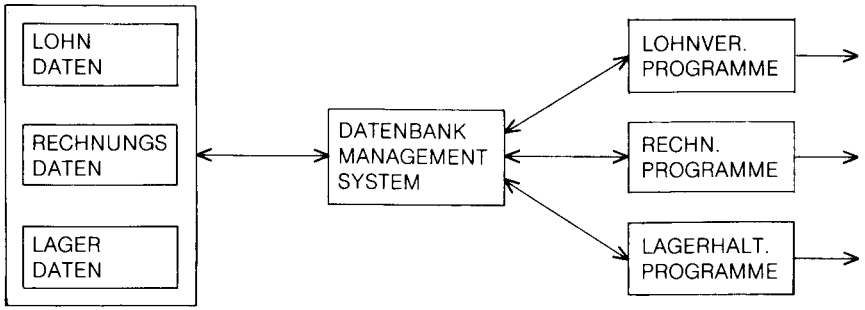
Beispiel 1.3:

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 II 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.

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: Dies 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 Auswert-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:

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 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-System 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.

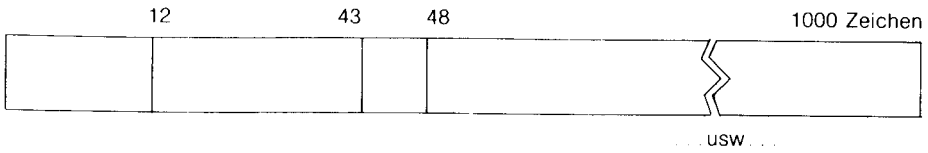
Neue Einträge zu einer Datenbank werden automatisch in den jeweils verwendeten Schlüsselfile eingetragen.

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.

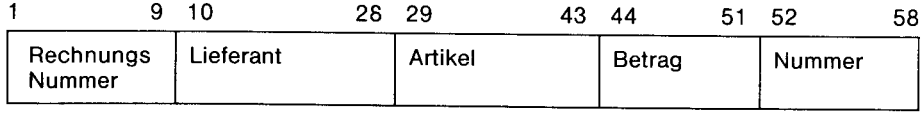
dBASE II beschränkt die Anzahl von Datensätzen in einer Datenbank auf 65 535, aber bei den für Mikrocomputern überhaupt verfügbaren Haupt- und Massenspeichern (Disketten oder Festplatten) dürfte dies kaum eine praktische Einschränkung bedeuten.

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 dazwischenliegende Anordnung.

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



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 bei dBASE II

Datei-Namen sind auf 8 Zeichen und 3 Zeichen für die Dateityp-Kennzeichnung (Extension) nach einem Punkt beschränkt. Sie können den Doppelpunkt in einem Dateinamen verwenden, aber dann können Sie diese Dateien nur unter dBASE II bearbeiten: Ihr Betriebssystem 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 aus Ihrem Betriebssystem mit einem Befehl wie PIP oder COPY 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: Das Betriebssystem beschneidet sie einfach auf die ersten 8 Zeichen. Wenn Sie Groß- und Kleinschreibung zur Benennung von Dateien benutzen, so verwandelt Ihr Betriebssystem 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 Dateityp-Kennzeichnung wird von dBASE II 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 auch mit einem Texteditor bearbeitet werden.

.CMD

Command Files oder

.PRG

Program Files (Bei Betriebssystemen. deren Befehls-Files mit .CMD gekennzeichnet sind, verwendet man bei dBASE II die Typ-Kennzeichnung .PRG)
.CMD-, bzw .PRG-Files sind: 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.

.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, Konstanten oder Variablen 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 mathematische Funktionen miteinander verknüpft werden. Die Ergebnisse befinden sich in jeweils zu definierenden Feldern.

Arithmetische Operatoren (erzeugen rechnerische Resultate)

( )	: Klammern, Zusammenfassung für Vorrangregelung
*	: 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
$	: Unterketten-Suchoperator (ist zeichenkette1 in zeichenkette2 enthalten?)

Zeichenketten-Operatoren (ergeben Zeichenketten als Ergebnis)

+	: Zeichenketten-Verkettung durch einfaches Anhängen
–	: Zeichenketten-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/zeichenkette>)	Umwandlung von Klein- in Großbuchstaben
TYPE (<ausdruck>)	Datentyp ('N' = numerisch, 'C' = Zeichen (character), 'L' = logisch, 'U' = undefiniert).
INT (<variable/ausdruck>)	Umwandlung in numerische Form
VAL (<variable/zeichenkette/unterkette>)	Umwandlung von Zeichenketten in numerisches Feld
STR (<ausdruck/variable/zahl>, <länge>, <stellen>)	numerische Form in Zeichenketten umwandeln
LEN (<variable/zeichenkette>)	Länge einer Zeichenkette
$ (<ausdruck/variable/zeichenkette>, <start>, <länge>)	Herauskopieren einer Unterkette
@ (<variable1/zeichenkette1>, <variable2/zeichenkette2>)	Unterketten-Suche (Position)
CHR (<zahl>)	Zahl in ASCII umwandeln
& <var>	Makro-Aufruf
FILE (<"dateiname"/var/ausdruck>)	Abfrage, ob Datei existiert
TRIM (<var/zeichenkette>)	nachfolgende Leerzeichen entfernen
RANK (<zeichenkette/var>)	ASCII-Wert eines Zeichens
TEST (<ausdruck>)	Zulässigkeit eines Ausdrucks prüfen

Zusammenfassung der Befehle von dBASE II

In der nun folgenden Zusammenfassung werden diese Abkürzungen benutzt:

<ausdr> = Ausdruck
<var> = Variable
<zeichk> = Zeichenkette
<koord> = Koordinaten

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.

? <ausdr> [liste]>
Zeigt einen Ausdruck (oder eine Liste, durch Kommata getrennt) an.

@ <koord> [SAY <ausdr> [USING 'bild']]
[GET <var> [PICTURE 'bild']]
Formatiert Bildschirm oder Drucker-Ausgabe.

ACCEPT [zeichk] **TO** <var>
Eingabe einer Zeichenkette von der Tastatur, ohne Anführungszeichen.

APPEND [BLANK]
Anfügung eines [leeren] Datensatzes

APPEND FROM <dateiname> [SDF] [FOR <ausdr>] [WHILE <ausdr>] [DELIMITED]
Fügt Datensätze nach Maßgabe der Optionen an eine Datenbank an.

CANCEL
Abbruch der Ausführung einer Programm-Datei.

CHANGE [<bereich>] FIELD <liste> [FOR <ausdr>]
Führt mehrfache Änderungen in einer Datenbank aus.

CLEAR [GETS]
Schließen der eröffneten Dateien und Löschen der erzeugten Speichervariablen in dBASE II. Mit GETS werden alle Get-Anweisungen gelöscht; die Ausgabe auf dem Bildschirm selbst bleibt erhalten.

CONTINUE
Setze eine LOCATE-Anweisung fort.

COPY [<bereich>] TO <dateiname> [STRUCTURE] [FIELD <liste>] [FOR <aus>] [SDF] [WHILE <ausdr>] [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 <ausdr>] [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 <ausdr>] [While <ausdr>]
Markiere bestimmte Datensätze als gelöscht.

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

DELETE FILE LIKE <kürzel dateiname??.dateitypkennzeichnung>
Löscht wie im Betriebsystem alle Dateien, deren Namen den Kürzel enthalten.(Der „*" darf nicht verwendet werden!)

DISPLAY [<bereich>] [FOR <ausdr>] [liste] [OFF] [Fields <liste>]
[WHILE <ausdr>]
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 ein dBASE II-Programm aus.

DO CASE <ausdr>
Führt für jeden Ausdruck eine oder mehrere Anweisungen aus.

DO WHILE <ausdr>
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 Anweisungsblock in einer DO WHILE-Anweisung.

ENDIF
Begrenzungs-Zeichen für den Anweisungsblock 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 kann ohne Anführungszeichen angegeben werden).

HELP
Anzeige von Kurzinformationen zu einzelnen dBASE II-Befehlen

GO oder **GOTO** [RECORD], oder [TOP], oder [BOTTOM], <n>
Gehe zu einer bestimmten Position (Datensatznummer) in einer Datenbank.

IF <ausdr>
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 <ausdr> [FIELDS <liste>]
Erzeugt eine Datenbank, die aus den einander entsprechenden (gleichen) Datensätzen zweier verschiedener Datenbanken besteht.

LIST [<bereich>] [FOR <ausdr>] [liste] [OFF] [WHILE <ausdr>]
[Fields <liste>]
Listet Datensätze aus.

LIST STRUCTURE
Listet Dateistruktur auf.

LIST MEMORY
Listet Speicherinhalt auf.

LIST FILES [ON <laufwerk>] [LIKE <dateimaske>]
Listet Dateienverzeichnis auf.

LOCATE [<bereich>] [FOR <ausdr>]
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 <auszuführende befehls-,programmliste>]
(Dieser Befehl ist auch unter CP/M-86 Version1.1 und MS-DOS ab Version 2.0 implementiert).
Beendet dBASE II-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 <ausdr> [, <feld> WITH <ausdr>...]
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 <ausdr>]
[WHILE <ausdr>]
Erzeugt Auswertungen der Datenbank-Informationen.

RESET [<laufwerk>]
Teilt dem Betriebssystem 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>]
[ALL EXCEPT <bereich>]
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 II-Parameter.

SET <parameter> TO <wert>
Umschaltung verschiedener dBASE II-Parameter auf vorbestimmten Wert.

SKIP [[-/+] <ausdr/nummer>]
Springt um eine entsprechende Anzahl von Datensätzen in der Datenbank vor- oder rückwärts.

SORT ON <schlüsselfeld> TO <dateiname> [ASCENDING]
[DESCENDING]
Erzeugt eine Datenbank, die nach dem Schlüssel-Feld sortiert ist.

STORE <ausdr> TO <var>
Legt einen Wert in einer Speichervariablen ab.

SUM [<bereich>] <feld [, liste]> [TO <var [, liste]>] [FOR <ausdr>]
[WHILE <ausdr>]
Bildet Gesamtsummen der Felder in einer Datenbank.

TEXT
Bringt den folgenden Textblock zur Anzeige.

TOTAL TO <dateiname> ON <schlüssel> [FIELDS <feld [, liste]>]
[FOR <ausdr>] [WHILE <ausdr>]
Erzeugt eine Datenbank mit (Teil)-Summen von Datensätzen.

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

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 ("I" = logged in) oder dem angegebenen Laufwerk gespeichert sind. Stattdessen 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 II-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 II-Befehle.

CHANGE ... FIELD
Editierung auf der Grundlage von Feldern statt Datensätzen.

@ <koord> [SAY ['nachricht']] [GET <var> [PICTURE]]
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 [+ <ausdr>]
springt um 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 <zeichk>
sucht einen Datensatz in einer indizierten Datenbank nach dem Wert des Schlüssels.

LOCATE FOR <ausdr>
CONTINUE
Findet den Datensatz, der mit der Bedingung <ausdr> übereinstimmt.

Ausgabe-Befehle:

?, DISPLAY, LIST
zeigen (Werte von) Ausdrücken, Datensätzen, Variable, Strukturen.

REPORT [FORM <formatname>]
erzeugt Datenbankauswertungen

@ <koord> **SAY** <var/ausdr/zeichk>
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 II-Programme werden in sog. "command files" (Befehls-Dateien) gespeichert, deren Namen die Erweiterung ".CMD" bzw. ".PRG" besitzen.)

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.

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

DO CASE
CASE <ausdr>
auszuführende Befehle
...
OTHERWISE

ENDCASE
Ähnlich wie IF ENDIF, aber ohne Verschachtelung.

DO WHILE <bedingungen>
[LOOP]
zu wiederholende Anweisungen <Abbruchkriterium>.

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

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 II gekauft haben.

(Diese Seite wurde
absichtlich nicht bedruckt)

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 ebensowenig 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 und Anwender-Programmen (dBASE II, Ihre Datenbänke und speziellen Datenbank-Programme), die alle auf Diskette gespeichert werden, klar.

So ist dBASE II ein bestimmtes Programm, das auf Ihrem Computer läuft und der Vermittlung durch das jeweilige Betriebssystem bedarf. 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 II 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“),“BACK-SPACE“, “CONTROL“ oder „CTRL“ oder ähnliches. 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.

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 Karteischrank 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“, “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).

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. „%" 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-Struktur 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.

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 Betriebssystem-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 des Betriebssystems 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 II verlassen und ins Betriebssystem zurückkehren.

Lesen Sie bitte die jeweilige Anleitung für Ihr Betriebssystem genau durch 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:

BITTE DATEINAMEN EINGEBEN:

(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 im Allgemeinen bis zu 8 Zeichen lang sein (diese Begrenzung besteht in den meisten Betriebssystemen). 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 II - Files enthält, kein Platz mehr ist, können Sie auch die Bezeichnung eines anderen Laufwerks voranstellen, z. B.:
> 'B:Namen'
> Sie können auch - wohlgemerkt auf Ihrer Arbeitskopie! - alle nicht benötigten Files, deren Namen auf „.DBF" (Datenbank-File), ".FRM" (Report-File), ".CMD", bzw. „.PRG" (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 Dateityp-Kennzeichnung ("extension") Ihres Betriebssystems und steht als Abkürzung für "Datenbank File". Diese Ergänzung des Dateinamens, um die Sie sich jetzt nicht weiter zu kümmern brauchen, fügt dBASE II automatisch an - auch, wenn Sie später die Datenbank wieder aufrufen, brauchen Sie nur „namen" einzugeben.

> 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 auch als Record bezeichnet (vergl. Abschnitt 1, Datenbank-Grundlagen). In unserem Beispiel hat jeder Datensatz 6 Felder. dBASE II 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

```
. create <RETURN>                    (= erzeuge Datenbank)
BITTE DATEINAMEN EINGEBEN: namen <RETURN>
SATZSTRUKTUR FOLGENDERMASSEN EINGEBEN:
  FELD NAME, TYP, LÄNGE, DEZIMALSTELLEN
  001
```

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

Unter dem Wort „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 für 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 (vgl. hierzu Abb. 2.2):

Abb. 2.2

```
. create
BITTE DATEINAMEN EINGEBEN: namen
SATZSTRUKTUR FOLGENDERMASSEN EINGEBEN:
     FELD      NAME, TYP, LÄNGE, DEZIMAL STELLEN
     001       name,c,20
     002       adresse,c,20
     003       postltzahl,c,4
UNZULÄSSIGER FELDNAME
     003       postltzahl,c,4
     004       stadt,c,20
     005       telefon,c,10
     006       bundesland,c,3
     007       <RETURN>
DATEN JETZT EINGEBEN?
j
```

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 "unzulässig" 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.

Die Datei <namen.dbf> ist sogleich für die Erfassung von Daten bereit, drücken Sie daher 'j' (= 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 drükken. 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 und alle Felder des Datensatzes in der linken oberen Ecke des Bildschirmes, 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 „J" - CHANGE/MODIFY" in der Anpassungsprozedur wählen (siehe Kapitel 1).

Abb. 2.3:

```
SATZNUMMER 00001

NAME       :                    :
ADRESSE    :                    :
POSTLTZAHL :    :
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 Vorsatz „<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:

Ein Feld nach oben

↑
E

Ein Zeichen nach links ←S D→ Ein Zeichen nach rechts

X
↓

Ein Feld nach unten

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 <DEL> 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 <W> ('<CTL>-W').

dBASE II 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 II-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 II-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' (benutze (die Datei) namen)
'edit <nummer>' (editiere (Datensatz Nr.:) <nummer>)

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

dBASE II 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:

```
SATZNUMMER 00003                                    GELÖSCHT

NAME            :Camphausen, Eva        :
ADRESSE         :Muellerstr. 7          :
POSTLTZAHL      :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 "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 EINFÜGEN 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> finden! Vielleicht müssen Sie auch <CTL>-"-", d. h. CONTROL-Taste und Bindestrich, verwenden!

<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).

<CTL>-W speichert ab und kehrt zur dBASE II-Kommandoebene zurück.
<CTL>-U schaltet die Löschmarkierung für einen Datensatz ein/aus.
>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.

'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:

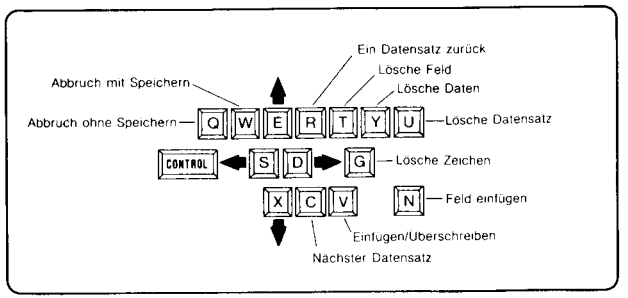
<CTL>-C schreibt den Datensatz auf die Diskette und geht zum nächsten Datensatz über.
<CTL>-R schreibt den Datensatz auf die Diskette und geht zum nächsten Datensatz über.
<RETURN> während sich der Cursor auf der ersten Position eines neuen Datensatzes befindet, kehrt zur dBASE II-Kommandoebene zurück.
<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 II 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 II, Ihre Eingabe als Befehl auszuwerten.

Um dem dBASE II-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 Vorbeirollen 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
*** UNBEKANNTER BEFEHL
edut 3
KORRIGIEREN UND WIEDERHOLEN? (J/N)? j
ÄNDERN VON: u
ÄNDERN NACH: i
edit 3
WEITERE KORREKTUREN? (J/N)? n
```

Wenn dBASE II einen Befehl nicht erkennt, so wiederholt es ihn mit einer entsprechenden Fehlermeldung. Wenn Sie ihn korrigieren wollen, brauchen Sie nicht den ganzen Befehl neu einzugeben. Als Antwort auf „ÄNDERN 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 „ÄNDERN NACH:" 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. Wenn die Breite Ihres Bildschirms für Ihren Befehl nicht ausreicht, 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, können Sie bei der Formulierung von Bedingungen für den selektiven Zugriff auf Daten die relationalen (Vergleichs-)Operatoren verwenden. Die Bearbeitung Ihrer Daten wird dadurch im Vergleich zu anderen Datenbanksystemen erheblich vereinfacht und beschleunigt.

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

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 [<bereich>] [FOR <ausdr>] [<liste>] [OFF]
(WHILE <ausdr>] [Feld <liste>]"

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, um 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 auf, 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 auslisten, 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'	(benutze die Datei „namen")
'list'	(liste alle Datensätze auf)
'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 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 im Betriebssystem üblichen Doppelpunkt weg!!

Abb. 2.8

```
. use namen
. list structure

STRUKTURDATEN FüR DATEI: NAMEN.DBF
ANZAHL DER SÄTZE: 00008
DATUM DER LETZTEN AKTUALISIERUNG: 11/25/82
PRIMÄRE DATEI
FELD        NAME          TYP     LÄNGE    DEZIMALSTELLEN
001         NAME          C         020
002         ADRESSE       C         020
003         POSTLTZAHL    C         004
004         STADT         C         020
005         TELEFON       C         010
006         BUNDESLAND    C         003
** GESAMT**                         00078

. list files
DATENBANK          # SÄTZE   LETZTE ÄNDERUNG
TEST        DBF    00002          00/00/00
ADRESSEN    DBF    00097          11/23/82
NAMBAK      DBF    00010          11/01/82
AUSGABEN    DBF    00000          10/19/82
AUFTRAEG    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:

DISPLAY [RECORD n] [ALL] [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 „SATZ 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- oder WHILE-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" Ihres Betriebssystems verwenden (soviel wie "JokerZeichen", 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 (analog für „.PRG"). Wenn Ihnen das nicht vertraut ist, schlagen Sie bitte in Ihrem Betriebssystem-Handbuch nach und benutzen Sie dieses Schema:

DISPLAY FILES LIKE <Joker-Zeichen>.

Mit anderen Worten: Das Sternchen „*" steht als abkürzende Stellvertretung für Teile eines Dateinamens, die Sie für den obigen Zweck der Bequemlichkeit halber nicht ausschreiben wollen. Der Teil endet vor dem (.) der Dateityp-Kennzeichnung oder beginnt danach. Es können beide Teilbereiche mit dieser Option verarbeitet werden.

(like = wie; alles was aussieht 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. Datensatz)
'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 lassen. 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
SATZ: 00005
. display

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

. skip
SATZ: 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 II 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):

'? 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'	(benutze 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 '?' über weitere dBASE II-Funktionen verfügen können und wie Sie damit auf dem Bildschirm Anweisungen an den jeweiligen Benutzer 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 <Dateiname> eingeben, dann tippen Sie das Kommando APPEND (hänge an):

'use namen'
'append'

Abb. 2.14

```
. use namen
. append

SATZNUMMER: 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, die aus Zeichenketten ("C", also nicht numerische „character"-Felder) bestehen, 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 letzen 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
*** ENDE DES dBASE II LAUFES ***
```

In der Befehlsart INSERT kehrt dBASE II, 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>-U ein.

Um mehr als einen Datensatz zu löschen benutzen Sie die Form: 'delete <bereich>'. Die Bereichsangabe wird genauso wie bei anderen dBASE II-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").

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'        (benutze 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 LÖSCHUNG(EN)
. delete record 4
00001 LÖSCHUNG(EN)
. 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 REAKTIVIERUNG(EN)
. 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	1642 Oberhof	456	BAY


```
. pack
"PACK" DURCHGEFÜHRT 00010 SÄTZE KOPIERT
. 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 hinzufügen 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 (SPRINGE) 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
BITTE DATEINAMEN EINGEBEN: ausgaben
SATZSTRUKTUR FOLGENDERMAßEN EINGEBEN:
FELD          NAME,TYP,LÄNGE, DEZIMALSTELLEN
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
DATEN JETZT EINGEBEN? N
```

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

Abb. 2.18

```
. create
BITTE DATEINAMEN EINGEBEN: auftraeg
SATZSTRUKTUR FOLGENDERMAßEN EINGEBEN:
FELD          NAME, TYP, LÄNGE, DEZIMALSTELLEN
001           kundnumr,c,9
002           artikel,c,20
003           menge,n,4
004           preis,n,7,2
005           betrag,n,9,2
006           ruecksend,1,1
007           auftrdat,c,6
008
DATEN JETZT EINGEBEN? N
```

Abschnitt 3: Arithmetische Funktionen und Manipulationen der Datenbank

(Diese Seite wurde
absichtlich nicht bedruckt)

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 II-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 II-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 II-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.

(Diese Seite wurde
absichtlich nicht bedruckt)

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 erkären. 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, z. B. "3", oder die logischen Werte „T" (True = Richtig) und „F" (False = Falsch).

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 II 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.

Zeichen (character) und Zeichenketten (character strings), bestehen aus allen druckbaren Zeichen einschließlich Leerzeichen (Zwischenraum), sie können ebenfalls Konstanten sein, müssen aber etwas anders behandelt werden.

Zeichenketten sind also einfach eine Aneinanderfügung von Zeichen (einschließlich Leerzeichen, Ziffern und Sonderzeichen), die bearbeitet oder als Daten benutzt werden können. Eine Unterkette ist ein Teil einer bestimmten Zeichenkette.

Wenn ein Zeichen (Character) oder eine Zeichenkette als Zeichenketten-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).

Variable 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
SATZ: 00004
. ? name
Dollinger, Erwin
. use
. ? name

*** SYNTAXFEHLER ***
  ?
? NAME
KORRIGIEREN UND WIEDERHOLEN? 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 jeder gültigen dBASE II-Bezeichnung 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 Eingabe-Daten getrennt von Datenfeldern zu behandeln. Beispielsweise können wir während einer Sitzung (mit dem dBASE II-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 der Zeichenkette (die eine Konstante darstellt) in der ersten Zeile von Beispiel 3.2 verwendet haben, weil wir das einfache Anführungszeichen als Apostroph im Inneren der Zeichenkette 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

*** SYNTAXFEHLER ***
        ?
? KONSTANTE/3
KORRIGIEREN UND WIEDERHOLEN? 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    VARIABLEN BENUTZT 00051 BYTES BELEGT
```

Wenn Sie in eine Variable Daten einspeichern, entnimmt dBASE II der Form dieser Daten automatisch den Datentyp. Von diesem Augenblick an können Sie die Datentypen nicht mehr mischen (etwa, indem Sie eine Zeichenkette 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 löschen.

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 (Zeichenketten-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.

Regeln: 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 löschen, indem Sie 'release <variablen-name>' befehlen, oder Sie können alle Speichervariablen löschen durch 'release all'.

Geben Sie folgendes ein (evtl. 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 als
> =	: größer oder gleich als

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 84 00 33  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 14 12 53  BAY
00002 Camphausen, Eva   Muellerstr. 7     Berlin 2          1000 84 00 33  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 einer Unterkette (Test auf
          Enthaltensein)
```

Die Eingabe:

Beispiel 3.5:

```
'list for (auftrnumr = 730 .or. auftrnumr = 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!)

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 Auftragsnummer gleich 730 (wahr oder falsch)
2.) Ist die Auftragsnummer 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 Klammerausdruck 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 (Auftragsnummer = 730 ODER Auftragsnummer = 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: „Auftragsnummer = 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 postitzahl > '5' .and. postitzahl <'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       14 12 53   BAY
00002 Camphausen, Eva     Muellerstr. 7      1000 Berlin 2         84 00 33   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       14 12 53   BAY
00002 Camphausen, Eva     Muellerstr. 7      1000 Berlin 2          84 00 33  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 ausgelistet. 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 $-Unterketten-Such-Operator ist sehr hilfreich durch seine Fähigkeit, zu entscheiden, ob eine bestimmte Zeichenfolge in einer Zeichenkette vorkommt. Sein Format lautet:

"<unterkette > $ <zeichenkette>"

Der Operator sucht die links stehende Unterkette in der Zeichenkette auf der rechten Seite. Einer oder beide Terme können Zeichenketten-Variable oder -Konstante sein. Das Ergebnis der Operation ist „wahr", wenn die Unterkette in der rechts stehenden Vergleichszeichenkette gefunden wird, "falsch", wenn sie nicht darin vorkommt. Verwechseln Sie den logischen Zeichenketten-Suchoperator, der die Frage „ist eine Unterkette in einer anderen Zeichenkette enthalten" mit wahr oder falsch beantwortet, nicht mit den nachfolgend besprochenen Zeichenketten-Operatoren, die Zeichenketten 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       14 12 53   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 Konstanten", 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'

Zeichenketten-Operatoren liefern Zeichenketten als Ergebnis.

+ = Zeichenketten-Verkettung (exakt)
- = Zeichenketten-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 Zeichenketten-Operatoren tun: "+" verknüpft Zeichenketten, wie sie vorliegen, "-" schiebt nachfolgende Leerzeichen ans Ende. Daher sammeln sich in Zeichenketten, die mehrmals mit "-" verknüpft wurden, alle nachfolgenden Leerzeichen an - es werden immer mehr. Es hilft auch nichts, zwischen '...die Adresse lautet' und „+ adresse" ein „-" zu setzen - dann werden die beiden Zeichenketten 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 aneinander. Das „plus"-Zeichen verbindet die beiden Zeichenketten genau so, wie sie vorliegen. Das „minus"-Zeichen dagegen schiebt nachfolgende Leerzeichen in der linken Zeichenkette an das Ende der neuen Zeichenkette. 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'
'J'
```


Abb. 3.8

```
. use ausgaben
. list structure
STRUKTURDATEN FÜR DATEI:  AUSGABEN.DBF
ANZAHL DER SÄTZE: 00000
DATUM DER LETZTEN AKTUALISIERUNG: 11/25/82
PRIMÄRE DATEI
    FELD        NAME          TYP    LÄNGE     DEZIMALSTELLEN
    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 LÖSCHT ALLE DATENSÄTZE ....WEITER (J/N) J
```

Wenn Sie mit „j" antworten, erscheint die Maske für Ihre Änderungen (Hier z. B. „AUFTRNUMR" in „AUFTR:NR"). Nach der Eingabe der Änderungen beenden Sie den dBASE II-Durchlauf mit quit:

```
. quit
*** ENDE DES dBASE II-LAUFES ***
```

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 (nicht die Erläuterungen in Klammern!):

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

Abb. 3.9

```
. use namen
. copy to temp
00010 SÄTZE KOPIERT
. use temp
. display structure
STRUKTURDATEN FÜR DATEI: TEMP.DBF
ANZAHL DER SÄTZE:  00010
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
   FELD        NAME          TYP      LÄNGE    DEZIMALSTELLEN
   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     14 12 53  BAY
00002 Camphausen, Eva    Muellerstr. 7    1000 Berlin 2       84 00 33  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'          (kopiere (nur) die Datenstruktur)
'use temp'
'display structure'
```

Abb. 3.10

```
. use namen
. copy to temp structure
. use temp
. display structure
STRUKTURDATEN FÜR DATEI: TEMP.DBF
ANZAHL DER SÄTZE: 00000
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
  FELD        NAME          TYP     LÄNGE    DEZIMALSTELLEN
  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
STRUKTURDATEN FÜR DATEI:  TEMP.DBF
ANZAHL DER SÄTZE: 00000
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
  FELD        NAME          TYP     LÄNGE    DEZIMALSTELLEN
  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 SÄTZE KOPIERT
. use neu
. display structure
STRUKTURDATEN FÜR DATEI:  NEU.DBF
ANZAHL DER SÄTZE: 00006
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
  FELD      NAME          TYP     LÄNGE     DEZIMALSTELLEN
  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", "TYP", "LÄNGE" und „DEZIMALSTELLEN" 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 ein Benutzer die Struktur einer neu anzulegenden Datei eingeben kann, ohne daß er die Bedienung von dBASE II lernen muß. Das Programm würde dann an seiner 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 II 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 (ohne die Erläuterungen in Klammern!):

'use namen'	(benutze „namen")
'copy to temp structure'	(kopiere die Struktur nach „temp")
'use temp'	(benutze „temp")
'modify structure'	(verändere die Struktur)
'J'	(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 LÖSCHT ALLE DATENSÄTZE ....WEITER (J/N) J
. display structure
STRUKTURDATEN FÜR DATEI: TEMP.DBF
ANZAHL DER SÄTZE: 00000
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
  FELD      NAME          TYP     LÄNGE     DEZIMALSTELLEN
  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 SÄTZE HINZUGEFÜGT
. list
00001 Adams, Peter        Bergstrasse 15    8000 Muenchen 2      14 12 53   BAY
00002 Camphausen, Eva     Muellerstr. 7     1000 Berlin 2        84 00 33   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
```

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 SÄTZE KOPIERT
. use namen
. display structure
STRUKTURDATEN FÜR DATEI:  NAMEN.DBF
ANZAHL DER SÄTZE: 00010
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
  FELD      NAME          TYP     LÄNGE     DEZIMALSTELLEN
  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      14 12 53   BAY
00002 Camphausen, Eva     Muellerstr. 7     1000 Berlin 2        84 00 33   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>'	(verwende <alterfile>)
'copy to <neuerfile> structure'	(kopiere Struktur auf <neuerfile>)
'use <neuerfile>'	(verwende <neuerfile>)
'modify structure'	(verändere Struktur)
'append from <alterfile>'	füge Daten aus <alterfile> hinzu)
'copy to <alterfile>'	(kopiere nach <alterfile> zurück)

Austausch von dBASE II-Dateien und Betriebssystem-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.

