Die Index-sequ

Das Arbeiten mit einer relativen Datei ist gar nicht so komfortabel wie man es eigentlich möchte. Der Zugriff auf Datensätze nur über die Datensatznummer ist nicht besonders benutzerfreundlich und in der Regel auch praxisfremd. Unter Zuhilfenahme einer sequentiellen Datei wird dieser Nachteil jedoch aufgehoben.

In den letzten beiden Ausgaben des 64'er wurden Ihnen die Grundzüge der sequentiellen und der relativen Datei vorgestellt und erläutert. Ich deutete auch schon an, daß erst die Verbindung dieser beiden Dateiformen zu einer befriedigenden Lösung führt. Aber auch hier muß man Abstriche machen. Durch diese Vermischung werden leider nicht nur die Vorteile, sondern auch ein Teil der Nachteile mit übernommen. Ich möchte diese noch einmal kurz anreißen.

Das Wesen der sequentiellen Datei liegt in der aufeinanderfolgenden (eben sequentiellen) Anordnung der Daten. Um mit dieser Datei arbeiten zu können, muß sie vollkommen in den Speicher des Computers geladen werden. Bei großen Datenbeständen kann das eine ganze Zeit lang dauern. Andererseits wird die maximale Größe durch den zu Verfügung stehenden Speicherplatz im Computer selbst bestimmt. Sind jedoch alle Datensätze erst einmal geladen, ist ein Bearbeiten der Daten sehr einfach und auch schnell. Dann können alle Änderungen, Ergänzungen und sonstige Manipulationen im Computer selbst durchgeführt werden. Und für zeitkritische Funktionen wie Sortieren und Suchen kann man die Maschinensprache sehr wirkungsvoll einsetzen.

Die relative Datei hingegen benötigt außer dem eigentlichen Programm nur sehr wenig Speicherplatz, nämlich nur noch denjenigen für die internen Variablen und für einen Datensatz. Der Rest aller Daten befindet sich auf Diskette. Und dort stehen einem fast 170 KByte zur Verfügung. Allerdings ist das Arbeiten mit einer rein relativen Datei nur in speziellen Fällen sinnvoll und praktisch, da bei ihr ein Datensatz nur über die Datensatznummer angesprochen werden kann. Ein weiteres Merkmal ist, daß die Datensatzlänge vorher festgelegt werden muß und eine Floppy unbedingt notwendig ist.

Indizierung als Lösung

Durch die Verbindung einer relativen Datei mit einer sequentiellen wird ein Kompromiß geschlossen. Einerseits wird die Suche nach einem Datensatz, beispielsweise nach einer Adresse, jetzt direkt über den Namen möglich. Andererseits muß gleich nach dem Programmstart die sequentielle Datei geladen und am Ende wieder gespeichert werden. Allerdings fällt sie wesentlich kürzer aus. In ihr braucht lediglich ein Feld der Datensätze enthalten zu sein, nämlich das Indexfeld, im Beispiel die Nachnamen, und nicht, wie bei der puren sequentiellen Form, alle Felder.

Gehen wir einmal von folgender Voraussetzung aus: Auf der Diskette sind bereits eine Anzahl von Adressen gespeichert. Im RAM des Computers befindet sich das Programm und in einem eindimensionalen Feld (im Beispiel die Variable IN$) alle bisher eingegebenen Nachnamen. Indem wir diese Variable mittels einer FOR-NEXT-Schleife durchsuchen, können wir auch einen gespeicherten Namen finden. Allerdings kommen wir damit noch nicht an seine Adresse heran, da diese ja in diesem Moment nicht im Computer sondern auf Diskette gespeichert steht. Mit einem kleinen Kniff schaffen wir aber auch diese Hürde. Wir speichern bei der Eingabe einer neuen Adresse einfach die Datensatznummer vor dem Nachnamen in die Variable IN$. Und zwar reservieren wir vor jedem Nachnamen vier Stellen für die zugehörige Datensatznummer. Ab der 5. Stelle beginnt also der jeweilige Nachname. Wollen wir nun einen Namen suchen, so müssen wir mit der MID$-Funktion die Feldvariable IN$ ab dem fünften Buchstaben durchkämmen. Sobald der gesuchte Name gefunden wurde, trennen wir die ersten vier Zeichen ab und erhalten damit die zugehörige Datensatznummer. Mit dieser Nummer können wir jetzt direkt auf die relative Datei auf der Diskette zugreifen und holen uns die komplette Adresse.