Im Betriebssystem beinhaltet das Standard-Daten-Format (abgekürzt als SDF unter 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 des Betriebssystems) um sich den File anzuschauen. Verlassen Sie dazu dBASE II mit „QUIT" und geben ein: TYPE SYSDATA.TXT.

Sie werden feststellen, daß Sie geradeso damit arbeiten können, als hätten Sie die Datei im Betriebssystem erzeugt.

Das Standard-Daten-Format erlaubt dBASE II auch, mit Daten aus Betriebssystem-Dateien 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 Zeichenketten 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).

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 Zeichenketten 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]


APPEND FROM <filename.TXT> [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).

Beachte: Obwohl dBASE II automatisch Dateityp-Kennzeichnungen (extensions) für die Dateien, die es selbst erzeugt, einsetzt, müssen sie die Dateityp-Kennzeichnung „.TXT" angeben, wenn Sie von einem Betriebssystem-File mit APPEND Daten zu einer Datenbank hinzufügen.

Beachte: 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 „auftraeg" zu ändern, so daß die Felder einander entsprechen.

Abb. 3.15

```
. use namen
. copy to temp sdf
00014 SÄTZE KOPIERT
. modify structure
MODIFY LÖSCHT ALLE DATENSÄTZE ....WEITER (J/N) J
. append from temp.txt sdf
00014 SÄTZE HINZUGEFÜGT
. display structure
STRUKTURDATEN FÜR DATEI:  NAMEN.DBF
ANZAHL DER SÄTZE: 00014
DATUM DER LETZTEN AKTUALISIERUNG: 11/30/82
PRIMÄRE DATEI
  FELD        NAME          TYP     LÄNGE    DEZIMALSTELLEN
  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 evtl. 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>.)] [WHILE <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 „AUFTRAEG" 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 ERSETZUNG(EN)
. 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: "VERÄNDERN?"

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 VERÄNDERN? 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

SATZ: 00001


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

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

SATZ: 00002


KUNDCODE:
VERÄNDERN? ' '
NACH        1002

KUNDCODE: 1002
ÄNDERN? <RETURN>
```

Hinweis: Wenn Sie nur eine kleine Anzahl von Datensätzen ändern wollen, empfiehlt sich die Verwendung des BROWSE-Befehls.

Abb. 3.18

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

SATZ: 00001
NAME: Adams, Peter
VERÄNDERN? Adams
NACH   Adam
NAME: Adam, Peter
VERÄNDERN? <RETURN>
ADRESSE: Bergstrasse 15
VERÄNDERN? 15
NACH   30
ADRESSE: Bergstrasse 30
VERÄNDERN? <RETURN>
STADT: Muenchen 2
VERÄNDERN? Muenchen
NACH   München
STADT: München 2
VERÄNDERN? <RETURN>

SATZ: 00002
NAME: Camphausen, Eva
VERÄNDERN? Cam
NACH   Kam
NAME: Kamphausen, Eva
VERÄNDERN? <RETURN>
ADRESSE: Muellerstr. 7
VERÄNDERN? Mue
NACH   Mü
ADRESSE: Müllerstr. 7
VERÄNDERN? <RETURN>
STADT: Berlin 2
VERÄNDERN? 2
NACH   3
STADT: Berlin 3
VERÄNDERN? <RETURN>

SATZ: 00003
NAME: Dirschedel, Fritz
VERÄNDERN? Fritz
NACH   Bernd
NAME: Dirschedel, Bernd
VERÄNDERN? <RETURN>
ADRESSE: Magnolienweg 20
VERÄNDERN? <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 alphanumerisch (Zeichenkette) 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 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 off'
'use namen'
'sort on name to temp'
'use temp'
'list off'
'copy to namen'
```


Abb. 3.19

```
. use namen
. sort on stadt to temp
SORT BEENDET
. use temp
. list off
Camphausen, Eva    Muellerstr. 7      1000 Berlin 2          84 00 33  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        14 12 53  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 BEENDET
. use temp
. list off
Adams, Peter       Bergstrasse 15     8000 Muenchen 2        14 12 53  BAY 1001
Camphausen, Eva    Muellerstr. 7      1000 Berlin 2          84 00 33  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 SÄTZE KOPIERT
```

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 Dateityp-Kennzeichnung „.NDX" 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.

Bei der Indizierung werden die Datensätze in der Datenbank also 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 SÄTZE INDIZIERT
. use namen index postleit
. list

00002 Camphausen, Eva  Muellerstr. 7      1000 Berlin 2          84 00 33 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        14 12 53 BAY 1001
00003 Dirschedel, Fritz Magnolienweg 20   8000 Muenchen 90       97 81 13 BAY 1003
00011 Haberer, Karl    Teufelsweg 13      8047                   12 45 7  BAY 1008
                                          Schrobenhausen
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 SÄTZE INDIZIERT
. 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      14 12 53  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        84 00 33  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 ausgelistet 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"

Hinweis: Obwohl für den Text des Indexfelds sowohl Klein- als auch Großbuchstaben zugelassen sind, empfiehlt es sich in der Praxis, ausschließlich Großbuchstaben zu verwenden!

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 neu indizieren.

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 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          84 00 33 BLN 1002

. find 8
. display

00001 Adams, Peter      Bergstrasse 15   8000 Muenchen 2        14 12 53 BAY 1001

. display next 3

00001 Adams, Peter      Bergstrasse 15   8000 Muenchen 2        14 12 53 BAY 1001
00003 Dirschedel, Fritz Magnolienweg 20  8000 Muenchen 90       97 81 13 BAY 1003
00011 Haberer, Karl     Teufelsweg 13    8047                   12 45 7  BAY 1008
                                         Schrobenhausen
```


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 „WERT NICHT 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 evtl. 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        14 12 53 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                   12 45 7  BAY 1008
                                         Schrobenhausen
00013 Jahn, Alois       Huehnersteig 3   6142 Oberhof           456      BAY 1010

. find 1006
WERT NICHT GEFUNDEN
. find BAY 1006
WERT NICHT GEFUNDEN
. 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> FOR <ausdruck>". 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 hinausschieben 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 SATZ 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: DATEIENDE ERREICHT.

Abb. 3.24

```
. use namen
. locate for name = 'Gol'
SATZ: 00008
. display
00008 Goldig, Nicole      Radweg 73    8400 Regensburg      29 42 31 BAY 1012

. locate for postltzahl > '6' .and. name < 'F'
SATZ: 00001
. display name, postltzahl
00001 Adams, Peter      8000
. continue
SATZ: 00003
. display name, postltzahl
00003 Dirschedel, Fritz   8000
. continue
SATZ: 00004
. display name, postltzahl
00004 Dollinger, Erwin   7432
. continue
DATEIENDE ERREICHT
```

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 "auftraeg", die wir bisher als Beispiele benutzt haben, enthalten zu wenig Daten, um Ihnen die volle Leistungsfähigkeit von dBASE II zu zeigen.
In den folgenden Beispielen wird die Datenbank "ausgaben.dbf" benutzt. Wenn Sie damit üben wollen, geben Sie bitte folgende Daten ein:

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  Winkeleisen          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
BITTE DATEINAMEN FÜR BERICHT EINGEBEN: auftrko
BITTE ANGEBEN: M=LINKER RAND, L=ZEILEN/SEITE, W=ZEILENBREITE L=70

MIT SEITENÜBERSCHRIFT? (J/N) J
BITTE SEITENÜBERSCHRIFT EINGEBEN: Kosten-Übersicht
LEERZEILE ZWISCHEN DEN SÄTZEN?(J/N)N
GESAMTSUMMEN ERFORDERLICH? (J/N) J
BERICHT MIT ZWISCHENSUMMEN? (J/N) N
SPALTE  BREITE  INHALT
001     10,scheck:dat
BITTE ÜBERSCHRIFT EINGEBEN: Datum
002     22,name
BITTE ÜBERSCHRIFT EINGEBEN: Lieferant
003     22,beschr
BITTE ÜBERSCHRIFT EINGEBEN: Beschreibung
004     4,auftr:nr
BITTE ÜBERSCHRIFT EINGEBEN: ANr.
GESAMTSUMMEN ERFORDERLICH? (J/N) N
(Diese Frage wird bei allen numerischen Feldern
einzeln gestellt)
005     12,betrag
BITTE ÜBERSCHRIFT EINGEBEN: Betrag
GESAMTSUMMEN ERFORDERLICH? (J/N) J
006     <RETURN>                          (Beenden der Report-Defintion)
        SEITE 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   Winkeleisen            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. Um den Report zu stoppen, drücken Sie die Taste <ESCAPE> (<ESC>).

Gleichzeitig speichert dBASE II das Berichts-Format in einem File mit der Dateityp-Kennzeichnung „.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>]
[WHILE <ausdruck>] [TO PRINT]"

Durch Eingabe von

'report form auftrko for auftr:nr = '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

SEITE  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 auftr:nr='4 ') usw., wie Sie es gerade 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 <ausdruck>] [TO <variable>]"

Der Befehl kann ohne, mit einigen oder allen Erweiterungen benutzt werden.

Ohne irgendwelche Angaben zählt er einfach alle Datensätze in der Datenbank. "bereich" kann auf einen oder eine bestimmte Anzahl von Datensätzen beschränkt werden, und „ausdruck" kann 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 zu erhalten, geben Sie folgenden Befehl ein:

"SUM <liste [<bereich>] [FOR <ausdruck>] [TO <variable(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 „ausdruck" nach dem Wort „FOR" speziell ausgewählt werden (Kunde <"ausdruck" oder „bereich" > 'Sem ' .AND.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 VARIABLEN BENUTZT   00012 BYTES BELEGT
```

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 <ausdruck>] [WHILE <ausdruck>]

Beachte: 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 unterdrükken und Gesamtsummen zu erhalten. Abbildung 3.29 zeigt, was mit unserer Datenbank „ausgaben" passiert:

Beispiel 3.10:

```
'use ausgaben'
'index on auftr:nr to posten'
'use ausgaben index posten'
'total on auftr:nr to temp fields betrag for auftr:nr > 699;
                                    .and. auftr:nr < 800'
'use temp'
'list'
```


Die neue Datei hat für jede Auftragsnummer einen Eintrag, und eine Gesamtsumme aller Kosten, bezogen auf die Auftragsnummer in unserer Datenbank "ausgaben". Ein Problem mit der neuen Datenbank liegt darin, daß nur zwei der Felder brauchbare Informationen 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, bei denen eine Gesamtsumme von Interesse ist.

Achtung: Das kann auch zu mysteriösen Fehlern führen, wenn sich - ohne daß Sie daran denken - 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.

Geben Sie nun zur weiteren Ausführung des TOTAL-Befehls ein:
'COPY TO temp FIELD auftr:nr, 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 (SORT-ierte oder INDEX-ierte) Information zu summieren.

Abb. 3.29

```
. use ausgaben
. index on auftr:nr to posten
00009 SÄTZE INDIZIERT
. use ausgaben index posten
. total on auftr:nr to temp field betrag for auftr:nr > 90;
                                        .and. auftr:nr < 800
00003 SÄTZE KOPIERT
. 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
DATEI WURDE GELÖSCHT
. use ausgaben index posten
. copy to temp field auftr:nr, betrag
00009 SÄTZE KOPIERT
. total on auftr:nr to temp field betrag
00003 SÄTZE KOPIERT
. use temp
. list
00001       98       1070.58
00002      103        420.73
00003      718       2598.79
```


Zusammenfassung von Abschnitt 3:

Dieser Abschnitt hat Ihre Kenntnis davon, was man mit dBASE II anfangen kann, schon erweitert.

Wir haben Ihnen gezeigt, wie verschiedene Operatoren (arithmetische, vergleichende (relationale) und Zeichenketten-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 Informationen überführen können (COUNT, SUM, REPORT, TOTAL).

Im nächsten Abschnitt werden wir zeigen, wie man unter 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

(Diese Seite wurde
absichtlich nicht bedruckt)

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 ... ENDIF), wie man ihn eine bestimmte Folge von Befehlen wiederholen läßt (DO WHILE ... ENDDO) und wie man bestimmte Aktivititäten 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 II-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 eine Betriebssystem-Datei mit der Dateityp-Kennzeichnung (extension) ".CMD", bzw. „PRG". Sie können das mit Ihrem gewohnten Texteditor oder jedem Textverarbeitungs-System tun, das ASCII-Code erzeugt. (Es geht aber auch ohne einen solchen, wie Sie bei der Behandlung des dBASE II-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.

(Diese Seite wurde
absichtlich nicht bedruckt)

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 <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 <CTRL>-W (<CTRL>-O auf dem Superbrain), um den Text abzuspeichern und zum dBASE II Punkt-Prompt zurückzukehren. Verlassen Sie dann das dBASE II-System ('quit'). Dann geben Sie ein:

```
'dbase test'
```

Wenn Sie das Programm genauso 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 5000 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 „.PRG"- bzw. „.CMD"-Dateien in Ihrem Betriebssystem aufrufen.

Wenn Sie dBASE II bereits gestartet haben (Punkt-Prompt ist sichtbar), rufen Sie ein Programm durch folgende Befehlsform auf:

```
. 'DO <programmname>'
```

wobei der <programmname> die Dateityp-Kennzeichnung „.CMD", bzw. ".PRG" hat!

Vorschlag: Vielleicht wollen Sie die Programm-Datei „dbase.com" umbenennen in „do.com", so daß Sie einfach eingeben: „do <programmname>", ohne erst darüber nachzudenken, ob Sie das Programm von Ihrem Betriebssystem, oder von dBASE II aus starten.

Um dies unter CP/M zu erreichen, schreiben Sie (nach dem Verlassen vom dBASE II-System natürlich):
A> 'REN DO.COM=DBASE.COM'

Unter CP/M-86 schreiben Sie nach dem Verlassen von dBASE II:
A> 'REN DO.CMD=DBASE.CMD'

Unter MS-DOS schreiben Sie nach dem Verlassen von dBASE II (genau in der umgekehrten Reihenfolge!):
A> 'REN DBASE.COM DO.COM'

Achtung: Achten Sie bitte darauf, daß Sie wirklich die Dateityp-Kennzeichnung verwenden, die in Ihrem Betriebssystem verwendet wird!

Natürlich müssen Sie dann in jedem Fall das dBASE II-System mit "do" starten (anstatt wie bisher mit „dbase").

Auswahl zwischen Alternativen und Fällen von Entscheidungen mit IF ... ELSE ENDIF

Die Wahl zwischen zwei Alternativen wird in dBASE II 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 II-System versteht:

Beispiel 4.2:

```
IF <bedingung> [ .AND. <bedingung2> .OR. <bedingung3> ... ]
        führe-diesen-befehl-aus
        [führe-den-nächsten-befehl-aus]
        [ ELSE ]
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. beschr = '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. 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 russische 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 FALLUNTERSCHEIDUNG
```

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 II programmieren müssen (vergleiche mit Beispiel 4.4):

Beispiel 4.6:

```
DO CASE
        CASE <bedingung 1>
             <befehl(e) 1>
        CASE <bedingung 2>
             <befehl(e) 2>
        CASE <bedingung 3>
             <befehl(e) 3>
        ...
        OTHERWISE
             <befehl 4> (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...ENDDO

Unermüdliche Wiederholungen einer Aktivität sind eines der kennzeichnendsten Leistungsmerkmale 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 abzubrechen. 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 sich 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 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 anderen 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 Auftrag A" geschrieben haben. „Auftrag A" ist eine weitere Programm-Datei (deren Name die Typ-Kennzeichnung „.CMD" ,bzw. bei einem entsprechend anderen Betriebssystem „.PRG", 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 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 Menue, 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 SYSTEM WARTET 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 = falsch).

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 den Benutzer 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 ist ACCEPT 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 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 Postltzahl'
' @  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 den Benutzer geradeso aussieht wie die alten Papier-Formulare, welche er 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> drükken. 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 Art automatische Cursorführung betreibt.

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", bzw. "versuch.PRG", der die folgenden Befehle enthält (verwenden Sie Ihren Texteditor oder MODIFY COMMAND. Bei MODIFY COMMAND geben Sie die Dateityp-Kennzeichnung „.CMD", bzw,. ".PRG" nicht an, da dBASE II dies selbst tut!!!):

Beispiel 4.12:

```
SET TALK OFF
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 „POSTLEITZAHL" GET PostltZahl
      READ

      SAY „ <S> bricht die Eingabe ab,"
      SAY „<RETURN> setzt sie fort."
      WAIT TO fortsetz
ENDDO
RETURN
```

Wenn Sie in 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 Betriebsystem 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:

(Natürlich nennen Sie es gegebenenfalls: „Beispiel.PRG")

```
********************************* 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 Unterprogramm 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 II die Ausführung von Befehlen (z. B. SKIP,
* d. h. Weiterschalten auf einen neuen Datensatz) auf dem Bildschirm. Dies
* kann verwirrend sein, daher geben wir SET TALK OFF ein
* (schalte „Rückmeldungen" 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 des Benutzers
* 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 abzubrechen.  '
  @ 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  m u s s  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 Bedienungsfehlern umgeht (Fehlermeldungen etc.).

Obwohl es nur ein fragmentarisches Programm ist, das keine wirklich nützliche Arbeit leistet, macht "Beispiel.CMD", bzw. "Beispiel .PRG" (s. o.) 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 einem „sauberen Bildschirm" 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 großgeschrieben, 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 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 erfolgen Fehlermeldungen.

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 abzubrechen, 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. "Adelheid" 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 Adelheid (ohne Anführungszeichen) dagegen verursacht eine Fehlermeldung:

SYNTAXFEHLER, EINGABE BITTE WIEDERHOLEN

und dBASE II fordert Sie auf, die Daten neu einzugeben.
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, bzw. Primäre/Sekundäre Datei, 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. dBASE II nennt die Dateien dann - 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 (PRIMÄR)-Bereich arbeiten, verwenden Sie „S." (um Datenobjekte im Hintergrund anzusprechen), wenn Sie im SECONDARY (SEKUNDÄR)-Bereich arbeiten, setzen Sie „P." vor die Namen, um Daten im Vordergrund anzusprechen.

Ein Beispiel dafür finden Sie in der Programmdatei "Zweidat.CMD" bzw. ".PRG" (im Abschnitt 4). Einzelne Datensätze in einer Datei im PRIMÄR-Bereich (PRIMARY) werden mit allen Datensätzen in einer anderen Datei verglichen, die sich im Hintergrund oder SEKUNDÄR-Arbeitsbereich (SECONDARY) befindet.

Als kleines Beispiel für die Verwendung von Vorder- und Hintergrund-Dateien folgt ein Programm, 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" (bzw. „.prg")

```
* Datei: zweidat.cmd                (bzw. .prg)
* 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" (oder .prg) (Beispiel 4.15) und „zeigaus.cmd" (oder .prg) (Beispiel 4.16), um die Datensätze darzustellen. Hier sind sie:

Beispiel 4.15: "zeignam.cmd" (bzw. „.prg")

```
* Datei: zeignam.cmd                (bzw. .prg)

@ 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" (bzw. „.prg")

```
* Datei: zeigaus.cmd                (bzw. .prg)

@ 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" (nur für 8 Bit Versionen) und wird in einem anderen Abschnitt des Handbuches besprochen.

Allgemein nützliche Systembefehle und Funktionen

MODIFY COMMAND <dateiname> läßt Sie das angegebene Programm (command file) 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 Bildschirmes zu sehen, geben Sie <CTRL>-B, um mit dem Bildschirm auf die rechte Seite zu "rollen". Mit <CTRL>-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 (den Benutzer) 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 Betriebssystem-Directory (dem Disk-Inhaltsverzeichnis) eingetragen ist. Versuchen Sie NIE eine Datei umzubenennen, die gerade bearbeitet wird!

QUIT TO '<auszuführende befehls-, programmliste>' ist auf den Betriebssystemen CP/M-86 ab Version 1.1 und MS-DOS ab Version 2.0 implementiert. Der Befehl gestattet es, von dBASE II aus automatisch im Betriebssystem die Ausführung anderer ".COM"- bzw. ".CMD"-Dateien zu starten. Jede Befehls-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 (Satz#) 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 ('Dateiende erreicht') erreicht wurde, "falsch", wenn das Ende noch nicht erreicht wurde.