... und so wird's gemacht

Anhand eines Beispiels soll das noch einmal gezeigt werden:
Inhalt der ersten Elemente von IN$:
IN$(1) =»1... MEIER«

uentielle Datei

IN$(2) =»2... MUELLER«
IN$(3) =»3... ANDERMANN«
IN$(4) =»*«

Gesuchter Name=Mueller, er steht in der Variablen N$ (Zeile 3060)
3060 INPUT"NACHNAME";N$
3070 N=LEN(N$) Länge des eingegebenen Namens

Die folgenden Programmzeilen suchen den Namen.
3090 FOR I=1 TO DATEIENDE
3100 IF IN$(I) = "*" THEN 3120
3110 IF MID$(IN$(I),5,N)=N$ THEN gefunden
3120 NEXT I

Zur Zeile 3100 kommen wir später noch. Angenommen, wir haben nicht den ganzen Namen eingegeben, sondern lediglich die ersten drei Buchstaben MUE. Dann werden in der Variable IN$ nur die 5.,6. und 7. Zeichen, das entspricht den ersten drei Buchstaben eines jeden Namens, verglichen. Dafür sorgt die Zeile 3070 und in 3110 das »,N« als letzter Parameter der MID$-Funktion. Somit könnte mit dieser Eingabe auch der Name MUECKE gefunden werden. Im Extremfall, wenn nichts eingegeben, also RETURN gedrückt wird, findet diese Funktion jeden Namen. Das heißt, man kann mit der Suchfunktion sich nicht nur einzelne Adressen holen, sondern auch die gesamte Datei, oder zum Beispiel alle Adressen, die mit M anfangen. Damit würde der Menüpunkt »G = Anzeigen gesamte Datei« überflüssig. Ich habe ihn nicht entfernt, da hier unabhängig von der sequentiellen Datei alle Datensätze direkt von der Diskette aus der relativen Datei gelesen werden. Man könnte diese Funktion benutzen, um eine eventuell (zum Beispiel durch Stromausfall) teilweise zerstörte sequentielle Datei zu reorganisieren.

Die Zeile 3100 wird dann verständlich, wenn man sich den Programmteil »Neue Datei anlegen« ab 11000 anschaut. In den Zeilen 11320 bis 11350 wird jedes Element der Variablen IN$ mit einem »*« vorbesetzt. Der Stern kennzeichnet einen leeren Datensatz. Damit kommen wir auch gleich zum Programmteil »Löschen Datensatz« ab 4000. Auch dort wird ein zu löschender Datensatz in IN$ mit einem »*« gekennzeichnet und zusätzlich der entsprechende Datensatz auf der Diskette mit Hex FF (= Dezimal 255 = das Zeichen PI). Auch das wollen wir uns anhand eines kleinen Beispiels näher betrachten:

Wir wollen den Herrn Mueller aus unserer Datei entfernen. Nachdem wir seine Adresse mit der Suchfunktion gefunden haben, geben wir ein »L« für Löschen ein. Danach springt das Programm nach 4000, schreibt in die Variable IN$ das »*« , überschreibt den Datensatz auf der Diskette mit CHR$(255) (=PI), löscht die einzelnen Datensatzfelder (NN$, NV$, und so weiter) und springt zurück nach 3350. Die Variable IN$ sieht jetzt so aus:
IN$(1) =»1... MAIER«
IN$(2) =»*«
IN$(3) =»3... ANDERMANN«»
IN$(4) =»*«

Der zweite Datensatz ist nun mit »*« als gelöscht und leer gekennzeichnet. Das bedeutet auch, daß er wieder mit einer neuen Adresse belegt werden kann. Sehen wir uns dazu den Programmteil »Neueingabe Adresse« ab 1800 an.

Nachdem wir eine neue Adresse eingegeben haben (die einzelnen Programmteile, die in 1830 bis 1850 aufgerufen werden, kennen Sie ja bereits aus den letzten beiden 64'er Ausgaben), muß jetzt ein leerer Platz gefunden werden, auf dem wir die neue Adresse abspeichern können. Wichtig sind jetzt die Zeilen 1880 bis 1910. In diesem Abschnitt wird IN$ so lange durchsucht, bis ein mit »*« gekennzeichnetes Element gefunden wird (1910). In unserem Beispiel ist das bereits das zweite Element (IN$(I)=»*«, bei I=2). Somit wird die neue Adresse als zweiter Datensatz auf Diskette gespeichert. In Zeile 1930 und 1940 wird die Variable IN$ auf den neuesten Stand gebracht. Zuerst wird die Datensatznummer (I) in einen String umgewandelt (1920). Dabei wird immer das (unsichtbare positive) Vorzeichen mit berücksichtigt. Da wir dieses nicht brauchen, wird es abgeschnitten (MID$(I$,2) und die Zahl mit Leerstellen auf insgesamt vier Zeichen Länge erweitert. Zeile 1940 verbindet dann die Datensatznummer mit dem Namen.

Vielleicht ein Wort noch zur MID$-Funktion. Normalerweise gehören zu dieser Funktion drei Parameter. So steht es auch kurz erläutert im Commodore-Handbuch. Der letzte Parameter gibt an, wieviele Zeichen ab einer definierten Stelle des Strings genommen werden sollen. Wird dieser Parameter weggelassen, werden ab der definierten Stelle alle restlichen Zeichen des Strings übernommen.

Probleme, die sich ergeben können

Es könnte sein, daß Ihnen die Dateistruktur nicht gefällt. Zum Beispiel möchten Sie die Länge der Datenfelder ändern. Das dürfen Sie nur machen, wenn die Datei neu eingerichtet werden soll. Bei einer bestehenden geht es nicht! Beim Verkürzen gibt es keine Probleme, auch keine beim Verlängern bis zu 88 Zeichen. Zu verändern sind dann die Zeile 30040 sowie die Längenangaben zwischen den Zeilen 5000 und 7100. Bedenken Sie dabei, daß die Variable BL$ mindestens so viele Leerstellen haben muß, wie das längste Datenfeld.

Schwieriger wird es, wenn Sie infolge einer Vergrößerung der Datensatzlänge über 88 Zeichen hinwegkommen. Das Problem hierbei ist der INPUT#-Befehl. Mit ihm kann man nämlich nur maximal 88 Zeichen aus einer Datei lesen. Es gibt zwei Methoden, diese Hürde zu überwinden. Erstens eine Maschinensprache-Routine, die den INPUT#-Befehl erweitert, so daß auch Datensätze mit bis zu 255 Zeichen gelesen werden können (schauen Sie doch einmal in das Floppy-Buch von Data Becker, dort finden Sie eine entsprechende Routine). Die zweite Möglichkeit ist die Verwendung des GET#-Befehls anstelle des INPUT#-Befehls. Dann ist die