Einige Worte über Programmierung und wie Sie Ihre Anwendung analysieren sollten

Hinweis: 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 „Programm" 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                          * Gehe zum ersten Datensatz
DO WHILE .NOT. EOF                * Wiederhole für ganze Datei
   IF Scheck:Nr=' '               * Wenn Rechnung unbezahlt
      DO SchrbScheck              * Schreibe einen Scheck
      DO Berichtige               * Bringe Datensätze a. neuen Stand
   ENDIF
SKIP                              * Gehe zum nächsten Datensatz
ENDDO
```

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 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 "SchrbScheck.CMD" und „Berichtige.CMD", aber das macht nichts (Gegebenfalls jeweils: „.PRG").

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 wiederspiegelt 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: Zusatzfunktionen zu Programmierung und Manipulation der Datenbank

(Diese Seite wurde
absichtlich nicht bedruckt)

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 Zeichenketten-Operatoren schwer oder gar nicht zu erreichen sind. Die Funktionen von dBASE II fallen in die gleichen drei Kategorien (numerisch, logisch und Zeichen(kette)), 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 Großschreibung 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 Zeichenketten-Konstante ("Hallo") einsetzen.

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 ausführliche Darstellung aller Funktionen und sämtlicher Anwendungsmöglichkeiten finden Sie im 3. Kapitel dieses Handbuches.

(Diese Seite wurde
absichtlich nicht bedruckt)

!(<variable/zeichenkette>)

ist eine Funktion, die Kleinschreibung in Großschreibung umwandelt. Sie verwandelt alle Zeichen von „a"..."z" in einer Zeichenkette oder Zeichenketten-Variablen in Großbuchstaben. Alle sonstigen Zeichen in der Zeichenkette 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 "ß" verwendet, so daß die lexikalische Reihenfolge von Fall zu Fall zu klären ist. Sollte dies zu Problemen führen, schreibt man besser „MUENCHEN" statt „München".

$ (<variable/zeichenkette>, <start>, <länge>)

ist die Unterketten-Funktion. Sie sucht aus einer Zeichenkette oder Zeichenketten-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 '810823' 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-Unterketten Operator, der in Abschnitt 3 beschrieben wird.

Der Unterschied ergibt sich aus der Form des Aufrufes:

```
? 'BAY' $ 'BAYERN'                (ergibt:)
.T.                               (wahr - die Zeichenkette ist enthalten)

? $ ('BAYERN', 1, 3 )             (ergibt:)
'BAY'                             (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
```

Die &-Funktion 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.

@ (<variable1/zeichenkette1>, <variable2/zeichenkette2>)

ist eine Funktion, welche eine Unterkette sucht. Sie können sich Ihre Arbeitsweise anhand der Frage "Wo befindet sich zeichenkette1 in zeichenkette2?" 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 die zuerst genannte Zeichenkette nicht als Teil der 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 eine Zeichenkette in einer anderen enthalten ist, können Sie den logischen Zeichenketten-Operator benutzen: zeichenkette1 $ zeichenkette2, siehe Abschnitt 3.
Sie werden dies in einem Programm nützlich finden, wenn der Computer nach einer Zeichenfolge sucht, ohne daß ein Benutzer 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 (26) Ihren Bildschirm löschen, CHR (14) negative Schriftdarstellung 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: ? 'zeichenkette' + 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. Ü.: 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.)

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 den Benutzer 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, der der ganzzahlige Anteil von DM 2333.75 ist.

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. INT (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/zeichenkette>)

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 Datenbank-Variable eingesetzt wird, die ein Zeichen-Feld enthält, dann ergibt diese Funktion die Größe des Feldes, nicht die aktuelle Länge des Inhalts (da alle unbenutzten Positionen von dBASE II mit Leerzeichen aufgefüllt 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 4.13 im Abschnitt 4, wo dBASE II den Datentyp einer Eingabe erkennt.

VAL (<variable/zeichenkette/unterkette>)

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 Unterketten-Operator benutzen: VAL ($(<zeichenkette>)). Das ist nützlich, wenn nur bestimmte Zeichen in der Zeichenkette die gewünschte Zahl darstellen - siehe bei der Funktion '$'.

STR (<ausdruck/variable/zahl>, <laenge>, [<dezimalstellen>])

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 Dezimalstellen. 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 eine Zeichenkette 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 Zeichenketten verwandelt und dann mit anderen Zeichenketten verknüpft (zusammengehängt), um neue Zeichenketten zu bilden, die dargestellt werden.

TRIM (<variable/ausdruck>)

entfernt die nachfolgenden Leerzeichen aus dem Inhalt einer Zeichenketten-Variablen. Das kann durch Angabe von

```
'STORE TRIM (<variable>) TO <neue-variable>
```

ausgeführt werden.

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 (von dBASE II voreingestellten) Standardwerte hervorgehoben.

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 „.TXT", 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>".

SET BELL **ON** 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 ON 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 bildschirmorientierten Editierung ein Eingabenfeld verläßt.
OFF Springt automatisch aus dem Feld heraus, wenn es voll ist.

SET CONSOLE **ON** Gibt alle Ausgaben auf Ihrem Bildschirm wieder.
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 abzubrechen.
OFF macht die <ESCAPE>-Taste unwirksam.

SET EXACT ON Erfordert, daß alle Zeichen in einem Vergleich zweier Zeichenketten exakt übereinstimmen.
OFF Erlaubt unterschiedlich lange Zeichenketten. Zum Beispiel wird 'ABCD' = 'AB' „wahr"
(betrifft 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 <zeichenkette> 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. Die Verbindung muß vor (!) dem Öffnen der Datenbank geschehen
	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 PRIMÄRE Datenbank im Vordergrund (PRIMARY) und die SEKUNDÄRE Datenbank im Hintergrund (SECONDARY)) 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 PRIMÄR-Datei (PRIMARY) und wertet jeden Datensatz in der SEKUNDÄR-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 Hintergrund-Datei für den ersten Datensatz in der Vordergrund-Datei ausgewertet worden ist, schreitet das System zum zweiten Datensatz der Vordergrund-Datei und wertet erneut alle Datensätze der Hintergrund-Datei 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          (Primäre Datei)
. use namen               (Datei aufrufen)
. select secondary        (Sekundäre Datei)
. use ausgaben            (Datei aufrufen)
. select primary          (In Vordergrund schalten)
. join to kundausg for kundcode = 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“ (bzw. jeweils „.PRG“) 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                        (dh. 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 (Ja,Nein)?“ get mauswert
read                                        (*** J 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 an Stelle einer führenden Null.
*	schreibt einen Stern auf den Bildschirm an Stelle einer führenden Null.

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.

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/zeichenkette> [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 einer wörtlich angegebenen Zeichenkette 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/MM/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 @ ..SAY:

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 Unterketten-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 Programmes 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).

(Diese Seite wurde
absichtlich nicht bedruckt)

3. Kapitel: Alle dBASE II-Funktionen auf einen Blick

3. Kapitel: Alle d BASE II-Funktionen

Markt&Technik
Verlag Aktiengesellschaft
Hans-Pinsel-Straße 2
8013 Haar bei München

Abschnitt 1: Einführung in die wichtigsten Funktionen

(Diese Seite wurde
absichtlich nicht bedruckt)

1: Einführung in die wichtigsten Funktionen

1.0 Einführung in dBASE II

Zum Starten des Datenbanksystems dBASE II legen Sie eine Kopie der Originaldiskette in ein verfügbares Diskettenlaufwerk ein und machen dieses Laufwerk zum Arbeitslaufwerk Ihres Betriebssystems, indem Sie den Buchstaben, der das Laufwerk bezeichnet (z. B. „B"), gefolgt von einem Doppelpunkt (:) und der RETURN-Taste, eingeben. Ihr Betriebssystem meldet sich hierauf mit dem Buchstaben des ausgewählten Laufwerks. Als nächstes geben Sie nun ein:

```
DBASE
```

Hierdurch wird das Datenbanksystem in den Arbeitsspeicher geladen und gestartet. Auf dem Bildschirm erscheint dann:

```
*** dBASE II/86      Ver 2.41          19 JUNE 1984          - EUROPEAN
```

Es folgt ein kurzer Hinweis auf die Bestimmungen des Urheberrechts (Copyright) und anschließend meldet sich dBASE II mit dem Punkt-Prompt.

Hinweis: Wenn Sie eine kurze Zusammenfassung der wichtigsten dBASE II Befehle und weiterer Hilfestellungen sehen wollen, geben Sie HELP ein.

In einem normalen Arbeitsdurchgang mit dBASE II beginnen Befehle 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 darunter 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 Befehlstextes mit einem Fragezeichen.

(Diese Seite wurde
absichtlich nicht bedruckt)

Steuerzeichen

Im dBASE-Befehlsmodus sind folgende Tastenkombinationen mit Steuercodes belegt:

ctrl-P	Schaltet den Drucker ein bzw. aus (siehe SET PRINT-Befehl)
ctrl-U	Löscht die aktuelle Befehlszeile
ctrl-X	Löscht die aktuelle Befehlszeile

DELETE/RUBOUT	Löscht das zuletzt eingegebene Zeichen
ctrl-H/BACKSPACE	wie DELETE/RUBOUT

ESCAPE — Bricht Befehle mit möglicherweise langen Laufzeiten ab; im einzelnen: COUNT, DELETE, DISPLAY, HELP, 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.

Beispiele für Fehlerkorrekturen:

```
.DISPRAY MEMORY
*** UNBEKANNTER BEFEHL
DISPRAY MEMORY
KORRIGIEREN UND WIEDERHOLEN? (J/N)? J
ÄNDERN VON  :PR
ÄNDERN NACH :PL
DISPLAY MEMORY
WEITERE KORREKTUREN (J/N)? (RETURN)

. STORE (2+2 TO X
*** SYNTAXFEHLER ***
?                                   (Zeichenfolge (2+2 als
STORE (2+2) TO X                    fehlerhaft markiert)
KORRIGIEREN UND WIEDERHOLEN? (J/N)? J
ÄNDERN VON  :+2
ÄNDERN NACH :+2)
STORE (2+2) TO X
WEITERE KORREKTUREN(J/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 TO-Klausel im QUIT-Befehl.

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 Betriebssystem-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 Dateinamens erfordert, kann die Angabe des Typs unterbleiben, wenn dieser mit der Voreinstellung durch den Befehl übereinstimmt.

Datenbank-Dateien	- .DBF	
Speichervariablen-Dateien	- .MEM	
Befehls-Dateien	- .CMD	(.PRG bei 16-Bit-Version)
Format-Dateien	- .FRM	
Text-(ASCII)-Dateien	- .TXT	
Schlüssel-Dateien	- .NDX	
Masken-Dateien	- .FMT	

Für nähere Informationen über die Benennung der Dateien bitte die Benutzerhandbücher der jeweiligen Betriebssysteme verwenden.

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.

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- 8085- 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 2.2 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 (bei einigen 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 (ab Vers. 1.1) oder MSDOS bzw. PCDOS (ab Vers.2.0)

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 (bei einigen Befehlen erforderlich).

Beispiele für Datenfeld-Bezeichnungen:

A	
A123456789	
ABC:DEF	
A:B:C:D:E	
ABCD:	unzulässig, Doppelpunkt nicht eingebettet
ABC,DEF	unzulässig, Komma nicht erlaubt

Datentypen - dBASE II läßt drei verschiedene Datentypen für die Inhalte von Datenbankfeldern zu, und zwar: alphanumerische Zeichenketten (z. B. 'ABCD'), numerische Werte (z. B. 2 oder das Ergebnis eines Ausdruckes wie 5*18) und logische Werte (.T. True,.F. False).

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 Zeichenkette 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 der Felder für die logischen Werte 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 Speichervariablen speichern. Speichervariable 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)

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.

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 Zeichenketten), und der Ausdruck 1 > 2 liefert den Wahrheitswert ".F." für „falsch".

Folgende Komponenten können in dBASE-Ausdrücken enthalten sein:

* Datenbank-Variablen (Datenbank-Feldbezeichnungen)
* Speichervariablen
* Konstanten
* Funktionen
* Operationen

VARIABLEN - dBASE II unterscheidet zwei Arten von Variablen: Datenbank-Variablen und Speichervariablen. Datenbank-Variablen sind alle Felder des aktuellen Datensatzes, da diese unmittelbar geändert werden können. Ihr jeweils letzter Wert bleibt in der Datenbank gespeichert. Speichervariablen 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 (Zeichenketten-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 Zeichenkette bestehend aus den Buchstaben "a", "b" und "c" bzw. den logischen Wert „wahr" (.T.). Demgegenüber repräsentieren Datenbankvariablen immer nur ihren jeweils aktuellen Wert.

Alphanumerische Konstanten müssen stets in Hochkomma ('), Anführungszeichen (") oder ekkigen Klammern ([,]) eingeschlossen sein. Enthält eine Zeichenkette eines dieser Sonderzeichen, so muß sie durch ein Paar der anderen Sonderzeichen eingeschlossen werden. Beispielsweise stellen die Zeichenketten 'abc[def]ghi' und [abc'def'ghi] korrekte alphanumerische Konstanten dar, 'abc'def'ghi' hingegen nicht.

Logische Werte 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 (Zeichenketten) und logische Funktionen haben Wahrheitswerte .T. (wahr) oder .F. (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 „*** SYNTAXFEHLER ***“).

CHR-Funktion (Umwandlung eines durch eine Zahl definierten Zeichens in das entsprechende ASCII-Zeichen)

CHR (<numerischer Ausdruck>)

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.
```

Datei-Funktion (Prüfung, ob eine Datei vorhanden ist)

FILE (<Zeichenkette>)

Mit dieser logischen Funktion erfolgt die Prüfung, ob die in <Zeichenkette> 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 Zeichenkette, 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 Zeichenkette 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()
15/08/82

. STORE DATE() TO DATUM
15/08/82
```

Großbuchstaben-Funktion (Konvertierung von Klein- in Großbuchstaben)

!(<Zeichenkette>)

Die alphanumerische Funktion wird durch ein Ausrufezeichen aufgerufen. Sie wertet die <Zeichenkette> 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 Zeichenkette)

LEN (<Zeichenkette>)

Die numerische Funktion liefert als Zahl die Länge der <Zeichenkette>. Jedes Zeichen belegt dabei eine Stelle.

Beispiele:

```
. STORE 'abc'  TO WORT

. ? LEN(WORT)
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 Zeichenkette)

STR(<numerischer Ausdruck> , <Länge> , [<Dezimalstellen>])

STR ist eine alphanumerische Funktion und wertet zunächst den <numerischen Ausdruck> aus und formt dessen Zahlenwert dann in eine Zeichenkette aus Ziffern um. Das Ergebnis ist eine Zeichenkette 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 Zahlen vorgegeben werden.

Beispiele:

```
. STORE STR(2 * 1.1111, 7, 3) TO ALPHA
2.222

. DISPLAY MEMORY
ALPHA    (C)        2.222
** GESAMT **    01 VARIABLEN BENUTZT   00007 BYTES BELEGT
```

Teil"STRING"-Funktion (Manipulation von Teilen einer Zeichenkette (Unterketten)).

$(<alphanumerischer Ausdruck> , <Anfang> , <Länge>)

Mittels dieser alphanumerischen Funktion, die durch das Dollar-Zeichen aufgerufen wird, wird ein Teil einer Zeichenkette zu einer eigenen Zeichenkette. $ liefert als Ergebnis eine Zeichenkette, die nach Erarbeitung der Zeichenkette des <alphanumerischen Ausdrucks> dort am <Anfang> beginnt und dann die mit <Länge> angegebene Zahl von Stellen umfaßt. Aus einer Zeichenkette läßt sich damit ein beliebiger, zusammenhängender Teil als Unterkette herausschneiden. <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 Zahlen 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 Zeichenkette in numerischen Wert)

VAL(<Zeichenkette>)

Mit der numerischen Funktion VAL wird eine Zeichenkette in den ihr entsprechenden ganzzahligen Wert umgewandelt. Die <Zeichenkette> 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 Zeichenkette nur am Anfang Ziffern aufweist, ergibt sich die Zahl aus diesen führenden Ziffern.

Eine andere Möglichkeit, Zeichenketten 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 Zeichenkette 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(<Zeichenkette>)

Diese Funktion liefert in Form einer ganzen Zahl die Position, die dem ersten Zeichen der <Zeichenkette> im ASCII-Code zugeordnet ist. Sie entspricht damit der ASC-Funktion vieler BASIC-Dialekte.

Beispiel:

```
. ? RANK("A")
65
```

Zeichenketten-Suchfunktion (Prüfung, ob eine Zeichenkette in einer anderen vorkommt)

@ (<Zeichenkette 1> , <Zeichenkette 2>).

Diese numerische Funktion, aufgerufen mit dem Sonderzeichen „@", liefert eine ganze Zahl. Sie gibt an, an welcher Stelle der <Zeichenkette 2> eine Folge von Zeichen beginnt, die derjenigen von <Zeichenkette 1> entspricht. Dabei wird nur die erste vorgefundene Übereinstimmung angezeigt; die Prüfung in <Zeichenkette 2> beginnt am Anfang, also von links nach rechts. Falls die erste Zeichenkette in der zweiten nicht enthalten ist, lautet das Ergebnis „0". Bei Gleichheit der beiden Zeichenketten liefert die Funktion den Wert „1".
Diese Funktion ist der Funktion zur Manipulation von Teilen einer Zeichenkette (siehe Teil"STRING"-Funktion) ähnlich. Jedoch gibt @ an, an welcher Stelle in einer Zeichenkette eine vorgegebene Zeichenkette 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 FEHLER ***") abfangen.

Diese Funktion liefert einen von Null verschiedenen Wert, wenn der <Ausdruck> korrekt ist, andernfalls ist das Resultat von Test Null.

Beispiel:

```
INPUT TO 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 Ausdrucks oder einer Variablen 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
```

```
. STORE '3' TO A
3
 . STORE 3 TO B
3
. ? A + B
*** SYNTAXFEHLER ***
?
? A + B
 . ? VAL(A) + B
6
```

In diesem Beispiel tritt ein Fehler auf, weil die 3 zum einen ein numerischer Wert zum anderen eine Zeichenkette ist. Im Rechner werden sie unterschiedlich dargestellt und dBASE II kann nicht mehr erkennen, ob eine arithmetische Addition oder das Aneinanderfügen von Zeichenketten 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
```

Vergleichsoperationen

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 Zeichenkette in anderer enthalten" (A$B liefert .T., falls A mit B identisch oder A in B enthalten ist)

TRIM-Funktion (Abschneiden von Leerstellen am Ende von Zeichenketten)

TRIM(<Variable> bzw. <Datenbankname>)

TRIM entfernt in einer Variablen oder einem Datenfeld die dort am Ende vorgefundenen Leerstellen. Der alphanumerische Wert dieser Funktion ist dann die so gekürzte Zeichenkette. 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 Bearbeiten von Zeichenketten. 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 Zeichenketten aneinandergefügt werden, nicht hingegen ein numerischer Wert an eine Zeichenkette. Im letzten Fall ist durch eine enstprechende Funktion (hier STR) die Zahl in eine Zeichenkette umzuwandeln:

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 Zeichenketten

Sie liefern als Wert eine Zeichenkette und sind durch folgende Zeichen anzugeben:

- \+ für Aneinanderfügen von Zeichenketten
- \- für Aneinanderfügen von Zeichenketten, wobei Leerstellen am Ende zu verbindender Zeichenketten an das Ende der Ergebniszeichenkette 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 Zeichenkette unverändert übernomen wurden.

```
. ? A - B
ABCDEFGH
```
Hier wurden die Leerstellen am Ende von A entfernt und zum Ende der Ergebniszeichenkette 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:

	Reihenfolge bei		
Stufe	arithmetischen Operationen	Operationen auf Zeichenketten	logischen Operationen
1	Klammern, Funktionen	Klammern, Funktionen	.NOT.
2	Vorzeichen (+,-)	Vergleiche $ (Zeichenkette in Zeichenkette enthalten)	.AND.
3	*, / (Mulitipl., Division)	+, - (Aneinanderfügen von Zeichenketten	.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.

6.0 Schnittstellen zwischen dBASE II 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 Konventionen des jeweiligen Betriebssystemes ist jeder Satz in einer Datei mit bestimmten Zeichen 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: Zeichenketten werden durch spezielle Zeichen (meist " oder ') eingeschlossen und Eintragungen sind durch Kommata 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 Hochkommata oder Anführungszeichen zur Begrenzung von Text-Eintragungen vorliegen. COPY führt bei DELIMITED die Begrenzung mit Hochkommata 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 Zeichenketten nur Hochkommata 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 <Dateiname>.DBF
.COPY TO <Dateiname>.TXT DELIMITED WITH "

.USE <Dateiname>.DBF
.APPEND FROM <Dateiname>.DAT SDF
```


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 Zeichenkette als Wert zugewiesen ist, so ersetzt dBASE II an derselben Stelle den Namen der Variablen (einschließlich &) durch diese Zeichenkette.

Damit läßt sich z. B. die wiederholte Eingabe einer Zeichenreihe vermeiden. Sie wird einmal einer Variablen zugewiesen, die dann mit ihrem Namen stellvertretend (als „Makro") für diese Zeichenkette auftritt. Erst wenn die Zeichenkette benötigt wird, erfolgt die Ersetzung.

Auf diese Art lassen sich auch Parameterwerte zwischen Befehlsdateien austauschen, die sich untereinander aufrufen. Falls zu einer Zeichenkette, die über Ersetzung einer Variablen verwendet wird, noch Zeichenketten 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 Zeichenkette. 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)
```

7.0 Gruppierung der Befehle nach ihrer Wirkung

In einer Sitzung mit dBASE II, d. h. nach dem Start bis zur Eingabe des Befehls zur Beeendigung, 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	erzeugt eine neue Datenbank-Datei.
COPY	kopiert Daten aus einer Datei in eine andere Datei.
MODIFY	erzeugt und/oder bearbeitet eineBefehlsdatei oder ändert die Struktur in einer Datenbank-Datei.
REPORT	erzeugt einen Bericht
SAVE	speichert die Speichervariablen auf der Diskette.
INDEX	erzeugt eine Schlüssel-Datei.
REINDEX	aktualisiert die vorhandene Schlüssel-Datei.
JOIN	erzeugt gemeinsame Ausgabe aus zwei Datenbank-Dateien.
TOTAL	erzeugt zusammengefaßte Kopien einer Datenbank, bestehend aus Daten bestimmter Felder oder Datensätze.

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	erzeugt eine neue Datenbank-Datei.
INSERT	Einfügen neuer Datensätze.

Verändern von Daten

Vorhandene Daten bzw. Sätze lassen sich mit folgenden Befehlen ändern.

CHANGE	feldweise Bearbeitung von Datenbanken.
BROWSE	Bildschirmorientierte Bearbeitung der Datenbank.
DELETE	markiert einen Satz zum Löschen.
EDIT	ruft die Bearbeitung von Daten in einer Datenbank auf.
PACK	eliminiert Datensätze, die zum Löschen markiert sind.
RECALL	löscht die Markierungen zum Löschen von Datensätzen.
REPLACE	ändert Informationen in einem Datensatz oder in einer ganzen Datenbankdatei Feld für Feld.
READ	erlaubt Datei-Bearbeitung mit formatiertem Bildschirm, nimmt die Daten aus @ GET-Anweisungen entgegen.
UPDATE	aktualisiert eine Datenbank im Batchbetrieb.

Befehle zur Unterstützung des Anwenders

Mit den folgenden Befehlen kann sich der Anwender aktuelle Übersichten anzeigen lassen.

DISPLAY	zeigt Dateien, Datensätze oder Strukturen, Speichervariable oder den Status an.
HELP	gibt dem Benutzer Hilfestellung auf dem Bildschirm.

Befehle zur Anzeige von Daten

Zur Ausgabe von Daten auf Peripheriegeräten dienen diese Befehle.

@	gibt formatierte Daten auf Bildschirm oder Drucker aus.
BROWSE	Bildschirmorientierte Bearbeitung der Datenbank.
COUNT	zählt die Datensätze in einer Datei, die eine angegebene Bedingung erfüllen.
DISPLAY	zeigt Datensätze oder Strukturen, Speichervariable oder den Status an.
READ	erlaubt Datei-Bearbeitung mit formatiertem Bildschirm, nimmt die Daten aus @ GET-Anweisungen entgegen.
REPORT	erzeugt einen Bericht.
SUM	berechnet die Gesamtsummen der Felder in einer Datenbank.
?	zeigt einen Ausdruck, eine Variable oder ein Feld an.
TEXT	gibt Textblöcke von einer Befehlsdatei aus.

Befehle zur Positionierung in Datenbankdateien

Der interne Satzzeiger wird duch folgende Befehle verändert.

CONTINUE	setzt die Suchaktion nach einem LOCATE-Befehl fort.
FIND	sucht einen Datensatz in einer indizierten Datei.
GOTO/GO	geht zu einer bestimmten Position in einer Datei.
LOCATE	findet einen Datensatz, der eine Bedingung erfüllt.
SKIP	springt vorwärts und rückwärts in der Datenbank.

Befehle mit Auswirkungen auf eine ganze Datei

Folgende Befehle beziehen sich jeweils auf eine komplette Datei.

APPEND	fügt Informationen von einer anderen dBASE-Datenbank oder Datei an die benutzte Datei an.
COPY	kopiert Daten aus einer Datei in eine andere Datei.
DELETE	löscht eine Datei.
DO	führt Befehlsdateien aus.
RENAME	Umbenennen von Dateien.
SELECT	schaltet zwischen primärer und sekundärer Datenbank um.
SORT	erzeugt eine Datei, die nach einem Schlüsselfeld sortiert ist.
USE	eröffnet eine Datenbank für nachfolgende Bearbeitung, bis der nächste USE-Befehl erscheint.

Befehle zum Einrichten / Ändern von Speichervariablen

Folgende Befehle erzeugen bzw. verändern Speichervariable, hier kurz Variable genannt.

ACCEPT	Eingabe einer Zeichenkette in eine bestimmte Speichervariable.
COUNT	zählt die Anzahl der Datensätze in einer Datei, die eine angegebene Bedingung erfüllen und weist den Wert einer Speichervariablen zu.
DISPLAY	zeigt Speichervariable an.
INPUT	erlaubt Eingabe von Ausdrücken in Speichervariable.
RESTORE	reaktiviert Speichervaribale aus einer Datei, ggf. in Ergänzung zu bereits vorhandenen Variablen.
SAVE	speichert Speichervariablen auf Diskette ab.
STORE	weist den Wert eines Ausdrucks einer Speichervariablen zu.
SUM	berechnet die Gesamtsummen der Felder in einer Datenbank.
WAIT	unterbricht die Programmausführung, bis eine Eingabe vom Benutzer erfolgt ist.

Spezielle Befehle zur Anwendung und zum Aufbau von Befehls-Dateien

Diese Befehle dienen der Steuerung des Ablaufs bzw. dem Aufbau von Befehlsdateien.

ACCEPT	Eingabe einer Zeichenkette in eine bestimmte Speichervariable.
CANCEL	beendet die Ausführung einer Befehlsdatei.
DO	führt Befehlsdateien oder strukturierte Schleifen aus.
IF	erlaubt eine bedingte Befehlsausführung.
ELSE	alternativer Weg bei der Befehlsausführung innerhalb IF.
ENDDO	beendet einen DO WHILE-Befehl.
ENDIF	beendet einen IF-Befehl.
INPUT	erlaubt Eingabe von Ausdrücken in Speichervariable.
LOOP	springt zum Anfang eines DO WHILE-Befehls.
MODIFY	erzeugt und/oder bearbeitet eine Befehlsdatei.
RETURN	beendet den Lauf einer Befehlsdatei.
SET	setzt dBASE-Kontrollparameter.
WAIT	unterbricht die Programmausführung, bis eine Eingabe vom Benutzer erfolgt ist.

Befehle zur Steuerung von Geräten

Sie steuern die Ausgabe auf Peripheriegeräten (z. B. Drucker).

EJECT	erzeugt Seitenvorschub auf dem Drucker.
ERASE	löscht den Bildschirm.

8.0 Bildschirmorientierte Bearbeitung

Die bildschirmorientierte Bearbeitung, die bei den Befehlen 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 hier der Cursor frei positionierbar ist. Dies bedeutet, daß die einzelnen Datenfelder in beliebiger Reihenfolge bearbeitet werden können.

Bei der folgenden Beschreibung der Tastenkombinationen für Modus der bildschirmorientierten Bearbeitung steht "CTRL" 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 Niederdrükken der CONTROL-Taste ein geringfügiger Vorsprung einzuräumen ist, d. h. "CTRL-X" bedeutet:

1. CONTROL-Taste drücken und niederhalten
2. Taste X drücken, wobei die CONTROL-Taste gedrückt bleibt

8.1 Tastenkombinationen zur Cursor-Steuerung

ctrl-X	bewegt Cursor abwärts zum nächsten Feld
ctrl-E	bewegt Cursor aufwärts zum vorherigen Feld
ctrl-D	bewegt Cursor ein Zeichen nach rechts (vorwärts)
ctrl-S	bewegt Cursor ein Zeichen nach links (rückwärts)
ctrl-G	löscht das Zeichen unter dem Cursor
<RUB>/ <DEL>	löscht das Zeichen links vom Cursor
ctrl-Y	löscht das laufende Feld rechts vom Cursor
ctrl-V	schaltet um zwischen Überschreib- und EINFÜGEN-Modus
ctrl-W	speichert Änderungen und geht zurück zum "." -Prompt
ctl-Q	bricht die bildschirmorientierte Bearbeitung 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

ctrl-U	schaltet die Markierung DELETE des Satzes EIN bzw. AUS
ctrl-C	schreibt laufenden Satz auf Diskette, geht zum nächsten Satz
ctrl-R	schreibt laufenden Satz auf Diskette, geht zum vorherigen
ctrl-Q	ignoriert Änderungen im Satz und geht zurück zum "."-Prompt
ctrl-W	speichert alle Änderungen und geht zurück zum „."-Prompt

8.3 Zusätzliche Tastenkombinationen für den MODIFY-Befehl

ctrl-T	löscht die laufende Zeile, zieht alle unteren Zeilen hoch
ctrl-N	einfügen einer neuen Zeile an der Cursorposition
ctrl-C	schiebt Bildschirm eine halbe Seite abwärts
ctrl-R	schiebt Bildschirm eine halbe Seite aufwärts
ctrl-W	speichert alle Änderungen und geht zurück zum „."-Prompt
ctrl-Q	ignoriert alle Änderungen und geht zurück

8.4 Zusätzliche Tastenkombinationen für die Befehle APPEND, CREATE und INSERT

ctrl-C ctrl-R	schreibt laufenden Satz auf Diskette, geht zum nächsten Satz
<RETURN>	beendet APPEND, wenn der Cursor in der ersten Position des ersten Feldes ist
ctrl-U	schaltet die Markierung DELETE des Satzes EIN bzw. AUS

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
<zeichenkette>	eine beliebige Zeichenkette Unter einer Zeichenkette 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, Zeichenketten 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 ein Komma voneinander getrennt sind
<feld>	die Bezeichnung eines beliebigen Datenfeldes
<felderfolge>	eine Folge von Datenfeldbezeichnungen, die jeweils durch ein Komma voneinander getrennt sind
<dateiname>	ein beliebiger Dateiname, der den Konventionen für Dateinamen entspricht (vgl. Abschnitt 3.0)

8.5 Zusätzliche Tastenkombinationen für den BROWSE-Befehl

ctrl-U	schaltet die Markierung DELETE des Satzes EIN bzw. AUS
ctrl-W	speichert den Datensatz und geht zum „."-Promt (bei Superbrain ctrl-0)
ctrl-C	schreibt laufenden Satz auf Diskette, geht zum nächsten Satz
ctrl-R	schreibt laufenden Satz auf Diskette, geht zum vorherigen
ctrl-B	verschiebt das Bildschirm-Fenster ein Feld nach RECHTS
ctrl-Z	verschiebt das Bildschirm-Fenster ein Feld nach LINKS

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 Speichervariable Speichervariable 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 Speichervariablen.
<variablenfolge>	eine Folge von Speichervariablen, die jeweils durch ein 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.