Die Index-Sequenti

Listing Index-sequentielle Datei

```
100 REM  ******************************
110 REM  * ADRESSENDATEI 64'ER/9       *
120 REM  * INDEX-SEQUENTIELL           *
130 REM  ******************************
140 :
150 :
160 GOSUB 30000:REM  INIT
170 CLOSE 1:OPEN 1,8,2,FR$+",L,"+CHR$(DL
)
180 CLOSE 3:OPEN 3,8,3,FI$+",S,R"
190 GOSUB 10000:REM :DISKFEHLER
200 IF A1<>0 THEN RUN
210 INPUT# 3,IN$:MX$=LEFT$(IN$,15)
220 MX=VAL(MX$)
230 :
240 IN$(0)=IN$
250 PRINT "■"
260 PRINT :PRINT :PRINT
270 PRINT "            INFORMATION"
280 PRINT :PRINT
290 PRINT "  BISHERIGE DATEIGROESSE: ";M
X
300 PRINT :PRINT
310 PRINT "        BITTE WARTEN"
320 I=0
330 I=I+1
340 :INPUT# 3,IN$(I):PRINT ""I;MX;IN$(I)
350 IF ST<>64 THEN 330
360 PRINT :PRINT
370 PRINT "  DRUECKEN SIE EINE TASTE"
380 POKE 198,0:WAIT 198,1
390 REM  ----------------------------
1000 REM  - MENUE                     -
1010 REM  ----------------------------
1020 :
1030 PRINT "■"
1040 PRINT :PRINT
1050 PRINT "       ADRESSENDATEI"
1060 PRINT "  RELATIV UND SEQUENTIELL"
1070 PRINT :PRINT
1080 PRINT "■ X = PROGRAMMENDE"
1090 PRINT
1100 PRINT "   G = ANZEIGEN GESAMTE DATE
I
1110 PRINT
1120 PRINT "   S = SUCHEN"
1130 PRINT
1140 PRINT "   N = NEUE ADRESSEN EINGEBE
N"
1150 PRINT
1160 PRINT "   ! = NEUE DATEI ANLEGEN"
1170 PRINT :PRINT :PRINT :PRINT
1180 PRINT "WAEHLEN SIE ";
1190 POKE 198,0
1200 GET R$:IF R$="" THEN 1200
1210 IF R$="X" THEN CLOSE 1:GOSUB 15000:
CLOSE 15:END
1220 IF R$="G" THEN GOSUB 3500:REM  ANZ.
1230 IF R$="S" THEN GOSUB 2570:REM  SUCH
1240 IF R$="N" THEN GOSUB 1800:REM  NEUE
ING.
1250 IF R$="!" THEN GOSUB 11000:REM  NEU
DATEI
1260 GOTO 390
1800 REM ----------------------------
1810 REM  SCHREIBEN /EINGABE SEQ/REL-
1820 REM ----------------------------
1825 NN$=".":NV$=".":OT$=".":TE$=".":PL$
=".":SR$="."
1830 GOSUB 2000:REM  AUSGABE 1 DATENSATZ
1840 GOSUB 6000:REM  EINGABE
1850 GOSUB 7000:REM  VERKETTEN
1860 REM  BESTIMMUNG SATZNUMMER
1870 LZ$="     ":LZ=4
1880 I=0
1890 I=I+1
1900 :
1910 IF IN$(I)<>"*" THEN 1890
1920 I$=STR$(I)
1930 I$=MID$(I$,2)+LEFT$(LZ$,LZ-LEN(I$)+
1)
1940 IN$(I)=I$+NN$
1950 RN$=STR$(I)
1960 GOSUB 14000:REM  SATZNR.AUFTEILEN
1970 GOSUB 8000:REM  SPEICHERN
1980 IF I>=MX THEN PRINT "■ DATEI VOLL":
GOTO 11500
1990 RETURN
2000 REM  ----------------------------
-
2010 REM  - AUSGABE  1 DATENSATZ
2020 REM  ----------------------------
2030 :
2040 PRINT "■■   ANZEIGE DATENSATZ";RN$
1
2050 PRINT :PRINT :PRINT :PRINT
2060 PRINT " NACHNAME    "NN$
2070 PRINT " VORNAME     "NV$
2080 PRINT " STRASSE     "SR$
2090 PRINT " PLZ         "PL$
2100 PRINT " ORT         "OT$