Symbole	Bedeutung
<geltungsbereich>	der Bereich der Datenbank, auf den der Befehl angewendet werden soll
	Als <Geltungsbereich> sind folgende Angaben zulässig:
ALL	- 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 logischen Wert „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 bei anderen 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 "Schlüsselwörter" 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. Schlüsselwörter werden durch ihre ersten vier Buchstaben eindeutig identifiziert. Dies bedeutet, daß die Abkürzung von Schlüsselwörtern 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 Schlüsselwortes entsprechen.

5. Zur Eingabe von Befehlen, Schlüsselwörtern, Feldbezeichnungen, Namen von Speichervariablen 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-Schlüsselwörter sind keine reservierten Wörter in dem Sinne, daß sie bei anderweitiger Verwendung Fehlermeldungen hervorrufen würden. Doch sei darauf hingewiesen, daß manche Schlüsselwörter, 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-Schlüsselwörtern 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 Schlüsselwort ENDDO erscheint vor ENDIF und führt so zu unvorhersagbaren Effekten bei der Ausführung der Befehlsdatei, ohne daß dBASE II einen Fehler meldet.)

Abschnitt 2: dBASE-Befehle in alpabetischer Reihenfolge

10. 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 Zeichenketten-<Variable> übergeben, so wird diese im Kellerspeicher abgelegt und das Registerpaar HL enthält die Adresse des ersten Bytes. Das Maschinenprogramm darf die Länge der Zeichenkette 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.

Hinweis: Die PEEK()-Funktion ist im Bereich von 1000 HEX (4096 Dezimal) bis 7A00 HEX (31232 Dezimal) nicht durchführbar. Es wird zwar ein Wert angezeigt, wenn der Befehl in diesem Bereich benutzt wird, aber der Wert ist nicht korrekt.

Beispiele:

```
. STORE PEEK(128) TO X
      229

. STORE 41012 TO I
      41012

. ? PEEK (I), PEEK (I-10), PEEK (I+4)
      7    0    11
```

Abschnitt 2: dBASE II-Befehle in alphabetischer Reihenfolge

?

[<Felderfolge>]
?? [<Felderfolge>]

Dieser Befehl stellt eine Sonderform des DISPLAY-Befehls dar und ist gleichbedeutend mit DISPLAY OFF <Felderfolge>. Mit ihm läßt sich der Wert eines Feldes oder einer Felderfolge anzeigen. Die <Felderfolge> kann Speichervariablen, 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 <Felderfolge> 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.

(Diese Seite wurde
absichtlich nicht bedruckt)

Beispiele:

```
. USE MITGLIED

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: MITGLIED.DBF
ANZAHL DER SÄTZE: 00015
DATUM DER LETZTEN AKTUALISIERUNG: 17/10/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     NAME          C        020
002     MITGL:NR      N        005
003     EINTR:JAHR    C        004
** GESAMT **                  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 bzw. PRG auf Laufwerk B vorhanden, kann sie durch den Befehl „DBASE B:START" vom Betriebssystem aus 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

STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00011
DATUM DER LETZTEN AKTUALISIERUNG: 15/10/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     ARTIKEL       C        020
002     ARTIKEL:NR    N        005
003     EINZ:PREIS    N        007      002
004     BESTAND       N        005
005     DATUM         C        008
** GESAMT **                  00046


Gib heutiges Datum an: 26/10/82
```

@

@ <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 hängen 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 23 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 numerische Speichervariablen 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 23 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 bei einem nachfolgenden READ-Befehl 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 Datenbank- oder Speichervariablen, die jedoch bereits existieren muß. 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 eine logische Variable, 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 Speichervariablen 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 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX'
    @ ZEILE,48 SAY D USING 'XXXXXXXXX'
    @ ZEILE,64 SAY P1 USING 'XXXXXXXXXXXXXXXX'
    @ ZEILE,88 SAY U USING 'XXXXXXXXX'
    @ ZEILE,104 SAY P2 USING 'XXXXXXXXXXXXXXXX'
    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> für die Anweisung USING „XX...X" 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 des nächsten Zeichens (Buchstabe, Ziffer, Sonderzeichen)	erlaubt die Eingabe eines beliebigen Zeichens (Buchstabe, Ziffer, Sonderzeichen)
A	wie X	erlaubt die Eingabe eines Buchstaben oder Leerzeichens
$ oder *	bewirkt die Ausgabe einer Ziffer oder eines „$" bz. "*" 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 '(bbbbb)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 'XXXXXXXXXXXXXXXXXX'
    @ 6,0 SAY „Gib Straße und Hausnummer ein“ ;
            GET STRASSE PICTURE 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
    @ 7,0 SAY „Gib Postleitzahl ein“ ;
            GET PLZ PICTURE '9999'
    @ 7,5 SAY „Gib Ortsnamen ein“ ;
            GET ORT PICTURE 'XXXXXXXXXXXXXXXXXX'
    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 die bildschirmorientierte Bearbeitung.

* 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 die bildschirmorientierte Bearbeitung ein, so daß die durch GET angegebenen Variablen editiert werden können.

ACCEPT

ACCEPT ["<Zeichenkette>"] TO <Speichervariable>

Dieser Befehl ermöglicht die Eingabe alphanumerischer Zeichenketten in eine Speichervariable. Im Unterschied zu INPUT braucht der Benutzer bei ACCEPT seine Eingaben jedoch nicht in Anführungszeichen einzuschließen, da ACCEPT jede Eingabe als alphanumerische Zeichenkette auffaßt, während INPUT den Variablentyp anhand der Schreibweise der Eingabe festlegt.

Existiert die angegebene <Speichervariable> noch nicht, so wird sie angelegt und die Eingabe des Benutzers in ihr gespeichert. Enthält der ACCEPT-Befehl vor dem Steuerwort TO eine in Anführungszeichen, Hochkomma oder eckige Klammern eingeschlossene <Zeichenkette> (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 <Speichervariable> 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
** GESAMT **            02 VARIABLEN BENUTZT     00026 BYTES BELEGT

. ACCEPT TO ALLES
:Alles Mögliche

. DISP MEMO
NAME          (C)       Hans Schneider
NAME:WEIT     (C)       Müller, Paul
ALLES         (C)       Alles Mögliche
** GESAMT **            03 VARIABLEN BENUTZT     00040 BYTES BELEGT
```


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, gelesen und an die Datenbank angehängt. Sind im Befehl weder die Schlüsselwörter 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 das Dateiende der Quelldatei erreicht 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 Zeichenketten 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 im <Ausdruck> von Datenbankfeldern verwendete Bezeichnungen in der aktivierten Datenbank existieren 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 Feld für Feld aufgefordert, die Daten des neu aufgenommenen Satzes einzugeben, dabei werden die Feldbezeichnungen der Datenbankstruktur den Eingabefeldern vorangestellt. 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 tabellarischer Form umgehen und den Bedürfnissen des Nutzers anpassen.

Bei der bildschirmorientierten Bearbeitung 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 aufgefüllt.

Beispiele:

```
. USE MITGLIED

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: MITGLIED.DBF
ANZAHL DER SÄTZE: 00005
DATUM DER LETZTEN AKTUALISIERUNG: 06/10/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      NAME          C         020
002      MITGL:NR      N         005
003      EINTR:JAHR    C         004
** GESAMT **                     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

. APPEND
SATZNUMMER 00006

NAME       : Weber, Peter
MITGL:NR   : 513
EINTR:JAHR : 82

SATZNUMMER 00007

NAME       : Richter, M. „Hansi"
MITGL:NR   : 515
EINTR:JAHR: 82

SATZNUMMER 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 SÄTZE HINZUGEFÜGT
```

```
. 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    ERSETZUNG(EN)

. 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
STRUKTURDATEN FÜR DATEI: KATALOG.DBF
ANZAHL DER SÄTZE: 00009
DATUM DER LETZTEN AKTUALISIERUNG: 17/10/08
PRIMÄRE DATEI
FELD      NAME          TYP      LÄNGE      DEZ. ST.
001       ARTIKEL       C          020
002       ARTIKEL:NR    N          005
003       PREIS         N          010        002
004       BESTAND       N          005
** GESAMT **                       00041

. CREATE KATALOG2
SATZSTRUKTUR FOLGENDERMASSEN EINGEBEN:
FELD      NAME,TYP,LÄNGE,DEZIMALSTELLEN
001       ARTIKEL,C,25
002       ARTIKEL:NR,N,5
003       PREIS,N,10,2
004       DATUM,C,8
005       (RETURN)
DATEN JETZT EINGEBEN? N

. USE KATALOG2

. APPEND FROM KATALOG
00009 SÄTZE HINZUGEFÜGT

. 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 ERSETZUNG(EN)
```

```
. 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      65.55   01.06.82
00006   Leiter             103      88.00   01.06.82
00007   Eimer               33       6.39   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 Kommata getrennt und Text wird von Zahlen durch die Einschließung in Hochkommata (') unterschieden.)

```
. USE BESTELLG
. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: BESTELLG.DBF
ANZAHL DER SÄTZE: 00008
DATUM DER LETZTEN AKTUALISIERUNG: 17/05/82
PRIMÄRE DATEI
FELD    NAME            TYP     LÄNGE     DEZ. ST.
001     NAME            C         020
002     ARTIKEL:NR      C         005
003     ANZAHL          N         005
** GESAMT **                      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 SÄTZE HINZUGEFÜGT

. 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
STRUKTURDATEN FÜR DATEI: AUSZAHLG.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 17/06/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      SCHECK:NR     N        005
002      EMPF          C        020
003      BETRAG        N        010       002
004      ANGEWIESEN    L        001
005      ABGEBUCHT     L        001
** GESAMT **                    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
STRUKTURDATEN FÜR DATEI: ABBUCHG.DBF
ANZAHL DER SÄTZE: 00003
DATUM DER LETZTEN AKTUALISIERUNG: 10/06/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      SCHECK:NR     N        005
002      BETRAG        N        010       002
003      ABGEBUCHT     L        001
** GESAMT **                    0017
```



```
. LIST
00001     1          45.60 .T.
00002     3        1230.56 .T.
00003     4         120.23 .T.
.

. APPEND FROM AUSZAHLG FOR ABGEBUCHT
00004 SÄTZE HINZUGEFÜGT

. 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

*** SYNTAXFEHLER ***
                                ?
APPEND FROM AUSZAHLG FOR ANGEWIESEN
KORRIGIEREN UND WIEDERHOLEN (J/N)? N
```

Das letze APPEND führte zu einem Fehler, da das Feld ANGEWIESEN in der Datenbank ABBUCHT nicht vorhanden ist.

BROWSE

BROWSE [FIELDS <Felderfolge>]

Der BROWSE-Befehl ist einer der wirkungsvollsten Befehle zur Datenänderung. 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 die bildschirmorientierte Bearbeitung 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 die bildschirmorientierte Bearbeitung zusammengefaßt:

ctrl-E,A	setzt den Cursor auf das vorangegangene Datenfeld zurück.
ctrl-X,F	rückt den Cursor auf das nächste Datenfeld vor.
ctrl-S	setzt den Cursor um ein Zeichen nach links.
ctrl-D	setzt den Cursor um ein Zeichen nach rechts.
ctrl-G	löscht das Zeichen an der Cursorposition.
DEL/RUB	löscht das Zeichen links vom Cursor.
ctrl-Q	bricht die bildschirmorientierte Bearbeitung ab und leitet zum dBASE-Befehlsmodus über. Die durchgeführten Änderungen bleiben dabei unwirksam, d. h. sie werden nicht gespeichert.
ctrl-W	speichert den aktuellen Datensatz auf der Diskette und beendet die bildschirmorientierte Bearbeitung (bei Superbrain: ctrl-O).
ctrl-R	speichert den aktuellen Datensatz auf der Diskette und zeigt den vorangegangenen Datensatz an.
ctrl-C	speichert den aktuellen Datensatz auf der Diskette und zeigt den nächsten Datensatz an.
ctrl-Z	verschiebt das Fenster um ein Datenfeld nach links.
ctrl-B	verschiebt das Fenster um ein Datenfeld nach rechts.
ctrl-U	setzt bzw. löscht die Löschmarkierung „*" des aktuellen Datensatzes.

Beispiel:

```
. BROWSE FIELDS 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 ctrl-Y, RETURN eingibt. Die Ausführung des CHANGE-Befehls kann durch Drücken der ESCAPE-Taste abgebrochen werden.

Beispiele:

```
. USE BESTAND

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00006
DATUM DER LETZTEN AKTUALISIERUNG: 23/12/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     ARTIKEL       C        020
002     ARTIKEL:NR    N        005
003     EINZ:PREIS    N        007      002
004     BESTAND       N        005
005     DATUM         C        008
** GESAMT **                  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

    SATZ: 00006

    BESTAND: 10
    NACH: 12

    . CHANGE ALL FIELD DATUM

    SATZ: 00001

    DATUM: 12.07.82
    VERÄNDERN? (RETURN)

    SATZ: 00002

    DATUM: 03.09.82
    VERÄNDERN? 03
    NACH: 04

    SATZ: 00003

    DATUM: 04.09.82
    VERÄNDERN? (ESCAPE)

    . CHANGE ALL FIELD EINZ:PREIS FOR (EINZ:PREIS - INT(EINZ:PREIS)) > 0

    SATZ: 00002

    EINZ:PREIS: 80.60
    NACH: 81

    SATZ: 00003

    EINZ:PREIS: 70.50
    NACH: 70

    SATZ: 00004

    EINZ:PREIS: 350.90
    NACH: 350

    SATZ: 00005

    EINZ:PREIS: 260.90
    NACH: 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]

Dieser Befehl deaktiviert die aktivierten GET-Klauseln. 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 und Formatdateien geschlossen und deaktiviert, sowie alle Speichervariablen gelöscht werden. Außerdem stellt CLEAR als Arbeitsbereich PRIMARY ein (siehe SELECT-Befehl).

Dieser Befehl sollte zu Beginn eines Programmes verwendet werden, um dem Programm einen bekannten (definierten) Anfangszustand zu geben.

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, jedoch gibt es hierbei Einschränkungen. Weitere Erläuterungen zu diesem 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 Zieldatei genannt wird. Existiert die Zieldatei noch nicht, so wird sie angelegt.

Gibt der Benutzer keine FIELD-Klausel an, so werden alle Felder der aktivierten Datenbank in die Zieldatei übernommen. Ist jedoch eine FIELD-Klausel vorhanden, so enthält die Zieldatei 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 Zieldatei übernommen.

Enthält der COPY-Befehl das Steuerwort DELIMITED, werden in der Zieldatei alle Felder durch Kommata voneinander getrennt und die Inhalte der Felder vom Typ Zeichenkette in Hochkommata (') eingeschlossen. Sollen die Zeichenketten 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 (variable Satzlänge). Außerdem wird der Wert alphanumerischer Felder nicht in Sonderzeichen eingeschlossen.

Will man die so erzeugte Zieldatei später wieder mit APPEND einlesen, ist bei der Wahl der Sonderzeichen zu beachten, daß APPEND nur Hochkommata, Anführungszeichen und Kommata 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 Zieldatei 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 Zieldatei 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
STRUKTURDATEN FÜR DATEI: A:MITGLIED.DBF
ANZAHL DER SÄTZE: 00008
DATUM DER LEZTEN AKTUALISIERUNG: 17/12/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     NAME          C        020
002     MITGL:NR      N        005
003     EINTR:JAHR    C        004
** GESAMT **                  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 TEMP
00008 SÄTZE KOPIERT

. COPY TO KLEIN500 FOR MITGL:NR < 500
00005 SÄTZE KOPIERT
. 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 SÄTZE KOPIERT

. 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 SÄTZE KOPIERT

. 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 MITGL4
00004 SÄTZE KOPIERT

. USE MITGL4

. 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
STRUKTURDATEN FÜR DATEI: BESTELLG.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 29/10/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      NAME          C         020
002      ARTIKEL:NR    C         005
003      ANZAHL        N         005
** GESAMT **                     00031

. COPY STRUCTURE EXTENDED TO BESTSTRU
00003 SÄTZE KOPIERT

. USE BESTSTRU

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: BESTSTRU.DBF
ANZAHL DER SÄTZE: 00003
DATUM DER LETZTEN AKTUALISIERUNG: 29/10/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      FIELD:NAME    C         010
002      FIELD:TYP     C         001
003      FIELD:LEN     N         003
004      FIELD:DEC     N         003
** GESAMT **                     00018

. LIST
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
STRUKTURDATEN FÜR DATEI: BESTELLG.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 17/12/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      NAME          C         020
002      ARTIKEL:NR    C         005
003      ANZAHL        N         005
** GESAMT **                     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 SÄTZE KOPIERT
```

(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 Textverarbeitungsprogrammen.

```
. COPY TO HOCHKOMM DELIMITED
00013 SÄTZE KOPIERT
```

(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 SÄTZE KOPIERT
```

(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 <Speichervariable>]
[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 <Speichervariablen> 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
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00011
DATUM DER LETZTEN AKTUALISIERUNG: 10/15/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      ARTIKEL       C         020
002      ARTIKEL:NR    N         005
003      EINZ:PREIS    N         007       002
004      BESTAND       N         005
005      DATUM         C         008
** GESAMT **                     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
ANZAHL DER SÄTZE: = 00011

. GOTO TOP

. COUNT FOR ARTIKEL:NR > 12000
ANZAHL DER SÄTZE: = 00004

. GOTO TOP

. COUNT FOR EINZ:PREIS < 100 NEXT 6
ANZAHL DER SÄTZE: = 00004

. GOTO TOP

. COUNT NEXT 6 FOR EINZ:PREIS < 100
ANZAHL DER SÄTZE: = 00004

. COUNT TO XX FOR BESTAND < 30
ANZAHL DER SÄTZE: = 00003
. ? XX
        3

. USE ARTIKEL

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: ARTIKEL.DBF
ANZAHL DER SÄTZE: 00010
DATUM DER LETZTEN AKTUALISIERUNG: 10/12/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      TITEL         C         020
002      AUTOR         C         015
003      REGISTER      C         008
004      ERSCHIENEN    C         008
** GESAMT **                     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

. COUNT FOR 'Anton' $ AUTOR
ANZAHL DER SÄTZE: = 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 „BITTE DATEINAMEN EINGEBEN:" zu dessen Eingabe auf. Dabei gilt für den Dateinamen neben den bekannten Konventionen des jeweiligen Betriebssystemes die zusätzliche Einschränkung, daß Sonderzeichen nur dort zugelassen sind, wo sie eine besondere Bedeutung haben (z. B. der Doppelpunkt in „B:" zur Bezeichnung des Laufwerks B).

Ist unter dem angegebenen Namen bereits eine Datei vorhanden, fragt dBASE den Benutzer „BEREITS BESTEHENDE DATEI LÖSCHEN?", worauf dieser entsprechend seinem Wunsch mit Y(es) oder N(o) antworten muß.

Wird eine neue Datei angelegt bzw. die obige Frage mit Y beant wortet, so fordert dBASE den Benutzer zur Eingabe der Struktur der neuen Datenbank auf. Dabei erscheint folgende Meldung:

```
SATZSTRUKTUR FOLGENDERMASSEN EINGEBEN:
FELD            NAME,TYP,LÄNGE,DEZIMALSTELLEN
001
```

Für jedes Feld seiner neuen Datenbank muß der Benutzer nun dessen Namen und einige zusätzliche Informationen, die durch Kommata getrennt werden, über dessen Beschaffenheit angeben. Als Name eines Feldes ist eine maximal zehn Zeichen lange Zeichenkette 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 TYP angegeben werden muß; im einzelnen:

C - Zeichenketten (alphanumerische Felder)
N - numerische Werte (Zahlen)
L - logische Werte (einstellig mit F für „falsch"
und T für „wahr")

Unter der Rubrik "LÄNGE" 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 „LÄNGE"), und die Anzahl der Stellen, die hinter dem Dezimalpunkt erscheinen sollen (einzugeben unter der Rubrik „DEZIMALSTELLEN"). Logische Felder 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 "DATEN JETZT EINGEBEN?" mit N(o), so ist der CREATE-Befehl beendet. Bei der Antwort Y(es) oder J(a) 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
BITTE DATEINAMEN EINGEBEN: A:MITGLIED
SÄTZSTRUKTUR FOLGENDERMAßEN EINGEBEN:
FELD        NAME,TYP,LÄNGE,DEZIMALSTELLEN
001         NAME,C,20
002         MITGL:NR,N,5
003         EINTR:JAHR,C,4
004         (RETURN)
DATEN JETZT EINGEBEN? J
```



```
SATZNUMMER:  00001
NAME       : Müller, Hans
MITGL:NR   : 103
EINTR:JAHR : 76

SATZNUMMER:  00002
NAME       : Schneider, Manfred
MITGL:NR   : 204
EINTR:JAHR : 78

SATZNUMMER:  00003
NAME       : Wetzel, Irene
MITGL:NR   : 304
EINTR:JAHR : 79

SATZNUMMER:  00004
NAME       : Lehmann, Paul
MITGL:NR   : 403
EINTR:JAHR : 81

SATZNUMMER:  00005
NAME       : Kern, Christa
MITGL:NR   : 406
EINTR:JAHR :  81

SATZNUMMER:  00006
NAME       :  (RETURN)

. USE A:MITGLIED

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: A:MITGLIED.DBF
ANZAHL DER SÄTZE: 00005
DATUM DER LETZTEN AKTUALISIERUFNG: 10/12/82
PRIMÄRE DATEI
FELD        NAME          TYP      LÄNGE      DEZ. ST.
001         NAME          C          020
002         MITGL:NR      N          005
003         EINTR:JAHR    C          004
** GESAMT **                        00030
```

DELETE

DELETE [<Geltungsbereich>][FOR <Ausdruck>][WHILE <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 gekennzeichneter 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
STRUKTURDATEN FÜR DATEI: AUSZAHLG.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 17/12/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     SCHECK:NR     N      005
002     EMPF          C      020
003     BETRAG        N      010      002
004     ANGEWIESEN    L      001
005     ABGEBUCHT     L      001
** GESAMT **                 00038
```



```
. 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 STUCTURE
STRUKTURDATEN FÜR DATEI: BESTSTRU.DBF
ANZAHL DER SÄTZE: 00004
DATUM DER LETZTEN AKTUALISIERUNG: 11/01/83
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     FIELD:NAME    C      010
002     FIELD:TYP     C      001
003     FIELD:LEN     N      003
004     FIELD:DEC     N      003
** GESAMT **                 00018

. LIST
00001   NAME          C      200
00002   ARTIKEL:NR    C      50
00003   ANZAHL        N      50
00004   DATUM         C      60

. CREATE NEUBEST FROM BESTSTRU

. USE NEUBEST

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: NEUBEST.DBF
ANZAHL DER SÄTZE: 00000
DATUM DER LETZTEN AKTUALISIERUNG: 11/01/83
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     NAME          C      020
002     ARTIKEL:NR    C      005
003     ANZAHL        N      005
004     DATUM         C      006
** GESAMT **                 00037
```

```
. 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 LÖSCHUNG(EN)

. 5
. DELETE NEXT 3
00003 LÖSCHUNG(EN)

. DELETE ALL FOR .NOT. ANGEWIESEN
00002 LÖSCHUNG(EN)

. 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 REAKTIVIERUNG(EN)

. 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>] [<Felderfolge>]
[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 <Felderfolge> an, werden alle Feldinhalte der ausgewählten Sätze zur Anzeige gebracht, andernfalls nur die Werte der aufgeführten Felder. 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 und dessen Voreinstellung von <Geltungsbereich> ALL ist. 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 Speichervariablen und ihre derzeitigen Werte angezeigt.

Mit Form d lassen sich alle auf dem voreingestellten bzw. explizit angegebenen <Laufwerk> vorhandenen Dateien vom Typ .DBF 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 üblichen Konventionen verfahren werden kann.

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
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00006
DATUM DER LETZTEN AKTUALISIERUNG: 23/12/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     ARTIKEL       C        020
002     ARTIKEL:NR    N        005
003     EINZ:PREIS    N        007      002
004     BESTAND       N        005
005     DATUM         C        008
** GESAMT **                   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)   Waldschmitdt & Söhne
UNBEZAHLT     (N)   123456.77
DATEIENDE     (L)   .T.
** GESAMT **        03 VARIABLEN BENUTZT 00026 BYTES BELEGT
```

```
. DISPLAY FILES ON A: LIKE *.FRM

SOLLIST .FRM


. DISPLAY FILES
DATENBANK          # SÄTZE     LETZTE ÄNDERUNG
MITGLIED  DBF      00015        17/12/82
NEUMITGL  DBF      00007        17/12/08
KATALOG   DBF      00009        17/12/08
BESTELLG  DBF      00013        17/12/82
AUSZAHLG  DBF      00013        17/12/82
LAGER     DBF      00007        17/12/82
TEST      DBF
KEINE dBASE II DATENBANK
```

Die letzte o. a. aufgeführte .DBF-Datei ist keine Datenbank aus dBASE II

```
. DISPLAY STATUS

AKTIVE DATENBANK - B:BESTAND .DBF
PRIMÄRE DATEI

SCHLÜSSEL: SCHLÜSSELBEGRIFF:
B:BESTAND .NDX ARTIKEL

INAKTIVE DATENBANK - B:PREIS .DBF
SEKUNDÄRE DATEI
```



```
SYSTEM WARTET
TAGESDATUM            - 17/09/83
AKTUELLES LAUFWERK    - B:
ALTERNATE    - ON
ACCENT       - 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 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, d. h. eine Auswahl aus n (>2) Alternativen zu realisieren im Gegensatz zum IF- Befehl, der eine Auswahl aus 2 Alternativen darstellt. 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:

```
DO CASE                                  IF ARTIKEL = 'Äpfel'
   CASE ARTIKEL = 'Äpfel'                   (Befehle für Äpfel)
      (Befehle für Äpfel)                ELSE
   CASE ARTIKEL = 'Birnen'        =         IF ARTIKEL = 'Birnen'
      (Befehle für Birnen)                    (Befehle für Birnen)
   OTHERWISE                                ELSE
      (Befehle für sonstiges)                  (Befehle für sonstiges)
ENDCASE                                     ENDIF
                                         ENDIF
```

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.

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), falls nicht anders angegeben. 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 logischen Wert „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 - ENDIF), voll in die DO WHILE-Schleife eingebettet sein müssen, d. h. falls innerhalb eines DO WHILE - ENDDO Befehls ein IF-Befehl sich befindet, darf das ENDDO nicht vor dem ENDIF sein!

Beispiel:

```
DO WHILE .NOT. EOF
   DISPLAY NAME
   .
   .
   .
   SKIP
ENDDO
```

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 logischen Wert liefert, ist zugelassen.

```
DO CASE
   CASE VAR1 + VAR2 = 12
      (Befehle für numerischen Vergleich)
   CASE .NOT. A
      (Befehle für logische Werte)
   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 in dem Modus der bildschirmorientierten Bearbeitung arbeitet oder nicht (siehe auch Befehl SET SCREEN).

Bei der bildschirmorientierten Bearbeitung 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 „BITTE SATZNUMMER EINGEBEN:" 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 II nicht in dem Modus für bildschirmorientierte Bearbeitung, erscheint nach dem Aufruf des EDIT-Befehls die Nachricht "ZUORDNUNG:", 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 „VERÄNDERN?". 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 „ZUORDNUNG:" 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 „ZUORDNUNG:".

Ein Datenfeld kann komplett gelöscht werden, indem man auf die Frage "VERÄNDERN?" die Tastenfolge ctrl-Y, RETURN eingibt. Anschließend kann der gesamte Feldinhalt neu eingegeben werden. Durch die Tastenkombination ctrl-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
STRUKTURDATEN FÜR DATEI: KUNDEN.DBF
ANZAHL DER SÄTZE: 00004
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
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
** GESAMT **                     00145

. DISPLAY ALL NAME,KUNDEN:NR
00001 Meier & Co                 2001
00002 Anlagenbau Schmidt         1004
00003 Schneider GmbH             8003
00004 Mikro GmbH                   89

. EDIT

SATZ EINGEBEN #,FELD (# ODER NAME),NEUER WERT
ZUORDNUNG: 1,1

NAME: Meier & Co
VERÄNDERN? & Co
NACH     & Sohn

NAME: Meier & Sohn
VERÄNDERN? (RETURN)

ZUORDNUNG: 4,1,Softmikro GmbH

ZUORDNUNG: ,8

KUNDEN:NR: 89
NACH: 95

ZUORDNUNG: (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 Zeichenkette 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

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

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 <Zeichenkette> oder ' <Zeichenkette> ' oder "<Zeichenkette> "

Mit diesem Befehl läßt sich auf schnelle Weise in einer indizierten Datenbank der erste Satz mit dem Schlüssel <Zeichenkette> auffinden. Befindet sich die Datenbank auf einer Floppy-Disk, beträgt die typische Zeit für die Ausführung eines FIND-Befehls unabhängig von der vorhandenen Satzzahl der Datenbank 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 Zeichenkette, so genügt es, im FIND-Befehl als <Zeichenkette> 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 <Zeichenkette> 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 Schlüssel-Datei durchsucht wird.

Sind die Feldinhalte der <Zeichenkette> numerisch, ermittelt FIND den ersten Satz, dessen Schlüssel arithmetisch mit dem als <Zeichenkette> angegebenen Wert übereinstimmt.

HINWEIS: Man beachte, daß als Suchbegriffe sowohl für numerische als auch alphanumerische Schlüssel stets Werte anzugeben sind, die in Hochkommata (') 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 Hochkommata/Anführungszeichen eingeschlossenen <Zeichenkette> enthalten sein.

Soll der Inhalt einer Speichervariablen als Schlüsselwert verwendet werden, ist diese Variable in Form einer Makro-Ersetzung, d. h. ihr Name mit vorangestelltem „&", nach FIND anzugeben. Eine numerische Speichervariable 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 <Zeichenkette> übereinstimmt, erscheint auf dem Bildschirm die Nachricht "WERT NICHT GEFUNDEN". 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, d. h. Zeichen für Zeichen, 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
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00009
DATUM DER LETZTEN AKTUALISIERUNG: 10/10/82
PRIMÄRE DATEI
FELD      NAME          TYP      LÄNGE      DEZ. ST.
001       ARTIKEL       C          020
002       ARTIKEL:NR    N          005
003       EINZ:PREIS    N          007        002
004       BESTAND       N          005
005       DATUM         C          008
** GESAMT **                      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     1.00  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 LÖSCHUNG(EN)
. SET DELETE ON
. FIND R
```



```
. 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
STRUKTURDATEN FÜR DATEI8: ARTIKEL.DBF
ANZAHL DER SÄTZE: 00010
DATUM DER LETZTEN AKTUALISIERUNG: 10/10/82
PRIMÄRE DATEI
FELD    NAME        TYP    LÄNGE    DEZ. ST.
001     TITEL       C        020
002     AUTOR       C        015
003     REGISTER    C        008
004     ERSCHIENEN  C        008
** GESAMT **              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
```

GO oder GOTO

a. GOTO RECORD <n>
b. GOTO TOP
c. GOTO BOTTOM
d. <n>
e. GOTO <Speichervariable>

Mit diesem Befehl läßt sich der (dBASE-interne) Zeiger auf einen Datensatz neu positionieren.

In den Fällen a und d wird dieser Zeiger auf den Datensatz 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 des Schlüssels richtet.

Fall e kann dazu benutzt werden, um mit Hilfe einer Speichervariablen 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
```


Ein Schlüssel kann auch aus Feldern kombiniert werden, um eindeutige Ergebnisse bei FIND zu erhalten:

```
. INDEX ON AUTOR+REGISTER TO AUTREG
00010 SÄTZE INDIZIERT

. 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"
```

Hinweis: Der FIND Befehl führt bei zusammengesetzten Schlüsseln zum Erfolg, wenn man die Leerzeichen zwischen den Teilen des Schlüssels berücksichtigt.

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
```

HELP

HELP [<Befehlswort>]

Mit dem HELP-Befehl lassen sich allgemeine Informationen über dBASE und kurze Beschreibungen über die Syntax einzelner Befehle und ihrer 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
```



```
. GOTO RECORD 6

. 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
```

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 Feldinhalten läßt sich jedoch - bei entsprechender Formulierung des <Ausdrucks> - auch absteigend sortieren. <Ausdrucks> darf kein numerisches Feld sein. Numerische Felder müssen mittels STR () Funktion umgewandelt werden.

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.

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 werden können oder nicht.

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
```

Beispiele:

```
. USE BESTAND

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00009
DATUM DER LETZTEN AKTUALISIERUNG: 10/10/82
PRIMÄRE DATEI
FELD    NAME        TYP     LÄNGE     DEZ. ST.
001     ARTIKEL     C         020
002     ARTIKEL:NR  N         005
003     EINZ:PREIS  N         007       002
004     BESTAND     N         005
005     DATUM       C         008
** GESAMT **                 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 SÄTZE INDIZIERT

. 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
SATZ: 00003
```



```
. DISPLAY
00003   Reifen        987     70.00  210  28.08.82

. SKIP -1
SATZ: 00006

. DISPLAY
00006   Rahmen        230   3460.00   12  03.05.82

. USE BESTAND
```

Mit diesem Kommando zur Anmeldung der Datenbank werden keine Schlüsseldateien aktiviert. 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 ARTIKEL

. LIST
00004   Achse       70023     50.00   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
STRUKTURDATEN FÜR DATEI: ARTIKEL.DBF
ANZAHL DER SÄTZE: 000010
DATUM DER LETZTEN AKTUALISIERUNG: 10/10/82
PRIMÄRE DATEI
FELD    NAME        TYP    LÄNGE    DEZ. ST.
001     TITEL       C      020
002     AUTOR       C      015
003     REGISTER    C      008
004     ERSCHIENEN  C      008
** GESAMT **               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 SÄTZE INDIZIERT
. 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 ["<Zeichenkette>"] TO <Speichervariable>

Dieser Befehl findet in Befehlsdateien Verwendung und dient dazu, vom Benutzer eingegebene Daten in Speichervariablen zu speichern. Existierte die angegebene <Speichervariable> zum Zeitpunkt des INPUT-Befehls noch nicht, so wird sie erzeugt und der vom Benutzer eingegebene Ausdruck in ihr gespeichert. Enthält der INPUT-Befehl eine in Hochkommata oder Anführungszeichen eingeschlossene <Zeichenkette> (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 <Speichervariablen> bestimmt dBASE anhand der Eingabe. Numerische Werte werden an numerische Variable zugewiesen, in Sonderzeichen eingeschlossene alphanumerische Zeichenreihen an alphanumerische Variable. 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 Zeichenkette nicht in Hochkommata 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!!!
** GESAMT **        05 VARIABLEN BENUTZT        00050 BYTES BELEGT

. 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. Es kann nur ein Datensatz in die Datenbank mit diesem Befehl eingegeben werden.

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 II 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
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00009
DATUM DER LETZTEN AKTUALISIERUNG: 10/10/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     ARTIKEL       C        020
002     ARTIKEL:NR    N        005
003     EINZ:PREIS    N        007      002
004     BESTAND       N        005
005     DATUM         C        008
** GESAMT **                   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

SATZNUMMER 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
. GOTO RECORD 10

. INSERT BEFORE
SATZNUMMER 00010

ARTIKEL     : Generator
ARTIKEL:NR  : 10015
EINZ:PREIS  : 340.
BESTAND     : 25
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 Generator     10015   340.00  25 12.10.82
00011 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 ERSETZUNG(EN)
```



```
. 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 1.000.000 Sätze aufnehmen. Es sind aber nur 65535 Sätze in einer Datenbank zugelassen.

Beispiele:

```
. USE KATALOG

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: KATALOG.DBF
ANZAHL DER SÄTZE: 00009
DATUM DER LETZTEN AKTUALISIERUNG: 10/12/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     ARTIKEL       C        020
002     ARTIKEL:NR    N        005
003     PREIS         N        010      002
004     BESTAND       N        005
** GESAMT **                   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
STRUKTURDATEN FÜR DATEI: BESTELLG.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 10/12/82
SEKUNDÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     NAME          C        020
002     ARTIKEL:NR    N        005
003     ANZAHL        N        005
** GESAMT **                   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
```

LIST

a. LIST [<Geltungsbereich>] [<Felderfolge>]
   [FOR <Ausdruck> WHILE <Ausdruck>] [OFF]
   [FIELDS <Felderfolge>]
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.


```
. SELECT PRIMARY

JOIN TO AUFWERT FOR P.ARTIKEL:NR = S.ARTIKEL:NR FIELDS NAME,
S.ANZAHL, PREIS

. USE AUFWERT

. LIST

00001   Müller & Co             2      12.00
00002   Tischlerei Schmitz      10     12.00
0O003   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 Reinhardts      30     13.00
00013   Hansen KG               1      1260.80
```

```
. 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
SATZ: 00003

. CONTINUE
SATZ: 00005

. DISP ARTIKEL
00005 Schubkarre

. CONTINUE
SATZ: 00006

. CONTINUE
SATZ: 00009

. CONTINUE
DATEIEINDE ERREICHT
. GO TOP

. LOCATE FOR PREIS > 100 NEXT 5
SATZ: 00003

. CONTINUE
SATZ: 00005

. CONTINUE
ENDE DES SUCH ('LOCATE')-BEREICHS
```


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 „DATEIENDE ERREICHT" 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 „ENDE DES SUCH ('LOCATE')-BEREICHS", 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
STRUKTURDATEN FÜR DATEI: KATALOG.DBF
ANZAHL DER SÄTZE: 00009
DATUM DER LETZTEN AKTUALISIERUNG: 17/12/08
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     ARTIKEL       C        020
002     ARTIKEL:NR    N        005
003     PREIS         N        010      002
004     BESTAND       N        005
** GESAMT **                   00041
```

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 gewünschten Tastenkombinationen die 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 eingeschränkter bildschirmorientierter Bearbeitung 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.

LOOP

Dieser Befehl wird innerhalb einer DO WHILE-Schleife 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
ENDDO
```

Bei der Cursersteuerung des MODIFY-Befehls gilt in analoger Weise die Syntax der bildschirmorientierten Bearbeitung. Im einzelnen sind dies:

ctrl-T	löscht die laufende Zeile, zieht alle unteren Zeilen hoch
ctrl-N	Einfügen einer neuen Zeile an der Cursorposition
ctrl-C	schiebt Bildschirm eine halbe Seite abwärts
ctrl-Y	löscht das laufende Feld rechts vom Cursor
ctrl-V	schaltet um zwischen Überschreib- und INSERT-Modus
ctrl-W	speichert alle Änderungen und geht zurück zum „.“-Prompt
ctrl-Q	ignoriert alle Änderungen und geht zurück

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 (einschließlich 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 (ctrl-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.

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
```

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.

Da der PACK-Befehl - insbesondere bei großen Dateien - viel Zeit kostet, sollte man ihn nur so selten wie nötig verwenden. Eine Alternative, zur Löschung markierter Sätze von der weiteren Verarbeitung auszuschließen, stellt der Befehl SET DELETE ON dar (siehe SET-Befehle).

Beispiele:

```
. USE BESTAND

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      ARTIKEL       C         020
002      ARTIKEL:NR    N         005
003      EINZ:PREIS    N         007       002
004      BESTAND       N         005
005      DATUM         C         008
** GESAMT **                     00046
```


NOTE

a. NOTE <Zeichenkette>
b. * <Zeichenkette>

Dieser Befehl ermöglicht die Einbettung von Kommentaren in Befehlsdateien. Im Gegensatz zum REMARK-Befehl erscheint jedoch die angegebene <Zeichenkette> bei Ausführung des NOTE-Befehls nicht auf dem Ausgabemedium.

Beispiele:

```
NOTE - Letzte Änderung: 30.10.82

*****************************
*   Eingabeschleife     *
*****************************
```

QUIT

QUIT [TO <Folge von Programmdateienamen>]

Dieser Befehl schließt alle Datenbankdateien, Befehlsdateien und Ausgabedateien und gibt die Kontrolle an das Betriebssystem zurück. Auf dem Bildschirm erscheint *** ENDE DES dBASE II-Laufes ***. DENKEN SIE an Ihre Datensicherung.

QUIT TO ist implementiert auf den Betriebssystemen CP/M-80, CP/M-86, Version 1.1 und MS-DOS, Version 2.0. Die unter <Folge von Programmdateinamen> angegebenen Programme werden der Folge nach vom Betriebssystem ausgeführt. Um zu dBASE II zurückzukehren, muß dBASE II in der <Folge von Programmdateinamen> spezifiziert werden.

Unter MS(PC)-DOS gelangt man nach Beendigung der <Folge von Programmdateinamen> wieder zurück zur dBASE II- Hauptbefehlsebene. Wird aus einem Programm der QUIT TO- Befehl aufgerufen, so wird nach dessen Ausführung bei der nächsten Code-Zeile des Programms fortgefahren. Speichervariable und Parameter werden nicht zerstört, jedoch werden alle aktivierten Dateien und die dazugehörigen Schlüssel- und Formatdateien geschlossen. Wenn Sie 'dBASE' am Ende der <Folge von Programmdateinamen> angegeben haben, startet ein neuer dBASE II Lauf. Die COMMAND.COM-Datei Ihres DOS-Betriebssystems muß sich in jedem Fall auf demselben Laufwerk befinden, von dem auch dBASE II aufgerufen wird.

Die Anzahl der angegebenen Programmdateinamen ist nicht beschränkt, allerdings darf kein Element dieser Folge die Länge von 254 Zeichen überschreiten.

Beispiel (je nach Betriebssystem verschieden):

```
. QUIT TO 'DIR B:','PIP PRN:=ALTERNAT.TXT','DBASE VOREINST'

        oder aber

. QUIT TO 'DIR B:','COPY ALTERNAT.TXT PRN'
```



```
. 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 LÖSCHUNG(EN)

. 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' DURCHGEFÜHRT 00011 SÄTZE KOPIERT

. 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
```

READ

READ [NOUPDATE]

Dieser Befehl aktiviert den Modus für bildschirmorientierte Verarbeitung 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. Speichervariablen 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 kann anschließend eine neue Bildschirmmaske erstellt werden 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 Speichervariablen sein. Speichervariablen müssen bereits vor Ausführung des @-Befehls existieren. Notfalls füllt man sie zur Initialisierung mit der gewünschten Stellenzahl entsprechender Anzahl von Leerzeichen (z. B. STORE ' ' TO VAR).

Die für die Eingabe und Änderung von Daten in der bildschirmorientierten Bearbeitung benötigten Tastencodes sind im 3. Kapitel, Abschnitt 1, Ziffer 8 angegeben. Zur Aktivierung der bildschirmorientierten Bearbeitung muß zuvor der Befehl SET SCREEN ON erfolgen (dies entspricht der Voreinstellung, wenn der Benutzer bei der Installation von dBASE die für die bildschirmorientierte Bearbeitung 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 SAY '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 'XXXXXXXXXXXXXXXXXXXXXXXXXXXX'
   @ 7,0 SAY „Gib Betrag an" ;  GET BETRAG PICTURE '9999999.99'
   @ 8,0 SAY „Scheck bereits eingelöst" ;
         GET EINGELÖST
   READ
  ENDDO
ENDIF
```

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
STRUKTURDATEN FÜR DATEI: NEUMITGL.DBF
ANZAHL DER SÄTZE: 00007
DATUM DER LETZTEN AKTUALISIERUNG: 17/12/08
PRIMÄRE DATEI
FELD    NAME        TYP    LÄNGE    DEZ. ST.
001     NAME        C        020
002     MITGL:NR    N        005
** GESAMT **                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 LÖSCHUNG(EN)
```


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 logischer Wert) keine PICTURE-Klausel angegeben werden muß.

Für weitere Beispiele sei auf den @-Befehl verwiesen.

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

NEUINDEXIERUNG DER INDEX-DATEI - B:BESTNDX .NDX
00012 SÄTZE INDIZIERT
```



```
. 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 REAKTIVIERUNG(EN)

. RECALL ALL
00002 REAKTIVIERUNG(EN)

. 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 LÖSCHUNG(EN)

. RECALL ALL FOR MITGL:NR > 500
00002 REAKTIVIERUNG(EN)

. 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
```

REMARK

REMARK <Zeichenkette>

Dieser Befehl ermöglicht - im Gegensatz zum NOTE-Befehl - die Ausgabe einer Kommentar-<Zeichenkette> auf dem Ausgabemedium. In der Regel findet der REMARK-Befehl nur in Befehlsdateien Verwendung.

Beispiel:

```
. REMARK ****** REMARK TEST ******
****** REMARK TEST ******
```


RELEASE

RELEASE [<Speichervariablenfolge>][ALL]
[ALL LIKE <Maske>][ALL EXCEPT <Maske>]

Dieser Befehl deaktiviert die in der <speichervariablenfolge> angegebenen Speichervariablen und stellt ihren Speicherplatz dem System wieder zur Verfügung. Das Steuerwort ALL bewirkt die Löschung aller Speichervariablen.

Die <Maske> entspricht der <Maske> des DISPLAY-Befehls, d. h. "?" und „*" können entsprechend den Betriebssystem-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
** GESAMT **        09 VARIABLEN BENUTZT 00045 BYTES BELEGT

. 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
** GESAMT **        06 VARIABLEN BENUTZT 00027 BYTES BELEGT

. RELEASE ALL EXCEPT VAR*

. DISPLAY MEMORY
VARAB       (C)     abcd
VARABY      (N)     3001.22
VARXXXXX    (C)     XYZ
** GESAMT **        03 VARIABLEN BENUTZT 00016 BYTES BELEGT
```

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).

Die Option NOUPDATE bietet sich insbesondere an, wenn bekannt ist, daß durch den REPLACE- Befehl keine Schlüsselfelder verändert werden.

Hinweis: Bei Verwendung von PRIMARY- und SECONDARY-Datenbankdateien kann der REPLACE-Befehl nur auf der Datei ausgeführt werden, die mittels SELECT zuletzt angesprochen wurde.

RENAME

RENAME <alter 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
```

Beispiele:

```
. USE AUSZAHLG

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: AUSZAHLG.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 10/10/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      SCHECK:NR     N          005
002      EMPF          C          020
003      BETRAG        N          010     002
004      ANGEWIESEN    L          001
005      ABGEBUCHT     L          001
** GESAMT **                     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.

. REPLACE ABGEBUCHT WITH T FOR EMPF = 'selbst'
00004 ERSETZUNG(EN)
```



```
. 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 ERSETZUNG(EN)

. DISPLAY
00013    15 selbst             800.00    .T. .F.

. USE BESTAND

. COPY ALL TO B:PREISNEU FIELD ARTIKEL, EINZ:PREIS
00011 SÄTZE KOPIERT

. USE PREISNEU

. DISPLAY STRUCT
STRUKTURDATEN FÜR DATEI: PREISNEU.DBF
ANZAHL DER SÄTZE: 00011
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      ARTIKEL       C          020
002      EINZ:PREIS    N          007     002
** GESAMT **                     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 ERSETZUNG(EN)

. REPLACE ALL EINZ:PREIS WITH EINZ:PREIS * 1.1
00011 ERSETZUNG(EN)

. 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 ERSETZUNG(EN)

. 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 II 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, wie die in der mit dem USE-Befehl aktivierten Datenbank enthaltenen Daten aufgelistet werden sollen. Zugelassen sind darüberhinaus Spaltenüberschriften, Berechnung von Summen über numerische Felder sowie Ausdrücke, deren Wert sich aus Datenfeldern, Speichervariablen 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 Angabe hierzu erfolgt, ist der <Geltungsbereich> entsprechend der Voreinstellung mit ALL festgelegt.

Das mit REPORT erzeugte Schema wird in einer Datei des Typs .FRM abgelegt. Fehlt die Angabe FORM <Formatdatei> bzw. ist die angegebene Datei nicht vorhanden, so geht dBASE II davon aus, daß ein neuer Bericht definiert werden soll. Nachdem im Dialog das Schema festgelegt wurde, legt dBASE II 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 Berichtsseite 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

STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00011
DATUM DER LETZTEN AKTUALISIERUNG: 10/15/82
PRIMÄRE DATEI
FELD      NAME          TYP      LÄNGE      DEZ. ST.
001       ARTIKEL       C          020
002       ARTIKEL:NR    N          005
003       EINZ:PREIS    N          007        002
004       BESTAND       N          005
005       DATUM         C          008
** GESAMT **                      00046

. REPORT
BITTE DATEINAMEN FÜR BERICHT EINGEBEN: BESTBERI
BITTE ANGEBEN: M=LINKER RAND, L=ZEILEN/SEITE, W=ZEILENBREITE
M=5,L=72,W=80
MIT SEITENÜBERSCHRIFT? (J/N) J
BITTE SEITENÜBERSCHRIFT EINGEBEN: BESTANDS - BERICHT
LEERZEILE ZWISCHEN DEN SÄTZEN? (J/N) J
GESAMTSUMMEN ERFORDERLICH? (J/N) J
BERICHT MIT ZWISCHENSUMMEN? (J/N) N

SPALTE    BREITE,INHALT
001       23,ARTIKEL+'...
BITTE ÜBERSCHRIFT EINGEBEN: < Artikel

002       10,EINZ:PREIS
BITTE ÜBERSCHRIFT EINGEBEN: Einzel-;Preis
GESAMTSUMMEN ERFORDERLICH? (J/N) N

003       8,BESTAND
BITTE ÜBERSCHRIFT EINGEBEN: >Bestand;------
GESAMTSUMMEN ERFORDERLICH? (J/N) N

004       12,EINZ:PREIS * BESTAND
BITTE ÜBERSCHRIFT EINGEBEN: Preis * Bestand
GESAMTSUMMEN ERFORDERLICH? (J/N) J

005       10,DATUM
BITTE ÜBERSCHRIFT EINGEBEN: <Datum

006       (RETURN)
(Ausgabe des Berichts)
```

Außer Datenfeldern können in einer Spalte auch Speichervariablen 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 „J“ auf LEERZEILE ZWISCHEN DEN SÄTZEN? (J/N) wird ein zweizeiliger Abstand zwischen den Zeilen für die Ausgabe eingestellt.

Bei MIT SEITENÜBERSCHRIFT bzw. BITTE ÜBERSCHRIFT EINGEBEN: 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 II - 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 „J“ auf GESAMTSUMMEN ERFORDERLICH? (J/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 ** GESAMT ** am Ende des Berichts ausgewiesen.

Seitennr. 00001
28/10/82

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
** GESAMT **			126645.60	

Wie das Beispiel zeigt, erfragt REPORT nacheinander für die Spalten (SPALTE) des Berichts die Anzahl der auszugebenden Stellen (BREITE) und den Namen, des dafür vorgesehenen Datenbankfeldes (INHALT). Dabei ist die Spaltenbreite im Bericht unabhängig von der aktuellen Länge des Datenbankfeldes definierbar. Im Beispiel wurde die Spalte 1 (SPALTE 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 Zeichenkette "..." 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 II das Feld in mehrere Zeilen in dieser Spalte auf. Ein 80-stelliges alphanumerisches Feld würde also in einer 50-stelligen Spalte zwei Zeilen benötigen.

BERICHT MIT ZWISCHENSUMMEN? (J/N) mit „J" liefert Zwischensummen über das bei BITTE FELDNAMEN FÜR ZWISCHENSUMMEN EINGEBEN: 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 BITTE ÜBERSCHRIFT FÜR ZWISCHENSUMME EINGEBEN: versehen werden. dBASE II ergänzt diesen Text zur besonderen Kennzeichnung mit einem Stern (*). Nach jeder Zwischensumme (in der Ausgabe als ** ZWISCHENSUMME gekennzeichnet) kann ein Seitenumbruch mit "J" auf SEITENVORSCHUB NACH ZWISCHENSUMMEN? (J/N) verlangt werden.

Nach wie vor lassen sich neben den Zwischensummen auch die Gesamtsummen bilden (siehe GESAMTSUMMEN ERFORDERLICH? (J/N)). Zusätzlich sind Berichte definierbar, die nur die Ergebnisse der Summenberechnungen ausweisen, also auf die Auflistung der Werte der einzelnen Datensätze verzichten.
Hierzu ist "J" bei NUR ZUSAMMENFASSUNG DES BERICHTS? (J/N) einzugeben.
Mit RETURN auf die Ausgabe einer neuen Spaltennummer wird der Dialog zur Festlegung des Berichtsschemas beendet (BERICHT erwartet dann die Eingaben zur Spaltenbreite und zum Inhalt). Es beginnt die Erstellung des Berichtes durch dBASE II, der dann auf dem Bildschirm, und bei Verwendung von TO PRINT auch auf dem Drucker erscheint.

Die Gestaltung des mit BERICHT definierten Berichtes kann zusätzlich mit anderen dBASE II-Befehlen geändert werden (näheres siehe SET-Befehl):
- 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 Seitennummerierung,
- SET MARGIN TO <n> Festlegung des linken Randes.

Im Gegensatz zu den Angaben in der <Formatdatei> bleiben diese Angaben nur während der dBASE II-Sitzung erhalten, müssen also mit jedem Start von dBASE II 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 II erzeugte <Formatdatei> kann mit Texteditoren bearbeitet werden (in dBASE II 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:

```
SEITENNR. 00001                                           Lager 25
11/10/82

                        BESTANDS - BERICHT

Artikel          Einzel-      Bestand      Preis *      Datum
-------           Preis       -------      Bestand

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
** GESAMT **                                106301.50
```

Beispiel mit Datenbank BESTELLG für Zwischen- und Gesamtsummen

```
. USE BESTELLG INDEX BESTELLG

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: BESTELLG.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 10/15/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST.
001     NAME          C      020
002     ARTIKEL:NR    C      005
003     ANZAHL        N      005
** GESAMT **                  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

BITTE ANGEBEN: M=LINKER RAND, L=ZEILEN/SEITE, W=ZEILENBREITE M=5
MIT SEITENÜBERSCHRIFT? (J/N) J
BITTE SEITENÜBERSCHRIFT EINGEBEN:
Summierung aus Anzahl der Bestellungen je Artikel-Nr
LEERZEILE ZWISCHEN DEN SÄTZEN? (J/N) N
GESAMTSUMMEN ERFORDERLICH? (J/N) J
BERICHT MIT ZWISCHENSUMMEN? (J/N) J
BITTE FELDNAMEN FÜR ZWISCHENSUMME EINGEBEN: Artikel:Nr
NUR ZUSAMMENFASSUNG DES BERICHTS: N
SEITENVORSCHUB NACH ZWISCHENSUMMEN: N
BITTE ÜBERSCHRIFT FÜR ZWISCHENSUMME EINGEBEN:
Bestellungen für Artikel-Nummer:
```