2110 PRINT " TELEFON     "TE$
2120 PRINT :PRINT :PRINT
2130 RETURN
2140 :
2500 REM ----------------------------
2510 REM      DRUCKEN
2520 REM ----------------------------
2530 :
2540 PRINT "■■ ■ NOCH NICHT DEFINIERT
  ":PRINT
2550 FOR J=1 TO 500:NEXT
2560 RETURN
2570 REM ----------------------------
3000 REM  SUCHEN SEQ/REL
3010 REM ----------------------------
3020 N$=""
3030 PRINT "■":PRINT :PRINT
3040 PRINT "      SUCHEN"
3050 PRINT :PRINT
3060 INPUT " NACHNAME";N$
3070 N=LEN(N$)
3080 S1=1
3090 FOR I=S1 TO MX
3100 :IF IN$(I)="*" THEN 3120
3110 :IF MID$(IN$(I),5,N)=N$ THEN 3190
3120 NEXT I
3130 IF S1>1 THEN PRINT "■■ SUCHE BEENDE
T■":GOTO 3150
3140 PRINT :PRINT N$" NICHT GEFUNDEN"
3150 PRINT :PRINT "DRUECKE TASTE"
3160 GET R$:IF R$="" THEN 3160
3170 RETURN
3180 RN$=LEFT$(IN$(I),4)
3190 GOSUB 14000:REM  SATZNR.AUFTEILEN
3200 GOSUB 9000:REM  LESEN
3210 GOSUB 5000:REM  AUFTEILEN
3220 GOSUB 2000:REM  ANZEIGEN
3230 PRINT :PRINT
3240 W$="                    WAEHLE  "
3250 PRINT "(W)EITERSUCHEN  (Z)URUECK"
3260 PRINT "(A)ENDERN       (L)OESCHEN"
3270 PRINT "(D)RUCKEN                "
3280 PRINT W$
3290 GET R$:IF R$="" THEN 3290
3300 W$="                    WARTEN  "
3310 PRINT "■ "W$"■"
3320 IF R$="Z" THEN 3170
3330 IF R$="W" THEN S1=I+1:GOTO 3090
3340 IF R$="A" THEN GOSUB 11430:GOTO 324
0
3350 IF R$="L" THEN GOSUB 3670:GOTO 3220
3360 IF R$="D" THEN GOSUB 2500
3370 PRINT "■■■■■";:GOTO 3240
3500 REM ----------------------------
3510 REM - LESEN GESAMTE DATEI
3520 REM ----------------------------
3530 RN=0
3540 RN=RN+1
3550 :HB=INT(RN/256)
3560 :LB=RN-HB*256
3570 :GOSUB 9000:REM  LESEN
3580 :IF ER=50 THEN PRINT "■   DATEI END
E ■■■":GOTO 3620
3590 :IF F=2 THEN PRINT "■■ NICHT BELEGT
:  DATENSATZ-NR. ";RN;"■■  ":GOTO 3540
3600 :GOSUB 5000:REM  AUFTEILEN
3610 :GOSUB 2000:REM  ANZEIGEN
3620 :PRINT "DRUECKE TASTE"
3630 :GET R$:IF R$="" THEN 3630
3640 :R$=""
3650 IF ER<>50 THEN 3540
3660 RETURN
4000 REM ----------------------------
4010 REM  LOESCHEN  DATENSATZ
4020 REM ----------------------------
4030 :
4040 IN$(I)="*"
4050 RC$="■"
4060 GOSUB 8000:REM  SPEICHERN
4070 NN$=".":NV$=".":OT$=".":TE$=".":PL$
=".":SR$="."
4080 :
4100 RETURN
5000 REM ----------------------------
5010 REM  AUFTEILEN DATENSATZ IN FELDER
5020 REM ----------------------------
5030 :
5050 NN$=MID$(RC$,1,15)
5060 NV$=MID$(RC$,16,15)
5070 SR$=MID$(RC$,31,20)
5080 PL$=MID$(RC$,51,4)
5090 OT$=MID$(RC$,55,15)
5100 TE$=MID$(RC$,70,12)
5110 RETURN
6000 REM ----------------------------
6010 REM - EINGABE NEUE DATEN         -
6020 REM ----------------------------
6030 :
6050 PRINT "■"
6060 PRINT "    EINGABE              "
6070 PRINT :PRINT :PRINT :PRINT
6080 INPUT " NACHNAME ";NN$:NN$=LEFT$(NN
$,15)
6090 INPUT " VORNAME  ";NV$:NV$=LEFT$(NV
$,15)
6100 INPUT " STRASSE  ";SR$:SR$=LEFT$(SR
$,20)
6110 INPUT " PLZ      ";PL$:PL$=LEFT$(PL
$,4)
6120 INPUT " ORT      ";OT$:OT$=LEFT$(OT
$,15)
6130 INPUT " TELEFON  ";TE$:TE$=LEFT$(TE
$,12)
6140 PRINT :PRINT
6150 PRINT "  ADRESSE OK (J/N) ?"
6160 GET R$:IF R$="" THEN 6160
6170 IF R$="N" THEN 6050
6175 IF R$<>"J" THEN 6160
6180 RETURN
6190 :
7000 REM ----------------------------
7010 REM      VERKETTEN DER FELDER   -
7020 REM ----------------------------
7030 :
7040 BL$="                           "
7050 RC$=NN$+LEFT$(BL$,15-LEN(NN$))
7060 RC$=RC$+NV$+LEFT$(BL$,15-LEN(NV$))
7070 RC$=RC$+SR$+LEFT$(BL$,20-LEN(SR$))
7080 RC$=RC$+PL$+LEFT$(BL$,4-LEN(PL$))
7090 RC$=RC$+OT$+LEFT$(BL$,15-LEN(OT$))
7100 RC$=RC$+TE$+LEFT$(BL$,12-LEN(TE$))
7110 RETURN
7120 :
8000 REM  ----------------------------
8010 REM  - SPEICHERN DATEN AUF DISK -
8020 REM  ----------------------------
8030 :
8080 PRINT# 15,"P"+CHR$(2)+CHR$(LB)+CHR$
(HB)+CHR$(1)
8100 PRINT# 1,RC$
8110 FS=1:REM  FLAG FUER SPEICHERN
8170 RETURN
8180 :
9000 REM  ----------------------------
9010 REM  - LESEN DATENSATZ VON DISK -
9020 REM  ----------------------------
9030 F=0
9040 PRINT# 15,"P"+CHR$(2)+CHR$(LB)+CHR$
(HB)+CHR$(1)
9050 INPUT# 15,ER
9060 IF ER=50 THEN 9110
9070 INPUT# 1,RC$
9080 IF RC$<>"■" THEN F=1:GOTO 9110
9090 F=2:REM  FREIER DATENSATZ
9100 :
9110 RETURN
10000 REM  ----------------------------
10010 REM  - DISKETTENFEHLER           -
10020 REM  ----------------------------
10030 PRINT "■"
10040 INPUT# 15,A1,A2$,A3,A4
10050 IF A1=0 THEN 10180
10060 IF A1=62 THEN GOSUB 10200:GOTO 101
80
10070 PRINT
10080 PRINT A1,A2$;A3;A4
10090 PRINT :PRINT
10100 PRINT "        DISKETTENFEHLER"
10110 PRINT :PRINT
10120 PRINT "     BEHEBEN SIE DEN FEHLER
 "
10130 PRINT "        UND DRUECKEN SIE"
10140 PRINT
10150 PRINT "            >> F <<"
10160 GET R$:IF R$="" THEN 10160
10170 PRINT "■"
10180 RETURN
10190 :
10200 PRINT "■"
10210 PRINT :PRINT :PRINT :PRINT
10220 PRINT " DIE DATEI        "FR$
10230 PRINT
10240 PRINT " ODER             "FI$
10250 PRINT
10260 PRINT " EXISTIEREN NICHT!"
10270 PRINT :PRINT
10280 PRINT "  L = DATENDISK EINLEGEN"
10290 PRINT
10300 PRINT "  N = DATEI NEU ANLEGEN"
10310 GET R$:IF R$="" THEN 10310
10320 IF R$="L" THEN RETURN
10330 IF R$="N" THEN GOTO 11000
10340 GOTO 10310
11000 REM ----------------------------
11010 REM   - NEUE DATEI ANLEGEN
11020 REM  ----------------------------
11030 :
11040 PRINT "■":PRINT
```

tielle Datei

```
11050 IF A1=0 THEN 11070
11060 PRINT " "
11070 PRINT "  ACHTUNG, DIE GESAMTE DISK
ETTE WIRD  "
11080 PRINT "  GELOESCHT ! "
11090 PRINT :PRINT
11100 PRINT " N = NEUE DATEI  X = ENDE"
11110 GET R$:IF R$="" THEN 11110
11120 IF R$="X" THEN  CLOSE 1:GOSUB 1500
0:CLOSE 15:END
11130 IF R$<>"N" THEN 11110
11140 PRINT :PRINT "       BITTE WARTEN"
11150 PRINT# 15,"N:RELATIVE DATEI"
11160 CLR :GOSUB 30000:REM  INIT
11170 CLOSE 1:OPEN 1,8,2,FR$+",L,"+CHR$(
DL)
11180 PRINT "WIEVIELE DATENSAETZE SOLL D
IE DATEI "
11190 PRINT "VERWALTEN? ";
11200 INPUT RN$:RN=ABS(INT(VAL(RN$)))
11210 IF RN<=MX THEN 11180
11220 HB=INT(RN/256)
11230 LB=RN-HB*256
11240 PRINT "BITTE WARTEN"
11250 PRINT# 15,"P"+CHR$(2)+CHR$(LB)+CHR
$(HB)+CHR$(1)
11260 PRINT# 1,CHR$(255)
11270 MX=RN
11280 MX$=STR$(RN)
11290 CLOSE 1
11300 :
11310 PRINT "
          "
11320 FOR I=AM+1 TO MX
11330 :IN$(I)="*"
11340 PRINT "          "MX;I,IN$(I)
11350 NEXT I
11360 FI$="@:"+FR$+"INDEX"
11370 CLOSE 3:OPEN 3,8,3,FI$+",S,W"
11380 IN$(0)=MX$
11390 FOR I=0 TO MX:PRINT# 3,IN$(I):NEXT

11400 CLOSE 3
11420 RUN
11430 :
11500 REM ------------------------------
11510 REM  DATEI ERWEITERN
11520 REM ------------------------------
11530 AM=MX
11540 MX=MX+50
11550 PRINT :PRINT "  ERWEITERN DER DATE
I"
11560 PRINT :PRINT "  BISHERIGE GROESSE=
 "AM
11570 PRINT :PRINT "  NEUE GRØESSE     =
 "MX
11580 RN=MX
11590 GOTO 11220
12000 REM ------------------------------
12010 REM  AENDERN  DATENSATZ
12020 REM ------------------------------
12030 GOSUB 2000:REM  ANZEIGE DATENSATZ
12040 GOSUB 6000:REM  EINGABE NEUE DATEN
12050 LZ$="     ":LZ=4
12060 I$=STR$(I)
12070 I$=MID$(I$,2)+LEFT$(LZ$,LZ-LEN(I$)
+1)
12080 IN$(I)=I$+NN$
12090 GOSUB 7000:REM  VERKETTEN
12100 GOSUB 8000:REM  SPEICHERN
12110 RETURN
12120 :
14000 REM  ------------------------------
14010 REM  - AUFTEILEN DATENSATZNUMMER
14020 REM  ------------------------------
14030 :
14080 RN=ABS(INT(VAL(RN$)))
14100 HB=INT(RN/256)
14110 LB=RN-HB*256
14130 RETURN
14140 RETURN
14150 :
15000 REM ------------------------
15010 REM  SPEICHERN SEQ DATEI  -
15020 REM ------------------------
15030 IF FS<>1 THEN 15120
15040 CLOSE 3:OPEN 3,8,3,"@:"+FI$+",S,W"
15050 GOSUB 10000:REM  FEHLERKANAL
15060 IF A1<>0 THEN 15040
15070 FOR I=0 TO MX
15080 :PRINT# 3,IN$(I)
15090 :PRINT "          "I;MX;IN$(I)
15100 NEXT I
15110 CLOSE 3
15120 RETURN
15130 :
15500 REM ------------------------
15510 REM  LESEN     SEQ DATEI  -
15520 REM ------------------------
15530 CLOSE 3:OPEN 3,8,3,"@:"+FI$+",S,R"
15540 GOSUB 10000:REM  FEHLERKANAL
```

```
15550 IF A1<>0 THEN 15530
15560 FOR I=1 TO MX
15570 :PRINT# 3,IN$(I)
15580 :PRINT " "I;MX;IN$(I)
15590 NEXT I
15600 CLOSE 3
15610 RETURN
15620 :
30000 REM ------------------------------
30010 REM  INITIALISIERUNG
30020 REM ------------------------------
30030 :
30040 DL=82:REM  DATENSATZLAENGE
30050 RN=1
30060 CLOSE 15:OPEN 15,8,15
30070 BL$="                    "
30080 BL=LEN(BL$)
30090 PRINT "":POKE 53281,0:POKE 53280,
0
30100 FR$="ADR.REL"
30110 FI$=FR$+"INDEX"
30120 DIM IN$(2000)
30130 RETURN
30140 STOP

READY.
```

Listing Index-sequentielle Datei (Schluß)

Liste der verwendeten Variablen

DL	=	Datensatzlänge
RN	=	Record-Nummer
RN$	=	Record-Nummer
BL$	=	Leerstellen (Blanks)
BL	=	Anzahl Leerstellen
FR$	=	Name der relativen Datei
FI$	=	Name der Index-Datei
A1 bis A4	=	Fehlermeldung der Floppy
IN$(i)	=	Inhalt der Index-Datei
IN$(0)	=	Größe der Datei
MX/MX$	=	Größe der Datei
AM	=	Größe der Datei vor Erweiterung
HB/LB	=	High-Low-Byte
ER	=	Fehlermeldung der Floppy bezogen auf relative Datei
RC$	=	ein gesamter Datensatz
NN$	=	Nachname
NV$	=	Vorname
SR$	=	Straße
OT$	=	Ort
PL$	=	Postleitzahl
TE$	=	Telefon
ST	=	Statusvariable (prüft auf Dateiende bei der Sequent. Datei)
S1	=	stellt fest, ob die gesamte Datei durchsucht wurde
F	=	stellt fest, ob ein Datensatz belegt ist
FS	=	wenn 1, dann wird vor Beenden des Programms die sequentielle Datei erneut abgespeichert.

Zeile 9070 INPUT#1,RC$ durch folgende Zeilen zu ersetzen:

```
9068 RC$=" "
9070 GET#1,W$
9072 RC$=RC$+W$
9074 IF W$=CHR$(255) THEN 9090
9076 IF W$=CHR$(13) THEN 9070
```

Den GET#-Befehl kann man natürlich auch dann einsetzen, wenn die Satzlänge kleiner als 88 Zeichen ist. Dann werden die Zeichen jedoch um einiges langsamer eingelesen als mit dem INPUT#-Befehl. Aber es ist mit dem GET#-Befehl möglich, bis zu 255 Zeichen in eine Variable einzulesen.

Ein Problem ganz anderer Art kann auftauchen, wenn Sie eine Floppy besitzen, die vor Dezember 1983 gekauft beziehungsweise hergestellt wurde und gleichzeitig mit einem VC 1526 drucken. Es kann möglich sein, daß bei Benutzung einer relativen Datei Fehler auftauchen, sobald dieser Drucker eingeschaltet ist. Probieren Sie das vorher aus! Abhilfe schafft meines Wissens nur ein neues DOS für die alte VC 1541.

Anregungen für Programmierer

Dieses Programm ist — bis auf die Druckerroutine, die sich jeder für seinen eigenen Drucker selber schreiben sollte — komplett und funktionsfähig. Sicherlich lassen sich noch eine ganze Reihe von Schönheitsreparaturen machen. Auch eine Erweiterung des Programms ist durchaus sinnvoll. Zum Beispiel könnte man ein Unterprogramm schreiben, das, ähnlich einem professionellen Dateiprogramm, es ermöglicht, sich die Datenfelder selbst zu definieren. Auch die Suche nach mehr als einem Feld ist durchaus sinnvoll. Wenn erwünscht, kann auch eine Sortierroutine eingefügt werden. Der modulare Aufbau erlaubt dies ohne Schwierigkeiten. So kann sich jeder, der etwas Programmiererfahrung hat, eine Dateiverwaltung aufbauen, die sogar professionellen Programmen etwas voraus hat: Sie ist auf die persönlichen Bedürfnisse abgestimmt und läßt sich auch jederzeit ändern. Wer das Programm kompiliert, kann auch bei sehr großen Dateien noch mit guten Suchgeschwindigkeiten rechnen. Dieses Beispielprogramm enthält alle Voraussetzungen für eigene Erweiterungen. Ich selbst habe obengenannte Funktionen für meine eigene Dateiverwaltung bereits realisiert. (gk)