```
SPALTE      BREITE,INHALT
001         23,NAME + '     :'
BITTE ÜBERSCHRIFT EINGEBEN: <Firma
002         7,ANZAHL
BITTE ÜBERSCHRIFT EINGEBEN: >Anzahl
GESAMTSUMMEN ERFORDERLICH? (J/N) J
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
** ZWISCHENSUMME **	4
* Bestellungen für Artikel-Nummer : 112	
Schneider GmbH :	2
Stahlbau :	3
** ZWISCHENSUMME **	5
* Bestellungen für Artikel-Nummer : 120	
Gartenbau Wild :	34
Sanitär Reinhardts :	30
** ZWISCHENSUMME **	64
* Bestellungen für Artikel-Nummer : 140	
Wilken und Sohn :	2
** ZWISCHENSUMME **	2
* Bestellungen für Artikel-Nummer : 331	
Fortschritt e.V. :	12
Karl Anders :	12
** ZWISCHENSUMME **	24

```
* Bestellungen für Artikel-Nummer  : 492
Hobby-Bau                         :                35
** ZWISCHENSUMME **
                                                   35

* Bestellungen für Artikel-Nummer  : 660
Hansen KG                         :                 1
** ZWISCHENSUMME **
                                                    1

* Bestellungen für Artikel-Nummer  : 680
Müller & Co                       :                 2
Tischlerei Schmitz                :                10
** ZWISCHENSUMME **
                                                   12

* Bestellungen für Artikel-Nummer  : 830
Heimbedarf Knobel                 :                 3
** ZWISCHENSUMME **
                                                    3
** GESAMT **
                                                  150
```

. REPORT FORM GESAMT

```
BITTE ANGEBEN: M=LINKER RAND, L=ZEILEN/SEITE, W=ZEILENBREITE M=20
MIT SEITENÜBERSCHRIFT? (J/N) J
BITTE SEITENÜBERSCHRIFT EINGEBEN:
            Zwischensumme je Artikel (ohne Besteller-Angaben)
LEERZEILE ZWISCHEN DEN SÄTZEN? (J/N) N
GESAMTSUMMEN ERFORDERLICH? (J/N) J
BERICHT MIT ZWISCHENSUMMEN? (J/N) J
BITTE FELDNAMEN FÜR ZWISCHENSUMME EINGEBEN: Artikel:Nr
NUR ZUSAMMENFASSUNG DES BERICHTS: J
SEITENVORSCHUB NACH ZWISCHENSUMMEN: N
BITTE ÜBERSCHRIFT FÜR ZWISCHENSUMME EINGEBEN: Artikel-Nummer:

SPALTE          BREITE,INHALT
001             20 NAME
BITTE ÜBERSCHRIFT EINGEBEN: (RETURN)
002             8, ANZAHL
BITTE ÜBERSCHRIFT EINGEBEN: (RETURN)
GESAMTSUMMEN ERFORDERLICH? (J/N) J
003 (RETURN)
```


(Nachfolgend Ausgabe des Berichts)

Zwischensumme je Artikel (ohne Besteller-Angaben)

```
* Artikel-Nummer:  103
** ZWISCHENSUMME **
                                    4

* Artikel-Nummer:  112
** ZWISCHENSUMME **
                                    5

* Artikel-Nummer:  120
** ZWISCHENSUMME **
                                   64

* Artikel-Nummer:  140
** ZWISCHENSUMME **
                                    2

* Artikel-Nummer:  331
** ZWISCHENSUMME **
                                   24

* Artikel-Nummer:  492
** ZWISCHENSUMME **
                                   35

* Artikel-Nummer:  660
** ZWISCHENSUMME **
                                    1

* Artikel-Nummer:  680
** ZWISCHENSUMME **
                                   12

* Artikel-Nummer:  830
** ZWISCHENSUMME **
                                    3

** GESAMT **                      150
```

RESTORE

RESTORE FROM <Dateiname> [ADDITIVE]

Dieser Befehl liest eine Datei, in die mit SAVE MEMORY TO <Dateiname> vorher Speichervariable abgespeichert worden ist. Alle Speichervariablen, 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 .MEM ergänzt.

Bei Verwendung der Option ADDITIVE bleiben die bereits vorhandenen Speichervariablen erhalten. Aus der angegebenen Datei werden zusätzlich so viele Variable wie möglich aufgenommen. Variable, 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 „AbCdEfGhIjKlMn...“ TO ALPHA
AbCdEfGhIjKlMn...

. 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)     AbCdEfGhIjKlMn...
ALT         (C)     alte Variablenbelegung
NEU         (C)     neue Variablenbelegung
** GESAMT **        04 VARIABLEN BENUTZT 00071 BYTES BELEGT
```


RESET

RESET [<Laufwerk>]

Die Anwendung des RESET-Befehles ist nur bei CP/M-80 und CP/M-86 notwendig, bei MS DOS, PC DOS und CCP/M-86 (ab. Vers. 3.1) kann er unterbleiben. 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 ctrl-C. RESET versucht, alle Dateien wieder zu aktivieren, die vor dem Diskettenwechsel in Benutzung waren.

Enthält der Befehl keine Laufwerksangabe, wird das gesamte System reinitialisiert. Wurde eine Diskette gewechselt, kann dies zu Ein-/Ausgabefehlern führen. Bei Angabe eines <Laufwerks> prüft dBASE II, 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 II-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 FROM VARALT.DAT
. DISPLAY MEMORY
EINS        (N)     1.11111
ALPHA       (C)     AbCdEfGhIjKlMn...
ALT         (C)     alte Variablenbelegung
** GESAMT **        03 VARIABLEN BENUTZT 00048 BYTES BELEGT

. RESTORE FROM VARNEU.DAT ADDITIVE
. DISPLAY MEMORY
EINS        (N)     2.22222
ALPHA       (C)     AbCdEfGhIjKlMn...
ALT         (C)     alte Variablenbelegung
NEU         (C)     neue Variablenbelegung
** GESAMT **        04 VARIABLEN BENUTZT 00071 BYTES BELEGT
```


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 II mit dem üblichen Zeichen zur Befehlseingabe, dem Punkt (Prompt).

Wird das Ende einer Befehlsdatei ohne RETURN erreicht, wirkt dies wie ein RETURN-Befehl.

Eine Befehlsdatei sollte immer ein RETURN-Befehl als letzten auszuführenden Befehl enthalten.

SAVE

SAVE TO <Dateiname> [ALL LIKE <Dateimaske>]

Die aktuell vorhandenen Speichervariablen werden mit Namen, Typ und Wert in der Datei <Dateiname> gespeichert und können später von dort wieder eingelesen werden (siehe RESTORE-Befehl). Von dBASE II 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 "*" kann entsprechend den üblichen Konventionen verwendet werden.

Beispiele:

```
. DISPLAY MEMORY
** GESAMT **                00 VARIABLEN BENUTZT 00000 BYTES BELEGT

. 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
** GESAMT **               03 VARIABLEN BENUTZT 00026 BYTES BELEGT
```



```
. SAVE TO ALLE.DAT

. SAVE TO ZWEI.DAT ALL LIKE EINS*

. RELEASE ALL

. DISPLAY MEMORY
** GESAMT **                00 VARIABLEN BENUTZT 00000 BYTES BELEGT

. RESTORE FROM ZWEI.DAT

. DISPLAY MEMORY
EINS          (N)         1.111
EINS:MAL:2    (N)         2.222
** GESAMT **               02 VARIABLEN BENUTZT 00014 BYTES BELEGT

. RESTORE FROM ALLE.DAT

. DISPLAY MEMORY
ALPHA         (C)         ABcdEFgh...
EINS          (N)         1.111
EINS:MAL:2    (N)         2.222
** GESAMT **               03 VARIABLEN BENUTZT 00026 BYTES BELEGT
```

SELECT

SELECT [PRIMARY]
[SECONDARY]

dBASE II gestattet, bis zu zwei Datenbanken 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 II 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 Datei im primären Bereich erscheint in den Meldungen als "Primäre Datei", die im sekundären Bereich entsprechend als "Sekundäre Datei" erscheint. 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 II.

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 positioniert werden. 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. Speichervariablen eingerichtet werden, wenn gleichzeitig mehr als zwei Datenbanken angesprochen werden sollen.

Beispiele:

```
. SELECT PRIMARY

. USE KATALOG

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: KATALOG.DBF
ANZAHL DER SÄTZE: 00009
DATUM DER LETZTEN AKTUALISIERUNG: 10/12/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      ARTIKEL       C         020
002      ARTIKEL:NR    N         005
003      PREIS         N         010       002
004      BESTAND       N         005
** GESAMT **                     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
STRUKTURDATEN FÜR DATEI: BESTELLG.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 10/12/82
SEKUNDÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST.
001      NAME          C         020
002      ARTIKEL:NR    N         005
003      ANZAHL        N         005
** GESAMT **                     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
```

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.ANZAHL * P.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 zusätzliche Angabe nicht notwendig.

SET

a. SET <Parameter 1> [ON]
   [OFF]
b. SET <Parameter 2> 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 <Parameter 1> 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 <Parameter 1> nur die dort aufgeführten Angaben zulässig sind. Die Voreinstellung 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.

<Parameter 1>	Einstellung	Wirkung
1.SET ALTERNATE	ON	speichert alle Bildschirmausgaben, (außer in der bildschirmorientierten Bearbeitung) in eine Datei. (Dem muß vorangehen: SET ALTERNATE TO <Datei>).
	OFF	unterbricht die Ausgabe in die Datei.
2.SET BELL	ON	Es ertönt ein akustisches Signal, wenn ein ungültiges Zeichen eingegeben wird oder die Feldgrenze überschritten wird.
	OFF	Schaltet das Signal aus.

3.SET CARRY	ON	In der bildschirmorientierten Bearbeitung mit APPEND werden Daten aus dem vorherigen Datensatz in den laufenden Satz geschrieben.
	OFF	Das Feld bleibt leer.
4.SET COLON	ON	Bei mit GET einzulesenden Datenfeldern werden die Feldgrenzen bei der Anzeige (@-Befehl) mit Doppelpunkten markiert.
	OFF	Keine Anzeige der Doppelpunkte zur Feldbegrenzung.
5.SET CONFIRM	ON	Es wird bei der bildschirmorientierten Bearbeitung automatisches Springen zum nächsten Feld verhindert, wenn das laufende Feld gefüllt ist.
	OFF	Das System wartet auf die Eingabe von <Return>, bevor der Cursor zum nächsten Feld springt.
6.SET CONSOLE	ON	Alle Eingaben erscheinen auf dem Bildschirm.
	OFF	schaltet die Bildschirmausgabe aus.
7.SET DEBUG	ON	Die mit ECHO und STEP erzeugte Ausgabe wird auf dem Drucker ausgegeben.
	OFF	Die Ausgabe erfolgt auf dem Bildschirm.
8.SET DELETED	ON	Die zum Löschen markierten Datensätze können mit keinem Kommando aufgerufen oder bearbeitet werden (z. B. <Bereich>, LIST, DISPLAY, COUNT).
	OFF	dBASE erkennt alle Datensätze. (bei COPY- und APPEND-Befehlen bleiben sie jedoch unberücksichtigt!).

9.SET ECHO	ON	Die Verfolgung einer Befehlsdatei durch Anzeige aller Kommandos auf dem Bildschirm wird ermöglicht.
	OFF	Keine Ausgabe der Kommandos.
10.SET EJECT	ON	Veranlaßt den REPORT-Befehl, einen Seitenvorschub auszuführen, bevor ein auszugebender Bericht zum Drucker geschickt wird.
	OFF	Verhindert den Seitenvorschub.
11.SET ESCAPE	ON	Der Benutzer kann die Ausführung einer Befehlsdatei abbrechen, indem er die ESCAPE-Taste drückt.
	OFF	Erlaubt keinen Abbruch mit der ESCAPE-Taste.
12.SET EXACT	ON	Erfordert genaue Eingaben der Zeichenketten in jeder Einzelheit (bei FOR<Ausdr>, FIND-Kommandos etc).
	OFF	Erlaubt die Eingabe voneinander abweichender Zeichenketten, z. B. unterschiedliche Länge: 'ABCDEF' = 'ABC'.
13.SET INTENSITY	ON	Ermöglicht eine inverse Darstellung oder eine andere Helligkeit in der Bildschirmanzeige (falls die Hardware dies zuläßt).
	OFF	Schaltet diese Einrichtung ab.
14.SET LINKAGE	ON	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).
	OFF	Führt zur voneinander unabhängigen Positionierung in der Primär- und Sekundärdatenbank.

15. SET PRINT	ON	Protokollierung von Anzeigen auf dem Drucker.
	OFF	Keine Protokollierung auf Drucker.
16. SET RAW	ON	Es werden bei DISPLAY- und LIST- Befehlen keine Leerstellen zwischen den Feldern eingefügt.
	OFF	Einfügen von Leerstellen zwischen den Feldern.
17. SET SCREEN	ON	Erlaubt bildschirmorientierte Bearbeitung mit APPEND, EDIT, INSERT, READ, und CREATE Kommandos.
	OFF	Schaltet bildschirmorientierte Bearbeitung aus.
18. SET STEP	ON	Unterstützt das Testen von Programmdateien, indem diese in Einzelschritten ausgeführt werden, d. h. nach jedem Befehl angehalten wird.
	OFF	Schaltet diese Testeinrichtung ab.
19. SET TALK	ON	Zeigt Ergebnisse der Kommandoausführung auf dem Bildschirm an.
	OFF	Unterdrückt die Anzeige.

Die Form b erlaubt folgende Einstellungen:

1. SET HEADING TO <Zeichenkette>

Hiermit wird intern in dBASE die <Zeichenkette> mit einer Länge von höchstens 60 Zeichen abgespeichert und bei mit REPORT veranlaßten Berichten als Teil der 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). Die angegebene Maskendatei wird ebenfalls bei den Befehlen APPEND und EDIT verwendet.

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 zunächst das Laufwerk mit den Systemdateien, das in der Regel der unter dem Betriebssystem gewählten Voreinstellung entspricht.
Die Angabe zu <Laufwerk> kann mit oder ohne den (sonst obligatorischen) Doppelpunkt erfolgen. Auch die Verwendung einer Speichervariablen mit vorangestelltem & (Macro) ist zugelassen. Die Voreinstellungen können unterschiedlich sein, d. h. der SET-Befehl ändert nicht die Voreinstellung des Bestriebssystems, 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. 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 (mit Ausnahme derjenigen Ausgaben die im Modus der bildschirmorientierten Bearbeitung ausgegeben werden, wie z. B. bei EDIT, APPEND, @SAY), die auch auf dem Bildschirm erscheinen. Mit dem Befehl wird die Datei zunächst nur eingerichtet und eröffnet. 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 Modus für bildschirmorientierte Bearbeitung erfolgen) erfolgen zusätzlich auch in der Datei und sind damit gespeichert. Die Protokollierung kann mit ON bzw. OFF ein - und ausgeschaltet werden, d. h. Protokolle lassen sich stückweise aufbauen. Ein nachfolgender Befehl SET ALTERNATE TO (auch ohne Dateiname) schließt die mit der letzten SET ALTERNATE TO <Dateiname> eröffnete Datei.

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)

. SET ALTERNATE ON
(Fortsetzung Protokollierung in B:PROTOK.TXT)

. SET ALTERNATE TO B:WEITER
(Protokollierung läuft weiter in B:WEITER.TXT)

5. SET DATE TO tt/mm/jj

Dieser Befehl erlaubt die Veränderung des beim Start von dBASE erfragten Datums. Anstelle von tt/mm/jj ist auch jede andere Reihenfolge von Tag, Monat und Jahr zulässig, da dBASE bei diesem Befehl keine Plausibilitätsprüfung des Datums durchführt. dBASE verwendet das Datum zur Protokollierung des jeweils letzten Änderungsstandes von Datenbankdateien und in von REPORT erzeugten Berichten.

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
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00011
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST
001     ARTIKEL       C      020
002     ARTIKEL:NR    N      005
003     EINZ:PREIS    N      007      002
004     BESTAND       N      005
005     DATUM         C      008
** GESAMT **                 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

. 5
```


6. SET INDEX TO [<Schlüsseldatei>], [<Schlüsseldatei>, ... , <Schlüsseldatei>]

Bis zu sieben <Schlüsseldateien> (bei mehreren Dateien jeweils durch Kommata 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. HINWEIS: 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 -2
SATZ: 00003

. DISPLAY
00003    Reifen          987        77.00    210    15.10.82

. SKIP 2 * 2
SATZ: 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
SATZ: 00010

. DISPLAY
00010    Generator      10015       340.00     25    12.10.82
. SKIP 2*2
SATZ: 00002

. DISPLAY
00002    Kolben         12221        90.00    233    15.10.82
. SKIP 3
SATZ: 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 einer internen Darstellung, d. h. deutsche Umlaute sind entsprechend in der Reihenfolge einsortiert. 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
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00011
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST
001     ARTIKEL       C        020
002     ARTIKEL:NR    N        005
003     EINZ:PREIS    N        007      002
004     BESTAND       N        005
005     DATUM         C        008
** GESAMT **                  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

. SORT ON ARTIKEL TO ARTSORT
SORT BEENDET

. 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 BEENDET
. 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
```

STORE

STORE <Ausdruck> TO <Speichervariable>

Der Wert von <Ausdruck> wird berechnet, wenn er sich nicht bereits direkt ergibt (z. B. Eingabe einer Zahl oder eines Textes) und in einer Speichervariablen mit Namen <Speichervariable> abgespeichert wird. 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 Speichervariable in Dateien abgespeichert werden und bleiben dann über den aktuellen Lauf von dBASE hinaus erhalten.
STORE dient nur der Einrichtung und Änderung von Speichervariablen. 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 + 'IjKlMn...' TO ALPHALANG
AbCdEfGh...IjKlMn...

. DISPLAY MEMORY
EINS        (N)  1.11111
ZWEI        (N)  2.22222
ALPHA       (C)  AbCdEfGh...
ALPHALANG   (C)  AbCdEfGh...IjKlMn...
** GESAMT **      04 VARIABLEN BENUTZT      00043 BYTES BELEGT
```



```
. USE BESTAND

. SORT ON ARTIKEL:NR TO ARTSORT DESCENDING
SORT BEENDET
. 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
```

SUM

SUM <Ausdruck>[<Ausdruck>][TO <Speichervariablenfolge>]
[<Geltungsbereich>][FOR <Ausdruck>]
[WHILE <Ausdruck>]

Mit SUM lassen sich numerische Datenfelder von Datensätzen bzw. damit gebildete <Ausdrükke> 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 Speichervariablen 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 Speichervariablen 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
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00011
DATUM DER LETZTEN AKTUALISIERUNG: 00/00/00
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST
001     ARTIKEL       C        020
002     ARTIKEL:NR    N        005
003     EINZ:PREIS    N        007      002
004     BESTAND       N        005
005     DATUM         C        008
** GESAMT **                   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

. SUM EINŻ: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
** GESAMT **         02 VARIABLES BENUTZT 00011 BYTES BELEGT

. 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 Zeilenfortsetzungssysmbol „;“ 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 Aus-
nahme ist das Zeilenfortsetzungssymbol „;“.
ENDTEXT
```


TOTAL

```
TOTAL ON <Schlüssel> TO <Datenbank> [ FIELD <Felderfolge>]
        [FOR <Ausdruck>] [ WHILE < 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.

Beispiel:

```
. USE SUMBEST

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: SUMBEST.DBF
ANZAHL DER SÄTZE: 00009
DATUM DER LETZTEN AKTUALISIERUNG: 15/10/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST
001      ARTIKEL:NR    C          005
002      ANZAHL        N          005
** GESAMT **                      00011
```

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.


```
. USE BESTELLG INDEX ARTIKEL

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: BESTELLG.DBF
ANZAHL DER SÄTZE: 00013
DATUM DER LETZTEN AKTUALISIERUNG: 15/10/82
PRIMÄRE DATEI
FELD    NAME          TYP    LÄNGE    DEZ. ST
001     NAME          C      020
002     ARTIKEL:NR    C      005
003     ANZAHL        N      005
** GESAMT **                  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 SÄTZE KOPIERT

. USE SUMBEST

. LIST
00001   103    4
00002   112    5    ( Hinweis: zwei Bestellungen werden hier addiert. )
00003   120   64    ( -" - )
00004   140    2
00005   331   24    ( -" - )
00006   492   35
00007   660    1
00008   680   12    ( -" - )
00009   830    3
```

Beispiel:

```
. USE PREISBST

. DISPLAY STRUCTURE
STRUKTURDATEN FÜR DATEI: PREISBST.DBF
ANZAHL DER SÄTZE: 00011
DATUM DER LETZTEN AKTUALISIERUNG: 28/10/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST
001      ARTIKEL       C       020
002      EINZ:PREIS    N       007       002
003      BESTAND       N       005
004      DATUM         C       008
** GESAMT **                   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
STRUKTURDATEN FÜR DATEI: BESTAND.DBF
ANZAHL DER SÄTZE: 00011
DATUM DER LETZTEN AKTUALISIERUNG: 28/10/82
PRIMÄRE DATEI
FELD     NAME          TYP     LÄNGE     DEZ. ST
001      ARTIKEL       C       020
002      ARTIKEL:NR    N       005
003      EINZ:PREIS    N       007       002
004      BESTAND       N       005
005      DATUM         C       008
** GESAMT **                   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
```

Von dBASE wird, wenn nichts anderes angegeben wird, bei der Schlüsseldatei der Dateityp .NDX angenommen, und bei der Datenbank der Dateityp .DBF. Im dritten Beispiel wird demnach die Datenbank MITGLIED.DBF mit der Schlüsseldatei MITGLIED.NDX eröffnet.

WAIT

WAIT [TO <Speichervariable>]

Dieser Befehl veranlaßt dBASE, so lange in einen Wartezustand überzugehen, bis an der Tastatur ein Tastendruck erfolgt. Auf dem Bildschirm erscheint dabei die Nachricht SYSTEM WARTET. Enthält der Befehl eine TO-Klausel, so wird das durch den Tastendruck erzeugte Zeichen in der angegebenen <Speichervariablen> gespeichert. Existiert die <Speichervariable> nicht, so wird sie bei diesem Befehl erzeugt.

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 < Speichervariablen > ein Leerzeichen abgelegt.

Beispiel:

```
. RELEASE ALL

. WAIT TO WEITER
SYSTEM WARTET 1

. DISP MEMO
WEITER    (C)     1
** TOTAL **       01 VARIABLEN BENUTZT 00001 BYTES BELEGT
```

Anhang

(Diese Seite wurde absichtlich nicht bedruckt)

Anhang

Anhang A: Programm-Beispiel „Buecher"

Markt&Technik
Verlag Aktiengesellschaft
Hans-Pinsel-Straße 2
8013 Haar bei München

Erläuterungen zum Beispiel „Buecher"

Unter dem Sammelnamen „Buecher" finden Sie auf der beim Kauf von dBASE II erhaltenen Diskette eine Sammlung von Befehlsdateien, mit der Sie am Beispiel eines Bibliothekskataloges ausprobieren können, wie einfach Ihnen dBASE II die Pflege von Datenbanken macht, wenn Sie erst einmal mit seinen Möglichkeiten vertraut sind.

Mit einem fiktiven Anfangsbestand von einigen Büchern, deren Kenndaten (wie Titel, Autor, etc. ...) in der Datenbank 'buecher.dbf' für Sie vorbereitet wurden, können Sie mit Ihrem Mikrocomputer einige der üblichen Vorgänge bei der Führung einer Biliotheks-Kartei nachvollziehen, wie Bücher hinzufügen, Eintragungen ändern, oder den Bestand nach bestimmten Kriterien auflisten.

Das Beispiel soll natürlich keine komplette Lösung für einen großen Bücherbestand ersetzen, aber es wird Ihnen deutlich machen, wie sinnvoll sich dBASE II in allen Bereichen anwenden läßt, in denen Datenmengen verschiedenster Größe im ständigen systematischen Zugriff gehalten und verändert werden müssen.

Der Aufruf des Beispiels

Vom dBASE II-Promptpunkt aus wird das Beispiel „Buecher" mit dem Kommando

```
DO BUECHER
```

aufgerufen.

Sie können das Beispiel aber auch direkt vom Prompt-Zeichen Ihres Betriebssystems starten, dazu tippen Sie dann nur

```
BUECHER
```

(Die Beispielsdateien enthalten nämlich eine kleine Zusatzdatei namens "buecher.bat", durch die dBASE II bei dieser Art des Beispielaufrufs automatisch gestartet wird.)

Bei beiden Methoden meldet sich nach kurzer Zeit das Hauptmenü "BIBLIOTHEKSKATALOG".

Zum Umgang mit dem Beispiel

Anhand seiner Menüsteuerung erklärt es sich weitgehend selbst.
D. h., Sie werden von Anfang an mit einer Liste von Wahlmöglichkeiten durch das gesamte Programm geführt.

(Diese Seite wurde
absichtlich nicht bedruckt)

Falls Daten in Felder (erkenntlich an der Begrenzung durch Doppelpunkte, bzw. gegebenenfalls an der Hervorhebung durch eine helle Eingabemaske) einzugeben sind, können zur Positionierung des Cursors oder zur Editierung folgende Tastenkombinationen verwendet werden (Die Angabe "CTRL-X" bedeutet gleichzeitiges Drücken der Control-Taste und des Zeichens 'X'):

CTRL-E : Cursor nach oben,
CTRL-X : Cursor nach unten,
CTRL-S : Cursor nach links,
CTRL-D : Cursor nach rechts,
CTRL-G : Löschung des Zeichens unter dem Cursor,
CTRL-V : INSERT ein- und ausschalten.

Anmerkungen zum Aufbau des dBASE II-Beispiels

Sie können anhand der im Handbuch nachfolgend ausgedruckten Listings sämtlicher im Beispiel verwendeter Befehlsdateien eine erste Vorstellung davon gewinnen, auf welche Weise Ihnen dBASE II durch die Möglichkeit, Ihre Anwendungen selbst zu programmieren, das umständliche Eintippen von oft wiederholten Vorgängen bei der Pflege Ihrer Datenbanken abnimmt.

Mit solchen Befehlsdateien definieren Sie schließlich selbst ideal angepaßte Befehle unter dBASE II und finden damit vielfältige Erweiterungsmöglichkeiten für den ohnehin schon leistungsfähigen Befehlswortschatz von dBASE II. Befehlsdateien setzen sich aus dBASE II-Befehlen (ggf. auch neuen, mittels anderer Befehlsdateien definierten) zusammen und gestatten beispielsweise die Programmierung eines beliebigen Eingabe-Dialogs mit dem Benutzer.

Die Anwenderlösungen wurden, der klar strukturierten dBASE II-Programmiersprache entsprechend, modular aufgebaut. Sie bestehen also aus Befehlsdateien, die sich über mehrere Stufen hinweg aufrufen. Ihr Weg ist dabei übersichtlich nachzuvollziehen. Dies erleichtert spätere Änderungen und Erweiterungen erheblich und erlaubt, einzelne Bausteine und ihre Funktion separat auszutesten. Es wird empfohlen, sich diesem Konstruktionsprinzip anzuschließen.

Struktur der Datenbank „buecher.dbf"

Nach Eingabe der beiden folgenden Befehlszeilen zeigt Ihnen dBASE II die Struktur der Datenbank:

```
.USE BUECHER
.DISPLAY STRUCTURE

STRUKTURDATEN FÜR DATEI: BUECHER.DBF
ANZAHL DER SÄTZE: 00025
DATUM DER LETZTEN AKTUALISIERUNG: 08/08/84
PRIMÄRE DATEI
FELD    NAME       TYP    LÄNGE    DEZ.ST
001     TITEL      C        080
002     AUTOR      C        030
003     GEBIET     C        020
004     RAUM       C        015
005     REGAL      C        003
** GESAMT **                00149
```

Programm-Listings für das Beispiel BIBLIOTHEKSKATALOG

Auf den folgenden Seiten finden Sie die Progammdateien (.PRG) für das Beispiel BIBLIOTHEKSKATALOG. Die verwendete Datenbank heißt also BUECHER.DBF. Wenn Sie mit den auf der Diskette gespeicherten Dateien ausprobieren wollen, wie dBASE II funktioniert, benötigen Sie noch die ebenfalls dort vorhandenen Formatdateien (.FMT) und Indexdateien (.NDX). Die Listings für die Programm-Dateien sind hintereinander aufgeführt und bearbeiten die folgenden Prozeduren:

```
Hauptprogramm Buecher.prg
Unterprogramm: Buecher1.prg   (Buch hinzufügen)
Unterprogramm: Buecher2.prg   (Eintrag ändern)
Unterprogramm: Buecher3.prg   (Eintrag löschen)
Unterprogramm: Buecher4.prg   (Buchtitel suchen)
Unterprogramm: Buecher5.prg   (Bücher nach Autoren anzeigen)
Unterprogramm: Buecher6.prg   (Bücher nach Sachgebieten anzeigen)
Unterprogramm: Buecher7.prg   (Bücher nach Standort anzeigen)
Unterprogramm: Buecher8.prg   (Verzeichnis aller Bücher ausdrucken)
```

```
* ═══════════ BUECHER.PRG ═══════════
*
* HAUPTMENUE                    public from ASHTON-TATE (1984)
*
* ═══════════════════════════════════
*
set talk off
set intensity on
set bell off
do while T
        erase
        store" "to option
        @  1,23 say „BIBLIOTHEKSKATALOG“
        @  5,20 say “1. Buch hinzufügen“
        @  6,20 say “2. Eintrag ändern“
        @  7,20 say “3. Eintrag löschen“
        @  9,20 say “4. Buchtitel suchen“
        @ 10,20 say “5. Bücher nach Autoren anzeigen“
        @ 11,20 say “6. Bücher nach Sachgebieten anzeigen“
        @ 12,20 say “7. Bücher nach Standort anzeigen“
        @ 13,20 say “8. Verzeichnis aller Bücher ausdrucken“
        @ 15,20 say “X. Ende“
        @ 17,20 say “Geben Sie bitte Ihre Wahl ein“ get option picture „!“
        read
        @ 23,0 say „Warten Sie bitte einen Augenblick . . . . .“
        do case
                case option = „X“
                        erase
                        quit
                case option = „1“
                        do buecher1
                case option = „2“
                        do buecher2
                case option = „3“
                        do buecher3
                case option = „4“
                        do buecher4
                case option = „5“
                        do buecher5
                case option = „6“
                        do buecher6
                case option = „7“
                        do buecher7
                case option = „8“
                        do buecher8
        endcase
enddo
```



```
* ═══════════ BUECHER1.PRG ═══════════
*
* CASE OPTION 1 : BUCH HINZUFÜGEN (MIT INDIZIEREN)
*
* ═══════════════════════════════════
*
use buecher index btitel
store 0 to hinzu
do while T
        store str(1,81) to freilassen
        store $(freilassen,1,40) to titelx1
        store titelx1 to titelx2
        store $(freilassen,1,30) to autorx
        store $(freilassen,1,15) to raumx
        store $(freilassen,1,3) to regalx
        store $(freilassen,1,20) to gebietx
        erase
        @  1,30 say „BUCH HINZUFÜGEN“
        @  5,0 say „Titel “ get titelx1
        @  6,0 say „  “ get titelx2
        @  7,0 say „Autor “ get autorx
        @  8,0 say „Sachgebiet “ get gebietx
        @  9,0 say „Raum “ get raumx
        @ 10,0 say „Regal “ get regalx
        text
```

```
*═══════════════════════BUECHER2.PRG═══════════════════════
*
*    CASE OPTION 2 : EINTRAG ÄNDERN
*
*═══════════════════════════════════════════════════════
*
use buecher index btitel
erase
do while T
        erase
        @  1,30 say „EINTRAG ÄNDERN“
        store str(1,81) to freilassen
        store $(freilassen,1,40) to titelx1
        @  5,0 say „Titel “ get titelx1;
           picture „!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!“
        @  9,0 say „(Versuchen Sie es mit einem Buchstaben,“
        @ 10,1 say „wenn Ihnen der Buchtitel nicht einfällt)“
        @ 12,0 say „Drücken Sie <RETURN>, um aufzuhören“
        read
        if titelx1 = ” ”
        erase
                @ 2,0 say „Warten Sie bitte einen Augenblick . . . .“
                return
        else
                store trim(titelx1) to titelx
                find &titelx
                if #=0
                        @ 13,0 say titelx
                        @ 14,0 say „Nicht gefunden“
                        @ 17,0 say „Drücken Sie irgendeine Taste, um weiterzumachen“
                        set cons off
                        wait
                        set cons on
                else
                        store „M“ to mehr
                        do while mehr = „M“
                                store $(titel,1,40) to titelx1
                                store $(titel,41,40) to titelx2
                                store autor to autorx
                                store raum to raumx
                                store regal to regalx
                                store gebiet to gebietx
                                erase
```



```
********************************* EINGABE-VERFAHREN *********************************
*                                                                                   *
*                          Drücken Sie bitte CTRL und gleichzeitig:                 *
*                                                                                   *
*    Zur Bewegung des Cursors              *    Zum                                 *
*                                          *                                        *
*    E  Cursor nach oben                   *    G  Löschen                          *
*    X  Cursor nach unten                  *    V  Insert Einschalten               *
*    S  Cursor nach links                  *    V  Insert Ausschalten               *
*    D  Cursor nach rechts                 *    C  Rückkehren ins Hauptmenü         *
*                                          *                                        *
*************************************************************************************
        endtext
        read
        if titelx1 = ” ”
        return
        else
        @ 23,0 say „Warten Sie bitte einen Augenblick . . . . .“
                append blank
                store titelx1+titelx2 to titelx
                replace titel with titelx, autor with autorx, gebiet with gebietx;
                raum with raumx, regal with regalx
                store hinzu +1 to hinzu
                if hinzu> 20
                        ? „Datenbank überfüllt.“
                        use buecher index btitel
                        store 0 to hinzu
                endif
        endif
enddo
```

```
        @  1,30 say „EINTRAG ÄNDERN“
        @  5,0 say „Titel          “+titelx1
        @  6,0 say „               “+titelx2
        @  7,0 say „Autor          “+autorx
        @  8,0 say „Sachgebiet     “+gebietx
        @  9,0 say „Raum           “+raumx
        @ 10,0 say „Regal          “+regalx
        @ 20,0 say „Drücken Sie C wenn Sie Obenstehendes
          verändern wollen“
        @ 21,0 say „ oder + um das nächste Buch anzusehen“
        @ 22,0 say „ oder - um ein Buch zurückzugehen“
        @ 23,0 say „ oder jede andere Taste, um ein anderes Buch
          anzusehen“
        store ” ” to naechstes
        set cons off
        wait to naechstes
        set cons on
        if naechstes = „+“ .or. naechstes = „-“
                if naechstes = „+“
                        skip
                else
                        skip -1
                endif
        else
                store „N“ to mehr
        endif
enddo
if naechstes = „C“ .or. naechstes = „c“
        erase
        @  1,30 say „EINTRAG ÄNDERN“
        @  5,0 say „Titel          “ get titelx1
        @  6,0 say „               “ get titelx2
        @  7,0 say „Autor          “ get autorx
        @  8,0 say „Sachgebiet     “ get gebietx
        @  9,0 say „Raum           “ get raumx
        @ 10,0 say „Regal          “ get regalx
        text
```



```
********************************* EINGABE-VERFAHREN *********************************
*                                                                                   *
*                          Drücken Sie bitte CTRL und gleichzeitig:                 *
*                                                                                   *
*    Zur Bewegung des Cursors              *   Zum                                  *
*                                          *                                        *
*    E Cursor nach oben                    *   G Löschen                            *
*    X Cursor nach unten                   *   V Insert Einschalten                 *
*    S Cursor nach links                   *   V Insert Ausschalten                 *
*    D Cursor nach rechts                  *                                        *
*                                          *                                        *
*************************************************************************************
                        endtext
                        read
                        store titelx1+titelx2 to titelx
                        replace titel with titelx, autor with autorx, gebiet with gebietx;
                        raum with raumx, regal with regalx
                endif
        endif
    endif
enddo


*=============================== BUECHER3.PRG ===============================
*
*    CASE OPTION 3 : EINTRAG LÖSCHEN
*
*============================================================================
*
use buecher index btitel
erase
store 0 to geloescht
do while T
        erase
        @  1,30 say „EINTRAG LÖSCHEN“
        store str(1,81) to freilassen
        store $(freilassen,1,40) to titelx1
        @  5,0 say „Titel “ get titelx1;
           picture „!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!“
        @  9,0 say „(Versuchen Sie es mit einem Buchstaben,“
        @ 10,1 say „wenn Ihnen der Titel nicht einfällt)“
        @ 12,0 say „Drücken Sie <RETURN>, um aufzuhören“
```

```
read
if titelx1 = " "
        erase
        @ 2,0 say „Warten Sie bitte einen Augenblick . . . .“
        if geloescht > 0
                ?
                ? „Jetzt werden die markierten Sätze mit PACK endgültig gelöscht“
                pack
        endif
        return
else
        store trim(titelx1) to titelx
        find &titelx
        if #=0
                @ 14,0 say titelx
                @ 15,0 say „Nicht gefunden“
                @ 18,0 say „Drücken Sie irgendeine Taste, um weiterzumachen“
                set cons off
                wait
                set cons on
        else
                store „M“ to mehr
                do while mehr = „M“
                        erase
                        @ 1,30 say „EINTRAG LÖSCHEN“
                        if *
                                @ 3,25 say „Bereits zur Löschung markiert“
                        endif
                        @  5,0 say „Titel          “+$(titel,1,40)
                        @  6,0 say „               “+$(titel,41,40)
                        @  7,0 say „Autor          “+autor
                        @  8,0 say „Sachgebiet     “+gebiet
                        @  9,0 say „Raum           “+raum
                        @ 10,0 say „Regal          “+regal
                        @ 20,0 say „Drücken Sie D wenn Sie Obenstehendes löschen
                          wollen“
                        @ 21,0 say „ oder + um das nächste Buch anzusehen“
                        @ 22,0 say „ oder - um ein Buch zurückzugehen“
                        @ 23,0 say „ oder jede andere Taste, um ein anderes Buch
                          anzusehen“
                        store " " to naechstes
                        set cons off
                        wait to naechstes
                        set cons on
```



```
                        if naechstes = „+“ .or. naechstes = „-“
                                if naechstes = „+“
                                        skip
                                else
                                        skip -1
                                endif
                        else
                                store „N“ to mehr
                        endif
                enddo
                if naechstes = „D“ .or. naechstes = „d“
                        delete
                        store geloescht + 1 to geloescht
                        ?
                        ? „Satz zur Löschung markiert“
                        store 0 to top
                        do while top < 30
                                store top + 1 to top
                        enddo
                endif
        endif
    endif
enddo
```

```
* ═══════════════════ BUECHER4.PRG ═══════════════════
*
*    CASE OPTION 4 : BUCHTITEL SUCHEN
*
* ═══════════════════════════════════════════════════
*
use buecher index btitel
do while T
        erase
        @  1,30 say „BUCHTITEL SUCHEN“
        store str(1,81) to freilassen
        store $(freilassen,1,40) to  titelx1
        @  5,0 say „Titel“     get titelx1;
           picture „!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!“
        @  9,0 say „(Versuchen Sie es mit einem Buchstaben,“
        @ 10,1 say „wenn Ihnen der Buchtitel nicht einfällt)“
        @ 12,0 say „Drücken Sie <RETURN>, um aufzuhören“
        read
        if titelx1 = ” ”
                erase
                return
        else
                store trim(titelx1) to titelx
                find &titelx
                if #=0
                        @ 14,0 say titelx
                        @ 15,0 say „Nicht gefunden“
                        @ 18,0 say „Drücken Sie irgendeine Taste, um weiterzumachen“
                        set cons off
                        wait
                        set cons on
                else
                        store „M“ to mehr
                        do while mehr = „M“
```



```
                                erase
                                @  5,0 say „Titel “+$(titel,1,40)
                                @  6,0 say „  “+$(titel,41,40)
                                @  7,0 say „Autor “+autor
                                @  8,0 say „Sachgebiet “+gebiet
                                @  9,0 say „Raum “+raum
                                @ 10,0 say „Regal “+raum
                                @ 21,0 say „Drücken Sie + um das nächste Buch anzusehen“
                                @ 22,0 say „ oder - um ein Buch zurückzugehen“
                                @ 23,0 say „ oder jede andere Taste, um ein anderes Buch
                                   anzusehen“
                                store ” ” to naechstes
                                set cons off
                                wait to naechstes
                                set cons on
                                if naechstes = ”+“ .or. naechstes = ”-“
                                        if naechstes = ”+“
                                                skip
                                        else
                                                skip -1
                                        endif
                                else
                                        store ”N“ to mehr
                                endif
                        enddo
                endif
        endif
enddo
```

```
*═══════════════════ BUECHER5.PRG ═══════════════════
*
*   CASE OPTION 5 : BÜCHER NACH AUTOREN ANZEIGEN
*
*═══════════════════════════════════════════════════
*
use buecher
do while T
        erase
        @  1,30 say „BÜCHER NACH AUTOREN ANZEIGEN“
        store str(1,81) to freilassen
        store $(freilassen,1,30) to autorx
        @  5,0 say „Autor              “ get autorx;
           picture „!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!“
        @  9,0 say „(Versuchen Sie es mit ALLE, wenn Ihnen der Autor nicht einfällt)“
        @ 11,0 say „Drücken Sie <RETURN>, um aufzuhören“
        read
        if autorx = ” ”
                erase
                return
        else
                store trim(autorx) to autorx1
                erase
                store len(autorx1) to zeichen
                if autorx1 = „ALLE“
                        list $(titel,1,33),$(autor,1,15),$(gebiet,1,10),$(raum,1,13),regal off
                else
                        disp $(titel,1,33),$(autor,1,15),$(gebiet,1,10),$(raum,1,13),regal;
                        for autorx1 =!($(autor,1,zeichen)) off
                endif
                ? „Drücken Sie irgendeine Taste, um weiterzumachen“
                set cons off
                wait
                set cons on
        endif
enddo
```



```
*═══════════════════ BUECHER6.PRG ═══════════════════
*
*   CASE OPTION 6 : BÜCHER NACH SACHGEBIETEN ANZEIGEN
*
*═══════════════════════════════════════════════════
*
use buecher
do while T
        erase
        @  1,30 say „BÜCHER NACH SACHGEBIETEN ANZEIGEN“
        store str(1,81) to freilassen
        store $(freilassen,1,20) to gebietx
        @  5,0 say „Sachgebiet         „ get gebietx;
           picture „!!!!!!!!!!!!!!!!!!“
        @  9,0 say „(Versuchen Sie es mit ALLE,“
        @ 10,1 say „wenn Sie sich nicht erinnern können)“
        @ 12,0 say „Drücken Sie <RETURN>, um aufzuhören“
        read
        if gebietx = ” ”
                erase
                return
        else
                store trim(gebietx) to gebietx
                erase
                store len(gebietx) to zeichen
                if gebietx = „ALLE“
                        list $(titel,1,33),$(autor,1,15),$(gebiet,1,10),$(raum,1,13);
                        regal off
                else
                        disp $(titel,1,33),$(autor,1,15),$(gebiet,1,10),$(raum,1,13),regal;
                        for gebietx =!($(gebiet,1,zeichen)) off
                endif
                ? „Drücken Sie irgendeine Taste, um weiterzumachen“
                set cons off
                wait
                set cons on
        endif
enddo
```

```
* ═══════════════ BUECHER7.PRG ═══════════════
*
*    CASE OPTION 7 : BÜCHER NACH STANDORT ANZEIGEN
*
* ═══════════════════════════════════════════
*
use buecher
do while T
        erase
        @  1,30 say „BÜCHER NACH STANDORT ANZEIGEN“
        store str(1,81) to freilassen
        store $(freilassen,1,15) to raumx
        store $(freilassen,1,3) to regalx
        @  5,0 say „Raum              “ get raumx picture „!!!!!!!!!!!!!!!“
        @  6,0 say „Regal             “ get regalx picture „!!!“
        @  9,0 say „(Versuchen Sie es mit ALLE, wenn Ihnen der Raum nicht einfällt)“
        @ 11,0 say „Drücken Sie zweimal <RETURN>, um aufzuhören“
        read
        if raumx = " "
                erase
                return
        else
                if regalx = " "
                        store trim(raumx) to ortx
                else
                        store raumx+regalx to ortx
                endif
                erase
                store len(ortx) to zeichen
                if ortx = „ALLE“
                        list $(titel,1,33),$(autor,1,15),$(gebiet,1,10),$(raum,1,13);
                        regal off
                else
                        if regalx = „ „
                                disp $(titel,1,33),$(autor,1,15),$(gebiet,1,10),$(raum,1,13);
                                regal;
                                for ortx =!($(raum,1,zeichen)) off
                        else
                                disp $(titel,1,33),$(autor,1,15),$(gebiet,1,10),$(raum,1,13),regal;
                                for raumx =!(raum) .and. regalx =!(regal) off
                        endif
                endif
                ? „Drücken Sie irgendeine Taste, um weiterzumachen“
                set cons off
                wait
                set cons on
        endif
enddo
```



```
* ═══════════════ BUECHER8.PRG ═══════════════
*
*    CASE OPTION 8 : BÜCHERVERZEICHNIS AUSDRUCKEN
*
* ═══════════════════════════════════════════
*
erase
? „BÜCHERVERZEICHNIS AUSDRUCKEN“
?
?
? „Schalten Sie bitte den Drucker ein“
?
? „Drücken Sie irgendeine Taste, wenn Sie ihn eingeschaltet haben“
?
set cons off
wait
set cons on
do while T
        erase
        store " " to option
        @  1,30 say „WÄHLEN SIE BITTE, WAS SIE DRUCKEN WOLLEN“
        @  9,0 say "1. Titelverzeichnis“
        @ 10,0 say "2. Autorenverzeichnis“
        @ 11,0 say "3. Sachgebietsverzeichnis“
        @ 12,0 say "4. Raum/Regal-Verzeichnis“
        @ 16,0 say "X. Ende“
        @ 18,0 say "Geben Sie bitte Ihre Wahl ein“ get option picture „!“
        read
        @ 23,0 say "Warten Sie bitte einen Augenblick . . . .“
        do case
                case option = "X“
                        erase
                        return
                case option = "1“
                        use buecher index b+titel
                        report form buecher to print
                case option = "2“
                        use buecher
                        ? „Indizieren nach Autoren“
                        index on!($(autor,1,12))+!($(titel,1,12)) to autoren
                        report form buecher to print
                case option = "3“
                        use buecher
```

```
                ? „Indizieren nach Sachgebieten“
                index on!($(gebiet,1,10))+!($(autor,1,10)) to sgebiet
                report form buecher to print
            case option = „4“
                use buecher
                ? „Indizieren nach Räumen und Regalen“
                index on!($(raum,1,4)+regal+$(autor,1,5)) to raeume
                report form buecher to print
        endcase
enddo
```

Anhang B: Register

Befehlsverzeichnis Seite

(Diese Seite wurde
absichtlich nicht bedruckt)

ANHANG B FEHLERMELDUNGEN

Ausdruck ist nicht numerisch
SUM erfordert einen numerischen Ausdruck für die Summierung.

Befehlsdatei nicht vorhanden
Dateiname/Laufwerksangabe prüfen.

Dateiende erreicht
Datenbankdatei weist Fehler auf; gegebenenfalls kopieren bzw. reindizieren.

Dateiende unerwartet erreicht
Indexdatei paßt nicht zur Datenbank. Eventuell neuindizieren.

Datei bereits vorhanden
Nicht benötigte Datei gleichen Namens löschen oder anderen Namen verwenden.

Datei kann nicht geöffnet werden
Prüfen, ob Datei vorhanden ist.

Datei ist bereits eröffnet
Mit USE oder CLEAR aktive Dateien abmelden.

Datei ist nicht vorhanden!
Prüfen, ob Datei vorhanden ist, z.B. mit DISPLAY FILES LIKE *.*

Dateinummer nicht zugeordnet
Prüfen, ob DBASEMSG.TXT vorhanden ist; Fehlermeldung an Händler

Datenelement nicht gefunden
Prüfen, ob Feldbezeichnung korrekt, bzw. Feld in Struktur vorhanden. REPLACE-Befehl neu eingeben.

Die „FIELD"-Angabe fehlt
CHANGE-Kommando neu eingeben.

Die „FOR"-Angabe fehlt
JOIN erneut mit „FOR"-Angabe eingeben.

Die „FROM"-Angabe fehlt
UPDATE erneut mit „FROM"-Angabe eingeben.

Die „ON"-Angabe fehlt
UPDATE oder INDEX mit „ON"-Angabe neu formulieren.

(Diese Seite wurde
absichtlich nicht bedruckt)

Die „TO"-Angabe fehlt
Kommando mit „TO"- Angabe neu eingeben.

Die „WITH"-Angabe fehlt
REPLACE mit „WITH"-Angabe neu eingeben.

Diskettenverzeichnis ist voll
Nicht benötigte Dateien löschen.

Diskette ist voll
Nicht benötigte Dateien löschen.

*** Division durch null

Do abgebrochen

Einfügung nicht möglich - keine Sätze in der Datenbank
INSERT durch APPEND ersetzen.

Ende des Such ('LOCATE')- Bereichs

Es fehlt der numerische Ausdruck zum summieren
SUM erfordert Angabe des/der zu summierenden Feldes(r).

Es wurde noch keine Formatdatei ausgewählt
Format-Datei aktivieren.

Formatdatei kann nicht geöffnet werden
".FMT"-Datei prüfen, z.B. durch Auflisten.

Indexdatei kann nicht geöffnet werden
Dateiname prüfen oder Datenbank neu indizieren.

Indexdatei paßt nicht zur Datenbank
Nur zur Datenbankdatei gehörende Index-Dateien verwenden.

Keine Datenbank eröffnet, bitte Dateinamen eingeben:

Keine dBASE II Datenbank

MACRO ist keine Zeichenfolge
Makroexpansion verlangt Variablen mit Zeichenreihe als Wert.

MAKRO nicht gefunden

Maximal 5 Felder für Summenbildung möglich
SUM nur mit maximal 5 Feldern.

Maximale Schachtelungsgrenze überschritten
Programmdateien überarbeiten und zusammenfassen.

Mehr als 7 indizierte Dateien sind aktiviert

Mit JOIN wurde versucht, mehr als 65.534 Sätze zu erzeugen
Die „FOR"-Bedingung einengen.

OVERLAY-Datei kann nicht eröffnet werden
DBASEOVR.COM fehlt.

Satz nicht in Indexdatei enthalten
Index-Datei nicht aktuell; Datenbankdatei reindizieren.

Satz überschreitet belegten Bereich der Datenbank
Datei überprüfen (ggf. COPY), bzw. reindizieren.

Satzlänge überschreitet Maximalwert von 1000 Bytes
Feldlängen kürzen, so daß Satzlänge <= 1000 Bytes.

Schlüssel haben nicht die gleiche Länge
UPDATE verlangt übereinstimmende Schlüssel.

Speicher für temporäre Variablen ist voll
Nicht benötigte temporäre Variablen löschen.

*** Syntaxfehler ***
Unbekanntes Kommando, neu formulieren.

Syntaxfehler bei der Formatdefinition
Ein @ SAY GET PICTURE wurde fehlerhaft formuliert.

Syntaxfehler, Eingabe bitte wiederholen

Syntaxfehler bei Ein- oder Ausgabe

*** Überlauf im nummerischen Feld

*** Unbekannter Befehl
Formulierung prüfen.

Ungültiger Datentyp
SORT kann nicht auf logischen Feldern sortieren.

Ungültiger Variablenname
Nur alphanumerische Zeichen und Doppelpunkte sind als Namen für Felder und Variable zugelassen.

Ungültiger Wert bei „GOTO"
Satz-Nr. muß im belegten Bereich der Datenbank und im Bereich zwischen 1 und 65.535 liegen.

Unterschiedliche Datentypen in Quelle und Ziel
Datentypen müssen übereinstimmen.

Unzulässige Dezimalstellen im Feld
Anzahl der Dezimalstellen bei Feld-Definition ist neu einzugeben.

Unzulässige Feldlänge
Feldlänge minimal 1, maximal 255 Stellen. BeiString-Befehlen (Zeichenketten) darf Stellenangabe den Wert 255 nicht übersteigen.

Unzulässiger Dateiname
Syntaxfehler im Datei-Namen.

Unzulässiger Feldname
Unzulässige Zeichen/Länge im Feld-Namen; neu eingeben.

Unzulässiger Feldtyp
Nur C (Character), N (Numerisch) oder L (Logisch) zugelassen.

Variable nicht vorhanden
Temporäre Variable erzeugen, bzw. Name korrigieren.

Verwendete Datenbank ist nicht indiziert
FIND ist nur auf indizierten Datenbankdateien möglich.
Index-Datei mit USE anschließen.

Wert nicht gefunden
Nicht unbedingt ein Fehler: der Schlüsselwert ist nicht vorhanden (= 0).

Zeichenfolge ist unzulässig

Zu viele Dateien eröffnet
Nur bis zu 16 Dateien (aller Typen) können gleichzeitig aktiviert werden.

Zu viele <RETURNs> benutzt
Struktur der Programmdateien überprüfen (insbes. Schachtelung, Schleifen).

Zu viele temporäre Variable
Höchstens 64 temporäre Variable sind möglich; nicht benötigte Variable löschen.

Zu viele Zeichen (Befehlszeile zu lang)
Feldlänge bzw. maximale Länge der Befehlszeile wurden überschritten.

(Diese Seite wurde
absichtlich nicht bedruckt)

dBASE II

Zusammenfassung der Veränderungen von Version 2.4 nach 2.41

Erweiterungen und Neue Befehle

§<Koordinaten> GET <Variable> PICTURE „AAAAAAAA" -- Leerschritte

Zusätzlich zu den bekannten Alpha-Zeichen, die bei der PICTURE-Klausel als Eingabe erlaubt waren, ist jetzt auch das Leerschritt-Zeichen zulässig.

CLEAR -- In Zusammenhang mit den FORMAT-Dateien

In Erweiterung der Funktion des CLEAR-Befehls werden mit ihm nun auch die FORMAT-Dateien geschlossen. Der Befehl SET FORMAT TO ohne weitere Angaben schließt auch weiterhin nur die FORMAT-Dateien.

CREATE -- Abbruch mit der <ESCAPE>-Taste

Vorher reagierte dBASE II auf die Benutzung der <ESCAPE>-Taste nach der Frage „Bitte Dateinamen eingeben:" nicht immer korrekt.

Nun erlaubt dBASE II nach der Frage: "Bitte Dateinamen eingeben:" den Abbruch des Vorgangs mittels der <ESCAPE>-Taste.

CREATE <Neue Datei> FROM <structure extended> -- Begrenzung

In der vorherigen dBASE II-Version war die Anzahl der neuen Dateien, die mit dieser Methode erzeugt werden konnten, auf 14 begrenzt.

Nun gibt es keine Begrenzung mehr.

INSTALL -- Freie Wahl der Feldbegrenzungszeichen

Für die maskenorientierte Verarbeitung können die Feldbegrenzungs-Zeichen jetzt frei gewählt werden. Vorbelegt sind weiterhin die Doppelpunkte. Anstelle dieser können z.B. zwei „[ ]“ oder „< >“ gewählt werden.

INS- und **DEL**-Tasten - sind nun möglich.

Die Einfüge- (INS) und Lösch- (DEL) Tasten arbeiten nun im dBASE II-Programmablauf genauso, wie im DOS-Betriebssystem selbst.

* INS schaltet die Einfügefunktion EIN oder AUS
* DEL löscht das Zeichen an der jeweiligen CURSOR-Position

JOIN, SORT, und **TOTAL** -- Abbruch mit der <ESCAPE>-Taste

Wenn die <ESCAPE>-Taste bei den Befehlen JOIN, SORT oder TOTAL gedrückt wird, erfolgt automatische das Schließen der Datei.

In der alten Version blieben die Dateien offen. Dadurch erfolgte jeweils die Meldung „Datei ist bereits eroeffnet“, wenn sie angesprochen wurden.

REPORT FORM <Formatdateiname> -- Fehler in der Formatdatei

Wenn der REPORT früher einen Fehler in der Formatdatei erkannte, wurde die Meldung „Syntaxfehler“ angezeigt und der Fehlerkorrektur-Dialog angeboten. Jeder Versuch ‚diesen Fehler zu beheben und weiter zu arbeiten, ergab allerdings die Meldung „Datei ist bereits eroeffnet“.
Nun wird der REPORT im Fehlerfalle abgebrochen, und es erscheint die Meldung „Fehler bei Formatdefinition“. Wenn der REPORT direkt von der Eingabezeile gestartet wurde, also vom Promptpunkt, erscheint dieser wieder. Beim Aufruf des REPORT aus einem Programm, wird der REPORT abgebrochen und das Programm mit dem nächsten Befehl fortgesetzt.

RESTORE FROM <MEM-Dateiname> ADDITIVE -- Speichervariablen mit gleichem Namen

Wenn eine Speichervariable mit dem selben Namen wie in der MEM-Datei existiert und der RESTORE-Befehl mit der Erweiterung ADDITIVE verwendet wird, werden die im Speicher befindlichen Variablen überschrieben.

SAVE TO <MEM-Dateiname> ALL EXCEPT <Bereich> - Neue Parameter

Zusätzlich zur Bereichseingrenzung ALL LIKE <BEREICH> gibt es nun auch die Möglichkeit, ALL EXCEPT <Bereich> anzugeben.

Korrekturen

@ <Koordinaten> GET <logische Variable> READ
-- Wert bleibt unverändert

In der vorherigen Version wurden logische Felder, die den Wert (.T.) enthielten, beim Drücken der <RETURN>-Taste ohne zuvorige Eingabe auf (.F.) gesetzt.
In der neuen dBASE II-Version bleibt der ursprüngliche Wert jetzt bestehen und wird nicht mehr automatisch auf (.F.) gesetzt.

APPEND FROM <Datenbank-Name> -- Abbruch mit <ESCAPE>

Vorher: Wenn man beim APPEND FROM-Befehl die <ESCAPE>-Taste zum Programmabbruch verwendete, wurde die benutzte Datenbank zerstört.

Nun werden alle Sätze korrekt angehängt.

APPEND FROM <Datenbank-Name> -- 1000 Zeichen-Sätze

Wenn die aktivierte und die FROM-Datenbank gleiche Strukturen mit je 1000 Byte-Sätze hatten, wurden nur leere Sätze angehängt.

Nun erfolgt die Übernahme der Daten korrekt.

APPEND FROM <Text-Dateiname>.TXT -- SDF-Parameter vergessen

Wenn versucht wurde, ein APPEND FROM von einer Text-Datei ohne Angabe des Parameters 'SDF' zu starten, meldete sich dBASE II mit „Keine dBASE II Datenbank".
Wenn nun versucht wurde, den Fehler mittels des Fehlerdialogs zu korrigieren, erschien die Meldung: „Datei ist bereits geoeffnet".
Nun erfolgt nach der Korrektur die tatsächlich gewünschte Befehlsausführung.

Befehlsdateien -- Mit anderen Textverarbeitungsprogrammen erstellt

In der vorherigen Version wurde das „Ende der Datei" nicht immer erkannt, wenn die Befehlsdateien mit anderen Textverarbeitungs-Programmen als MODIFY COMMAND durchgeführt wurden. Dieses hatte zur Folge, daß der letzte Teil eines 128-Byte-Blockes nicht erkannt, bzw. ausgeführt wurde.

Nun erfolgt die Programmausführung auch, wenn der letzte Block kleiner als 128 Byte ist.

CREATE [<Datenbank-Name>] -- Mit MS(PC)-Dos Versionen 2.01 & 2.1

Die vorherigen dBASE II-Versionen brachten die Meldung: "Dateiende unerwartet erreicht", wenn der CREATE-Befehl gestartet wurde. Die genannten DOS-Versionen erlauben nämlich Umlaute in den Dateinamen.

dBASE II unterstützt nun diese Möglichkeiten selbst und die Fehlermeldung bleibt aus.

INDEX -- Veränderungszeiten

Jetzt haben die langen Wartezeiten bei der INDEX-Veränderung ein Ende. Das gilt zum Beispiel für die Befehle: APPEND, EDIT und READ.

INSERT - Mit einer FORMAT-Datei

Wenn in der vorherigen Version ein INSERT-Befehl im Zusammenhang mit dem SET FORMAT TO -Befehl durchgeführt werden sollte, erschien nach Abschluß der Eingabe die Meldung: "Satz überschreitet belegten Bereich der Datenbank" .
Nun erfolgt keine Fehlermeldung mehr, und dBASE II führt den Befehl korrekt aus.

JOIN TO <neuer Datenbankname> for P.<Schlüssel> = S.<Schlüssel>
-- Doppelte Felder

Wenn beim Benutzen des JOIN-Befehls in beiden Datenbanken der gleiche Feldname existierte, wurden doppelte Felder angelegt. Wenn in der neuen Datenbank nun mehr als 32 Felder pro Satz erzeugt wurden, stürzte das ganze System ab und es war keine Bearbeitung mehr möglich.

Jetzt erzeugt JOIN keine doppelten Felder mehr, und die neue Datei enthält nur die ersten 32 Felder.

Logische Felder -- In einer großen Datenbank-Struktur

Wenn in älteren Versionen die Datenbank-Struktur nahe an der Grenze von 1000 Zeichen pro Satz kam und ein logisches Feld hinter einem numerischen Feld angelegt war, zerstörte der LIST-Befehl die Datei-Struktur.

Nun ist der Zugriff auf diese Struktur ohne Probleme gewährleistet.

QUIT TO <auszuführende Befehls-, Programmliste> -- 16 Bit Systeme

QUIT TO ist implementiert auf den Betriebssystemen CP/M-86, Version 1.1 und MS-DOS, Version 2.0. Der Befehl funktioniert in der CP/M-86 Version exakt genauso, wie in der 8-Bit Version 2.0: Um zu dBASE II zurückzukehren, muß dBASE in der <auszuführenden Befehls-, Programmliste> genauer spezifiziert werden.
Unter MS(PC)-DOS gelangt man nach Beendigung der <auszuführenden Befehls-, Programmliste> wieder zurück zur dBASE II-Hauptbefehlsebene. Wird aus einem Programm der QUIT TO -Befehl aufgerufen, so wird nach dessen Ausführung bei der nächsten Code-Zeile des Programms fortgefahren. Memory-Variable und Parameter werden nicht zerstört, jedoch werden alle aktivierten Dateien und die dazugehörigen Index- und Formatdateien geschlossen. Wenn dBASE in der <auszuführenden Befehls-, Programmliste> spezifiziert wurde, startet ein neuer dBASE II - Lauf. Die COMMAND.-COM-Datei Ihres DOS-Betriebssystems muß sich in jedem Fall auf demselben Laufwerk befinden, von dem auch dBASE II aufgerufen wird.

REPLACE <numerischer Feldname> with <nummer> --Größer als 50 Stellen

In der alten Version wurde die mit USE aktivierte Datei zerstört, wenn die <Nummer> mehr als 50 Stellen hatte. Nun wird die <Nummer>, die in das Feld geschrieben wird, auf 34 Stellen gekürzt. Nur die ersten 10 Ziffern sind wichtig, der Rest sind Nullen. Die vollständige Nummer waird im Feld rechts justiert.

SKIP <negative memvar> -- Negative <memvar> wird beachtet

SKIP hat früher die <negative memvar> als einen positiven Wert interpretiert. Die Interpretation erfolgt jetzt korrekt.

SKIP VAL ("-1") -- Negative Ziffer wird beachtet

VAL ("-1") wurde früher von SKIP als ein positiver Wert interpretiert. Das Minus-Zeichen wird nun beachtet. SKIP springt die angegebene Anzahl von Sätzen zurück.

System Date -- Format ändern

Wie in den früheren Versionen ist die Wahl des Datumformates im INSTALL-Programm verfügbar: amerikanische Schreibweise (MM/TT/JJ) oder europäische Schreibweise (TT/MM/JJ). Bis jetzt konnte das Datum vorher nur einmal ausgewählt werden. In der neuen Version kann das Datum jederzeit im INSTALL-Programm wieder geändert werden.

TOTAL ON <Schlüsselfeld> TO <Dateiname> -- Feldlängendifferenzen

Die beiden genannten Datenbanken brauchten früher identische numerische Felder, da sonst die Summenbildung unkorrekt ausgeführt wurde.
Nunmehr können die Felder der TO-Datenbank, in die summiert werden soll, länger als die Felder der USE-Datei sein. Ist das numerische Feld lang genug, um die Gesamtsumme aufzunehmen, wird der Befehl korrekt ausgeführt.

UPDATE mit **REPLACE** -- Numerischer Überlauf

Wird jetzt ein UPDATE mit REPLACE von einem numerischen Feld in ein kleineres Feld der Master-Datei ausgeführt, wird die Zahl nicht mehr gekürzt. Es werden zum Zeichen des numerischen Überlaufs Sterne (*****) in das Feld miteingebracht.

Zeilenumbruch auf dem Bildschirm -- Verlust von Zeichen

Beim automatischen Zeilenumbruch nach 80 Zeichen wurde bei manchen Bildschirmen bisher das 81. Zeichen unterdrückt.

Nun werden alle Zeichen angezeigt.

Zusätzliche Dokumentation

@ <Koordinaten> GET <numerische Variable> PICTURE „<Format>" Überlauf

Wenn die <Format>Länge kleiner als die <num. Variable> ist, wie in dem nachfolgenden Beispiel gezeigt:

```
STORE 123.45 TO Zahl
@ 5,5 GET Zahl PICTURE „99.99"
READ
```

erscheint die Meldung:

```
**34
```

Die Sterne zeigen den numerischen Überlauf an.

Wenn eine Teilzahl verwendet wird, so muß das <Format> groß genug gewählt werden, um die Null vor dem Komma und den Dezimalpunkt aufnehmen zu können.

```
STORE .12 TO Dezimal
@ 5,5 GET Dezimal PICTURE „#.##"
```

@ <Koordinaten> GET <num. Feld> PICTURE „<Format>" Überlauf

Hat das <num. Feld> Dezimalstellen, wird die Dateneingabe nicht korrekt ausgeführt, wenn über den Dezimalpunkt geschrieben werden soll. Das rührt daher, daß ein Fließkomma nicht wie in einer Memory-Variablen überschrieben werden kann. Benutzen Sie eine Speichervariable für die Eingabe und den REPLACE-Befehl für die Datenübergabe in ein Datenbankfeld.

@ <Koordinaten> SAY <Ausdr.> USING „<Format>" Erweiterungen

dBASE II erweitert nicht mehr die Zeichenkette in dem USING-Format wie es in der Version 2.3 üblich war. Zum Beispiel:

```
STORE '120883' TO mdatum
@ 5,5 SAY mdatum USING '99/99/99'
```

Als Ergebnis erscheint: '12/88/ ' Um die Ausgabe richtig zu erzeugen, muß die $-Funktion zur Erzeugung einer Teilzeichenkette verwendet werden.

```
@  5,5 SAY $(mdatum,1,2) + '/' + $(mdatum,3,2) + '/' +
$(mdatum,5,2)
```

Als Ergebnis erscheint nun: '12/08/83'

DELETE FILE <Parameter> -- Gültige bzw. ungültige Befehlsform

Die Befehlserweiterungen LIKE <Bereich> und EXCEPT <Bereich> sind nicht erlaubt. Die einzig erlaubte Form für die Parameter ist die Benutzung des Fragezeichens in der gleichen Konvention, wie im Betriebssystem. Der Stern (*) ist jedoch nicht erlaubt. Zum Beispiel:

```
DELETE FILE LIKE TE??.DBF
```

löscht alle Dateien die mit „TE" beginnen und als Erweiterung (Typkennzeichnung) „DBF" haben.
Hingegen bringt die Benutzung von:

```
DELETE FILE LIKE TE*.DBF
```

die Fehlermeldung: „Datei nicht gefunden"

FUNKTIONSTASTEN -- Aufgerufene Befehle lassen sich nicht abbrechen

Wenn Befehle von den Funktionstasten aus gestartet werden, ist kein Abbruch mit Hilfe der Taste <ESCAPE> möglich, wenn ein Semikolon (;) als Abschluß des Befehls einprogrammiert ist. Wenn mehr als ein Befehl in dieser Art gestartet wurde, erfolgt die Abarbeitung, bis alle Befehle ausgeführt sind. Erst danach erfolgt eine Rückkehr zum Eingabe-Promptpunkt von dBASE II.

Das Semikolon (;) am Ende eines Befehls erzeugt ein automatisches <RETURN> und startet die Befehlsausführung. Wenn ein Abbruch erforderlich ist, muß der entsprechende Befehl über die normale Tastatur eingegeben werden, und darf nicht von den Funktionstasten aus gestartet werden.

GO/GOTO <Parameter>-- Auf einen zur Löschung markierten Satz positionieren

Der GO/GOTO-Befehl positioniert den dBASE II-internen Zeiger auf einen (Daten-) Satz auch, wenn dieser zur Löschung markiert wurde. Der Datensatz wird ebenfalls angezeigt, wenn der Befehl SET DELETED ON gegeben wurde. Dasselbe gilt für GO TOP. Wie Sie am nachfolgenden Beispiel sehen, kann man unter Verwendung dieser Eigenschaft zur Löschung markierte Sätze finden:

```
IF *
    skip
ELSE
    DISPLAY
ENDIF
```

INSERT BLANK -- Mit einer geöffneten INDEX-Datei

Mit dem Befehl INSERT BLANK kommt man bei einer geöffneten INDEX-Datei in den bildschirmorientierten Editiermodus (FULL SCREEN). Um einen leeren Satz einzufügen, während eine INDEX-Datei geöffnet ist, sollte der Befehl APPEND BLANK verwendet werden.

LOCATE <Bereich> FOR <log. Ausdr.> 'WHILE' nicht erlaubt.

Der Gebrauch der Befehlserweiterung WHILE anstelle von FOR ist nicht erlaubt. Bei seiner Verwendung erfolgt eine Fehlermeldung.

MODIFY STRUCTURE -- Bekannte Fehler in der PC-DOS Version

Die nachfolgend aufgelistete Befehlsfolge erzeugt eine undefinierte Bildschirmanzeige und das System stürzt ab:

```
        USE <Datenbankname>
MODIFY STRUCTURE
<CONTROL C>
<CONTROL-C>
<CONTROL-N>
<CONTROL-T>
<CONTROL-Q oder -W>
MODIFY STRUCTURE
```

Ein Einfügen eines WAIT-Befehls vor dem zweiten MODIFY STRUCTURE-Befehl behebt dieses Problem.

PEEK <Adresse> -- Bekannter Fehler

Die PEEK()-Funktion ist im Bereich von 1000 HEX (4096 Dezimal) bis 7A00 HEX (31232 Dezimal) nicht durchführbar. Es wird zwar ein Wert angezeigt, wenn der Befehl in diesem Bereich benutzt wird, aber der Wert ist nicht korrekt.

Relative Adressierung auf dem Drucker-- nicht implementiert.

Der Gebrauch des Dollar-Zeichens in der Koordinatenangabe beim „ @.,. SAY"-Befehl wird nicht unterstützt, wenn der Befehl SET PRINT ON gegeben wurde. Die relative Adressierung ist nur für dem Bildschirm zulässig.

REPLACE P.<Feld> with S.<Feld> -- Selektion des entsprechenden Bereiches

Der Befehl REPLACE erlaubt den Austausch eines Wertes in einem Feld der Datenbank, die geöffnet ist und sich nach dem entsprechenden SELECT-Befehl im aktuellen Zugriff befindet. Dabei muß man auf die richtige Verwendung von „P" und "S" achten. Das erste nachfolgende Beispiel ist möglich:

```
SELECT PRIMARY
REPLACE P.<Feld> WITH S.<Feld>
```

Das nächste ist jedoch nicht möglich:

```
SELECT PRIMARY
REPLACE S.<Feld> WITH P.<Feld>
```

SET COLOR TO <n1>,<n2> -- ohne Angabe von <n1>

Die Version 2.4 führte den Befehl SET COLOR TO <N2> auch aus, wenn die Angabe von <n1> fehlte. Diese Befehlsform sollte jedoch nie verwendet werden, obwohl Version 2.4 dies ermöglichte. Die Version 2.41 verändert die Farbattribute nun nicht eher, bis beide Parameter angegeben worden sind. Auch wird keine Syntax-Fehlermeldung erzeugt.

SET LINKAGE ON -- Unkorrekte Anwendung

Die Verbindung muß vor dem Öffnen der Datenbank geschehen. Wenn die Dateien geöffnet sind, und dann erst der Befehl SET LINKAGE ON gegeben wird, erfolgt keine Verbindung, und Befehle wie etwa LIST, die sequentiell über die Datei gehen, arbeiten fehlerhaft.

SORT ON <Schlüssel> TO <Sortier-Dateiname> -- Nach Satzanfügen

Wenn nach dem Anfügen von (Daten-)Sätzen der SORT-Befehl benutzt werden soll, muß die Datei erst geschlossen und wieder eröffnet werden. Erst danach kann sortiert werden. Dieser Vorgang läuft sehr einfach ab, wie das nachfolgende Beispiel zeigt:

```
USE Datei2
APPEND
USE Datei2
SORT ON <Schlüssel-Feld> TO <Sortier-Dateiname>
```

STORE -- Zeichenketten länger als 254 Zeichen

Der Befehl STORE <Zeichenkette1>+<Zeichenkette2> TO <Zeichenkette3> erlaubt die Zusammenstellung von bis zu 254 Zeichen in der <Zeichenkette3>. Die Meldung „Syntaxfehler" erscheint, wenn diese Kette länger als 254 Zeichen wird.

Wenn in einer Zeichenkette von 254 Zeichen ein Fehler existiert, der behoben werden soll, erscheint die Meldung "Syntaxfehler" ebenso, wenn die Kette bei der Korrektur länger wird. Damit das verhindert wird, muß der fehlerhafte Teil erst gelöscht werden, bevor eine einwandfreie Korrektur möglich ist.

STORE <Ausdruck> TO<Speichervarieble> -- Veränderung des Variablentyps

Weder die Funktion STR() noch VAL() können zur Veränderung des Variablentyps verwendet werden, wenn die Variable mit ihrem eigenen Namen, also quasi auf sich selbst, gespeichert wird. Das Ergebnis einer solchen Vorgehensweise wird immer "0" sein.

```
STORE „123" TO char
STORE VAL(char) TO char
```

oder

```
STORE 123 TO num
STORE STR(num,3) TO num
```

Wenn der Speicher-Variablentyp geändert werden soll, müssen Sie eine neue Variable mit der Funktion erzeugen und dann die alte Variable überschreiben:

```
STORE „123" TO char
STORE VAL(char) TO char2
STORE char2 TO char
```

TOTAL ON <Schlüsselfeld> TO <Dateiname> -- Teilzeichenkette nicht erlaubt

Die Teilzeichenketten-Funktion ist nicht als Parameter im Schlüsselfeld erlaubt. Wenn eine Summenfunktion (TOTAL) über eine Teilzeichenkette erzeugt werden soll, müssen Sie dazu ein zusätzliches Feld an die Datenbank anhängen. In dieses Feld werden dann per Teilzeichenketten-Funktion die Werte aus dem Schlüsselfeld übertragen. Danach wird dieses Feld mit einem Index belegt, der als Zugriffsweg verwendet werden kann.

TOTAL ON <Schlüsselfeld> TO <Dateiname> -- Felder müssen groß genug für die Summe sein

Wenn die TO-Datenbank nicht existiert, müssen die Felder der benutzten Datenbank groß genug sein, um die Summe abspeichern zu können. Die Struktur der neu erstellten TO-Datenbank wird nicht verändert und kann deshalb falsche Ergebnisse enthalten, wenn die Zielfelder zu klein sind.
Mit der Veränderung der Feldlänge für die TO-Datenbank kann man verhindern, daß dieses Problem auftaucht.

Stichwortverzeichnis

(Diese Seite wurde
absichtlich nicht bedruckt)

Anhang C: Dienstprogramme

Dienstprogramm ZIP
Bildschirm- und Druck-Maskengenerator

(Diese Seite wurde
absichtlich nicht bedruckt)

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 einen 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>, <ZEILENVORSCHUB>, <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.

(Diese Seite wurde
absichtlich nicht bedruckt)

(Für den englischen Begriff ZIP Talker (TM) wird in diesem Handbuch durchgehend das Wort „Dialogzeile“ verwendet.)

Hinweis: Beachten Sie bitte, daß dieser Masken-Generator zur Zeit nur in der 8-Bit-Version verfügbar ist!

Anpassen von ZIP an Ihren Rechner

Die Systemdiskette oder Zip-Diskette, 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 z. B. wird mit den 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 schikken. 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 der Zeichen in der Reihenfolge und 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:			EINGABE	Wenn angegebener Code:			EINGABE:
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-[*
12	C	FF	CTL-L	28	1C	FS	CTL-**
13	D	CR	<RETURN>	29	1D	GS	CTL-]*/ESC
14	E	SO	CTL-N	30	1E	RS	CTL-↑
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, drükken 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 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 II-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 ebensogut 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 II 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 Filtetyp angibt.

In einem Format-File hätte ZIP lediglich die ... SAY-Anweisungen eingefügt sowie alle in eckigen Klammern eingeschlossenen Einträge. Die SET-Anweisungen wären nicht erschienen, und der erste Kommentar hätte gelautet: * SAMPLE.FMT.

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-Werte 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 Bildschirmes 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.

Das Betriebssystem 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          (Das Befehls-Symbol wird auf der Hilfstafel nicht gebraucht)
9
```


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.

Beachte: 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 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.

Beachte: 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 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 entsprechen (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 im Betriebssystem 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 des Betriebssystems.

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 Ihres Betriebssystems 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-benutze eine 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 Bildschirmtafel 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. eingeben:

```
//
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 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 6.3:

```
                            ***** 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
```


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
```

dBASE-Dienstprogramm zur Umwandlung des europäischen Zeichencode

Mit diesem Dienstprogramm können Sie den internen dBASE-Code für die europäischen Sonderzeichen in den jeweiligen Zeichen-Code Ihres Computers umwandeln und umgekehrt.

Ein anschauliches Beispiel für seine Verwendung zeigt die Umwandlung einer Datei in eine Form, in der sie von anderen Programmen, bearbeitet werden kann.

Die Bewerkstelligung einer solchen Umwandlung beginnt man damit, daß man die betreffende Datei mit COPY in das Standard-Daten-Format (SDF) umkopiert. Die .SDF-Datei wird dann mit dBCNV bearbeitet und man erhält als Ergebnis eine ASCII-Datei. Sie enthält die maschineninternen Zeichencodes, mit deren Hilfe die europäischen Sonderzeichen von anderen Programmen auf dem Bildschirm ausgegeben und bearbeitet werden können.

Sie könnten dann beispielsweise eine Datei, die den Buchstaben „Ä" enthält, so verändern, daß aus dem dBASE-internen Code, der bei diesem Buchstaben 'hex ODE' lautet, der Code wird, den etwa der IBM-PC dafür verwendet, nämlich 'hex 84'.

Sie rufen dBCNV auf folgende Art auf:

- Sie benutzen die ganze Eingabezeile hinter dem Eingabe-Prompt Ihres Betriebssystems als Befehlszeile. Das Format für die Befehlszeile sieht bei dBCNV so aus:

'dbcnv <eingabedatei.ext> <ausgabedatei.ext> <richtung> [<overlay-datei.ext>]'

(".ext" steht für Dateityp-Kennzeichnung ("extension")

Die Parameter für die Bestimmung der (Umwandlungs-)'richtung' sind folgendermaßen definiert:

dte - wandelt internen dBASE-Code in externen Maschinencode um.
etd - wandelt externen Maschinencode in internen dBASE-Code um.

Sie dürfen in der Eingabezeile für den dBCNV-Befehl eine beliebige Anzahl von Parametern festlegen; das Programm wird den Benutzer gegebenenfalls nach zusätzlich benötigten Parametern fragen. Eine einzige Ausnahme ergibt sich, wenn Sie die als Option benennbare 'overlay-datei' näher spezifizieren wollen. In diesem Fall müssen alle entsprechenden Parameter unbedingt in der Befehlseingabezeile angeführt werden.

Beachten Sie bitte: Die Overlay-Datei, üblicherweise 'DBASEOVR.COM'genannt, wird von dBCNV unbedingt benötigt und muß sich deshalb auf dem Arbeitslaufwerk befinden, falls Sie die Overlay-Datei nicht näher spezifiziert haben.
Für die Runtime-Version von dBASE II muß jedoch die OVERLAY-Datei „DBRUNOVR.COM" spezifiziert werden.

Während eines Bearbeitungsvorgangs wird jedesmal eine Protokolldatei ("log file") angelegt, die den Namen 'dbcnv.log' trägt.

Die Protokolldatei ('DBCNV.LOG'), wird bei jedem dBCNV-Lauf angelegt und protokolliert alle Fehler-Hinweise, die während der Umwandlung ausgegeben werden. Fehler-Hinweise werden dann ausgegeben, wenn Ihr Computer das Zeichen nicht unterstützt, für das gerade der Code übersetzt werden soll. Bei der Umwandlung von dBASE-Code in Maschinen-Code wird dann in der Ausgabe-Datei ein Standardwert verwendet. Bei der Umwandlung von Maschinen-Code in dBASE-Code wird an entsprechender Stelle ein Sternchen ('*') eingesetzt.
- Zur Konvertierung von dBASE-Code zu Maschinen-Code können Sie auch folgendermaßen vorgehen:

Sie geben beim Eingabe-Prompt Ihres Betriebssystems ein:

'dbcnv <RETURN>

Das Konvertierungsprogramm meldet sich dann mit seinem Namen und fragt Sie nach der Datei, die Sie bearbeiten wollen.

dBASE II-European Code Converter Vx.x ....

Name der Eingabedatei:

Name der Ausgabedatei:

Internen dBASE-Code nach Maschinen-Code umwandeln? J/N: