Kleinstrechner

Tendenzen
und Theorien

Datenverwaltung

Informationen
und Ideen

Kleincomputer
in der Psychologie

Programme
und Projekte

Pixelgrafik

Spaß
und Spiel

13

Kleinstrechner-TIPS

Heft 13

Mit 29 Bildern

Herausgegeben von
Prof. Dr.-Ing. Hans Kreul
Prof. Dr. sc. techn. Thomas Horn
Doz. Dr.-Ing. Wilhelm Leupold

VEB Fachbuchverlag Leipzig

Inhalt

ISBN 3-343-00582-7

Zum Inhalt

Das vorliegende Heft enthält ein vielfältiges Angebot von Beiträgen zur Verarbeitung von Algorithmen mittels Computers, zur Computertechnik, zur Anwendung von Programmiersprachen und mit Beispielprogrammen für interessante Einsatzfälle von Computern.

JUNEK zeigt an prägnanten Beispielen die Vorteile und Möglichkeiten von rekursiven Methoden in der Programmierung unter besonderer Berücksichtigung der Anwendung von BASIC.

ANDRASCH gibt ein Rahmenprogramm in BASIC an, das die Verwaltung kleiner Datenmengen durch den Amateur ermöglicht.

RENO FILLA, Schüler einer 10. Klasse, stellt seine Überlegungen vor, wie die Graphik des KC 85/2 verbessert werden kann.

Von OELSCHLÄGEL und SCHULZ wird das iterative Lösen von zwei zahlentheoretischen Problemen mit geringem Speicherplatzbedarf vorgeführt. Das angegebene BASIC-Programm kann an Schulen und in Arbeitsgemeinschaften genutzt werden.

FINCK und BERNDT zeigen in ihrem Beitrag, daß Kleincomputer auch in der Psychodiagnostik oder für psychologische Experimente sinnvoll eingesetzt werden können. (Das Programm kann von Interessenten bei den Autoren angefordert werden.)

Mittels des von BIENER vorgestellten BASIC-Programmes können Commodore-Dateien in den KC 87 eingelesen werden.

Von HAMM wird am Beispiel des LDIR-Befehls ein Weg zur Einstimmung auf das Programmieren in Maschinensprache gezeigt, der z.B. durch Anwenden auf Bereiche des Bildwiederholspeichers sehr effektiv für das Löschen des Bildschirms einsetzbar ist.

PÖSCHEL beschreibt in einem BASIC-Programm für den KC 85/1, das z.B. in Computerarbeitsgemeinschaften nachvollzogen werden kann, die Umwandlung von römischen in Dezimalzahlen und umgekehrt.

SCHÖNFELDER erläutert die Voraussetzungen und zeigt Möglichkeiten einer graphischen Ausgabe.

HAMM gibt Möglichkeiten zur Realisierung sowohl einer Grobgraphik als auch von Graphik mit besserer Nutzung des vorhandenen Rasters beim KC 85/1 an.

Wie FORTH zum Entwurf schneller Computerspiele eingesetzt werden kann, wird von KUCKWA an Hand eines Beispiels gezeigt.

Wir hoffen, mit dem breiten Spektrum dieses Heftes unseren Lesern, für deren Zuschriften wir uns erneut bedanken, weitere Anregungen für die Beschäftigung mit Computern gegeben zu haben.

Wilhelm Leupold

Rekursive Methoden in der Programmierung

Rekursion [recurrere (lat). zurückführen] ist die Rückführung eines Problems $P(n)$ der Verschachtelungstiefe n auf ein gleichartiges Problem der Tiefe $n-1$. Sind ein solcher Rückführungsalgorithmus und ein Anfangsschritt ($n = 0$) gegeben, so kann das Problem nach den Prinzipien der mathematischen Rekursion als gelöst betrachtet werden, ohne daß eine direkte, durch eine Folge von Anweisungen gegebene Beschreibung des Lösungsweges vorliegt.
Rekursion stellt damit ein qualitativ anderes algorithmisches Herangehen an die Probleme dar, als es die streng sequentielle Programmierung ist. Rekursive Problemlösetechniken sind folglich besonders auf solche Probleme anwendbar, die durch eine hochgradige Verschachtelung von relativ einfachen und gleichartigen Teilproblemen gekennzeichnet sind. Derartige Strukturen treten insbesondere bei Aufgabenstellungen der künstlichen Intelligenz wiederholt auf. Dazu gehören z.B. Labyrinth-Probleme, Theorembeweiser, Formelmanipulatoren und Sprachübersetzer. Die Hauptmethode für die Lösung dieser Aufgaben sind sog. *backtracking-Algorithmen*, deren Kernstück die Rekursion ist. Für die professionelle Bearbeitung solcher Probleme sind die Sprachen der LISP-Familie und PROLOG besonders geeignet. Anfangsschwierigkeiten beim Erlernen dieser Sprachen sind häufig auf eine ungenügende Schulung rekursiver Denkweisen zurückzuführen [2]. In der Informatikausbildung aller Stufen sollte diesem Problem daher mehr Aufmerksamkeit geschenkt werden. Diesem Ziel dient auch der vorliegende Beitrag. Es ist die Absicht des Autors, eine Palette typischer, aber einfacher Probleme darzustellen, die eine Schlüsselrolle für die aktive Nutzung rekursiver Techniken spielen. Nach Meinung des Autors eignen sich hierfür vor allem geometrische Aufgaben.

1. Rekursive Programmierung in BASIC

Programmiersprachen wie PASCAL oder LOGO unterstützen die rekursive Programmierung in hervorragender Weise, und ihnen sollte bei der Bearbeitung größerer Probleme der Vorrang gegeben werden. Die Mehrzahl unserer Computerfreunde verfügt aber bislang nur über BASIC. Daher werden in diesem Artikel vor allem solche Aufgaben behandelt, die eine übersichtliche Bearbeitung in dieser Sprache noch zulassen.
Als Beispiel für rekursive Verfahren wird häufig die *Ermittlung des größten gemeinsamen Teilers* (GGT) zweier natürlicher Zahlen A und B angeführt. Da die Zahlen-

paare (A, B) und $(A-B, B)$ offensichtlich den gleichen größten gemeinsamen Teiler haben, ist der folgende, nach EUKLID (4. Jh. v.u.Z.) benannte Algorithmus zur Berechnung des größten gemeinsamen Teilers verwendbar.

GGT (A, B)
  Wenn $A = B$, dann drucke A.
  Wenn $A > B$, dann tue GGT $(A-B, B)$.
  Wenn $A < B$, dann tue GGT (B, A).
Ende.

Der Selbstaufruf der Prozedur GGT innerhalb des Programms ist das typische Merkmal rekursiver Programmierung. Als Maß für die Rekursionstiefe dient die Differenz $|A-B|$. Bei Erreichen des untersten Niveaus, gegeben durch $A = B$, wird der Wert direkt ausgegeben. Der Rücklauf aus den Selbstaufrufen heraus ist ohne Aktionen. Man spricht daher von einem leeren Rücklauf. Damit gehört dieses Programm zu den einfachsten Vertretern rekursiver Technik.
Versuchen wir nun die Umsetzung in ein BASIC-Programm. Grundlage hierfür ist die GOSUB/RETURN-Anweisung. Damit kann der Algorithmus gemäß Bild 1 notiert werden. Wir wollen die Verwendung des GOSUB-Befehls etwas

```
 10 REM ********************
 20 REM * GGT rekursiv     *
 30 REM ********************
 40
 50 INPUT "a,b=";a,b
 60 GO SUB 100
 70 PRINT "GGT=";a: STOP
 80
100 REM Rekursion
110 IF a<b THEN  LET h=a: LET a=b: LET b=h
120 IF a>b THEN  LET a=a-b
130 IF a<>b THEN  GO SUB 100
140 RETURN
```

Bild 1. Größter gemeinsamer Teiler

näher analysieren. Gegenüber der gut bekannten Verwendung dieses Befehls, z.B. im Zusammenhang mit Menütechniken, erhält man durch die Selbstaufrufe eine neue Qualität. Die Verarbeitung eines GOSUB-Befehls veranlaßt den BASIC-Interpreter, die der entsprechenden Zeile folgende Zeilennummer auf einen Stapelspeicher, den sogenannten RETURN-Stack, zu legen. Jeder weitere GOSUB-Befehl führt zu einer weiteren Ablage. Umgekehrt führt jeder RETURN-Befehl zur Entnahme einer Zeilennummer vom Stapelspeicher und zum Ansprung dieser Zeile. Sehr gut verfolgen läßt sich die Stapelarbeit am Programm »STAPEL« in Bild 2. Von gleicher Art ist das Programm »TRICHTER« in Bild 3, das vielleicht noch überraschender arbeitet. Dem Leser sei empfohlen, mit diesen Programmen weiter zu experimentieren. Insbesondere sollte die Wirkung von Zeilenvertauschungen untersucht und erklärt werden.

```
 10 REM **********
 20 REM * STAPEL *
 30 REM **********
 40
 50 LET K=5: GO SUB 100
 60 PRINT "ENDE": STOP
 70
100 REM REKURSION
110 IF K=0 THEN  RETURN
120  LET K=K-1: PRINT "ABSTIEG=";K
130  GO SUB 100
140 PRINT "AUFSTIEG=";K
150 LET K=K+1
160 RETURN
```

Bild 2. Listing STAPEL

```
10 REM ************
20 REM * TRICHTER *
30 REM ************
40
50 LET A$="TRICHTER"
60 LET K=LEN (A$): GO SUB 100: STOP
70
100 REM REKU
110 IF K=0 THEN  RETURN
120 PRINT LEFT$(A$,K)
130 LET K=K-1
140 GO SUB 100
150 LET K=K+1
160 PRINT LEFT$(A$,K)
170 RETURN
```

```
TRICHTER
TRICHTE
TRICHT
TRICH
TRIC
TRI
TR
T
T
TR
TRI
TRIC
TRICH
TRICHT
TRICHTE
TRICHTER
```

Bild 3. Trichter

Der möglichen Rekursionstiefe ist durch den maximal verfügbaren RETURN-Stack eine Grenze gesetzt, die man durch folgendes Programm ermitteln kann:

```
10 LET I = 0
20 LET I = I + 1
30 GOSUB 20
```

Dieses Programm bricht mit einer Fehlermeldung »Speicherüberlauf« ab, und man kann die erreichte Tiefe durch PRINT I feststellen. Man mache sich an diesem Programm nochmals den Unterschied zum GOTO-Sprung klar! Eine Zeile 30 GOTO 20 würde erst bei numerischen Überlauf (I > 1E 35) abbrechen.
Einige einschränkende Bemerkungen zur Verwendung von BASIC sind nun angebracht. Über die Bereitstellung der GOSUB/RETURN-Anweisung hinaus erfolgt in dieser Sprache keine weitere Unterstützung rekursiver Programmierung. Da BASIC nur globale Variablen kennt, entstehen insbesondere bei Rekursionen mit Wertrückgabe zusätzliche Schwierigkeiten. Man ist zur Einrichtung eines »Variablenstacks« gezwungen. In [4] wird versucht, allgemeine Konstruktionsvorschriften hierfür zu geben. Wir wollen uns diesem Problem an ausgewählten Beispielen später zuwenden und uns zunächst auf Probleme ohne Wertrückgabe konzentrieren. Hier findet sich bereits ein breites Anwendungsfeld.

2. Rekursion mit Verzweigung

Die eigentliche Domäne rekursiver Programmierung ist die Bearbeitung verzweigter Probleme. Ein lehrreiches Übungsfeld ist dabei die *Erzeugung selbstähnlicher Figuren.* Eine Schlüsselrolle spielt die Konstruktion eines binären Baumes (Bild 4c). Bezeichnet N die Rekursionsstufe (in Bild 4c ist $N = 6$) und ist S die Stammhöhe, so läßt sich ein Programm für einen Baum der Stufe N unter Verwendung von zwei Selbstaufrufen folgendermaßen beschreiben:
BAUM (N, S)

Wenn $N > 0$ DANN
  BEGINN
    GEHE S SCHRITTE VORWÄRTS.
    DREHE + 45°, TUE BAUM ($N-1$, $S/2$).
    DREHE − 90°, TUE BAUM ($N-1$, $S/2$).
    GEHE S SCHRITTE RÜCKWÄRTS.
  ENDE

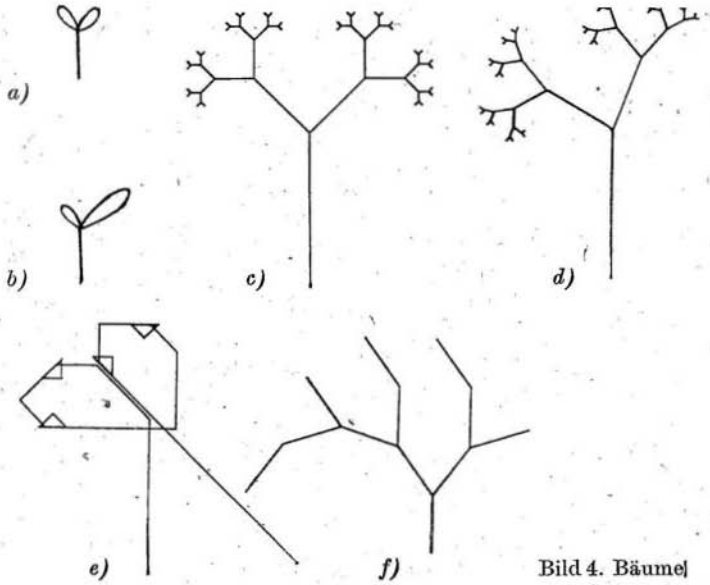

Bild 4. Bäume

Einige Bemerkungen zum Verständnis dieses Programmes sind angebracht. Die bei nichtrekursiven Programmen erprobte Methode eines »Trockentestes« versagt bei rekursiven Programmen total und hilft weder bei der Fehlersuche noch bei der Erklärung des Programmablaufes. Die mit der Potenz 2^n aufsteigende Zahl der Verzweigungen führt schnell in die Situation des Tausendfüßlers, der, als er über die Reihenfolge der zu bewegenden Beine nachdachte, unweigerlich in das Stolpern geriet. Der Schlüssel für das Verständnis liegt vielmehr in der bewußten Anwendung des zu Beginn des Beitrages beschriebenen Prinzips der mathematischen Rekursion. Demzufolge sind die beiden Selbstaufrufe BAUM ($N-1$, $S/2$) als Ganzheit zu verstehen, und die Idee des Programmes besteht darin, einen Zweig mit zwei Knospen gemäß Bild 4a zu konstruieren. Die Stufe 0 besteht in der leeren Aktion. Die Umsetzung in ein BASIC-Programm ist in Bild 5 gegeben. Überraschend, wie kurz sich ein solches Programm in einer vollrekursiven Sprache wie LOGO, einem LISP-Abkömmling, darstellen läßt.

Für die korrekte Arbeitsweise rekursiver Programme ist es entscheidend, daß am Ausgang jedes Rekursionsschrittes der gleiche Zustand wiederhergestellt wird, wie er am Eingang des Programmaufrufes vorlag. Im BAUM-Programm bedeutet das die Erhaltung der Richtung und der Zweiglänge S. So führt beispielsweise die Streichung der Programmzeile 16Ø zum Bild 4e. Fehlerhaft arbeitende rekursive Programme sind daher vor allem auf Zustandserhaltung zu überprüfen. Das Programm BAUM ist wegen seiner Einfachheit besonders gut zum Experimentieren geeignet. Bild 4d ergibt sich z.B. durch Antragen unterschiedlicher Winkel nach links und rechts.

Bevor wir uns weiteren geometrischen Figuren zuwenden, betrachten wir eine alphanumerische Variante des Baummodells. Die Aufgabe besteht in der Erzeugung aller Folgen der Länge n aus den Ziffern 0 und 1. Denkt man sich diese

Ziffern durch »links« bzw. »rechts« ersetzt, so sieht man, daß jeder Folge der Ziffern 0 und 1 eindeutig ein Weg im Baum von der Wurzel zu einer Spitze entspricht und umgekehrt. Programme zur Erzeugung des Baumes bzw. zur Erzeugung aller Folgen müssen also sehr ähnlich sein. Anstelle des Bildschirms, der gleichzeitig in gewissem Sinn als Stack dient, muß jetzt jedoch eine Stringvariable A$ zum Sammeln der Ziffernfolge eingerichtet werden. Wir können auf ein Listing des Programmes verzichten, da die Idee im nächsten Programm wiederkehrt.

Bekannt ist folgende Aufgabe. Gegeben sei eine *Treppe der Höhe H*. Beim Aufsteigen dürfen nach freier Wahl kurze Schritte (= 1 Stufe) oder lange Schritte (= 2 Stufen) gemacht werden. Gesucht sind alle möglichen Schrittfolgen, die oberste Stufe zu erreichen. Die Aufstellung des Programmes erfolgt wiederum rekursiv, eine einzelne Rekursionsstufe besteht darin, einen Zweig mit »unsymmetrischen« Knospen gemäß Bild 4b zu programmieren. Eine Ausführung in BASIC ergibt das Programm gemäß Bild 6. In Bild 4f ist der zugehörige Baum für $N = 4$ dargestellt. Häufig ist nur nach der Anzahl $A(N)$ der möglichen Wege gefragt. Diese Anzahl kann ebenfalls leicht rekursiv beschrieben werden: Indem man die beiden Möglichkeiten für den ersten Schritt gesondert betrachtet, ergibt sich die Anzahlformel

$$A(N) = A(N-1) + A(N-2), \quad A(1) = 1, \quad A(0) = 0.$$

Die Berechnung dieser Funktion in PASCAL ist einfach, in BASIC erschwert das Fehlen des Variablenstacks die Arbeit. Das obige Beispiel ist übrigens ein Spezialfall einer FIBONACCI- oder LUCAS-Folge, und der mathematisch interessierte Leser kann unter diesem Stichwort, z.B. in [3], eine Reihe weiteren Materials finden. Hierzu gehört auch die bekannte Aufgabe zur Vermehrung einer Kaninchenfamilie.

```
 10 REM ******************
 20 REM * BINAERER BAUM  *
 30 REM ******************
 40
 50 LET S=50: REM STRECKENLAENDE
 60 LET PHI=90: REM RICHTUNG
 70
 80 INPUT "STUFE =";N
 90 LET K=0: LET X=100: LET Y=20: REM ANFANGSPUNKT
 95 GO SUB 100: STOP
 97
100 REM Rekursion
110 IF K=N THEN  RETURN
120 GO SUB 500: REM VORWAERTS
130 LET S=S/2: LET K=K+1
140 LET PHI=PHI+45: GO SUB 100: REM KNOSPE LINKS
150 LET PHI=PHI-90: GO SUB 100: REM KNOSPE RECHTS
160 LET PHI=PHI+45: REM AUSGANGSRICHTUNG
170 LET S=S*2: LET K=K-1
175 REM RUECKWAERTS
180 LET PHI=PHI+180: GO SUB 500: LET PHI=PHI-180
190 RETURN
200
500 REM STRECKE ZEICHNEN
510 LET A=S*COS (PI*PHI/180)
520 LET B=S*SIN (PI*PHI/180)
530 LINE X,Y,X+A,Y+B
535 LET X=X+A: LET Y=Y+B
540 RETURN
```

Bild 5. Listing BAUM in BASIC und LOGO

```
TO BAUM :N :S
IF :N = 0 [STOP]
FORWARD :S LEFT 45
BAUM ( :N - 1 ) ( :S / 2 )
RIGHT 90
 BAUM ( :N - 1 ) ( :S / 2 )
LEFT 45
BACK :S
END
```

```
10 REM **************
11 REM *  TREPPE     *
12 REM **************
15
20 LET HO=5: REM HOEHE
40 LET H=0: LET A$="": REM SCHRITTFOLGE
50 GO SUB 100: STOP
60
100 REM Rekursion
110 IF H=HO THEN  PRINT A$: RETURN
120 LET H=H+1: REM 1-FACH SCHRITT
130 LET A$=A$+"1": GO SUB 100
140 LET H=H-1: LET A$=LEFT$(A$,H)
150 IF H+2>HO THEN  GO TO 200
160 LET H=H+2: REM 2-FACH SCHRITT
170 LET A$=A$+"-2": GO SUB 100
190 LET H=H-2: LET A$=LEFT$(A$,H)
200 RETURN
```

Bild 6. Listing TREPPE

Anstelle binärer Verzweigung können natürlich auch dreifach (ternär) verzweigte Bäume gebildet werden. Die notwendigen Ergänzungen im Programm (Bild 5) sind offensichtlich. Andere Erscheinungsbilder solcher Aufgaben sind die als SIERPINSKI-Teppiche bekannten Bilder (Bild 7). Statt des Zweigmodells ist die zu programmierende Grundstruktur das Dreieck in Bild 7a, die verdickten Ecken symbolisieren die Selbstaufrufe. Die Umsetzung der Idee in ein Programm wird dem Leser nicht schwerfallen.

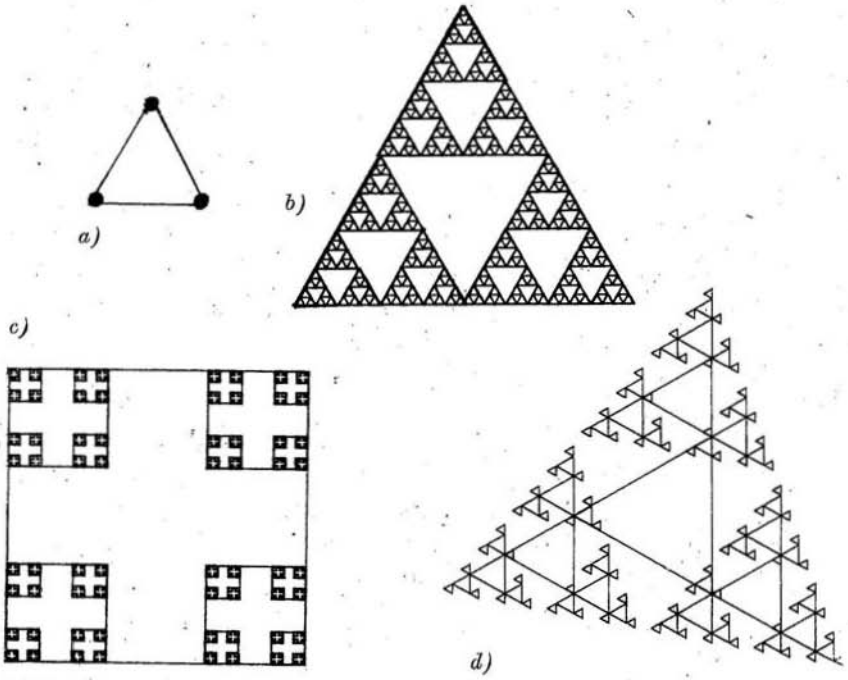

Bild 7. SIERPINSKI-Teppiche

3. Rekursion mit Wertrückgabe

Natürlich kam man auch bei den bisherigen Beispielen nicht ohne Wertrückgabe aus. Insbesondere erforderte die Bedingung der Zustandserhaltung eine saubere Übergabe der globalen Variablen. Hier soll aber nun das schwierigere Problem besprochen werden, bei dem man auf jeder Rekursionsstufe eigene »lokale« Variablen besitzt. In BASIC ist das nur zur realisieren, indem man ein Feld von Variablen anlegt, wobei zu jeder Rekursionsstufe ein gleichnamiger Feldindex gehört. Diese technische Schwierigkeit weist sehr deutlich auf die Grenzen von BASIC hin, und wir wollen uns daher auf die Behandlung zweier interessanter Beispiele beschränken. Ein Ende des vergangenen Jahrhunderts durch den Erfinder zahlreicher mathematischer Spiele, EDOUARD LUCAS, als *Turm von Hanoi* bekannt gemachte Aufgabe besagt: Gegeben sind drei Plätze A, B und C. Auf dem Platz A sind n Scheiben mit immer kleiner werdendem Durchmesser gestapelt. Durch Umlagern einzelner Scheiben ist der Turm von A nach C zu bringen. Dabei darf niemals eine größere auf einer kleineren Scheibe liegen. Der Platz B kann zur Zwischenspeicherung genutzt werden. Man finde (optimale) Lösungen. Die Idee der rekursiven Beschreibung des Transportweges für einen Turm der Höhe N führt auf folgendes Programm:

1. Transportiere die obersten $N-1$ Scheiben vom Ausgangsplatz zur Ablage.
2. Transportiere die N-te Scheibe vom Ausgangsplatz zum Ziel.
3. Transportiere die obersten $N-1$ Scheiben von der Ablage zum Ziel.

Wie im Programm BAUM kann das Problem also durch einen zweimaligen Selbstaufruf gelöst werden. Bild 8 enthält die Umsetzungen in ein BASIC- bzw. ein LOGO-Programm. Im BASIC-Programm dient das Feld PS (I, N) für I = 1, 2, 3 für die auf der Stufe N aktuelle Festlegung von Ablage, Zwischenspeicher und Ziel.

```
 10 REM ***********
 20 REM *  HANOI  *
 30 REM ***********
 40 CLS : INPUT "HOEHE=";NO
 50 LET N=NO+1: DIM P$(3,N)
 60 LET P$(1,N)="A": LET P$(2,N)="B"
 65 LET P$(3,N)="C"
 70 GO SUB 100: STOP
 80
100 REM REKU
110 IF N=1 THEN  RETURN
120 LET N=N-1
130 LET P$(1,N)=P$(1,N+1)
140 LET P$(2,N)=P$(3,N+1)
150 LET P$(3,N)=P$(2,N+1)
160 GO SUB 100
170 PRINT P$(1,N+1);P$(3,N+1);">";
180 IF N=NO THEN  PRINT "HALBZEIT"
190 LET P$(1,N)=P$(2,N+1)
200 LET P$(2,N)=P$(1,N+1)
210 LET P$(3,N)=P$(3,N+1)
220 GO SUB 100
230 LET N=N+1
240 RETURN
```

```
TO HANOI :HOEHE :A :B :C
IF :HOEHE = 0 [STOP]
HANOI ( :HOEHE - 1 ) :A :C :B
TYPE ( WORD :A :C ) TYPE ">
HANOI ( :HOEHE - 1 ) :B :A :C
END
```

Bild 8. Listing HANOI in BASIC und LOGO

Schließlich soll noch folgende interessante Aufgabe behandelt werden. Sie ist unter dem Namen *Risikospiel* bekannt. Denken wir uns einen Spieler, der eine Serie von Einzelspielen, jedes von ihnen mit einer Gewinnchance $0 < p < 1$ durchführt. Der Spieler startet mit einem finanziellen Vermögen $0 \leqq x \leqq 1$. Um sein Ziel $x = 1$ zu erreichen, spielt er folgende Strategie. Ist $0 < x < 1/2$, so setzt er alles, im Fall $1/2 \leqq x < 1$ setzt er den noch fehlenden Betrag $1 - x$. Die Aufgabe besteht darin, die Wahrscheinlichkeit zu ermitteln, mit der der Spieler bei einem gegebenen Startvermögen $x = x_0$ sein Ziel $x = 1$ erreicht. Bezeichnet $f(x)$ diese Wahrscheinlichkeit, so ist folgende Rückführung f auf den Ausgang des nächsten Spieles möglich:

a) Es ist $f(0) = 0$ und $f(1) = 1$.
b) Für $0 < x < 1/2$ ist $f(x) = p \cdot f(2x)$.
c) Für $1/2 \leqq x < 1$ ist $f(x) = p + (1 - p) f(x - (1 - x))$.

```
 10 REM **********
 20 REM * RISIKO *
 30 REM **********
 40
 50 DIM F(100): LET P=0.6: LET Q=1-P
 60 FOR T=0 TO 1 STEP 0.1
 70 LET N=1: LET X=T: REM STARTVERMOEGEN
 80 GO SUB 100: PRINT T,F
 90 NEXT T: STOP
 95
100 REM REKURSION
110 IF X=0 OR X=1 THEN  LET F=X: RETURN
120 IF X<0.5 THEN  LET F(N)=1: LET X=2*X: GO TO 140
130 LET F(N)=2: LET X=2*X-1
140 LET N=N+1
150 GO SUB 100
160 LET N=N-1
170 IF F(N)=1 THEN  LET F=P*F
180 IF F(N)=2 THEN  LET F=P+Q*F
190 RETURN
```

Bild 9. Listing RISIKO

In Bild 9 ist die Umsetzung in ein BASIC-Programm angegeben. Die Zuweisungen $F(N) = 1$ bzw. $= 2$ stehen für die Fälle b) bzw. c), die Auswertung dazu erfolgt in Zeile 170, 180. Das Programm scheint allen Ansprüchen an ein rekursives Verfahren zu genügen, als Rekursionsanfänge dienen $x = 0$ und $x = 1$. Bei näherer Betrachtung erkennt man aber, daß die Rückführung auf diese Zahlen nicht in jedem Fall möglich ist. So würde z. B. ein Startvermögen von $x = 1/3$ zu einem endlosen Zyklus führen. Man findet leicht weitere Beispiele. Dennoch läuft das Programm z. B. auf einem KC.
Die Menge der unzulässigen Startpunkte ist sehr »dünn« und wird von den verfügbaren Maschinenzahlen nicht getroffen. Ein seltener Umstand also, bei dem die Rechnerungenauigkeit nützlich wird.

4. Abschließende Bemerkungen

Es war das Ziel dieses Beitrages, an prägnanten Beispielen in die rekursive Technik einzuführen und eine Methodologie für ihre bewußte Anwendung zu entwerfen. Um einem breiten Leserkreis die Möglichkeit des Experimentierens auf Kleinstrechnern zu geben, wurde der Programmierung in BASIC ein großer Platz einge-

Fortsetzung Seite 17 und 20

Kleine Datenverwaltung in BASIC

Kommerzielle Datenbanken bieten dem Anwender umfangreiche Funktionen zur Verwaltung von Datenlisten. Es wurde versucht, ein Rahmenprogramm in BASIC zu erstellen, welches es dem Amateur ermöglicht, kleine Datenmengen zu verarbeiten. Das Programm ist erstellt auf einem Rechner mit Diskettenlaufwerk und Display mit 24 Zeilen mal 80 Spalten.
Dem erfahrenen Programmierer wird es aber keine Mühe machen, das Programm an seinen Rechner anzupassen. Ob es sich lohnt, ein Kassettenlaufwerk zu benutzen, muß er dann selbst entscheiden.
Die Zeilennummern wurden so gesetzt, daß das Einfügen von rechnerspezifischen Befehlen möglich ist,
z.B.:

- Kassettengerät einschalten
- Kassettengerät stoppen
- Kassette wechseln

usw.

Das vorliegende Programm verarbeitet und erstellt sequentielle Dateien nach dem Prinzip der »Karteikarten«.

Programmbeschreibung

In Zeile 10 erfolgt der Aufruf des Hauptprogramms ab 2630. Es werden acht Funktionen angeboten.
In den Zeilen 2890 bis 2910 wird die Tastatur nach der gewählten Funktion abgefragt und alle Tasten, außer die numerischen von 1 bis 8, werden ignoriert.
Durch Drücken von »8« wird die Abbruchroutine ab 2990 gestartet. Die Variable SP bezeichnet den Zustand des Speichers. Ist sie größer 0, d.h., eine Datei ist im Speicher, so kann durch Drücken von »CR« (Enter) wieder das Hauptprogramm aufgerufen werden. Dadurch wird verhindert, daß das Programm beendet wird, ohne daß eine editierte Datei auf dem Datenträger abgespeichert wurde. Durch Betätigen jeder anderen Taste wird das Programm beendet.
In der Zeile 2950 werden die jeweiligen Unterprogramme entsprechend den Tasten 1 bis 7 aufgerufen. Bei Rückkehr aus diesen Routinen wird in 2970 ein akustisches Signal generiert, und das Menü erscheint wieder vollständig auf dem Bildschirm.

UP 1: Zeilen 30 bis 550

Mit der Variablen SP wird überprüft, ob eine Datei im Speicher ist. Ab Zeile 150 wird der gesamte Speicherbereich gelöscht, und es kann eine neue Datei erstellt werden. Wir definieren die Anzahl (A) von »Karteikarten« und die Menge (B) der je Karteikarte zu unterscheidenden Kriterien. Jetzt werden die Felder für die Datei (D$) und für die Kriterien (S$) dimensioniert.

Von Zeile 210 bis 270 werden die Bezeichnungen der Kriterien abgefragt und in 290 bis 370 auf gleiche Länge formatiert und mit einem Doppelpunkt abgeschlossen.
Die von uns eingegebenen Werte werden dann aufgelistet, und es kann entschieden werden, ob die Datei mit diesen Vorgaben erstellt werden soll (390 bis 490). Durch Drücken von CR wird die Variable SP auf 1 gesetzt (Datei im Speicher).
Jede andere Taste ermöglicht die Wiederholung der Eingaberoutine.
Durch das CLEAR-Statement in Zeile 150 ist es nun aber nicht mehr möglich, das Unterprogramm mit RETURN zu verlassen. Die Rückkehr in das Hauptprogramm wird in Zeile 530 durch »GOTO 2630« gewährleitet.

UP 2: Zeilen 570 bis 850

Dieses Unterprogramm ermöglicht das Einlesen einer Datei von Diskette.
Von 590 bis 670 ist es wiederum möglich, in das Hauptprogramm zurückzukehren, falls sich eine Datei im Speicher befindet (SP = 1). Nach Eingabe des Dateinamens werden nach Diskettenzugriff die Werte der Variablen A und B (vgl. UP 1) eingelesen und die Felder D$ und S$ dimensioniert.
Nach Übergabe der Kriterien und der Daten wird die Diskettendatei geschlossen.
Die Variable SP erhält den Wert 1, und durch »GOTO 2630« wird das Unterprogramm verlassen (beachte CLEAR in 690).

UP 3: Zeilen 870 bis 1110

Das Programm verarbeitet alle ASCII-Zeichen größer Code 35.
(35="#")
Zur Ermittlung der ersten freien Karteikarten wird das erste Zeichen des ersten Satzes auf der Karteikarte überprüft.
Wird eine Karte gefunden, auf der dieses Zeichen dem Code 35 entspricht, so wird der Variablen N der aktuelle Wert der FOR-NEXT-Schleife zugewiesen und die Schleife durch I=A (A=Größe der Datei) beendet. Sollte N dem Wert von A entsprechen, so wird in Zeile 990 eine Information darüber auf dem Bildschirm ausgegeben.
Nach Abarbeitung der Warteschleife W wird das UP verlassen. In den Zeilen 1010 bis 1110 können die Daten auf die Karteikarte eingetragen werden. Die laufende Kartennummer wird angezeigt, und die Kriterien von 1 bis B werden abgefragt.
Bei Eingabe des Code-Zeichens 35 ("#") auf der ersten Position des ersten Kriteriums wird das UP verlassen.

UP 4: Zeilen 1130 bis 1390

In der aktiven Datei können Karteikarten gelöscht werden. Es werden alle Karteikarten von 1 bis A aufgelistet.
Bei Erreichen der ersten freien Karte wird das UP beendet. Durch Drücken der Taste "J" wird die Karte als gelöscht markiert. Die Taste "E" verläßt das UP, "CR" listet die nächste Karteikarte. Ist eine oder sind mehrere Karteikarten gelöscht, so ist die Variable LV in Zeile 1310 auf den Wert 1 gesetzt worden, dadurch wird dann in 1370 sofort das UP zur neuen Sortierung der Datei aufgerufen (vgl. UP 8).

UP 5: Zeilen 1410 bis 1850

Dieses UP ermöglicht es, eine bestehende Datei zu vergrößern, zu verkleinern oder einzelne Kriterien in eine andere Datei zu überführen.
Nach Eingabe des Namens der neuen Datei erwartet das Programm die Eingaben der zu verarbeitenden Kriterien, z.B.:

Reihenfolge ihrer Abarbeitung ?
3215 (CR)

```
10 GOTO 2630
30 REM UP NEUE DATEI
50 IF SP<=0 THEN GOTO 150
70 CLS:PRINT "***** Datei noch nicht gesichert *****"
90 PRINT :PRINT "Zurueck ins Hauptmenue ? (J=CR/N) "
110 J$=INKEY$:IF J$="" THEN GOTO 110
130 IF J$=CHR$(13) THEN RETURN
150 CLS:CLEAR:INPUT "Name der neuen Datei :";N$
170 INPUT "Anzahl der Dateikarten :";A
190 INPUT "Saetze pro Dateikarte  :";B
210 DIM D$(A,B),S$(B)
230 FOR I=1 TO B
250 PRINT I;:INPUT ".Satz :";S$(I)
270 NEXT I
290 L=LEN(S$(1)):FOR I=2 TO B
310 SL=LEN(S$(I))
330 IF SL>L THEN L=SL
350 NEXT I:L=L+1
370 FOR I=1 TO B:S$(I)=S$(I)+SPACE$(L-LEN(S$(I)))+":":NEXT I
390 CLS:PRINT "Format der Datei ";N$:PRINT
410 PRINT "Anzahl der Dateikarten =";A
430 PRINT "Saetze pro Dateikarte  =";B
450 PRINT :FOR I=1 TO B
470 PRINT S$(I):NEXT I:PRINT
490 PRINT "Richtig ? (J=CR/N) "
510 J$=INKEY$:IF J$="" THEN GOTO 510
530 IF J$=CHR$(13) THEN SP=1:GOTO 2630
550 GOTO 30
570 REM UP DATEI LADEN
590 IF SP<=0 THEN GOTO 690
610 CLS:PRINT "***** Datei noch nicht gesichert *****"
630 PRINT :PRINT "Zurueck ins Hauptmenue ? (J=CR/N) "
650 J$=INKEY$:IF J$="" THEN GOTO 650
670 IF J$=CHR$(13) THEN RETURN
690 CLS:CLEAR:INPUT "Name der Datei :";N$
710 OPEN "I",#1,N$
730 INPUT #1,A
750 INPUT #1,B:DIM D$(A,B),S$(B)
770 FOR I=1 TO B:INPUT #1,S$(I):NEXT I
790 FOR I=1 TO A:FOR J=1 TO B
810 INPUT #1,D$(I,J)
830 NEXT J,I
850 CLOSE:SP=1:GOTO 2630
870 REM UP DATEI ERWEITERN
890 FOR I=1 TO A
910 IF MID$(D$(I,1),1,1) >CHR$(35) THEN GOTO 950
930 N=I:I=A
950 NEXT I
970 CLS
990 IF N=A THEN PRINT "***** keine Dateikarten frei *****":
    FOR W=1 TO 3000:NEXT :RETURN
1010 PRINT "Dateikarte Nr.:";N
1030 FOR I=1 TO B:PRINT S$(I);:INPUT D$(N,I)
1050 IF MID$(D$(N,1),1,1)="#" THEN GOTO 1110
1070 NEXT I
1090 PRINT :N=N+1:GOTO 990
1110 RETURN
1130 REM UP KARTEN LOESCHEN
1150 CLS:LV=0:FOR I=1 TO A
1170 IF MID$(D$(I,1),1,1)<=CHR$(35) THEN PRINT :
     PRINT "******** Dateiende ********":I=A:FOR W=1 TO 2000:
     NEXT W:GOTO 1350
```

```
1190 PRINT "Dateikarte Nr.";I:FOR J=1 TO B
1210 PRINT S$(J);D$(I,J):NEXT J
1230 PRINT :PRINT "Loeschen ? (J/N=CR/Ende=E)"
1250 J$=INKEY$:IF J$="" THEN GOTO 1250
1270 IF J$=CHR$(13) THEN GOTO 1330
1290 IF J$="E" OR J$="e" THEN GOTO 1370
1310 IF J$="J" OR J$="j" THEN LV=1:FOR J=1 TO B:D$(I,J)="#" :NEXT J
1330 PRINT
1350 NEXT I
1370 IF LV>0 THEN GOSUB 2390
1390 RETURN
1410 REM UP SORTIEREN IN ZIELDATEI
1430 CLS:FOR I=1 TO B:PRINT S$(I);" Nr.";I:NEXT I
1450 PRINT :INPUT "Name der neuen Zieldatei:";NN$
1470 PRINT "Eingabe der zu verwendenden Saetze,in der "
1490 INPUT "Reihenfolge ihrer Abarbeitung";R$
1510 IF LEN(R$)>B THEN PRINT "Fehler":GOTO 1470
1530 B1=LEN(R$)
1550 PRINT "Alte Anzahl von Dateikarten =";A;" bleibt erhalten? (J=CR/N)"
1570 J$=INKEY$:IF J$="" THEN GOTO 1570
1590 IF J$=CHR$(13) THEN GOTO 1630
1610 PRINT:INPUT "Neue Anzahl der Dateikarten =";A1:GOTO 1650
1630 A1=A
1650 CLS:PRINT "Kopiere Dateikarte Nr.:"
1670 OPEN"O",#1,NN$
1690 PRINT #1,A1
1710 PRINT #1,B1
1730 FOR I=1 TO B1:N=VAL(MID$(R$,I,1)):PRINT #1,S$(N):NEXT I
1750 FOR I=1 TO A1:LOCATE 1,25:PRINT I
1770 FOR J=1 TO B1:N=VAL(MID$(R$,J,1))
1790 IF I<=A THEN PRINT #1,D$(I,N):GOTO 1830
1810 PRINT #1,"          "
1830 NEXT J:NEXT I
1850 CLOSE:RETURN
1870 REM UP SUCHEN IN DER DATEI
1890 CLS
1910 IF B<2 THEN J=1 :GOTO 1990
1930 PRINT "Suchen nach :":FOR I=1 TO B:PRINT S$(I);" --";I:NEXT I
1950 J$=INKEY$:IF J$="" THEN GOTO 1950
1970 J=VAL(J$):IF J<1 OR J>B THEN GOTO 1950
1990 PRINT :PRINT S$(J);:INPUT "=";DS$
2010 FOR I=1 TO A
2030 IF D$(I,J)=DS$ THEN PRINT:PRINT  "Dateikarte Nr.:";I:
     FOR P=1 TO B:PRINT S$(P);D$(I,P):NEXT P:GOTO 2070
2050 NEXT I:GOTO 2150
2070 PRINT :PRINT "Weiter ? (J/CR)"
2090 J$=INKEY$:IF J$="" THEN GOTO 2090
2110 IF J$=CHR$(13) THEN GOTO 2050
2130 I=A:GOTO 2050
2150 RETURN
2170 REM UP SPEICHERN DER DATEI
2190 IF SP<1 THEN CLS:PRINT "****** Datei schon gesichert ******":
     FOR W=1 TO 6000:NEXT :RETURN
2210 OPEN"O",#1,N$
2230 PRINT #1,A
2250 PRINT #1,B
2270 FOR I=1 TO B:PRINT #1,S$(I):NEXT I
2290 FOR I=1 TO A:FOR J=1 TO B
2310 PRINT #1,D$(I,J)
2330 NEXT J,I:CLOSE
2350 SP=0
2370 RETURN
2390 REM KARTEN AUFRUECKEN
2410 CLS:PRINT "Kopiere Dateikarte Nr.:"
```

```
2430 FOR I=1 TO A
2450 LOCATE 1,25:PRINT I
2470 IF MID$(D$(I,1),1,1)>CHR$(35) THEN GOTO 2590
2490 FOR N=I+1 TO A
2510 IF MID$(D$(N,1),1,1)<=CHR$(35) THEN GOTO 2570
2530 FOR J=1 TO B:D$(I,J)=D$(N,J):D$(N,J)="#":NEXT J
2550 N=A
2570 NEXT N
2590 NEXT I
2610 RETURN
2630 REM HAUPTPROGRAMM MIT MENUE
2650 CLS:PRINT TAB(32);"Dateiverwaltung"
2670 PRINT STRING$(80,"M"):PRINT
2690 PRINT TAB(25);"Neue Datei erstellen_____1":PRINT
2710 PRINT TAB(25);"Datei laden_____2":PRINT
2730 PRINT TAB(25);"Eintragen _____3":PRINT
2750 PRINT TAB(25);"Dateikarte loeschen_____4":PRINT
2770 PRINT TAB(25);"Sortieren in Zieldatei_____5":PRINT
2790 PRINT TAB(25);"Suchen in der Datei_____6":PRINT
2810 PRINT TAB(25);"Speichern der Datei_____7":PRINT
2830 PRINT TAB(25);"Ende der Bearbeitung_____8":PRINT
2850 PRINT STRING$(80,"M")
2870 PRINT TAB(25);"Funktion ?"
2890 F$=INKEY$:IF F$="" THEN GOTO 2890
2910 F=VAL(F$):IF F<1 OR F>8 THEN GOTO 2890
2930 IF F=8 THEN GOTO 2990
2950 ON F GOSUB 30,570,870,1130,1410,1870,2170
2970 BEEP:GOTO 2650
2990 IF SP<=0 THEN GOTO 3090
3010 CLS:PRINT "***** Datei noch nicht gesichert *****"
3030 PRINT :PRINT "Zurueck ins Hauptmenue ? (J=CR/N) "
3050 J$=INKEY$:IF J$="" THEN GOTO 3050
3070 IF J$=CHR$(13) THEN GOTO 2970
3090 CLS:PRINT "****** Ende Dateiverwaltung ******":PRINT :END
```

In 1510 wird überprüft, ob mehr Kriterien angegeben wurden als auf den Karteikarten vorhanden sind.
Von 1510 bis 1630 kann die Anzahl der Karteikarten vergrößert oder verkleinert werden.
Dann wird die neue Datei mit ihren Werten für Größe (A1), Anzahl der Kriterien (B1) und in der Reihenfolge der Kriterien, im Beispiel 3–2–1–5, auf Diskette abgespeichert.

UP 6: Zeilen 1870 bis 2150

Dieses UP sucht nach Zeichenketten für bestimmte Kriterien in der Datei. Nach Auflistung der vorhandenen Kriterien kann eines davon gewählt werden (1930 bis 1990).
Nach Abfrage der gesuchten Zeichenkette (D$) werden die Karteikarten durchmustert und bei Übereinstimmung angezeigt. Durch Betätigen von "CR" kann die Suche fortgesetzt werden. Jede andere Taste, Dateiende oder keine Übereinstimmung, führt zum Rücksprung in das Hauptprogramm.

UP 7: Zeilen 2170 bis 2370

Die aktive Datei wird auf die Diskette abgespeichert und die Variable SP auf 0 gesetzt (SP=0, Datei gesichert).

UP 8: Zeilen 2390 bis 2610

Dieses UP wird vom UP 4 aufgerufen, wenn die Variable LV den Wert 1 hat. Die Karteikarten von 1 bis A werden überprüft.
Entspricht das erste Zeichen des ersten Kriteriums dem Code 35 oder kleiner,

so werden auf diese Karteikarte die Zeichenketten der Karteikarte übertragen, auf der diese Bedingung nicht zutrifft. Alle gelöschten Karteikarten werden dadurch entfernt und die verbleibenden Karteikarten hintereinander plaziert, d.h. »aufgerückt«.

Auf eine Routine für alphabetisches Sortieren wurde aus Gründen der Laufzeit und der Speicherkapazität verzichtet. Durch das UP 6 ist das Auffinden von Zeichenketten leicht möglich.
Auf jeder Karteikarte sollten nur bis zu 9 Kriterien verwendet werden, da UP 5 nicht unterscheiden kann, ob das Kriterium 1 oder 10 verwendet werden soll.
Wird eine Datei im UP 5 vergrößert, so wird durch das Freihalten der letzten Karteikarte in den anderen UP's gewährleistet, daß Zeile 1810 leere Karteikarten generiert.
Da bei steigender Datenmenge auch steigender Speicherplatz zur Verfügung gestellt werden muß, folgende Empfehlungen:

- erstes Festlegen von A möglichst klein
- Karteikarten beschreiben
- bei Erreichen des Dateiendes Datei sichern
- Datei schrittweise mit UP 5 vergrößern, dabei mit neuem Namen bezeichnen
- mit neuer Datei weiterarbeiten.

Ist der Rechner dann nicht mehr in der Lage, eine Datei zu verarbeiten, haben wir die letzte Sicherheitskopie auf der Diskette vorliegen.
Dadurch gehen uns bei Speicherüberlauf immer nur wenige Daten verloren.
Zu einer bestehenden Datei können dann weitere Teildateien erstellt werden,
z.B.
LISTE 1, LISTE 2, . . . usw.

Natürlich erhebt das vorliegende Programm (s. S. 14ff.) keinen Anspruch auf Vollkommenheit, aber es ist vielleicht in dem einen oder anderen Anwendungsfall eine willkommene Hilfe für den Amateur.

Autor:
Andreas Andrasch
Akademie der Wissenschaften
Zentralinstitut für Optik und Spektroskopie

Fortsetzung von Seite 11

räumt. Nachdrücklich muß aber nochmals betont werden, daß diese Sprache sehr enge Grenzen zieht und nur einen Notbehelf darstellt.
Daraus ergibt sich insbesondere eine starke Selbstbeschränkung bei der Behandlung rekursiver Verfahren mit Wertrückgabe.
Keine Berücksichtigung fanden daher rekursive Funktionen, rekursive Sortierverfahren und rekursive Dateiverwaltung.
Die Kombination der in diesem Beitrag dargestellten Methoden mit speziellen Auswahlkriterien führt zu rekursiven Problemlöseverfahren, den sogenannten back-tracking-Algorithmen, deren Beschreibung einem späteren Beitrag vorbehalten bleiben muß.

Fortsetzung auf Seite 20

Pixelgraphik – mehr Komfort für den KC 85/2

Die Pixelgraphik des KC 85/3 hat gegenüber der des KC 85/2 den Vorteil, mehrere Funktionen zu besitzen, die sie erheblich komfortabler macht. So z. B. die LINE- bzw. CIRCLE-Funktion. Mit den hier beschriebenen kleinen BASIC-Programmen läßt sich das verbessern.

1. LINE-Funktion

Die ersten drei Zeilen des Programms (S. 19) dienen der Parametereingabe. Es werden zwei Punkte $P_1(x_1, y_1)$ und $P_2(x_2, y_2)$ festgelegt. Sollte x_2 kleiner als der Wert x_1 sein, so werden beide Argumente vertauscht; dasselbe geschieht mit den y-Werten. Ab Zeile 90 werden die Formeln linearer Gleichungssysteme genutzt. Zeile 90 berechnet den Anstieg m der Funktion, in Zeile 100 wird der Schnittpunkt n des entstehenden Graphen mit der y-Achse ermittelt. Mit diesen beiden Ergebnissen steht bereits die Funktionsgleichung desjenigen Graphen fest, der die Punkte P_1 und P_2 verbindet. Die allgemeine Gleichung einer linearen Funktion lautet

$$y = f(x) = mx + n.$$

Würde man nun den Wert x_1 schrittweise so lange erhöhen, bis er gleich x_2 ist, so könnte man die dazugehörenden Funktionswerte (y-Werte) berechnen. Dies geht jedoch nicht, wenn $x_1 = x_2$, da es eine solche Funktion nicht gibt, wenn dabei noch unterschiedliche y-Werte auftreten. Deshalb wird dieser Fall in Zeile 80 heraussortiert und ab Zeile 190 gesondert behandelt. Doch zurück zum eigentlichen Programm. In Zeile 110 wird die Erhöhung des x-Wertes um jeweils 1 durch eine FOR-NEXT-Schleife realisiert. In Zeile 120 wird der zum jeweiligen x-Wert gehörige y-Wert berechnet und der Variablen YJ zugewiesen. Leider sind jedoch die Funktionswerte zweier benachbarter x-Werte selbst oft nicht benachbart (Bild 1). Deshalb müssen die entstehenden Lücken »aufgefüllt« werden. Siehe dazu auch Bild 2. In der Zeile 130 wird der Funktionswert des nächstfolgenden x-Wertes berechnet

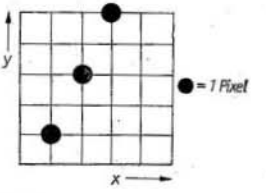

Bild 1. Nichtbenachbarte Funktionswerte bei benachbarten Argumenten

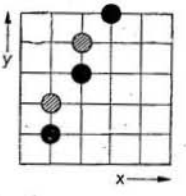

Bild 2. Das »Auffüllen mit Punkten«

und gerundet YZ zugewiesen. (Übrigens wurde auch YJ vorher gerundet.)
SY in Zeile 140 ermittelt das Vorzeichen der Differenz von YZ und YJ: SY ist 1, wenn YZ größer als YJ ist; SY ist —1, wenn das Umgekehrte der Fall sein sollte. Dies ist notwendig, denn das Auffüllen mit Punkten soll unterhalb des folgenden y-Wertes enden, wenn der Graph steigend ist, YZ also größer als YJ ist. Oberhalb von YZ soll es aufhören, wenn der Graph fallend, also YZ kleiner als YJ ist. Dieser Endpunkt wird in Zeile 140 mit YS betitelt. Die Zeilen 160 und 180 übernehmen das »Auffüllen« und das Beenden des Programms. In Zeile 170 wird auf Zeile 250 verwiesen – dort findet die Kontrolle der Setzbarkeit des bzw. der errechneten Punkte statt. Ab Zeile 190 wird der Sonderfall der senkrechten Linie mit $x_1 = x_2$ behandelt – eine Y-FOR-Next-Schleife sorgt dafür. Selbstverständlich kann in dieses Programm noch eine spezielle Farbgebung »eingebaut« werden, z.B.: 35 INPUT" Farbcode: ";F. In den PSET-Anweisungen muß dann noch ,F angefügt werden.
Bild 3 stellt eine fertig gezeichnete Linie dar.

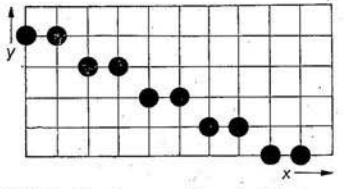

Bild 3. Fertig gezeichnete Linie

2. CIRCLE-Funktion

In diesem Programm wird nach den Formeln

$$y = r \sin t + d$$

und

$$x = r \cos t + c$$

ein Kreis beliebigen Radius berechnet und ausgegeben. Das CIRCLE-Programm ermittelt jedoch nur die Koordinaten für die Punkte des rechten oberen Viertels; durch Kombinieren

```
10      CLS:REM +++ LINE-FUNKTION +++
20      INPUT" P1: ";X1,Y1
30      INPUT" P2: ";X2,Y2
40      IF X1 <=X2 GOTO 80
50      A=X1:B=Y1
60      X1=X2:Y1=Y2
70      X2=A:Y2=B
80      IF X1=X2 GOTO 180
90      M=(Y1-Y2)/(X1-X2)
100     N=Y1-M*X1
110     FOR X=X1 TO X2
120     YJ=INT(M*X+N+0.5)
130     YZ=INT(M*(X+1)+N+0.5)
140     SY=SGN(YZ-YJ):YS=YZ-1*SY
150     IF SY=0 THEN SY=1
160     FOR Y=YJ TO YS STEP SY
170     GOSUB 250:IF J=1 THEN PSET X,Y
180     NEXT Y:NEXT X:END
190     REM --- SENKRECHT ---
200     FOR Y=Y1 TO Y2 STEP SGN(Y2-Y1)
210     GOSUB 250:IF J=1 THEN PSET X,Y
220     END
250     REM --- KONTROLLE ---
260     J=0:IF X<=0 AND X>=319 THEN J=J+1
270     IF Y<=0 AND Y>=255 THEN RETURN:ELSE J=0
280     RETURN
290     REM ----------------
```

```
10 CLS:REM +++ CIRCLE +++
20 INPUT" MITTELPUNKT :"; C, D
30 INPUT" RADIUS :";R
40 A=1/R:T=0
50 X=R+COS(T):Y=R+SIN(T)
60 X=INT(X+.5):Y=INT(T+.5)
70 X1=C+X : X2=C-X : Y1=D+Y : Y2=D-Y
80 PSET X1, Y1 : PSET X2, Y1
90 PSET X2, Y2 : PSET X1, Y2
100 T=T+A : IF T=PI/2 THEN END : ELSE GOTO 50
110 IF I2=1ANDJ 2=1 THEN PSET X2, Y2
120 IF I1=1AND J2=1 THEN PSET X1, Y2
130 T=T+A : IF T=90 THEN END : ELSE GOTO 50
200 I1=0 : I2=0 : J1=0 : J2=0
210 IF X1=0 ANDX1=319 THEN I1=1
220 IF X2=0 ANDX2=319 THEN I2=1
230 IF Y1=0 AND Y1=255 THEN J1=1
240 IF Y2=0 AND Y2=255 THEN J2=1
250 RETURN
```

der Ergebnisse mit den Mittelpunktkoordinaten (c, d) lassen sich die restlichen 3 Viertel sehr leicht berechnen. Dies geschieht in Zeile 70. Um die verschiedenen x- und y-Werte herauszubekommen, wird der Winkel T schrittweise von 0 auf 90 erhöht. Die Schrittweite wird in Zeile 40 errechnet und entspricht dem Winkel, dessen Bogenlänge 1 ist. Sie wird mit A bezeichnet und ist natürlich vom Radius R abhängig, der in Zeile 30 abgefragt wird. Zeile 50 berechnet die Grundwerte der Koordinaten, die in Zeile 60 gerundet und in Zeile 70 unterschiedlich verwendet werden. Bild 4 zeigt die Entstehung des Bildes.

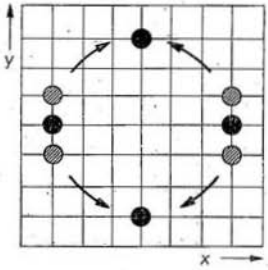

Bild 4. Entstehung des Kreisbildes $c = 4, d = 4, r = 3$; errechnet: 3, 1, folglich zu setzende Punkte: (7, 5), (1, 5), (1, 3), (7, 3)

Autor:
Reno Filla
Schüler der 10. Klasse
19. POS Dessau-Waldersee

Fortsetzung von Seite 17

Literatur

[1] Barron, L.: Rekursive Techniken in der Programmierung. – Leipzig: Teubner Verlag, 1971

[2] Birnstiel, H.: Rekursion – eine faszinierende Beschreibungsmöglichkeit. – In: Mikroprozessortechnik 3 (1988) 9, S. 277—278

[3] Stowasser, R.: Rekursive Verfahren. – Hannover: Schroedel Verlag, 1978

[4] Wagenknecht, C.: Rekursion mit BASIC. – In: Wiss. Ztschr. Päd. Hochschule Dresden **20** (1986), S. 39—44

Autor:
Prof. Dr. Heinz Junek
Pädagogische Hochschule
„Karl-Liebknecht" Potsdam
Sektion Mathematik
Potsdam
1571

Anwendung des Kleinrechners zur Bestimmung des größten gemeinsamen Teilers und zur Bestimmung von Lösungen im Komplexen

1. Einleitung

Meist wird die Benutzung von Kleinrechnern bei der Behandlung von numerischen Problemen, die der Analysis und Algebra entstammen, dargestellt, wobei nur reelle Lösungen bestimmt werden. In dieser Arbeit soll eine kleine Erweiterung dieser üblichen Aufgabenstellungen erfolgen. Und zwar wird die Möglichkeit des Einsatzes eines Kleinrechners bei der Lösung einer zahlentheoretischen Aufgabe und bei einer Aufgabe, die ins Komplexe führt, vorgestellt. Dabei handelt es sich um die Bestimmung des größten gemeinsamen Teilers (ggT) zwei ganzer Zahlen und um die iterative Bestimmung der Wurzeln komplexer Zahlen.

2. Bestimmung des ggT zweier ganzer Zahlen

Der ggT zweier natürlicher Zahlen kann über die Primzahlzerlegung dieser Zahlen ermittelt werden. Will man mit dem Rechner arbeiten, ist es günstig, den EUKLIDischen Algorithmus zu benutzen. Da der ggT auch als Linearkombination der gegebenen natürlichen Zahlen mit ganzzahligen Koeffizienten darstellbar ist, ergibt sich zudem der Wunsch, auch diese Koeffizienten mit dem Rechner zu ermitteln. Weiterhin ist man natürlich an einem Algorithmus interessiert, der mit wenig Speicherplatz auskommt. Die gewünschten Merkmale hat der sog. BERLEKAMP-Algorithmus ([2]). Dabei wird der ggT auf der Grundlage des EUKLIDischen Algorithmus berechnet (Folge $\{r_k\}$). Zusätzlich werden zwei Folgen $\{p_k\}$ und $\{q_k\}$, die die gesuchten ganzzahligen Koeffizienten der Linearkombination liefern, erzeugt.
Der BERLEKAMP-Algorithmus lautet:
Es seien a und b zwei natürliche Zahlen mit $a > b$!
Der Algorithmus funktioniert auch im Fall $a < b$, aber dann wird ein Iterationsschritt mehr benötigt.
Man setze

$r_0 := a,\ r_1 := b$

$p_0 := 0,\ p_1 := 1$

$q_0 := 1,\ q_1 := 0.$

Man berechne

$$r_k = a_{k+1}\, r_{k+1} + r_{k+2} \text{ mit } a_{k+1} = \frac{r_k}{r_{k+1}}$$

$p_{k+2} = a_{k+1}\, p_{k+1} + p_k$

$q_{k+2} = a_{k+1}\, q_{k+1} + q_k, \quad k = 0, 1, 2, \ldots :$

Das Verfahren wird abgebrochen, sobald $r_k = 0$.
Die Eigenschaften der drei so erzeugten Folgen $\{r_k\}$, $\{p_k\}$ und $\{q_k\}$ sind im folgenden Satz zusammengefaßt.

Satz

Es gibt ein $n \geqq 0$ mit $r_n \neq 0$ und $r_{n+1} = 0$. Für dieses n gilt:

a) $r_n = \mathrm{ggT}\,(a, b)$

b) $bp_n - aq_n = (-1)^{n+1}\, r_n$

Zum Beweis des Satzes vergleiche man [1].
Der Begriff des ggT soll noch auf ganze Zahlen erweitert werden.

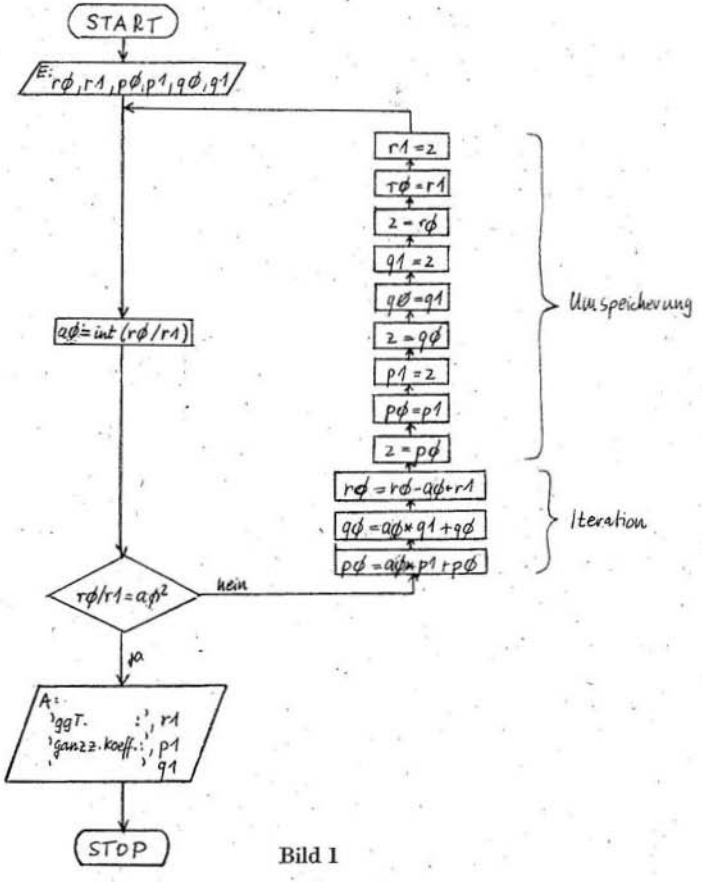

Bild 1

Definition

a, b, c seien ganze Zahlen, c heißt ggT von a und b, falls

a) $c \mid a$ und $c \mid b$,

b) ist d gemeinsamer Teiler von a und b, so teilt d auch c.

ggT (a, b) bezeichnet jetzt die Menge der größten gemeinsamen Teiler von a und b: Falls c die obige Definition erfüllt, gilt ggT $(a, b) = \{-c, c\}$.
Der BERLEKAMP-Algorithmus behält für ganze Zahlen a, b seine Anwendbarkeit, falls $b = 0$ ist. Lediglich die Aussage a) des Satzes muß abgeändert werden, sie lautet jetzt:

ggT $(a, b) = \{r_n, -r_n\}$.

Der Programmablaufplan des BERLEKAMP-Algorithmus ist in Bild 1 angegeben. Auf der Grundlage des Programmablaufplanes kann das folgende BASIC-Programm angegeben werden. Durch entsprechende Kommentare ist der Dialog mit dem Programm von selbst verständlich.

Programm

```
10    CLS
20    PRINT
30    PRINT
40    PRINT " Berechnung des ggT !"
50    PRINT " ====================="
60    PRINT
70    PRINT " BERLEKAMP-Algorithmus "
80    PRINT
90    PRINT " Eingabe:"
100   PRINT
110   INPUT " A = " ; R0
120   INPUT " B = " ; R1
130   P0 = 0
140   P1 = 1
150   Q0 = 1
160   Q1 = 0
170   N = 0
180   N = N + 1
190   G = R0/R1
200   A0 = INT (G)
210   PRINT " Quotient: "; G; TAB (23); "ganzer Teil:"; A0
220   IF G = A0 THEN 400
230   P0 = A0*P1 + P0
240   Z = P0
245   P0 = P1
250   P1 = Z
260   Q0 = A0 * Q1 + Q0
270   Z = Q0
275   Q0 = Q1
280   Q1 = Z
290   R0 = R0 - A0 * R1
300   Z = R0
305   R0 = R1
310   R1 = Z
320   GOTO 180
400   PRINT
410   PRINT
420   PRINT " ggT der eingeg. Zahlen:" ; R1
430   PRINT
```

```
440     PRINT " ganzz. Koeff.:   p(n) = " ; P1
450     PRINT "                  g(n) = " ; Q1
460     PRINT
470     PRINT "                    n  = " ; N
480     PRINT
485     PRINT
490     PRINT " Wollen Sie weitere Werte für "
500     PRINT " a und b eingeben ? "
505     PRINT
510     INPUT " JA = 1, NEIN<>1 " ; I
520     IF I = 1 THEN 530  ELSE 550
530     CLS
540     GOTO 70
550     PRINT
560     PRINT  " ENDE ! "
570     END
```

3. Iterative Bestimmung der Wurzeln komplexer Zahlen

Ein Standardverfahren zur Bestimmung der Lösungen einer Gleichung

$f(x) = 0$

ist das NEWTON-Verfahren. Ausgehend von einer Anfangsnäherung (Schätzung) x_0 der gesuchten Lösung werden nach der Iterationsvorschrift

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)},\ f'(x_k) \neq 0,\ k = 0, 1, 2, \ldots, \tag{1}$$

weitere Näherungswerte berechnet, die unter bestimmten Voraussetzungen zur gesuchten Lösung konvergieren. (Über Lösungsverfahren für nichtlineare Gleichungen kann man z. B. in [3] nachlesen.) Das NEWTON-Verfahren läßt sich auch auf komplexwertige Funktionen anwenden, z. B. auf

$f(z) = z^2 - c,$

$c = a + ib$ komplex, (2)

was zur Bestimmung von $\sqrt{c}$ führt. Wendet man die allgemeine Iterationsvorschrift (1) auf die spezielle Funktion (2) an, so erhält man

$$z_{k+1} = z_k - \frac{z_k^2 - c}{2 z_k} = \frac{1}{2}\left(z_k + \frac{c}{z_k}\right),\ z_k \neq 0,\ k = 0, 1, 2, \ldots. \tag{3}$$

Die Näherungswerte sind nun auch komplexe Zahlen:

$z_k = x_k + iy_k, \quad k = 0, 1, \ldots$.

Die Iterationsvorschrift (3) war bereits HERON VON ALEXANDRIA, der im 1. Jahrhundert u. Z. lebte, bekannt und wird deshalb auch HERONsche Vorschrift genannt. Auf dem Rechner werden Real- und Imaginärteil der Näherungswerte getrennt berechnet. Aus (3) folgt

$$x_{k+1} + iy_{k+1} = \frac{1}{2}\left(x_k + iy_k + \frac{a + ib}{x_k + iy_k}\right).$$

Erweitert man den Bruch auf der rechten Seite mit $x_k - iy_k$, so findet man

$$x_{k+1} + iy_{k+1} = \frac{1}{2}\left(x_k + iy_k + \frac{ax_k + by_k + i\,(bx_k - ay_k)}{x_k^2 + y_k^2}\right).$$

Der Vergleich von Real- und Imaginärteilen auf der linken und rechten Seite liefert

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{ax_k + by_k}{x_k^2 + y_k^2}\right) \tag{4}$$

$$y_{k+1} = \frac{1}{2}\left(y_k + \frac{bx_k - ay_k}{x_k^2 + y_k^2}\right) \tag{5}$$

$k = 0, 1, 2, \ldots$.

Die Beziehungen (4) und (5) beschreiben das Verfahren. Konvergiert die Folge der x_k gegen $\hat{x}$, die der y_k gegen $\hat{y}$, so gilt $\sqrt{c} = \hat{x} + i\hat{y}$. Das Verfahren konvergiert nicht für jeden Startwert $z = x_0 + iy_0$.
Der Programmablauf wird in Form eines Struktogramms (Bild 2) angegeben.

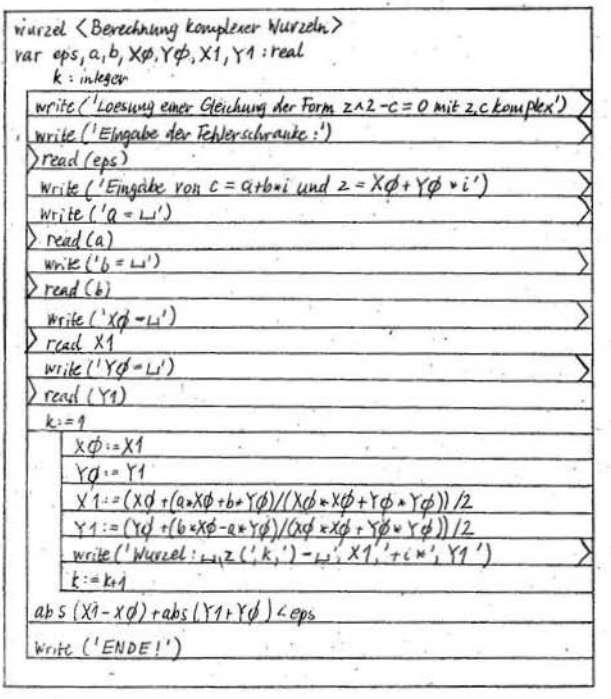

Bild 2

Ein **BASIC-Programm** lautet:

```
10  CLS
20  PRINT
30  PRINT
40  PRINT " BERECHNUNG KOMPLEXER WURZELN "
50  PRINT " ============================ "
60  PRINT
70  PRINT " Lösung der Gleichung z^2 - c = Ø; c, z in C"
80  PRINT
90  PRINT
```

```
100   PRINT " Eingabe: "
110   PRINT
120   INPUT " Fehlerschranke: " ; EPS
122   PRINT
130   INPUT " Realteil von c:" ; A
140   INPUT " Imaginärteil von c:" ; B
142   PRINT
150   INPUT " Realteil von z0:" ; X
160   INPUT " Imaginärteil von z0:" ; Y
162   PRINT
170   PRINT
180   K = 0
190   K = K+1
200   X1 = 1/2* (X+(A* X+B*Y)/(X 2 + Y 2))
210   Y1 = 1/2* (Y+(B* X-A*Y)/(X 2 + Y 2))
220   PRINT "Wurzel nach " ; K ; " Iterationen:"
230   PRINT " Z(";K; ") = "; X1; " + I*"; Y1
232   PRINT
240   IF ABS (X1-X)+ ABS(Y1-Y)< EPS THEN 350
250   PRINT "BETRAG" ; ABS (X1-X)
255   PRINT
260   PRINT
270   X = X1
280   Y = Y1
290   GOTO 190
350   PRINT "BETRAG:" ; ABS (X1-X)
352   PRINT
360   PRINT "Damit ist die gewünschte "
370   PRINT "Genauigkeit erreicht !"
380   PRINT
385   PRINT
390   PRINT "Wollen Sie weitere Wurzeln"
400   INPUT " berechnen ? ( JA = 1   NEIN<>1)" ; I
410   IF I = 1 THEN 420  ELSE 450
420   CLS
430   GOTO 60
450   PRINT
460   PRINT " RECHNUNG BEENDET "
462   PRINT
470   END
```

Zum Abschluß bringen wir ein

Beispiel:

Gibt man ein
Fehlerschranke: EPS = 0.00001
Realteil von c: A = 4
Imaginärteil von c: B = 5
Realteil von z0: X = 2
Imaginärteil von z0: Y = 2,
dann liefert das Verfahren folgende Näherungsfolge

$Z(1) = 2.125 + i \cdot 1.125$
$Z(2) = 2.28412 + i \cdot 1.09223$
$Z(3) = 2.28069 + i \cdot 1.09615$
$Z(4) = 2.28069 + i \cdot 1.09616.$

Mit Z (4) bricht das Verfahren wegen Unterbietung der Fehlerschranke ab.

Literatur

[1] Baptist, P.: Berlekamp-Algorithmus und programmierbarer Taschenrechner. – In: Elemente der Mathematik 36 (1981) 5, S. 126—131

[2] Berlekamp, E. L.: Algebraic Coding Theorie, – New York: McGraw-Hill, 1968

[3] Oelschlägel, D.; Matthäus, W.-G.: Numerische Methoden. – Leipzig: B. G. Teubner, 1985

Autoren:

Prof. Dr. rer. nat. habil. Dieter Oelschlägel
cand. math. C. Schulz

Technische Hochschule „Carl Schorlemmer"
Leuna-Merseburg
Sektion Mathematik

Nutzung von Kleincomputern in der Psychologieausbildung

1. Einleitung

Durch die rasante Entwicklung der Computertechnik entstehen für die Forschung und für die Gestaltung von Unterrichtsmitteln auch in der Psychologie neue Möglichkeiten.

Solche Möglichkeiten bieten sich z. B. in der Statistik (wahlweise-obligatorische Ausbildung), bei der Darbietung und Erläuterung psychodiagnostischer Verfahren oder psychologischer Experimente an [1].

Der Wissenschaftsbereich Psychologie der Pädagogischen Hochschule Güstrow beschäftigt sich im Rahmen seiner Forschungen u. a. mit Fragen der computergestützten Diagnostik. Dabei sind sowohl Neuentwicklungen als auch Applikationen bewährter diagnostischer Verfahren Gegenstand unserer Arbeit.

Wir möchten an einem Beispiel den Grundaufbau eines solchen Vorgehens erläutern und darstellen.

2. Zum Programminhalt

Ausgewählt wurde der Konzentrationstest d2 von BRICKENKAMP [2]. Dieser Aufmerksamkeits-Belastungstest ist ein Durchstreichtest, der die Schnelligkeit und Genauigkeit der Unterscheidung ähnlicher visueller Reize mißt. Er liegt bisher als Papier-Bleistift-Variante vor.

In 14 Zeilen (je Zeile 47 Zeichen) sind alle d mit zwei Punkten durchzustreichen (d̈, ḋ, d̤). Für die Bearbeitung einer Zeile stehen 20 Sekunden zur Verfügung.

Als Parameter für die Einschätzung der Leistung werden die Gesamtzeichen, die Fehler (hier wird noch zwischen Auslassungsfehlern (Typ 1) und Verwechslungsfehlern (Typ 2) unterschieden), die Differenz zwischen Zeichenmenge und Fehler sowie die Schwankungsbreite (Differenz zwischen größter und kleinster Zeilenleistung) genutzt.

Dieser Test ist zur Messung der Konzentrationsfähigkeit bei Jugendlichen und Erwachsenen geeignet. Es liegen Normwerte vor.

3. Zum Programmaufbau

Bild 1 zeigt eine Übersicht des von uns entwickelten Programms. Dabei ließen wir uns von dem traditionellen Aufbau psychologischer Tests leiten (d. h., der Aufbau gliedert sich in Instruktion, Durchführung und Auswertung).

Ein Hauptkriterium bei der Erarbeitung war eine hohe Nutzerfreundlichkeit. Deshalb wurde vorwiegend in Menütechnik und mit einem hohen Aufwand an Bildschirmarbeit programmiert. So konnte eine einfache Handhabe des Programms gewährleistet werden. Besonders problematisch war die Zeichendarstellung (Größe der Zei-

Titelbild

Instruktion

Aufgabenerläuterung mit Übungsbeispiel

Hauptmenü

1. Instruktion
2. Neuer Proband
3. Einzelauswertung
4. Teilgruppenauswertung
5. Gesamtauswertung
6. Vergleich von 2 Gruppen
7. Daten-Eingabe
8. Daten-Ausgabe
9. Service

Dateneingabe

- Kassette
- Tastatur
- Korrektur

Datenausgabe

- Kassette

Service

Programmkopie

Einzelauswertung

Eingabe der Probanden-Nr.

Teilgruppenauswertung

Eingabe von Umfang und Probanden der Teilstichprobe

Berechnung und Ausgabe der Testweite
(Roh- und Standardwerte)

- Gesamtzeichen (GZ)
- Fehlerprozent (F%)
- Gesamttestwert (GZ-F)
- Schwankungsbreite

Verlaufsanalyse
(grafisch, Zeile 1—14)

- Gesamtzeichen
- Fehler (Typ 1/Typ 2)

Drucken

Gesamtauswertung

Vergleich von 2 Gruppen

Eingabe von Umfang und Probanden beider Teilstichproben

Bild 1. Programmübersicht Test d2

chen) und die Tastenabfrage (Reaktionsgeschwindigkeit). Hier mußte teilweise auf Maschinencode zurückgegriffen werden.
Das Durchstreichen wurde durch Betätigung einer beliebigen Taste, außer den beiden Cursortasten ←, →, erreicht. Die beiden genannten Tasten ermöglichten die Bewegung des Zeilpfeiles und damit die Zeichenabfrage. Für die Zeichendarstellung wurde ein BASIC-Programm entwickelt, das die Kreation von Zeichen ermöglicht. Dabei werden 4 Cursortasten so aufgespaltet, daß die Pixel-Setzung überschaubar erfolgen kann. Die so geschaffenen Teilzeichen lassen sich dann zu einem größeren zusammensetzen. Dieses Programm kann von Interessenten bei den Autoren angefordert werden. Bild 2 zeigt einen Teil der Instruktion mit einem Übungsbeispiel, um dem Leser eine Vorstellung der Testvorlage, wie sie auf dem Bildschirm für den Probanden zu sehen ist, zu vermitteln.
Ein weiterer Schwerpunkt der Bearbeitung lag in einer umfangreichen Auswertung. Bild 3 zeigt die Auswertung eines einzelnen Probanden.
Bild 4 zeigt einen Vergleich von zwei Probandengruppen bzw. von zwei Probanden.

INSTRUKTION

Der folgende Versuch prüft , wie gut Du Dich konzentrieren kannst.
Deine Aufgabe besteht darin,alle >d<, die zwei Punkte haben ,wegzulöschen.
Dabei können die beiden Punkte unterschiedlich angeordnet sein :

- ein >d< hat beide Punkte oben,
- das zweite >d< hat die beiden Punkte unten und
- ein drittes je einen unten und einen oben.

INSTRUKTION

Hier ein Uebungsbeispiel:

dddddppdpdpppddpddpddddpdp

dddpdppddddddpddpdpddddddpd

AUFGABE: Alle >d< mit 2 Punkten wegloschen !

Zeichen löschen ..... (TASTE)
Pfeil nach rechts ... (+)
Pfeil nach links .... (+)

Nach 20 Sekunden wird abgebrochen !

Bild 2. Instruktion (Ausschnitt) mit Übungsbeispiel

EINZELAUSWERTUNG VON PROBAND-NR. 34

	R-WERT	%	%-RANG	C-WERT
ZEICHEN	367		43.21	5
FEHLER	17	4.6	75	4
GZ - GF	350		41.59	5
SCHWANK.	36		90	2

VERLAUFSANALYSE

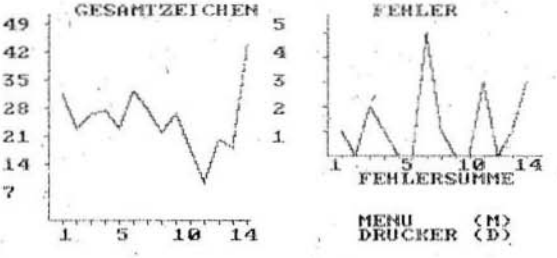

Bild 3. Darstellung (Rechnerausdruck) einer Einzelauswertung

VERGLEICH ZWEIER TEILGRUPPEN

1. Teilgruppe

	R-WERT	%	%-RANG	C-WERT
ZEICHEN	391	6.1	56.36	5
FEHLER	24		90	2
GZ - GF	367		51.15	5
SCHWANK.	11		50	5

2. Teilgruppe

	R-WERT	%	%-RANG	C-WERT
ZEICHEN	390	5.6	55.47	5
FEHLER	22		90	2
GZ - GF	368		51.37	5
SCHWANK.	13		50	5

WEITER (W)
DRUCKER (D)

VERGLEICH ZWEIER TEILGRUPPEN

VERLAUFSANALYSE

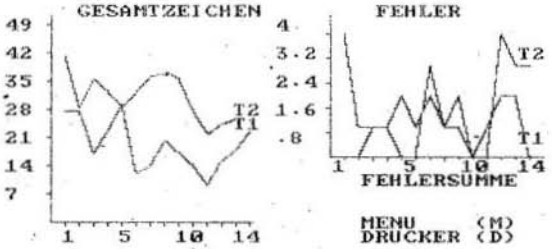

Bild 4. Darstellung (Rechnerausdruck) des Vergleichs von zwei Probandengruppen

Die Ergebnisdarstellung erfolgt sowohl in Rohwerten, %-Rangwerten und in C-Werten. Dadurch wird die Interpretation der Ergebnisse wesentlich erleichtert.

Neben der Erfassung der traditionellen Parameter für diesen Test sowohl als Rohwerte als auch in Normwerten ist bereits mit dem Kleincomputer eine Verlaufsanalyse äußerst günstig zu realisieren. Diese Verlaufsanalyse entspricht dem Trend in der Psychologie nach stärkerer Hinwendung zur Prozeßdiagnostik. Die Darstellung von Zeichenmenge und Fehlerverteilung erhöht die diagnostische Aussagekraft des Verfahrens. Damit liegt der Nutzen des Kleincomputers für entsprechende Aufgaben nicht nur im Zeitgewinn (obwohl er besonders bei Gruppenauswertungen enorm ist), sondern sein Einsatz wirkt sich auch qualitativ aus.

4. Zum Einsatz des Programms

Auf den diagnostischen Nutzeffekt des Programms wurde bereits hingewiesen. Dieses Programm kann sowohl in der Vorlesung (bereits mit Erfolg geschehen) als auch in Übungen eingesetzt werden. Besonders zur Erläuterung und Illustration eines Tests in der

Psychodiagnostik (einschließlich der Instruktion, Durchführung und Auswertung) liegen die Potenzen dieses Programms. Die Zuschauer können über Monitore sowohl den Verlauf der Bearbeitung der Tests als auch deren sofortige Auswertung verfolgen. Weiterhin kann durch ein Unterprogramm der Vergleich und die Diskussion der Leistungen zweier Probanden z.B. in einer Übung erfolgen.
Gleichzeitig kann dieses Programm auch für Forschungszwecke genutzt werden. Dabei kann die Bearbeitung des Tests direkt erfolgen (realisiert durch ein Unterprogramm), oder der Versuchsleiter gibt über ein anderes Unterprogramm die Werte (Zeichenmenge, Fehler je Zeile) ein. Hier wird das Programm nur zur Auswertung genutzt. Das ist z.B. nach einer umfangreichen Papier- und Bleistift-Version denkbar. Der Einbau der Drucker-Routine erleichtert die Datengewinnung ebenfalls beträchtlich.

Zur Zeit laufen Untersuchungen zum Vergleich der Computervariante und der Papier-Bleistift-Variante. Erste Befunde zeigen eine weitestgehende Übereinstimmung.

Zusammenfassend läßt sich feststellen, daß mit solchen Programmen einerseits die Ausbildung effektiver wird (in Form neuer Lehrmittel) und zum anderen auch in diesem Fachgebiet empirische Routinearbeit enorm reduziert werden kann.

Kurzdokumentation

Kurztitel:	TEST d2
Programmname:	Aufmerksamkeits-Belastungstest
Fachgebiet:	Psychologie/Diagnostik
Computer:	KC 85/3 mit RAM
Speicherplatzbedarf:	17 kByte
Programmiersprache:	BASIC/CODE

Literatur

[1] Finck, W.: Kleincomputer am Arbeitsplatz. – In: Psychologie für die Praxis, Heft 1 (1988)

[2] Brickenkamp, R.: Test d2 – Aufmerksamkeits-Belastungstest (Handanweisung). – Göttingen, 1975

Autoren:
Dr. Wolfgang Finck
Mathias Berndt
Pädagogische Hochschule Güstrow
Wissenschaftsbereich Psychologie

Einlesen von Kassettendateien der Commodore-Computer am KC 87

Nachdem in MP 4/88 ein Verfahren zum Einlesen von Dateien des ZX Spectrum vorgestellt wurde, soll nun auch ein entsprechendes Programm zur Veröffentlichung kommen, welches das Einlesen von Dateien der ebenfalls weit verbreiteten Computer C 64, C 16 und plus 4 ermöglicht. Das ist besonders für BASIC-Programme interessant, da die Kleincomputer der KC 85-Reihe und Commodore weitgehend den gleichen BASIC-Dialekt verwenden.

1. Aufzeichnungsverfahren bei Commodore-Computern

Grundlage dieses Beitrages ist das Standardaufzeichnungsverfahren der Commodore-Computer. Daneben gibt es eine Vielzahl von teilweise recht unterschiedlichen beschleunigten Verfahren (Turbo-Tapes), die jedoch im folgenden keine Berücksichtigung finden.

Zur Gewährleistung der Datensicherheit weist das Commodore-Aufzeichnungsverfahren eine hohe Redundanz auf, was allerdings mit einer geringen Übertragungsgeschwindigkeit verbunden ist. Sie beträgt etwa ein Viertel der beim KC 85 erreichten Datenrate.

Commodore benötigt für ein Bit zwei Perioden einer Rechteckschwingung. Eine Null wird durch eine Schwingung höherer Frequenz, gefolgt von einer Schwingung niederer Frequenz, dargestellt. Für eine Eins sind die Schwingungen vertauscht. Das Byte beginnt mit einem Synchronbit auf einer dritten Frequenz. Den acht Datenbits folgt ein Paritätsbit (s. Tafel 1).

Beim Dateiaufbau muß man im wesentlichen Programmdateien und sequentielle Datenfiles unterscheiden. Programme werden ungeblockt, Datenfiles in Blöcken zu je 192 Bytes übertragen. Ein Block setzt sich aus einem Vorton, einem Countdown, dem Daten-

	C 64	C16/C116/plus4
Vorton	3,4 kHz	2,4 kHz
Synchronbit	1,7 kHz	600 Hz
"hohe Frequenz"	3,4 kHz	2,4 kHz
"niedrige Frequenz"	2,5 kHz	1,2 kHz

Tafel 1. Übertragungsfrequenzen des Commodore-Kassettenaufzeichnungsverfahrens (experimentell am KC 87 ermittelt)

teil sowie der Wiederholung von Countdown und Datenteil zusammen. Zu Beginn jeder Datei wird ein Kopfblock übertragen, der Dateityp und Dateinamen, bei Programmen auch Anfangs- und Endadresse enthält.

2. Beschreibung des Einleseprogramms

Zum physischen Einlesen von Commodore-Dateien wurde auf dem KC 87 ein 463 Bytes langes Maschinenprogramm erstellt. Dabei wurden die Initialisierung von PIO und CTC sowie die Interruptroutine vom Betriebssystem des KC 87 übernommen. Das Programm belegt den Speicherbereich von 400H bis 5CEH. Die eingelesene Datei beginnt auf der Adresse 600H.
Auf der Adresse 45BH beginnt die Routine LPER zum Messen der Halbperiodenlänge einer Schwingung auf dem Kassetteninterface. Dort löst jede Flanke einen Interrupt aus, bei dem der CTC-Zählerstand auf die RAM-Zelle 6AH geschrieben wird. LPER berechnet daraus die echte Taktanzahl und übergibt sie an das aufrufende Programm. Das ist im Normalfall die auf 46BH beginnende Bitleseroutine IBIT. Das aus vier Halbperioden gebildete Bit wird über das Carry-Flag an die Routine IBYTE auf 480H übergeben. Diese sammelt die 8 Bit eines Bytes und führt die Paritätskontrolle aus. Ein Paritätsfehler wird über das Carry-Flag signalisiert, das gelesene Byte im Akku bereitgestellt.
Mit BLKR wurde die auf 4A8H beginnende Blockleseroutine bezeichnet. Hier wird zunächst der dem Block vorangehende Countdown gesucht. Danach erfolgt das Lesen der Daten in den vorgesehenen Speicherbereich. Registrierte Paritätsfehler inkrementieren einen Fehlerzähler. Bei Fehlerfreiheit wird mit Erreichen der maximalen Blocklänge die Routine abgeschlossen. Anderenfalls wird mit dem Lesen der Wiederholung eine Fehlerkorrektur versucht. Dazu werden die jetzt eingelesenen Daten mit den bereits gespeicherten verglichen. Bei auftretenden Differenzen gilt das in der Wiederholung gelesene Byte, sofern es paritätsrichtig ist. Das Hauptprogramm beginnt auf der Adresse 538H. Zunächst wird die Initialisierung entsprechend den vom Betriebssystem benutzten Routinen vorgenommen. Nach dem Einlesen des Kopfblockes wird der Dateiname angezeigt. Je nach Dateityp werden nun der Programmblock oder bis zum Abbruch Datenblöcke eingelesen. Das Programm endet mit einem Warmstart des Betriebssystems.

3. Bedienung

Für die volle Funktion des Programms ist es wichtig, daß es nach dem Laden nicht automatisch startet. Der Programmstart erfolgt aus dem Betriebssystem mit dem Kommando LDC64, wenn eine Datei des C 64 gelesen werden soll, für Dateien der Computer C 16, C 116 und plus 4 mit dem Kommando LDC16. Auf diese Weise wird die für den Computertyp zutreffende Zeitkonstante zur Erkennung des Synchronbits eingestellt.
Nach der Ausschrift »START TAPE« ist die Kassette mit der zu lesenden Commodore-Datei zu starten und die »ENTER«-Taste zu betätigen. Bei Programmen erfolgt nun das Einlesen bis zum Programm- oder Speicherende, bei Datenfiles muß der Abbruch manuell erfolgen. Zu diesem Zweck ist die »CTRL«-Taste zu drücken. Da die Tastatur während des Einlesens nicht vollständig dekodiert wird, führen auch einige andere Tastenbetätigungen zum Programmabbruch.

4. Erweiterungsmöglichkeiten

Mit dem physischen Einlesen eines BASIC-Programmes von einem Commodore-Computer ist leider die Laufähigkeit auf dem KC 87 noch nicht gesichert. Wichtigste Nachfolgearbeit ist die Umrechnung der Tokens nach Tafel 2. Das kann manuell mit Hilfe eines Monitors oder über ein entsprechend erweitertes Programm (Tafel 3) erfolgen. Schwierigkeiten gibt es bei Befehlen für Farbe, Grafik, Sound und Dateibehandlung. Dazu enthält Commodore-BASIC V3.5 in den C 16-kompatiblen Computern zahlreiche leistungsfähige Befehle, für die es beim KC 87 keine Entsprechung gibt. An solchen

Schlüsselwort	Token Commod V 3.5	Token KC 85	Schlüsselwort	Token Commod V 3.5	Token KC 85
END	80	80	SGN	B4	B6
FOR	81	81	INT	B5	B7
NEXT	82	82	ABS	B6	B8
DATA	83	83	USR	B7	B9
INPUT	85	84	FRE	B8	BA
DIM	86	85	POS	B9	BC
READ	87	86	SQR	BA	BD
(LET)	88	87	RND	BB	BE
GOTO	89	88	LOG(ln)	BC	BF
RUN	8A	89	EXP	BD	C0
IF	8B	8A	COS	BE	C1
RESTORE	8C	8B	SIN	BF	C2
GOSUB	8D	8C	TAN	C0	C3
RETURN	8E	8D	ATN	C1	C4
REM	8F	8E	PEEK	C2	C5
STOP	90	8F	LEN	C3	C8
ON	91	91	STR$	C4	C9
WAIT	92	93	VAL	C5	CA
cLOAD	93	A2	ASC	C6	CB
cSAVE	94	A3	CHR$	C7	CC
DEF	96	94	LEFT$	C8	CD
POKE	97	95	RIGHT$	C9	CE
PRINT	99	9E	MID$	CA	CF
CONT	9A	9F	INSTR	D4	D8
LIST	9B	A0	ELSE	D5	D4
CLeaR	9C	A1	TRON	D8	D1
SYS(call)	9E	9D	TROFF	D9	D2
NEW	A2	A4	SCNCLR(cls)	E8	99
TAB	A3	A5	DELETE	F7	DA
TO	A4	A6	RENUMBER	F8	D9
FN	A5	A7	PI	FF	C7
SPC	A6	A8			
THEN	A7	A9	Bei einigen Befehlen sind gleiche Funktionen durch unterschiedliche Schlüsselwörter realisiert. In solchen Fällen wurden die am KC 85 zusätzlich eingegebenden Zeichen klein geschrieben bzw. das für den KC 85 geltende Schlüsselwort in Klammern hinter das Commodore-Schlüsselwort geschrieben.		
NOT	A8	AA			
STEP	A9	AB			
+	AA	AC			
-	AB	AD			
*	AC	AE			
/	AD	AF			
^	AE	B0			
AND	AF	B1			
OR	B0	B2			
>	B1	B3			
=	B2	B4			
<	B3	B5			

Tafel 2. Umrechnungstabelle der Standard-BASIC-Tokens

Tafel 3. Quellprogramm zum Einlesen von Commodore-Dateien am KC 87

```
                                ORG   400H
0005                    BDOS    EQU   5           ;SYSTEMADRESSEN
0036                    EORAM   EQU   36H
006A                    ARB     EQU   6AH
006B                    BLNR    EQU   6BH
FB0A                    INIC1   EQU   0FB0AH
0028                    LMAX    EQU   28H         ;SYNC-LAENGE C64
                                                  ;
0400 C3 0538            COLA:   JP    LDC64       ;KOMMANDOTABELLE
0403 4C 44 43 36                DEFB  'LDC64      '
0407 34 20 20 20
040B 00                         DEFB  0
040C C3 053C                    JP    LDC16
040F 4C 44 43 31                DEFB  'LDC16      '
0413 36 20 20 20
0417 00                         DEFB  0
0418 0A0D               STEX:   DEFW  0A0DH       ;TEXTE
041A 53 54 41 52                DEFB  'START TAPE'
041E 54 20 54 41
0422 50 45
0424 0A0D                       DEFW  0A0DH
0426 00                         DEFB  0
0427 0A0D               ETEX:   DEFW  0A0DH
0429 45 52 52 4F                DEFB  'ERROR'
042D 52
042E 0A0D                       DEFW  0A0DH
0430 00                         DEFB  0
0431 0A0D               RTEX:   DEFW  0A0DH
0433 52 45 41 44                DEFB  'READY'
0437 59
0438 0A0D                       DEFW  0A0DH
043A 00                         DEFB  0
043B 0A0D               BTEX:   DEFW  0A0DH
043D 42 52 45 41                DEFB  'BREAK'
0441 4B
0442 0A0D                       DEFW  0A0DH
0444 00                         DEFB  0
                                                  ;
0445 C5                 OTEX:   PUSH  BC          ;TEXTAUSGABE
0446 0E 09                      LD    C,9
0448 CD 0005                    CALL  BDOS
044B C1                         POP   BC
044C C9                         RET
                                                  ;
044D E5                 OFIN:   PUSH  HL          ;FILENAME ANZEIGEN
044E 21 05E9                    LD    HL,FINA+10H
0451 36 00                      LD    M,0         ;ENDE MARKIEREN
0453 2E D9                      LD    L,LOW(FINA)
0455 EB                         EX    DE,HL
0456 CD 0445                    CALL  OTEX
0459 E1                         POP   HL
045A C9                         RET
                                                  ;
045B 3A 006A            LPER:   LD    A,(ARB)     ;IMPULSDAUER MESSEN
045E B7                         OR    A
045F 28 FA                      JR    Z,LPER      ;WARTEN AUF FLANKE
0461 D6 B0                      SUB   0B0H
0463 2F                         CPL               ;ECHTE CTC-ZYKLEN
0464 F5                         PUSH  AF
0465 AF                         XOR   A
0466 32 006A                    LD    (ARB),A     ;ARB ZURUECKSETZEN
0469 F1                         POP   AF
046A C9                         RET
046B C5                 IBIT:   PUSH  BC          ;BIT LESEN
046C CD 045B                    CALL  LPER        ;LAENGE HALBPERIODE1
046F 4F                         LD    C,A
0470 CD 045B                    CALL  LPER        ;LHP2
0473 81                         ADD   A,C
0474 4F                         LD    C,A         ;LHP1+LHP2
0475 CD 045B                    CALL  LPER        ;LHP3
0478 47                         LD    B A
```

```
0479 CD 045B            CALL LPER       ;LHP4
047C 80                 ADD  A,B        ;LHP3+LHP4
047D 91                 SUB  C          ;-(LHP1+LHP2)
047E C1                 POP  BC
047F C9                 RET             ;CY=1
                                        ;
0480 C5          IBYTE: PUSH BC         ;BYTE LESEN
0481 06 08              LD   B,8        ;BITZAEHLER
0483 CD 045B     IB1:   CALL LPER
0486 FE 28       CP1:   CP   LMAX       ;WARTEN AUF SYNC-BIT
0488 38 F9              JR   C,IB1
048A CD 045B     IB2:   CALL LPER
048D FE 28       CP2:   CP   LMAX       ;3.HALBPERIODE SYNC
048F 30 F9              JR   NC,IB2
0491 CD 045B            CALL LPER       ;4.HALBPERIODE SYNC
0494 CD 046B     IB3:   CALL IBIT       ;DATENBIT LESEN
0497 CB 19              RR   C
0499 10 F9              DJNZ IB3
049B CD 046B            CALL IBIT       ;PARITAETSBIT
049E 78                 LD   A,B
049F 1F                 RRA
04A0 A9                 XOR  C
04A1 E2 04A5            JP   PO,IB4
04A4 37                 SCF             ;PARITAETSFEHLER
04A5 79          IB4:   LD   A,C        ;DATENBYTE IN A
04A6 C1                 POP  BC
04A7 C9                 RET
                                        ;
04A8 E5          BLKR:  PUSH HL         ;BLOCKLESEROUTINE
04A9 C5                 PUSH BC
04AA E5                 PUSH HL
04AB 21 0000            LD   HL,0      ;FEHLERZAEHLER=0
04AE 22 05D2            LD   (ECON),HL
04B1 E1                 POP  HL
04B2 DB 91       BLK1:  IN   A,(91H)   ;TASTATURABFRAGE
04B4 FE 7F              CP   7FH       ;BREAK Z.B. MIT CTRL
04B6 20 0A              JR   NZ,BLK2
04B8 11 043B     BLKX:  LD   DE,BTEX   ;ABBRUCHMELDUNG
04BB CD 0445            CALL OTEX
04BE C1                 POP  BC
04BF E1                 POP  HL
04C0 F1                 POP  AF
04C1 C7                 RST  0         ;SYSTEMWARMSTART
04C2 CD 0480     BLK2:  CALL IBYTE     ;COUNTDOWN SUCHEN
04C5 FE 82              CP   82H
04C7 20 E9              JR   NZ,BLK1
04C9 CD 0480            CALL IBYTE
04CC FE 81              CP   81H
04CE 20 E2              JR   NZ,BLK1
04D0 CD 0480            CALL IBYTE     ;BLOCKTYP
04D3 FE 02              CP   2
04D5 20 09              JR   NZ,BLK4
04D7 DB 91       BLK3:  IN   A,(91H)
04D9 FE 7F              CP   7FH       ;BREAK?
04DB 28 DB              JR   Z,BLKX
04DD CD 0480            CALL IBYTE     ;DATENBYTE LESEN
04E0 77          BLK4:  LD   M,A
04E1 22 05D0            LD   (XADR),HL
04E4 30 09              JR   NC,BLK5
04E6 E5                 PUSH HL
04E7 2A 05D2            LD   HL,(ECON)
04EA 23                 INC  HL        ;FEHLERZAEHLER+1
04EB 22 05D2            LD   (ECON),HL
04EE E1                 POP  HL
04EF 23          BLK5:  INC  HL        ;PUFFERZEIGER+1
04F0 0B                 DEC  BC
04F1 79                 LD   A,C
04F2 B0                 OR   B         ;ENDE?
04F3 20 E2              JR   NZ,BLK3
04F5 2A 05D2            LD   HL,(ECON)
04F8 7C                 LD   A,H
```

```
04F9 B5                  OR   L          ;0 FEHLER?
04FA C1                  POP  BC
04FB E1                  POP  HL
04FC C8                  RET  Z          ;FEHLERFREI: ENDE
04FD DB 91       BLK6:   IN   A,(91H)
04FF FE 7F               CP   7FH        ;BREAK?
0501 C8                  RET  Z          ;KEINE KORREKTUR
0502 CD 0480     BLK7:   CALL IBYTE      ;COUNTDOWN SUCHEN
0505 FE 02               CP   2
0507 20 F4               JR   NZ,BLK6
0509 CD 0480             CALL IBYTE
050C FE 01               CP   1
050E 20 ED               JR   NZ,BLK6
0510 C5                  PUSH BC
0511 CD 0480             CALL IBYTE      ;BLOCKTYP
0514 FE 02               CP   2
0516 20 0B               JR   NZ,BLK9
0518 DB 91       BLK8:   IN   A,(91H)
051A FE 7F               CP   7FH        ;BREAK?
051C 28 18               JR   Z,BLK11
051E CD 0480             CALL IBYTE      ;DATENBYTE LESEN
0521 38 0D               JR   C,BLK10
0523 BE          BLK9:   CP   M          ;VERGLEICH
0524 28 0A               JR   Z,BLK10
0526 77                  LD   M,A        ;KORREKTUR
0527 E5                  PUSH HL
0528 2A 05D2             LD   HL,(ECON)
052B 2B                  DEC  HL         ;FEHLERZAEHLER-1
052C 22 05D2             LD   (ECON),HL
052F E1                  POP  HL
0530 23          BLK10:  INC  HL         ;PUFFERZEIGER+1
0531 0B                  DEC  BC
0532 79                  LD   A,C
0533 B0                  OR   B          ;ENDE?
0534 20 E2               JR   NZ,BLK8
0536 C1          BLK11:  POP  BC
0537 C9                  RET
                         ;
0538 3E 28       LDC64:  LD   A,28H      ;HAUPTPROGRAMM
053A 18 02               JR   LDO
053C 3E 68       LDC16:  LD   A,68H      ;LMAX EINSTELLEN
053E 32 0487     LDO:    LD   (CP1+1),A
0541 32 048E             LD   (CP2+1),A
0544 11 0418             LD   DE,STEX
0547 CD 0445             CALL OTEX       ;"START TAPE"
054A 0E 01               LD   C,1
054C CD 0005             CALL BDOS
054F FE 03               CP   3
0551 CA 0000             JP   Z,0        ;ABBRUCH
0554 F3                  DI
0555 CD FB0A             CALL INIC1      ;INITIALISIERUNG
0558 D3 93               OUT  (93H),A
055A D3 8A               OUT  (8AH),A
055C 3E 05               LD   A,5
055E D3 80               OUT  (80H),A
0560 3E B0               LD   A,0B0H
0562 D3 80               OUT  (80H),A
0564 3E 0F               LD   A,0FH
0566 D3 8A               OUT  (8AH),A
0568 3E 0A               LD   A,0AH
056A D3 8A               OUT  (8AH),A
056C 3E E7               LD   A,0E7H
056E D3 8A               OUT  (8AH),A
0570 FB                  EI
0571 01 0015             LD   BC,15H     ;LAENGE
0574 21 05D4             LD   HL,HEAD    ;ZIELADRESSE
0577 CD 04A8             CALL BLKR       ;KOPF LESEN
057A CD 044D             CALL OFIN       ;FILENAME ANZEIGEN
057D 3A 05D4             LD A,(HEAD)     ;BLOCKTYP
0580 FE 01               CP   1          ;BASIC-PROGRAMM?
0582 20 1E               JR   NZ,LD1
```

```
0584 ED 5B 05D5          LD   DE,(AADR)
0588 2A 05D7             LD   HL,(EADR)
058B ED 52               SBC  HL,DE      ;DATEILAENGE
058D 44                  LD   B,H
058E 4D                  LD   C,L
058F 2A 0036             LD   HL,(EORAM)         ;RAM-ENDE
0592 11 0600             LD   DE,DATA    ;PUFFERANFANG
0595 ED 52               SBC  HL,DE      ;MAX. LAENGE
0597 E5                  PUSH HL
0598 ED 42               SBC  HL,BC      ;VERGLEICH
059A 30 03               JR   NC,LD2
059C C1                  POP  BC         ;DATEI ZU LANG
059D 18 06               JR   LD3
059F E1            LD2:  POP  HL         ;VERGESSEN
05A0 18 03               JR   LD3
05A2 01 00C0       LD1:  LD   BC,0C0H    ;GEBLOCKTE DATEI
05A5 21 0600       LD3:  LD   HL,DATA    ;PUFFERANFANG
05A8 CD 04A8       LD6:  CALL BLKR       ;DATEN EINLESEN
05AB 3A 05D4             LD   A,(HEAD)
05AE FE 01               CP   1          ;BASIC-PROGRAMM
05B0 28 0C               JR   Z,LD4      ;LESEN BEENDET
05B2 09                  ADD  HL,BC      ;NAECHSTER BLOCK
05B3 E5                  PUSH HL
05B4 09                  ADD  HL,BC
05B5 ED 5B 0036          LD   DE,(EORAM)
05B9 ED 52               SBC  HL,DE      ;KONTROLLE UEBERLAUF
05BB E1                  POP  HL
05BC 38 EA               JR   C,LD6
05BE 11 0431       LD4:  LD   DE,RTEX    ;ENDE
05C1 2A 05D2             LD   HL,(ECON)
05C4 7C                  LD   A,H
05C5 B5                  OR   L
05C6 28 03               JR   Z,LD5      ;FEHLERFREI:"READY"
05C8 11 0427             LD   DE,ETEX    ;SONST:"ERROR"
05CB CD 0445       LD5:  CALL OTEX
05CE C7                  RST  0
05CF 00                  NOP
                                         ;
05D0 0000          XADR: DEFW 0          ;AKTUELLE ADRESSE
05D2 0000          ECON: DEFW 0          ;FEHLERZAEHLER
05D4 00            HEAD: DEFB 0          ;DATEITYP
05D5 0000          AADR: DEFW 0          ;ANFANGSADRESSE
05D7 0000          EADR: DEFW 0          ;ENDADRESSE
05D9               FINA: DEFS 10H        ;DATEINAME
0600               DATA  EQU  600H       ;DATENPUFFER
                         END
```

Stellen sollten REM-Befehle eingesetzt werden. Ein weiteres Problem stellen die in Commodore-BASIC V3.5 verwendbaren Integer-Variablen dar. Diese müssen für den KC 87 in normale Real-Variablen umgewandelt werden, wodurch der Speicherbedarf aber steigt.

Hat man das aufbereitete Programm im Speicher, muß schließlich noch der für KC 85-Rechner gültige Dateiaufbau hergestellt werden. Dazu wird dem Programm dreimal der Code D3H vorangestellt. Danach muß der acht Byte lange Dateiname folgen. Die zwei Bytes unmittelbar vor dem Programmtext müssen die hexadezimale Programmlänge enthalten. Das Programmende wird durch drei Bytes 0 und den Code 03 gekennzeichnet. Zum Abspeichern kann im einfachsten Fall das Programm OS-SAVE benutzt werden. In ein komfortables Programm, das die obengenannten Umwandlungen ausführt, sollte aber auch das Abspeichern integriert werden. Ein solches Programm erreicht einschließlich einer LIST-Funktion eine Größe von etwa 2 KBytes.

Prinzipiell ist das Einlesen von Com-

Speicherausdruck Programm zum Einlesen von Commodore-Dateien an KC 87

```
0400: C3 38 05 4C 44 43 36 34 20 20 20 00 C3 3C 05 4C  .8.LDC64   .<.L
0410: 44 43 31 36 20 20 00 0D 0A 53 54 41 52 54 20 0A  DC16   .START .
0420: 54 41 50 45 20 0D 0A 00 0D 0A 45 52 52 4F 52 0D  TAPE  ...ERROR..
0430: 00 0D 0A 52 45 41 44 59 0D 0A 42 52 45 0D 0A 00  ...READY..BRE...
0440: 41 4B 00 0A D9 EB CD 0E 09 CD 45 00 C9 E5 21 E9  AK.....E.....!.
0450: 05 36 00 2E F5 AF 32 6A 00 F1 C9 3A 00 B7 28 03  .6../..2j...:..(.
0460: FA D6 B0 2F F5 CD 6A 00 F1 C9 C5 CD 5B 04 4F 06  .../..j.....[.O.
0470: CD 5B 08 81 4F CD 5B 04 47 CD 5B 04 80 91 C1 C9  .[..O.[.G.[.....
0480: C5 CD 06 08 CD 5B 04 FE 04 28 38 F9 CD 5B 04 FE  .....[...(8...[..
0490: F9 CD 5B 04 CD 6B 04 CB 19 10 F9 CD 6B 00 04 78  ..[..k......k..x
04A0: A9 E2 A5 04 37 79 C1 C9 E5 C5 E5 21 00 00 04 22  ....7y.....!...."
04B0: 05 E1 DB 91 FE 7F 82 20 20 A 11 3B 80 CD 04 FE  ..... ...;.....
04C0: F1 C7 CD 80 04 FE 09 20 DB 91 FE 7F 28 DB CD 80  ..... ....(.....
04D0: CD 80 C7 04 FE 05 30 20 E5 09 2A D2 05 7C B5 D2  ......0 ..*..|..
04E0: 77 22 D0 05 E2 04 FE 02 20 0B DB 91 05 23 22 C8  w"...... ....#"..
04F0: 7F C8 CD 80 04 FE 02 07 E5 2A D2 20 F4 CD 80 7F  .....*. .. .....
0500: C5 CD 80 04 FE 0A C1 C9 3E 00 F5 2A D2 05 F4 CD  ........>....*..
0510: 04 38 0D BE 20 E2 04 CD 45 00 2A A8 04 4D 52 E1  .8.. ...E.*..MR.
0520: 04 23 0B 79 B0 20 11 18 04 CD 45 00 3E 04 3A A8  .#.y. .....E.>.*.
0530: 04 32 8E 04 11 18 04 CD 45 00 0E 01 CD 05 E1 38  .2......E......8
0540: 03 CA 00 00 F3 CD 0A FB D3 93 D3 8A 3E E7 D3 8A  ........>.....>..
0550: 3E B0 15 00 3E 0F D5 CD A8 04 CD D3 8A 04 3A D4  >...>.....:.....
0560: FB 01 20 1E ED 5B D5 05 E5 ED 42 30 03 C1 18 06  .. ..[....B0....
0570: FE 01 20 11 00 00 06 21 00 36 00 CD A8 04 3A D4  .. ....!.6....:.
0580: 36 00 11 C0 09 ED 5B 36 00 ED 52 E1 38 EA 11 31  6.....[6..R.8...1
0590: 18 03 01 C0 09 E5 7C B5 28 03 11 27 04 CD 45 04  ......|.(..'..E.
05A0: 28 0C 09 E5 7C D2 05 28 03 11 27 04 C7 00 C7 00  (...|..(..'.....
05B0: 04 2A D2 05 7C B5 28 03 11 27 04 CD 45 04 C7 00  .*..|.(..'..E...
05C0:
```

Tafel 4

modore-Dateien auch am KC 85/3 möglich (s. Tafel 4). Dazu müssen die Betriebssystemaufrufe angepaßt, die Initialisierung sowie die Zeitkonstanten zur Erkennung des Synchronbits geändert werden. Für C 64-Dateien kann eine kleine Hardwareänderung erforderlich werden, um die obere Grenzfrequenz des Kassetteninterfaces zu erhöhen.

(Literatur Seite 44)

Autor:
Bertold Biener

LDIR – ein Weg zur Einstimmung auf die Programmierung in Maschinensprache?

1. Problemstellung

Aus verschiedenen Gründen tritt der Wunsch oder auch die Notwendigkeit der Vermittlung von Vorstellungen sowie erster Fähigkeiten und Fertigkeiten bezüglich der Nutzung von Elementen der Maschinenprogrammierung auch bei Nutzern von Kleincomputern auf. Hier stellt sich nun die Frage, wie soll bzw. wie kann man anschaulich, fachlich richtig und in begrenzter Zeit Grundlagen der Arbeit mit einem Mikroprozessorsystem vermitteln. Die Erfahrungen zeigen, daß besonders die begrenzte »Anschaulichkeit« des Programmlaufes, die hohe Anzahl von einzelnen Anweisungen im Vergleich mit dem »sichtbaren Effekt« und der Aufwand für die Dialogfähigkeit des Programms gewisse Hemmschwellen bewirken bzw. verstärken.

Im folgenden soll der Versuch unternommen werden, am Beispiel der Nutzung des LDIR-Befehls einen Weg zur Einstimmung auf die Programmierung in Maschinensprache darzustellen. Dabei werden Erfahrungen vor allem aus der Arbeit mit Schülern an Kleincomputern zusammengefaßt. Die Mehrzahl der Schüler verfügte über Kenntnisse aus der Arbeit mit BASIC. Es muß auch betont werden, daß der dargestellte Weg sich vordergründig auf eine »Einstimmung« und Nutzung der Elemente in BASIC-Programmen orientiert (vgl. z. B. [1]) und nicht eine exakte und umfassende Einführung das Anliegen ist.

2. Der LDIR-Befehl

Nachdem man sich über den Aufbau eines Mikroprozessors prinzipiell informiert und seine Registerstruktur kennengelernt hat, wird man sich einzelnen Befehlen zuwenden. Dabei werden Rechenoperationen, Transport-, Sprung- und Unterprogrammbefehle im Mittelpunkt des Interesses stehen. Dabei wird sehr schnell die zentrale Bedeutung des Registers A als Akkumulator deutlich. So erfordert das Füllen einer Speicherzelle (z. B. 60416, d. h. 0EC00H) mit einem Wert (z. B. 32, d. h. 20H) die Befehlsfolge:

```
LD A, 20H
LD (0EC00H), A
```

Für den ungeübten Nutzer entsteht nun die Frage nach dem erreichten Effekt. Beim KC 85/1 würde damit in der ersten Zeile und Spalte ein Leerzeichen auf dem Bildschirm ausgegeben werden. Durch die Verwendung des LDIR-Befehls (vgl. [2], S. 130f.) könnte ein wesentlich größerer Effekt – z. B. das Löschen des gesamten Bildschirmes – erzielt werden.

Für die Anwendung des LDIR-Befehls

wird im Register HL die Anfangsadresse und in BC die Blocklänge des Bereiches abgelegt, der dann zu der in DE gespeicherten Adresse transportiert werden soll. LDIR bewirkt dann das wiederholte Laden der Speicherzellen ab »DE« mit den Werten ab »HL«, wobei die Adressen immer wieder inkrementiert (also um 1 vergrößert) werden, und zwar so oft, wie der Wert in BC angibt. Damit bewirkt die Befehlsfolge

```
LD HL, 0100H
LD DE, 0200H
LD BC, 0010H
LDIR
```

das Laden der Speicherzelle 0200H mit dem Wert aus 0100H, 0201H mit 0101H usw. bis zu 020FH mit 010FH.

3. Anwendung des LDIR-Befehls

Durch die Anwendung des LDIR-Befehls auf Bereiche des Bildwiederholspeichers lassen sich die Wirkungen und auch die Schnelligkeit dieses Maschinenprogrammteils gut erkennen. So bewirkt z.B. das folgende Programm das Kopieren der oberen Bildschirmhälfte in die untere.
(Durch die unterschiedliche Speicherorganisation des KC 85/2 und KC 85/3 bzw. des KC 85/1 und des KC 87 ergeben sich Abweichungen, die im folgenden durch zweifache Darstellung der Programme – links für den KC 85/2 und KC 85/3 sowie rechts für den KC 85/1 und KC 87 – berücksichtigt werden sollen.)

```
LD HL, 08000 H      LD HL, 0EC00 H
LD DE, 093FF H      LD DE, 0EDE0 H
LD BC, 01400 H      LD BC, 001E0 H
LDIR                LDIR
```

Wenn man die erste Speicherzelle des Bildwiederholspeichers mit 20H (d.h. 32, also dem ASCII-Code des Leerzeichens) und dann jede Speicherzelle mit dem Inhalt ihres Vorgängers lädt, so entsteht die Wirkung des Bildschirm-Löschens. Das Programm lautet dafür:

```
LD A, 20 H          LD A, 20 H
LD (08000 H), A     LD (0EC00 H), A
LD HL, 08000 H      LD HL, 0EC00 H
LD DE, 08001 H      LD DE, 0EC01 H
LD BC, 027FF H      LD BC, 003BF H
LDIR                LDIR
```

Wenn man dieses Ergebnis z.B. mit CLS im BASIC oder SHIFT HOME vergleicht, so muß man aber beachten, daß dort nicht nur der Bildwiederholspeicher – allerdings mit 0 – einheitlich gefüllt wird. Es wird der Farbattributspeicher einheitlich geladen, und die durch das aktuelle Fenster gegebenen Grenzen werden beachtet.
Durch die Veränderung von LD A, ... kann nun auch erreicht werden, daß der Bildschirm einheitlich mit einem Zeichen, dessen ASCII-Code eingesetzt wurde, beschrieben wird.

4. Realisierung von Maschinenprogrammen für den KC 85/2

In [3] werden Speicherbereiche benannt, die vom Anwender genutzt werden können. Weiterhin kann der Anwender durch die Beantwortung der Frage MEMORY END?: nach dem Kaltstart des Interpreters den für BASIC-Programme zur Verfügung stehenden Speicherbereich begrenzen und den folgenden Bereich bis zum Ende des Anwender-RAM für Maschinenprogramme oder die Abspeicherung von Daten (z.B. Bildschirminhalten) nutzen. Des weiteren gibt es noch verschiedene Möglichkeiten der Integration von Maschinenprogrammteilen in BASIC-Programme. Verwiesen sei hier z.B. auf [4].
Wir wollen uns zunächst auf die Nutzung des Bereiches von 0 bis 15FH für das Maschinenprogramm beschränken.

Falls kein Editor/Assembler zur Verfügung steht, so müssen wir folgende Arbeitsschritte durchlaufen:

a) Erarbeitung des Maschinenprogramms

b) Übersetzung der Mnemoniks in den Objektcode

c) Eingabe des Objektcodes mit MODIFY in die Speicherzellen

Das Programm kann dann aus dem BASIC mit CALL . . . aufgerufen werden. Dabei ist die Adresse dezimal einzugeben. Im Beispiel bedeutet dies:

Mnemonik	*Objektcode*
LD A, 20 H	3E 20
LD (08000 H), A	32 00 80
LD HL, 08000 H	21 00 80
LD DE, 08001 H	11 01 80
LD BC, 027FF H	01 FF 27
LDIR	ED B0

Dabei ist der Abschluß des Maschinenprogramms noch durch RET, also Return erforderlich.

Falls man nun versucht, ein solches Programm zu starten, so wird man enttäuscht sein. Da alle Steuerfunktionen des Computers über den Mikroprozessor und dessen Register laufen, werden durch die Ladebefehle wichtige Informationen für die weitere Arbeit nach dem Return zerstört. Es kommt also darauf an, diese zu retten. Die Maschinenprogrammierung unterstützt dies durch PUSH und POP. PUSH »drückt« den Wert des entsprechenden Registerpaares in den LIFO-Speicher und POP bringt den obersten Wert des LIFO-Speichers in das entsprechende Registerpaar. Dabei bedeutet LIFO-Speicher, daß der zuletzt eingespeicherte Wert als erster bei der Entnahme zur Verfügung steht, also last-in-first-out-Speicher. Damit muß das Maschinenprogramm durch

PUSH AF	F5
PUSH BC	C5
PUSH DE	D5
PUSH HL	E5

und

POP HL	E1
POP DE	D1
POP BC	C1
POP AF	F1
RET	C9

»eingerahmt« werden.

Auch wenn das Programm nun »fehlerfrei läuft«, so ist der Effekt auf dem Bildschirm nicht sichtbar. Die Ursache liegt wiederum in der Speicherorganisation. Im BASIC wird der Zugriff zu den Speicherzellen von 08000 H ab durch VPEEK und VPOKE ermöglicht, wobei durch das V das »Zuschalten« des IRM gekennzeichnet wird. Für uns heißt das, daß unser Maschinenprogramm nicht »läuft«, weil der IRM weder beschrieben noch gelesen werden kann. Dazu muß nach den PUSH-Befehlen und vor den POP-Befehlen noch mit Hilfe von Aufrufen im Betriebssystem enthaltener »Unterprogramme« durch

CALL 0F018 H	CD 18 F0

bzw.

CALL 0F01B H	CD 1B F0

der IRM zu- bzw. abgeschaltet werden. Damit ist das Maschinenprogramm aus dem BASIC aufrufbar. Falls ein Aufruf aus dem CAOS gewünscht wird, so muß nur der bei den KC übliche Vorsatz vorangestellt werden. Dazu ist der Prolog 7F 7F, der Name unter dem das Programm aufgerufen werden soll, und der Epilog 01 notwendig. Wenn man also im CAOS-Menü das Wort CLS einfügen will und dies die obige Wirkung haben soll, so muß man vor den PUSH-Befehlen folgendes eingeben:

7F	Prolog
7F	
43	C
4C	L
53	S

01	Epilog
...	Maschinenprogramm
C9	Return

5. Realisierung von Maschinenprogrammteilen vom BASIC aus

Oftmals ist es sinnvoll, das Maschinenprogramm unmittelbar in der Arbeit mit BASIC zu erstellen und zu nutzen.

Dazu kann man z.B. folgenden Weg gehen:

a) Nach dem Kaltstart des BASIC-Interpreters wird ein Speicherbereich reserviert (s. unter 4.).

b) Das Maschinenprogramm wird in Mnemoniks erarbeitet, mit Hilfe von Tabellen in Hexadezimalzahlen übersetzt und dann in Dezimalzahlen umgerechnet.

Beispiel:

Mnemoniks	*Objektcode*	*Dezimalcode*	*RAM-Adressen*
DEFW prolog	7F 7F	127 127	000 001
DEFM 'CLS'	43 4C 53	67 76 83	002 003 004
DEFB epilog	01	1	005
PUSH AF	F5	245	006
PUSH BC	C5	197	007
PUSH DE	D5	213	008
PUSH HL	E5	229	009
CALL 0F0 18 H	CD 18 F0	205 24 240	00A 00B 00C
LD A, 20 H	3E 20	62 32	00D 00E
LD (08000 H), A	32 00 80	50 0 128	00F 010 011
LD HL, 08000 H	21 00 80	33 0 128	012 013 014
LD DE, 08001 H	11 01 80	17 1 128	015 016 017
LD BC, 027FF H	01 FF 27	1 255 39	018 019 01A
LDIR	ED B0	237 176	01B 01C
CALL 0F01B H	CD 1B F0	205 27 240	01D 01E 01F
POP HL	E1	225	020
POP DE	D1	209	021
POP BC	C1	193	022
POP AF	F1	241	023
RET	C9	201	024

c) Mit Hilfe von DATA-Zeilen und READ kann dann das Maschinenprogramm eingelesen werden.

Beispiel:

```
... DATA 127, 127, 67, 76, 83, 1, 245, 197, 213, 229, 205, 24, 240
... DATA 62, 32, 50, 0, 128, 33, 0, 128, 17, 1, 128, 1, 255, 39
... DATA 237, 176, 205, 27, 240, 225, 209, 193, 241, 201
... FOR I=0 TO 36
... READ Z
... POKE I, Z
... NEXT
```

(Durch die Angabe eines Offsets in POKE I + ..., Z kann das Maschinenprogramm auch an andere Stellen des RAM geladen werden.)

d) Nach dem Start des BASIC-Programms kann unser Maschinenprogramm dann mit CALL 6 aufgerufen werden. Die ersten sechs Byte dienen der Namensfestlegung für den nun

ebenfalls möglichen Aufruf aus dem CAOS mit CLS ENTER.
(Falls man im obigen Programm POKE I + 5000, Z verwendet hat, so erfolgt der Aufruf aus dem BASIC nun mit CALL 5006.)

6. Anregungen zur weiteren Nutzung

Mit Hilfe der betrachteten Elemente der Programmierung in Maschinensprache lassen sich bereits wirksam BASIC-Programme »verbessern«, d.h. besonders schneller und damit effektiver gestalten. So ist es keine Schwierigkeit, »bewegte Bilder« zu erzeugen, indem man z.B. mit Hilfe eines BASIC-Programms einzelne Bilder erzeugt, mit einem Maschinenprogramm in dem durch MEMORY END reservierten Bereich abspeichert und dann in schneller Folge wiederum mit Maschinenprogrammen auf den Bildschirm holt. Dabei wird in jedem Fall nur das obige Programm verwendet, in dem die Inhalte der Register HL (Quelle), DE (Ziel) und BC (Länge) variiert werden müssen.
Bei geeigneter Wahl der Registerinhalte und ggf. unter Verwendung von LDDR – für LDIR – kann man das betrachtete Programm auch benutzen, um den Bildschirminhalt nach unten, links und rechts zu »scrollen«. (Dabei muß aber die Organisation des Pixel-RAM beachtet werden.)

Durch die Anwendung des Programms auf den RAM-Bereich des Farbattributspeichers lassen sich weitere interessante Effekte erreichen.
Die Betrachtungen konnten nur schlaglichtartig einzelne Aspekte der Maschinenprogrammierung streifen und sind als Anregung und mögliche Hilfe zur Einarbeitung gedacht. Dabei ist es selbstverständlich, daß weitere Möglichkeiten (z.B. das Nutzen der Unterprogramme des Betriebsystems im »direkten Dialog« über MODIFY und das CAOS) bestehen und unter entsprechenden Zielstellungen effektiv zur Einarbeitung nutzbar sind.

Literatur

[1] Hamm, G.: Programmverbesserung durch „Tricks". – In: Mikroprozessortechnik 2 (1988)
[2] Barthold, H.; Bäurich, H.: Mikroprozessoren – Mikroelektronische Schaltkreise und ihre Anwendung (Teil 1). – In: elektronica, Band 222/223. – Berlin: Militärverlag, 1985
[3] Kirves, K.-D.: Modul M027 Development. – In: Mikroprozessortechnik 1 (1987) 8. – S. 247—249
[4] Kirves, K.-D.: Maschinenprogramme. – In: Mikroprozessortechnik 1 (1987) 10. – S. 317—318

Autor:
Dr. rer. nat. Gerhard Hamm

Zum Beitrag „Einlesen von Kassettendateien der Commodore-Computer am KC 87"

Literatur

[1] Betriebssystem KC 85/1. – VEB Robotron-Meßelektronik „Otto Schön". – Dresden, 1986
[2] Völz, H.: Subroutinen des BASIC-Interpreters von den KC-Rechnern. – In: Mikroprozessortechnik, Heft 6. – Berlin, 1987
[3] Besenthal, W.; Muus, J.: Alles über den plus 4. – Haar, 1986

Römische Zahlen mit dem KC 85/1

Das Programm verwendet zur Umwandlung von dezimalen in römische Zahlen und umgekehrt im wesentlichen die Arbeit mit Strings.
In Zeile 1 wurde das schnelle CLS eingefügt. Dazu wurde in BASIC nach der Zeilennummer ein .! und 70mal der Doppelpunkt eingegeben. Daran ist das Programm laut Listing anzufügen. Mit einem geeigneten Monitorprogramm ist dann von 406H bis 419H (1030 bis 1049 dezimal) das Maschinenprogramm (vgl. Bild 1) einzugeben.

```
>>> HEXMONITOR <<<
Startadr.(hex.): 400
Endadr.(hex):    44F
mit Pruefsumme
0400   00 4D 04 01 00 9C 11 3A -->139
0408   21 01 EC 2B 11 01 EC 01 -->238
0410   C0 03 36 20 ED B0 0E 12 -->2D6
0418   16 01 1E 01 C3 14 F3 C9 -->2C9
0420   3A 3A 3A 3A 3A 3A 3A 3A -->1D0
0428   3A 3A 3A 3A 3A 3A 3A 3A -->1D0
0430   3A 3A 3A 3A 3A 3A 3A 3A -->1D0
0438   3A 3A 3A 3A 3A 3A 3A 3A -->1D0
0440   3A 3A 3A 3A 3A 3A 3A 3A -->1D0
0448   3A 3A 3A 3A 00 6B 04 02 -->159
OK
>
```

Bild 1. Hexdump für CLS

Sollte man darauf verzichten wollen, ist im Programm statt CALL 1032 jeweils CLS zu verwenden.

Umwandlung dezimale in römische Zahlen

Um möglichst elegant arbeiten zu können, werden zunächst größere Zahlen als 999 extra umgewandelt. Dabei wird bei Zahlen ab 4000 die Anzahl der »M« als dezimales Vielfaches ausgegeben.
Dann ist es möglich, mit einer Schleife die nächste Ziffer abzuspalten und umzuwandeln.
In Zeile 70 wurde im ersten Befehl .0001 addiert, da sich zeigte, daß der Computer sonst die Ziffer 9 nicht erkennt.

Umwandlung römische in Dezimalzahlen

In Zeile 190 wird zu Z$ »RR« addiert, um die folgende Schleife für alle Werte benutzen zu können.
Es wird von »M« bis »I« untersucht, ob die römische Zahl vorkommt (Schleife mit I, Wert ist P).
Kommt sie vor, wird untersucht, ob die nächste und übernächste vor ihr steht (römisch 4 und 9, Werte R und S) bzw. wie oft sie vorkommt (Schleife mit K).
Fehler in römischen Zahlen bei der Eingabe werden vom Programm nicht erkannt.
Auf S. 46 steht ein Programm zur Umwandlung von römischen in Dezimalzahlen.
Ich wünsche allen Anwendern viel Erfolg!

```
>LIST
 5 !  Umwandlung roemische in Dezimalzahlen ----------------------
 10 CALL1032:CL=1032:CLEAR:GOTO290
 20 !DEZ.->ROEM.
 30 L=LEN(Z$):RO$="":D=0
 40 IFL>5THENRO$=LEFT$(Z$,L-3)+"*M":Z$=MID$(Z$,L-2)
 50 D=VAL(Z$):K=3:IFD>999THENGOSUB150
 55 IFINSTR("*",RO$)<>0ANDD<>0THENRO$=RO$+"+"
 60 FORI=2TO0STEP-1:C$="I"
 70 T=INT(D/10^I+.0001):IFT=0THENK=K+2:NEXT:GOTO140
 80 D=D-T*10^I
 90 IFT=9THENC$=R$(K)+R$(K-2):GOTO130
 100 IFT=4THENC$=R$(K)+R$(K-1):GOTO130
 110 IFT>=5THENC$=R$(K-1):T=T-5:IFT=0THENGOTO130
 120 C$=C$+STRING$(T,R$(K))
 130 RO$=RO$+C$:K=K+2:NEXT
 140 PRINT"roemisch:";RO$:RETURN
 150 T=INT(D/1000)
 160 IFT>3THENRO$=STR$(T)+"*M":ELSERO$=STRING$(T,"M")
 170 D=D-T*1000:RETURN
 180 !roem.->dez.
 190 Z$=Z$+"RR":Q=0
 200 FORI=1TO7:FORK=0TO1STEP0
 210 L=LEN(Z$):P=INSTR(R$(I),Z$)
 220 IFL=1ORP=0THENK=1:NEXT:NEXT:GOTO280
 230 R=INSTR(R$(I+1),Z$):S=INSTR(R$(I+2),Z$):T=1
 233 IFR=0THENR=L+1
 237 IFS=0THENS=L+1
 240 IFP>RTHENT=0.8:L=L-1:K=1
 250 IFP>STHENT=0.9:L=L-1:K=1
 260 Q=Q+T*A(I):Z$=RIGHT$(Z$,L-1)
 270 NEXT:NEXT
 280 PRINT"dezimal:";Q:RETURN
 290 FORI=1TO9:READA(I),R$(I):NEXT :PRINT"UMWANDLUNGEN"
 300 PRINTAT(3,2);"1 - dezimal -> roemisch"
 310 PRINTAT(5,2);"2 - roemisch -> dezimal":PRINTAT(7,2);"3 - ENDE"
 320 PRINTAT(10,2);"AUSWAHL":POKE DEEK(45),32
 330 I$=INKEY$:IFI$=""THEN330:ELSEN=VAL(I$)
 340 IFN=3THENCALL1032:END
 350 CALL1032:IFN<1ORN>3THENRUN
 360 IFN=1THENPRINT"dezimal:";:ELSEPRINT"roemisch:";
 370 INPUTZ$
 380 ONNGOSUB30,180,430
 390 PRINT:PRINT"NOCHMALS(J/N)"
 400 I$=INKEY$:IFI$=""THEN400
 410 IFI$="J"THEN340:ELSERUN
 420 DATA 1000,"M",500,"D",100,"C",50,"L",10,"X",5,"V",1,"I",0,"R",0,"R"
 430 END
OK
>
```

Autor:
Diplomlehrer Volker Pöschel
Pädagogischer Mitarbeiter im
Haus der Jungen Pioniere „Bruno Kühn"
Gotha

Drucken von pseudographischen Bildern

Moderne Nadeldrucker bieten heute allgemein die Möglichkeit der graphischen Ausgabe. Bei wenigen Typen ist direkt ein dem jeweiligen Rechner angepaßter pseudographischer Zeichensatz bereits im Drucker verfügbar (z. B. bei Druckern für Commodore-Rechner) bzw. er läßt sich über bestimmte Steuerfolgen in den Zeichengenerator des Druckers laden.
Die in diesem Artikel getroffenen Aussagen beziehen sich auf die Druckertypen K6311 und K6312. Die Weiterentwicklungen ab K6313 lassen sich prinzipiell in gleicher Weise ansteuern. Sie besitzen jedoch den Vorteil, daß sich sowohl Druckkopf als auch die Druckerwalze wesentlich feiner positionieren lassen, wodurch sich einige programmtechnische Probleme einfacher gestalten.

1. Technische Voraussetzungen

Die wichtigste Voraussetzung ist die Graphikfähigkeit des Druckers, welche beim K6311 durch die Einzelnadelsteuerung gegeben ist. Bei diesem Druckertyp ist es möglich, acht der neun Nadeln des Druckkopfes über ein Byte anzusprechen. Es erfolgt damit der gemeinsame Anschlag von maximal acht Punkten. welche übereinander stehen. Das Weiterschalten der Spaltenposition erfolgt automatisch nach jedem Anschlag, d. h., ein Drucken am Ort für Doppelanschlag ist nicht möglich. Die Zeilenschaltung kann über Linefeed erfolgen, was zwölf Punkten entspricht. Da sich bei diesem Kommando jeweils 4 nicht erreichbare Punktreihen ergeben, muß der Feinschritt genutzt werden, der sechs Punkte weiterschaltet. Damit ist es nur sinnvoll, mit einem Wagenlauf sechs Linien zu drucken. Die zweite technische Voraussetzung ist ein rücklesbarer Zeichengenerator im Bildschirmsystem. Dies ist notwendig, da das Kopierprogramm vom Bildwiederholspeicher nur den Zeichencode bekommt. Es muß anschließend also möglich sein, auf die diesem Zeichencode entsprechende Punktmatrix zuzugreifen, um sie für den Drucker aufzubereiten. Diese Möglichkeit kann auch dadurch geschaffen werden, daß der Zeichengenerator im Arbeitsspeicher des Rechners als Duplikat abgelegt wird.

2. Ansteuerung des Druckers

Am günstigsten für die Druckeransteuerung wäre die zeichenweise Aufbereitung und Ausgabe. In diesem Fall wäre auch ohne Probleme ein Druck von Bildern, die größer als das Bildschirmformat sind, aus dem Programm heraus möglich. Dieser Variante stehen beim K6311 jedoch zwei Gründe entgegen:

a) Die Bildschirmzeichen haben ein Punktformat von 8 × 8 oder 8 × 6. Der Drucker realisiert bei größeren Bildern aber nur 6 × *N* Punkte. Damit müßten die einzelnen Positionen zweimal angelaufen werden und eine Verrechnung der überlappenden Teile erfolgen.

b) Der Drucker führt nach jedem Aufruf des Graphikmodus eine Kopfbewegung als Sichtautomatik durch. Diese blockiert eine Zeit von 1 bis 2 Sekunden, was bei einer Zeile bereits über eine Minute ergibt. Aus diesem Grund sollte man eine komplette Zeile mit einem Ruf ausdrucken, wodurch der Drucker mit maximaler Geschwindigkeit arbeitet.

Der Aufruf des Graphikmodus ist relativ einfach. Er besteht aus der Kennzeichnung, der Auswahl und der Länge des Graphikstrings. Der Drucker nimmt dann innerhalb dieses Modus keine anderen Steuerkommandos mehr an, was vor allem bei Programmfehlern zu seltsamen Erscheinungen führen kann. Die Bytekette für eine Ausgabe hat, in Assembler geschrieben, folgendes Aussehen:

```
TB1:  DB 1 BH              ;Kennzeichen Steuerbefehl
      DB 4 BH              ;Auswahl Graphikmode
      DA TB1E—#—1          ;Länge der Bytefolge
      DB 1. Punktspalte
      :
      :
TB1E: DB letzte Punktspalte
```

3. Druckertreiber für Pseudographik-Hardcopy

Mit diesem Programm sollte ein Bildschirmabzug bei Verwendung von pseudographischen Zeichen realisiert werden. Der Bildschirm hat im konkreten Fall zwei Betriebsarten, welche sich in der Zeilenzahl unterscheiden. Bei Pseudographik arbeitet er mit 32 × 32 Zeichen.
Wenn man die bisherigen Aussagen für den Druckertreiber zusammenfaßt, ergeben sich für das Copy-Programm folgende Randparameter:

a) Es müssen jeweils ganze Zeilen eingelesen und gedruckt werden.

b) Im Treiber sind drei Zeilen für die Umrechnung auf sechs Punktlinien erforderlich, welche als vier Druckzeilen ausgegeben werden. Es ist somit eine Puffermatrix von 256 × 3 Byte (= 256 × 24 Punkte) aufzubauen.

c) Die Elemente des Zeichengenerators sind um 90° zu drehen und zu spiegeln. Diese Maßnahme ist erforderlich, da der Zeichengenerator des Bildschirms in einem Byte eine Punktzeile speichert, der Drucker aber eine Punktspalte benötigt.

Diese Aktivitäten wiederholen sich für einen Bildschirmabzug mehrmals, so daß sich eine Anzahl ineinander verschachtelter Schleifen ergibt (Bild 1). Das Programm, welches auf Seite 51 ff. vollständig wiedergegeben wird, ist trotz der relativ großen Befehlsmenge in einem Durchlauf noch so schnell, daß es den Drucker nicht merklich behindert. Die insgesamt resultierende Druckgeschwindigkeit entspricht der eines Graphikausdruckes und beträgt etwa 90 Sekunden.
Bei dieser Bildgröße ist es noch möglich, den Maßstab zu ändern, da die Punktzahl einer Linie bei A4-Format etwa 600 Punkte beträgt. Bei einer Verdopplung des Bildes ergibt sich demzufolge ein formatfüllender Ausdruck einschließlich einiger Randzeichen.

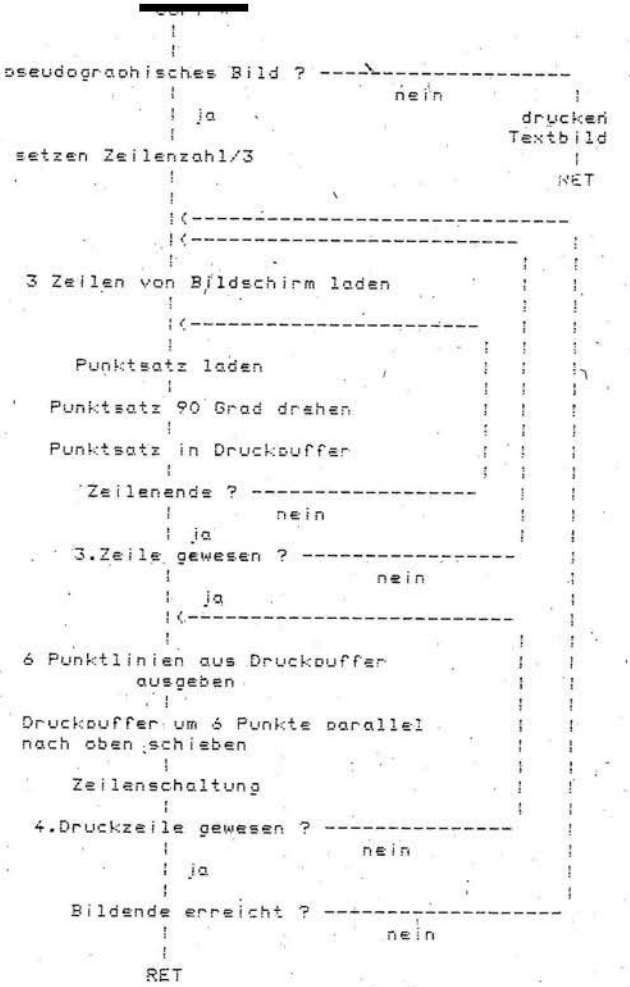

Bild 1. Ablauf des pseudographischen COPY

4. Druck maßstabgerechter Zeichnungen

Für technische Aufgabenstellungen, wie zum Beispiel beim Ausdruck von Leiterplattenentwürfen, ist es erforderlich, daß das Bild einen exakten Maßstab besitzt. Für dieses konkrete Beispiel war dabei der absolute Maßstab untergeordnet, da das entstehende Bild auf fototechnischem Wege weiterbearbeitet wird. Es ergeben sich für die Ausgabe von Bildern für technische Zwecke neue Probleme, welche beim normalen Abzug untergeordnet sind. Das waren unter anderem:

a) Der absolute Druckmaßstab ist feststehend und kann nur in ganzzahligen Schritten vergrößert werden.

b) Der Nadeldurchmesser beträgt etwa 0,34 mm, wodurch man zu gebrochenen Verhältnissen gegenüber der normalerweise metrischen Darstellung kommt. So ergaben sich bei einem Leiterplatten-

system 1,25 mm Original (= 1 Kästchen zu 8 × 8 Punkten) etwa 2,75 mm im Bild, was einem Grundmaßstab von 2,2:1 entspricht.

c) Die Druckachsen besitzen beim K6311 nicht den gleichen Maßstab. Dabei kann noch hinzukommen, daß dieser Unterschied im Rahmen der Fertigungstoleranzen schwankt. Messungen ergaben ein Verhältnis zwischen den Achsen von 1:1,0435 für X:Y.

d) Die Konstanz der Achsenverhältnisse wird wesentlich vom Papierzug bestimmt.

e) Der Kontrast des Druckbildes reicht nicht für eine fototechnische Weiterverarbeitung.

f) Die Qualität des Farbbandes und des verwendeten Papiers haben starken Einfluß auf die Verwendbarkeit der Zeichnung.

An den ersten beiden Punkten läßt sich im Prinzip nichts ändern. Man erreicht den Ausdruck von Bildern mit kleineren Maßstäben lediglich durch Reduzierung der Bildpunktzahl mittels Informationsverdichtung.

Die Korrektur der Achsenverhältnisse erfolgt sinnvollerweise in der x-Achse, da sich diese am einfachsten beeinflussen läßt. Dazu wurde der Treiber so modifiziert, daß jeder 23. Punkt doppelt ausgegeben wird. Man bekommt dabei zwar geringfügige optische Fehler in das Druckbild, diese fallen aber kaum auf. Denkbar wäre auch eine Interpolation zwischen den benachbarten Punkten zur Bestimmung des Korrekturpunkts.

Der konstante Papierzug ist vor allem für ein in sich lineares Bild erforderlich. Die Schwankungen entstehen dabei durch das Spiel des Walzenantriebs und die unterschiedlichen Zugkräfte beim Abwickeln einer Rolle (Rucken der Rolle beim Weiterschalten) bzw. durch die unterschiedlichen Kräftevektoren bei der Abnahme vom Papierstapel. In der Praxis hat es sich bewährt, nur so viel Papier einzuspannen, wie für den Druck benötigt wird (einschließlich Vor- und Abspann). Dieses läßt man über die Tischkante herunterhängen und belastet es am Ende mit wenigen Gramm. Diese Belastung muß so gewählt werden, daß der Walzenantrieb nicht überlastet wird und das Papiergewicht eine untergeordnete Größe erreicht. Bewährt hat sich hier das Anbringen von zwei Krokodilklemmen (oder kleinen Klammern) am Blattanfang.

Die erforderliche Kontrasterhöhung erreicht man am einfachsten durch einen doppelten Anschlag der Punkte. Da der Drucker aber nach jedem Anschlag eine Position weiterrückt, muß jede Spalte zweimal angefahren werden. Dies erreicht man, indem die Zeile zweimal übereinander ausgegeben wird. Die Wiederholgenauigkeit ist dabei sehr hoch, da der Drucker bei Graphikausgabe keinen Vorwärts-Rückwärts-Druck ausführt, sondern immer nur in einer Richtung arbeitet.

Ein wesentlich schwierigeres Problem stellt die Papierauswahl in Kombination mit einem geeigneten Farbband dar. Es muß dabei erreicht werden, daß die Druckfarbe in sich minimal verläuft, aber nicht zur Veränderung der Konturen führt. Wählt man ein zu trockenes Farbband und sehr glattes Papier, so sieht man die Konturen der einzelnen Nadeln, wogegen bei zu feuchten Farbbändern die Konturen verlaufen. Auf Transparentpapier zum Beispiel entsteht nach etwa 10 Minuten ein die Linie um 100 % ihrer eigenen Ausdehnung überschreitender grauer Rand, der durch die Trägeröle des Farbbandes gebildet wird. Von Bedeutung ist auch die Papierdicke, da es sich, wenn es zu dünn ist, durch die Trägeröle und die mechanische Bean-

spruchung des Druckes verzieht oder Wellen bildet. Zusammengefaßt ist eine experimentelle Optimierung zwischen:

- Saugfähigkeit des Papiers,
- Oberflächenbeschaffenheit des Papiers,
- Ölanteil im Farbband,
- Papierstärke

durchzuführen. In der Praxis hat sich gezeigt, daß bei Verwendung normaler Druckerfarbbänder auf Zeichenkarton eine ausreichende Qualität erreicht wird. Dem Optimum noch näher kommt Spezialtransparentpapier für Plotter, da es vor allem wesentlich dünner ist und für derartige Einsatzfälle ausgelegt wurde.

```
        PN      COPY
;
;  H A R D C O P Y - P G M
;
;       G R A P H I C    2 : 1
;
;
;---------------------------------------
;
START:  CALL    DSTR
        DB      4,7FH
        DB      0DH,0AH,20H
        JMP     ANF
;
;---------------------------------------
;
;  BILDSCHIRMPOSITION LESEN
CIN:    LD      A,B     ; ZEILENPOSITION
        OUT     1
        LD      A,C     ; SPALTENPOSITION
        OUT     0
        IN      3       ; DATEN LESEN
        RET
;
;.......................................
;
;  DRUCKER - INTERFACEPROGRAMM
;
;  OUT 60  DATEN (NEG.)
;  OUT 61  0 RUF
;          3 RESET
;  IN  61  0 ON/OFF (ON = 0)
;          1 /END
;          2 /FEHLER (1 = ERROR)
;          3 BUSY (0 KEINE DATENANNAHME)
;
DRAUS:  PUSH    AF
        IN      61H
        AND     1
        JRNZ    DRA-*   ; OFF-LINE
;
DRB:    IN      61H
        AND     8
        JRZ     DRB-*   ; NICHT RDY
;
        POP     AF
        PUSH    AF
        AND     7FH
        CPL
        OUT     60H
        LD      A,1
        OUT     61H     ; RUF EIN
        XOR     A
        OUT     61H     ; RUF AUS
;
DRA:    POP     AF
        RET
;
;.......................................
;
;  AUSGABE VON ZEICHENKETTEN
;
DSTR:   EX      (SP),HL
        PUSH    BC
        LD      B,M     ; ANZAHL
        INC     HL
;
DSTRA:  LD      A,M
        INC     HL
        CALL    DRAUS
        DJNZ    DSTRA-*
;
        POP     BC
        EX      (SP),HL
        RET

;---------------------------------------
;
;  LADEN 1 BILDZEILE
;
;  (BC) NICHT ZERSTOEREN
;
BIZEI:  EXX
        LD      HL,LPUF
        LD      D,32
        LD      C,0     ; ZEILE IN B

BIZEA:  CALL    CIN
        AND     7FH
        LD      M,A
        INC     HL
        INC     C
        DEC     D
        JRNZ    BIZEA-* ; 1 ZEILE LESEN
;
        INC     B
        EXX
        RET
;
;
;.......................................
;
;  PUNKTSATZ LADEN
;
PULA:   EXX
        PUSH    BC
        LD      H,0
        LD      L,A
        ADD     HL,HL
        ADD     HL,HL
        ADD     HL,HL   ; * 8
        LD      DE,0F800H       ; ZGEN
        ADD     HL,DE
        PUSH    IX
;       LD      IX,HPUF
        LD      B,8
PULAB:  LD      A,M
        RLA
        RL      (IX)
        RLA
```

```
        RL      (IX+1)
        RLA
        RL      (IX+2)
        RLA
        RL      (IX+3)
        RLA
        RL      (IX+4)
        RLA
        RL      (IX+5)
        RLA
        RL      (IX+6)
        RLA
        RL      (IX+7)
        INC     HL
        DJNZ    PULAB-$
;
;  PUNKTMATRIX IN HPUF
        POP     IX
        POP     BC
        EXX
        RET
;
;.....%.....
;
;  NACHSCHIEBEN PUNKTKETTE  3 PUNKTE
;
NAPU:   EXX
        PUSH    BC
        LD      HL,ZPUF
        LD      B,0
NAPUA:  LD      C,3
NAPUB:  INC     H
        INC     H
        SLA     M
        DEC     H
        RL      M
        DEC     H
        RL      M
        DEC     C
        JRNZ    NAPUB-$ : 3 PUNKTE
;
        INC     L
        DJNZ    NAPUA-$ ; 256 SPALTEN
;
        POP     BC
        EXX
        RET
;
;.....
;
NL:     CALL    DSTR
        DB      9
        DB      0DH
        DB      1BH,5BH,31H,65H : 0.5 LF
        DB      1BH,5BH,35H,61H
        RET
;
;.....
;
;  3 BILDZEILEN AUSGEBEN
;
BLOKA:  CALL    BIZEI
        LD      HL,LPUF
        LD      B,32
;
BLOKB:  LD      A,M
        CALL    PULA
        PUSH    HL
        PUSH    BC
;
        LD      HL,HPUF
        LD      BC,8
        LDIR
;
        POP     BC
        POP     HL
        INC     HL
        DJNZ    BLOKB-$
        RET
;
;
BLOK:   LD      DE,ZPUF
        CALL    BLOKA
        CALL    BLOKA
        JR      BLOKA
;
;.....
;
;  LISTEN 1 ZEILE ( 3 PUNKTE )
;
DUMP:   CALL    NL
;
        PUSH    BC
        LD      HL,ZPUF
        LD      B,0
        LD      A,8
DUMPC:  OR      M
        JRNZ    DUMPB-$
        INC     HL
        DJNZ    DUMPC-$
        POP     BC
        RET
;
;  ZEILE BELEGT
DUMPB:  POP     BC
        CALL    DSTR
        DB      4
        DB      1BH,4BH
        DA      512
;       PUSH    BC
        LD      B,0
        LD      HL,ZPUF
;
DUMPA:  LD      C,M
        XOR     A
        BIT     7,C
        JRZ     4
        OR      060H
        BIT     6,C
        JRZ     4
        OR      18H
        BIT     5,C
        JRZ     4
        OR      06H
;
        CALL    DRAUS
        CALL    DRAUS
        INC     HL
        DJNZ    DUMPA-$
;
        POP     BC
        RET
;
;-----------------------------------
;
;  GRAPHISCHE AUSGABE
;
ANF:    EQU     $
GRAPH:  EXX
        LD      BC,0
        EXX
;
        CALL    NL      ; CR-LF
;
        LD      B,10
GRP1:   PUSH    BC
        CALL    BLOK
        LD      C,8
```

```
GRP2:   CALL    DUMP
        CALL    NAPU
        DEC     C
        JRNZ    GRP2-*
;
        POP     BC
        CALL    BREAK   : TASTATURSTATUS
        CMP     3
        JRZ     GRP6-*  : ABBRUCH
        DJNZ    GRP1-*  : ZEILE 0...29
;
        CALL    BLOK
        LD      HL,ZPUF+512
        LD      B,0
GRP3:   LD      M,0
        INC     HL
        DJNZ    GRP3-*
;
        LD      B,6
GRP4:   CALL    DUMP
        CALL    NAPU
        DJNZ    GRP4-*  : ZEILE 30...31
;
;
GRP6:   CALL    DSTR
        DB      6
        DB      0DH,0CH,0AH
        DB      0AH,0AH,20H
        RET
;
;
;*****************************************
;
;
ZPUF:   BER     770     : DRUCKPUFFER
HPUF:   BER     8       : PUNKTPUFFER
LPUF:   BER     32      : ZEILENPUFFER
;
;
;
        END
;
```

Autor:

Dr.-Ing. Gert Schönfelder

Informatik-Zentrum des Hochschulwesens
an der Technischen Universität Dresden

Hinweise für Autoren

Herausgeber und Verlag danken den Lesern für das Interesse an den »Kleinstrechner-TIPS«, das sich in zahlreichen Zuschriften und Veröffentlichungsangeboten äußert. Beim Einsenden von Artikeln bitten sie folgende Hinweise zu beachten:

- In den »Kleinstrechner-TIPS« werden Artikel aus den auf der 4. Umschlagseite angegebenen Gebieten veröffentlicht.
- Manuskripte sind zweizeilig mit schwarzem Farbband mit Schreibmaschine zu schreiben, ä, ö und ü dürfen nicht durch ae, oe und ue ersetzt werden.
- Bilder sollten auf getrennten Blättern gezeichnet sein. Eine Bildunterschriftenliste ist beizufügen.
- Rechnerausdrucke (z.B. Programme) sollen tiefschwarz mit guter Ausnutzung des Formats (engzeilig, kein zu breites Kommentarfeld usw.) gedruckt sein.
- Die Autorenangabe soll enthalten: akadem. Grad, Vorname, Name, Tätigkeit, Arbeitsstelle, Privatanschrift.

Manuskripte mit einem Umfang von nicht mehr als 15 Schreibmaschinenseiten (einschließlich Bilder und Programme) sind in zwei Exemplaren an einen der Herausgeber zu senden (Anschrift s. S. 64).

Grafik mit dem KC 85/1

Obwohl das im KC 85/1 implementierte BASIC über keine Anweisungen zur Darstellung einzelner Bildpunkte verfügt, ergeben sich doch verschiedene Möglichkeiten für die Erzeugung von Grafiken. Im folgenden soll dies ausgeführt und diskutiert werden. Zur Veranschaulichung werden Listungs bzw. Hardcopy z.B. zur Darstellung von Kreisen und der Sinusfunktion mit Hilfe der verschiedenen Möglichkeiten wiedergegeben.

1. Quasigrafik

Der im KC 85/1 vorhandene Zeichensatz unterstützt besonders mit dem ASCII-Code von 128 bis 255 die Erzeugung von »Bildern«. Da je Bildschirmposition ein Zeichen und nicht 8 mal 8 Punkte gewählt werden müssen, ergeben sich damit zeitliche Vorteile. Da aber eine Berechnung der Zeichen i.allg. nicht sinnvoll möglich ist, müssen also 24 Zeilen mit jeweils 40 Zeichen, d.h. 960 Positionen, konkret definiert werden.

Beispiel:

```
Demonstrationsprogramm zur
Quasigrafik
M. Hamm, Kl. 2, Sept. 87
G. Reinhardt, Kl. 4, Sept. 87
```

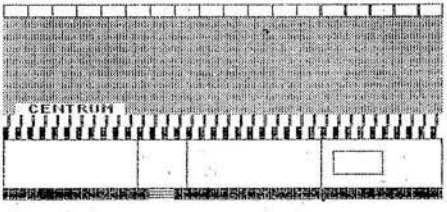

Bild 1 (linke Spalte)

2. Grobgrafik

Mit Hilfe der PRINT AT-Anweisung läßt sich eine »Grobgrafik« realisieren, die auf der Grundlage berechneter Werte die Darstellung z.B. einer Funktion ermöglicht. Dabei steht ein Raster von 40 Spalten (x-Koordinate) und 24 Zeilen (y-Koordinate) zur Verfügung. Selbstverständlich entstehen dabei keine »glatten« Kurven.

Beispiel:

```
1 ! Demonstrationsprogramm zur
2 ! Grobgrafik fuer Kreis
3 ! Dr. Hamm, Sept. 87
10 WINDOW 0,23,0,39:CLS
20 XM=15:YM=11:R=8:RH=R/SQR(2)
30 FOR X=0 TO RH
40   Y=INT(SQR(R*R-X*X)+.5)
50   PRINTAT(23-(YM+Y),XM+X);"*"
60   PRINTAT(23-(YM-Y),XM+X);"*"
70   PRINTAT(23-(YM-Y),XM-X);"*"
80   PRINTAT(23-(YM+Y),XM-X);"*"
90   PRINTAT(23-(YM+X),XM+Y);"*"
100  PRINTAT(23-(YM-X),XM+Y);"*"
110  PRINTAT(23-(YM-X),XM-Y);"*"
120  PRINTAT(23-(YM+X),XM-Y);"*"
130 NEXT
140 END
OK
```

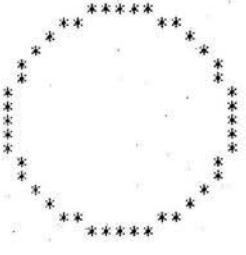

Bild 2

```
1 ! Demonstrationsprogramm zur
2 ! Grobgrafik fuer Sinusfunktion
3 ! Dr. Hamm, Sept. 87
10 WINDOW 0,23,0,39:CLS
20 FOR I=0 TO 39
30 PRINTAT(12-8*SIN(I/3)-.5,I);"*"
40 NEXT
50 END
OK
```

Bild 3

3. 1/16 – Grafik

Im ASCII-Code des KC 85/1 sind von 208 bis 223 Zeichen enthalten, die jeweils 2 mal 2 Punkte je Zeichenposition (8 mal 8 Punkte) ausgeben. Damit wird das 8 mal 8 Punkteraster in 16 Teile geteilt. Es gilt dabei folgende Zuordnung:

208	209	210	211
212	213	214	215
216	217	218	219
220	221	222	223

Die berechneten Werte können jetzt genauer in Bildschirmpositionen umgesetzt werden, da jetzt ein Raster mit 160 horizontalen (x-Koordinate) und 96 vertikalen (y-Koordinate) Unterteilungen verfügbar ist. Allerdings ist dies nur beschränkt nutzbar, da je Bildschirmposition nur eines der 16 Zeichen darstellbar ist.

Beispiel:

```
1 ! Demonstrationsprogramm zur
2 ! 1/16-Grafik am Bsp. Kreis
3 ! Dr. Hamm, Sept. 87
10 CLS
20 XM=100:YM=50:R=45:RH=R/SQR(2)+4
30 FOR XO=0 TO RH STEP 4
40 YO=INT(SQR(R*R-XO*XO)+.5)
50 X=XM+XO:Y=YM+YO:GOSUB 1000
60         Y=YM-YO:GOSUB 1000
70 X=XM-XO:        GOSUB 1000
80         Y=YM+YO:GOSUB 1000
90 X=XM+YO:Y=YM+XO:GOSUB 1000
100        Y=YM-XO:GOSUB 1000
110 X=XM-YO:       GOSUB 1000
120        Y=YM+XO:GOSUB 1000
130 NEXT
140 END
999  !
1000 ! UP PSET
1001 ! 0<= X <=159
1002 ! 0<= Y <= 95
1010 GY=INT(Y/4)
1020 RY=INT((Y/4-GY)*4)
1030 GX=INT(X/4)
1040 RX=INT((X/4-GX)*4)
1050 PRINTAT(23-GY,GX);CHR$(220-RY*4+RX)
1060 RETURN
OK
>
```

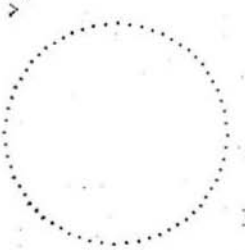

Bild 4

```
1 ! Demonstrationsprogramm zur
2 ! 1/16-Grafik am Bsp. Sinusfunktion
3 ! Dr. Hamm, Sept. 87
10 CLS
20 FOR X=0 TO 159
30 Y=INT(40*SIN(X/10)+45+.5)
40 GOSUB 1000
50 NEXT
60 END
999  !
1000 ! UP PSET
1001 ! 0<= X <=159
1002 ! 0<= Y <= 95
1010 GY=INT(Y/4)
1020 RY=INT((Y/4-GY)*4)
1030 GX=INT(X/4)
1040 RX=INT((X/4-GX)*4)
1050 PRINTAT(23-GY,GX);CHR$(220-RY*4+RX)
1060 RETURN
OK
>
```

Bild 5

4. 2/16 – Grafik

Die 2/16-Grafik nutzt das gebotene Raster nur sehr unvollständig, da je Bildschirmposition nur eines der 16

theoretisch möglichen Zeichen dargestellt werden kann. Wenn es gelingt, auf einer Bildschirm-Position mindestens zwei Zeichen quasi gleichzeitig darzustellen, so würde dies in der Mehrzahl der Fälle bereits eine durchaus befriedigende Darstellung ergeben. Dies läßt sich erreichen, indem man zwei Bilder generiert, die aus den Mengen der beiden gewünschten Zeichen bestehen. Mit Hilfe eines Maschinenprogrammteils kann man jetzt die beiden Bilder abwechselnd darstellen, und für den Beobachter entsteht der Eindruck der scheinbaren Gleichzeitigkeit. (Eventuell kann die geringfügig stärkere Unruhe des Bildes als störend empfunden werden.)

Beispiel:

Da für die Ablage der beiden Bilder und des Maschinenprogrammteils ein entsprechender RAM-Bereich benötigt wird, wäre folgendes Vorgehen möglich:

- Start des BASIC-Interpreters
- Beantwortung von MEMORY SIZE?: durch die Eingabe von 4000 ENTER

  (Damit wird der RAM-Bereich für das Abspeichern des BASIC-Programmtextes begrenzt.)
- Eingabe und Start des folgenden Programms

```
10 GOTO 50000
10000 !
10004 CLS
10010 FOR X=0+II TO 39
10020 Y=10+9*SIN(X/3)
10030 GOSUB 11000
10040 NEXT
10050 RETURN
11000 !  *** PKT-SETZ-PROGRAMM ***
11010 !  X,Y --> PUNKT
11020 GY=INT(Y):GX=INT(X)
11030 RY=Y-GY  :RX=X-GX
11040 DY=INT(RY*4):DX=INT(RX*4)
11050 PRINTAT(23-GY,GX);CHR$(208+DX+4*(3-DY))
11060 RETURN
50000 POKE-4152,32:WINDOW:CLS
50010 PRINTAT(11,12);"Bitte warten !"
50020 !
50030 !   CLS
50040 DATA 245,197,213,229
50050 DATA 62,32,50,0,236,33,0,236
50060 DATA 17,1,236,1,192,3,237,176
50070 DATA 225,209,193,241,201
50080 !
50090 !  BS 1 ---> 3800 H
50100 DATA 245,197,213,229
50110 DATA 33,0,236,17,0,56,1,192,3
50120 DATA 237,176
50130 DATA 225,209,193,241,201
50140 !
50150 !  BS 2 ---> 3C00 H
50160 DATA 245,197,213,229
50170 DATA 33,0,236,17,0,60,1,192,3
50180 DATA 237,176
50190 DATA 225,209,193,241,201
50200 !
50210 !      ---> BS 1
50220 DATA 245,197,213,229
```

```
50230 DATA 33,0,0,34,221,54,1,192,3
50240 DATA 33,0,56,17,0,236,237,176
50250 DATA 1,192,3,33,0,60,17,0,236
50260 DATA 237,176
50270 DATA 33,221,54,52,194,251,54
50310 DATA 225,209,193,241,201
50320 !
50330 !   MP einlesen
50333 RESTORE 50040
50340 FOR I=0 TO 108
50350    READ Z
50360    POKE 14000+I,Z
50370 NEXT
50380 CALL 14000
50390 !
50400 !   Bilderzeugung
50410 II=0:GOSUB 10000
50420 CALL 14025
50430 CALL 14000
50440 II=.5:GOSUB 10000
50450 CALL 14045
50460 CALL 14000
50470 !
50480 !   Bildwiedergabe
50500 CALL 14065
50510 PRINTAT(22,10);"Wiederholung (J/N)?"
50520 A$="":A$=INKEY$:IFA$="J"THEN50500:ELSEIFA$="N"THEN50530:ELSE50520
50530 POKE-4152,16:CALL14000:END
OK
>
```

5. dynamische Bilder

Die konsequente Fortführung des Gedankens der Nutzung von Maschinenprogrammen zum schnellen Wechsel des Bildschirminhalts führt zwangsläufig zu der Frage, ob bzw. wie es möglich ist, »Trickfilme« mit Hilfe des Computers ablaufen zu lassen. Dabei benötigt jedes Einzelbild im Speicher genau 960 Byte. In Abhängigkeit vom Umfang des »Steuerprogramms« ergeben sich folgende Richtwerte für die Anzahl der Bilder für den KC 85/1 mit ROM-BASIC:

- ohne RAM-Expander rund 15 Bilder
- mit einem RAM-Expander rund 30 Bilder
- mit zwei RAM-Expandern rund 50 Bilder

Damit ist es also möglich, in Verbindung mit Maschinenprogrammen kurze »Tricksequenzen« mit Hilfe des KC 85/1 darzustellen.

Zusammenfassend kann man feststellen, daß der KC 85/1 über spezifische Möglichkeiten zur Erzeugung bildlicher Darstellungen verfügt. Bei geeigneter Wahl der Möglichkeiten kann man gezielt z.B. die laufzeitlichen Vorteile nutzen und damit wirkungsvolle Effekte erreichen.

Zu detailierteren Fragen sei u.a. auf die Literatur verwiesen.

Literatur

[1] Heimcomputer Z 9001. – Programmierhandbuch. – Dresden: Robotron-Meßelektronik „Otto Schön"

[2] Hamm, G.: LDIR – ein Weg zur Einstimmung auf die Programmierung in Maschinensprache. – In: Kleinstrechner-TIPS, Heft 13. – Leipzig; Fachbuchverlag, 1990

Autor:

Dr. rer. nat. Gerhard Hamm

FORTH –
Ein universelles Programmiersystem zum Entwurf schneller Computerspiele

Computerspiele üben auf Alt und Jung eine besondere Faszination aus. Im Rahmen der außerunterrichtlichen Tätigkeit werden sie bereits in vielfältiger Art und Weise eingesetzt. In allen Bereichen haben wir die Feststellung machen können, daß es bezüglich der Beschäftigungsdauer zwei Gruppen von Spielen gibt, solche, die lange Zeit interessant bleiben, und andere, welche nach einmaliger Beschäftigung ihren Reiz verlieren. Worin unterscheiden sich beide Spieltypen? Folgende Gründe lassen sich für das Interessantbleiben eines Spiels vermuten:

- Vorhandensein einer Bestenliste;
- sich dem Spieler anpassender Schwierigkeitsgrad (d.h. niedrige Einstiegsschwelle vermittelt Erfolgserlebnisse, dann Steigerung der Schwierigkeit);
- gute Grafik und Soundeffekte;
- originelle und fesselnde Spielidee;
- Schnelligkeit des Spielverlaufs u.a.

Aus letzterem Grund sind viele Spiele in Assemblercode geschrieben. Mit diesem Beitrag soll an Hand eines Beispiels gezeigt werden, daß auch die Programmiersprache FORTH dazu geeignet und auf Grund ihrer Konzeption geradezu dazu prädestiniert ist. Da sich beim Programmieren von Computerspielen viele Teilaufgaben wiederholen (z.B. die Bewegung von Spielfiguren, auch Player oder Sprites genannt), kann man sich in FORTH einen eigenen Wortschatz zur Lösung solcher Aufgaben definieren.

Als Programmierbeispiel nehmen wir folgendes *Spiel:*

Ein Ball soll an den Grenzen eines Rechtecks reflektiert werden. Am unteren Bildschirmrand befindet sich ein Schläger, welcher den Ball im Spiel halten soll. Er wird mit den Tasten (→) und (←) gesteuert. Folgende Überlegungen sollen die *Methodik des Programmentwurfs* in FORTH (Top-down-planen, Buttom-up-programmieren) verdeutlichen. Alle Speicheradressen beziehen sich auf den KC85/1 und CFORTH1.1 von Herrn Dr. BEIERLEIN, welches als Kassettenversion vorliegt [2]. Beginnen wir mit der Ballbewegung. Einen entsprechenden Algorithmus gibt das Struktogramm Bild 1 an, welcher als BASIC-Programm im Systemhandbuch des KC 85/1 realisiert ist. Dabei wird der PRINT AT-Befehl benutzt. Überlegen wir, wie man in FORTH einen ähnlichen Befehl programmieren kann, welcher durch Angabe der Zeile und Spalte den Ball zeichnet. Die Absolutadresse des Bildelements im Bildschirmspeicher ergibt sich nach folgender Formel:

ABSADR=BS+Z*ZZ+S

ABSADR-Absolutadresse im Bildschirmspeicher
BS – Beginn des Bildschirmspeichers
Z – Zeilennummer
S – Spaltennummer
ZZ – Zeichen je Zeile

Für den KC 85/1 bedeutet das konkret: ABSADR=−5120+Z*40+S.
Entwickeln wir ein Wort ABSADR, welches die Werte für S und Z auf dem Stack erwartet, die Absolutadresse berechnet und diese auf dem Stack zurückläßt. Das Wort ABSADR hat demnach folgende Stacknotation: (Z S−−ABSADR). Das Stackdiagramm in Bild 2 soll die Funktion des Wortes deutlich machen. Als Zahlenbeispiel wurde Z=4 und S=7 gewählt. Tragen wir dieses Wort auf SCR#1 ein und versehen ihn mit einer entsprechenden Kommentarzeile als Überschrift. Dazu geht man in CFORTH1.1 wie folgt vor.

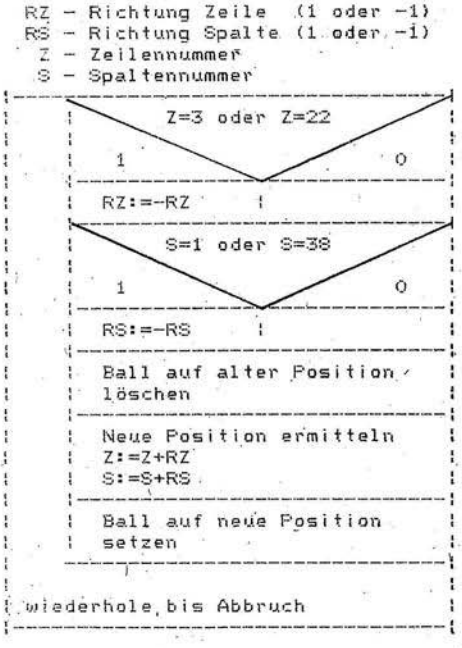

Bild 1. Struktogramm zur Ballbewegung

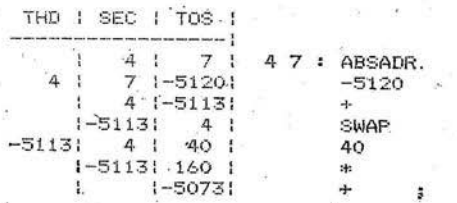

Bild 2. Stackdiagramm zum Wort ABSADR

```
( SCR#1==BALLMOVEMENT=========)

: ABSADR -5120 + SWAP 40 * + ;

                                ;S
```

Bild 3. Definition des Wortes ABSADR auf SCR # 1

EDITOR (ENTER)
1 CLEAR (ENTER)
1 EDIT (ENTER)

Nun beschreiben wir den Screen, wie es Bild 3 zeigt.
Das Wort ;S dient als Stopzeichen für den Compiler. Lassen wir nun unseren Screen compilieren

1 LOAD (ENTER)
und probieren das Wort aus
2 4 ABSADR . (ENTER)

Es ist auch möglich, die ermittelte Absolutadresse mit dem Wort FILL »weiterzuverarbeiten«. Es dient zum Füllen von Speicherbereichen und hat folgende Stacknotation
(anfang laenge asc-wert ——).

3 12 ABSADR 1 207 FILL (ENTER)

Somit ist es möglich geworden, den Ball über zwei Argumente zu steuern. Definieren wir als nächstes zwei Variablen für S und Z und zwei Wörter, welche diese Variablen abfragen und den Ball an der entsprechende Stelle setzen bzw. löschen. Bild 4 zeigt die entsprechenden Wörter dazu.

```
( SCR#1==BALLMOVEMENT=========)

22 VARIABLE Z  6 VARIABLE S
: ABSADR -5120 + SWAP 40 * + ;
: BSET   Z @ S @ ABSADR 1 207 FILL ;
: BRESET Z @ S @ ABSADR 1 32 FILL ;

                                ;S
```

Bild 4. Erweiterung des SCR # 1

Die alten Wörter kann man aus dem Wörterbuch mit

FORGET ABSADR (ENTER)

löschen, damit sich das Wörterbuch nicht unnötig durch Doppeldefinitionen aufbläht. Mit

1 LOAD (ENTER) wird die neue Variante von SCR#1 compiliert.

Nun können wir die neuen Wörter testen.

10 S!	(ENTER)	Variable S wird mit 10 belegt
12 Z!	(ENTER)	Variable Z wird mit 12 belegt
BSET	(ENTER)	Ball an dieser Position setzen
BRESET	(ENTER)	Ball an dieser Position zurücksetzen

Das Wort NEWARD hätte in der FORTH-Notation folgende Gestalt.

```
: NEWARD Z @ RZ @ + Z !
         S @ RS @ + Z ! ;
```

Auch für die Definition des Wortes RICHTUNG ist noch auf SCR#1 Platz.

```
: RICHTUNG Z @ 3 = Z @ 22 = OR
           IF RZ @ MINUS RZ ! THEN
           S @ 1 = S @ 38 = OR
           IF RS @ MINUS RS ! THEN ;
```

An Hand des Stackdiagramms in Bild 5 kann man sich die Funktionsweise des Wortes RICHTUNG verdeutlichen. Es setzt eine Belegung der Variablen Z mit 10

```
THD | SEC | TOS |
----------------|
    |     |     |     : RICHTUNG
    |     |  10 |     Z @
    |  10 |   3 |     3
    |     |   0 |     =       ( flag1)
    |   0 |  10 |     Z @
  0 |  10 |  22 |     22
    |   0 |   0 |     =       ( flag2)
    |     |   0 |     OR      ( OR-Verknüpfung
    |     |     |               beider flags
    |     |     |     IF
    |     |  -1 |      RS @
    |     |   1 |      MINUS
    |     |     |      RS !   ( 1 in RS speichern)
    |     |     |     THEN ;
```

Bild 5. Stackdiagramm zum Wort RICHTUNG

und RZ mit −1 voraus. Die gesamten Unterwörter zur Ballbewegung werden noch zu dem Wort BMOVE zusammengefaßt, so daß der erste Screen den in Bild 6 gezeigten Inhalt hat.

```
( SCR#1==BALLMOVEMENT======== )
22 VARIABLE Z   6 VARIABLE S
 1 VARIABLE RZ  1 VARIABLE RS
: ABSADR -5120 + SWAP 40 * + ;
: BSET   Z @ S @ ABSADR
                 1 207 FILL ;
: BRESET Z @ S @ ABSADR
                 1  32 FILL ;
: NEWADR Z @ RZ @ + Z !
         S @ RS @ + S ! ;
: RICHTUNG Z @ 3 = Z @ 22 = OR
       IF RZ @ MINUS RZ ! THEN
          S @ 1 = S @ 38 = OR
       IF RS @ MINUS RS ! THEN ;
: BMOVE RICHTUNG BRESET NEWADR
  BSET ;                  -->
```

Bild 6. Kompletter SCR # 1 mit den Wörtern zur Ballbewegung

Nun wird der Screen erneut compiliert. Probieren wir das Wort BMOVE aus.

```
: TEST BEGIN BMOVE ?TERMINAL UNTIL ; (ENTER)

TEST (ENTER)
```

Nachdem wir uns von dem Geschwindigkeitsvorteil, welchen FORTH zu bieten hat, überzeugt haben, schaffen wir eine etwas »sichtbarere« Testvariante.

: WAIT 100 0 DO LOOP ; (ENTER)
: TEST1 BEGIN BMOVE WAIT ?TERMINAL UNTIL ; (ENTER)

TEST 1 (ENTER)

Als nächstes soll ein Wort entwickelt werden, welches einen Tennisschläger in der unteren Bildschirmzeile (−4240 bis −4204) durch Betätigen der Tasten (←) und (→) bewegt. Der logische Aufbau des Wortes wird im Struktogramm in Bild 7 dargestellt.

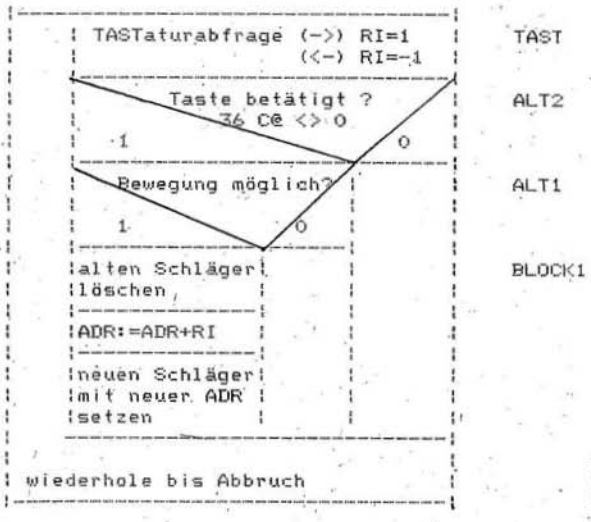

Bild 7. Struktogramm zur Schlägerbewegung

Da sich zur Eingabe das Wort KEY nicht eignet, denn es wartet jeweils auf eine Tastaturbetätigung, benutzt man die Speicherzelle 36, welche den Code für die zuletzt betätigte Taste enthält. Wird keine Taste betätigt, so steht hier eine Null. Man beginnt mit dem Wort BLOCK1, welches die Teilaufgaben Löschen des alten Spielers, Umschreiben der Adresse und Setzen des neuen Spielers realisiert. Die Alternative ALT1 prüft, ob eine weitere Bewegung des Schlägers möglich ist, d.h. ob gilt $-4240 < ADR < -4204$. Das Wort ALT1 realisiert diese Alternative. Alle Wörter, welche die Schlägerbewegung realisieren, stehen auf SCR#2, welcher in Bild 8 wiedergegeben ist.
2 LOAD (ENTER) compiliert auch diese Wörter ins Wörterbuch. Probieren wir sie im Zusammenspiel aus.
: TEST2 BEGIN BMOVE PLMOVE WAIT 36 C @ 3 = UNTIL ; (ENTER)

Erzeugt das Abbruchflag, wenn (STOP) gedrückt wird.

TEST 2 (ENTER)

Damit unser Ball auch mal ins Aus geraten kann, muß am unteren Spielfeldrand ein Test durchgeführt werden, ob sich unter dem Ball ein Schläger befindet. Wie erkennt man nun, daß sich der Ball am unteren Rand befindet? Eine mögliche

```
( SCR#2==PLAYERBEWEGUNG====== )

-4210 VARIABLE ADR 1 VARIABLE RI

: BLOCK1 ADR @ DUP 4  32 FILL
  RI @ + DUP ADR ! 4 127 FILL ;
: TAST 36 C@
   DUP 8 = IF -1 RI ! ELSE
   DUP 9 = IF  1 RI ! ELSE
             THEN THEN DROP ;
: ALT1 ADR @ RI @ + -4240 >
       ADR @ RI @ + -4204 < AND
       IF BLOCK1 THEN ;
: ALT2 36 C@ IF ALT1 THEN ;
: PLMOVE TAST ALT2 ;
                              -->
```

Bild 8. Kompletter SCR # 2 mit den Wörtern zur Schlägerbewegung

Überlegung ist folgende. Beginnt man mit der Ballbewegung in Zeile 21, so sind es immer 36 Schritte der Balles bis zur Rückkehr in Zeile 21. Durch eine Zählschleife bis 36 wird der Ball einmal durchs Spielfeld gebracht. Danach kann getestet werden, ob sich der Schläger unter dem Ball befindet.

```
: EINMAL 36 0 DO PLMOVE BMOVE WAIT LOOP ; (ENTER)
: TEST 4 BEGIN -1 RZ ! 21 Z ! EINMAL
  Z @ 1 + S @ ABSADR C @ 32 = UNTIL ;  (ENTER)
```

Erzeugt Abbruchflag

TEST 4 (ENTER)

Beim Testen des Spielverlaufs wünscht man sich, daß man den Schläger schneller bewegen könnte. Verraten wird auch hierzu eine mögliche Lösung. Der Zeitverlauf in der Hauptschleife läßt sich durch das Wort WAIT steuern, allerdings wird bei jedem Aufruf von BMOVE auch das Wort PLMOVE aufgerufen. Das Zeitverhältnis zwischen der Ausführung dieser beiden Wörter läßt sich einstellen, indem man nur bei jedem dritten Aufruf von PLMOVE das Wort BMOVE ausführen läßt. Bild 9 zeigt das entsprechende FORTH-Programm dazu.

```
( SCR#3==HAUPTSCHLEIFE======= )

: WAIT 50 0 DO LOOP ;
: TENNIS
  -4222 ADR !  0 RI !
  BEGIN
   21 Z !  -1 RZ !
   108 0 DO PLMOVE
        I 3 MOD 0= IF BMOVE THEN
      WAIT
      LOOP
  Z @ S @ ABSADR BRESET
  Z @ 1 + S @ ABSADR C@
  32 =
  UNTIL ;
( c '88 U. Kuckwa)              ;S
```

Bild 9. SCR # 3 mit dem Wort TENNIS

: WAIT
: TEST

TES"
Al
r

abschließend noch das Wort SPIEL, welches TENNIS zehnmal

(ENTER)
'LS I . ." . Runde" TENNIS LOOP ; (ENTER)

... von Soundeffekten und einer Spielerwertung läßt sich aus dem ... Grundgerüst ein attraktives Bildschirmspiel gestalten.
...alisierung dieses Spieles wurde bei uns mit fortgeschrittenen Schülern aus ...en Klassen 9 und 10 im Rahmen eines Computerclubs vollzogen.

Literatur

[1] HUTTERER, G.: Orientierungen für eine erziehungswirksame Gestaltung des inhaltlichen und methodischen Herangehens an die Beschäftigung mit dem Computer in der außerunterrichtlichen Tätigkeit. – Halle, 1987, S. 5—20

[2] CFORTH – ein einheitliches F.I.G.-FORTH für alle in der DDR produzierten Kleincomputer. – In: radio fernsehen elektronik. – Berlin 37 (1988) 6, S. 339

[3] KÜHNEL, C.: FORTH – ein Softwarekonzept für Micro- und Minicomputer. – In: Kleinstrechner-TIPS, Heft 10. – Leipzig, 1989

Autor:
Uwe Kuckwa
Pädagogischer Mitarbeiter an der
„Station Junger Chemiker"
Windmühlenstraße
Leuna
4220

ISBN 3-343-00582-7

1. Auflage
Lizenznummer 114/210/4/90
LSV 1083
Verlagslektor: Helga Fago
Printed in GDR
Satz und Druck:
Messedruck Leipzig, Bereich Borsdorf
III/18-328
Redaktionsschluß: 15. 12. 1989
Bestellnummer: 547651 2
00780

Kleinstrechner-TIPS/Hrsg. von
Hans Kreul u. a. – Leipzig: Fachbuchverl.,
H. 13. – 1. Aufl. – 1990. – 64 S.: 29 Bild.

Anschrift des Verlages:
VEB Fachbuchverlag Leipzig
PSF 67
Leipzig
DDR – 7031

Herausgeber:
Prof. Dr.-Ing. *Hans Kreul*
Bruno-Schröter-Str. 1
Zittau
DDR – 8800
Prof. Dr. sc. techn. *Thomas Horn*,
Doz. Dr.-Ing. *Wilhelm Leupold*
Informatik-Zentrum des Hochschulwesens
an der Technischen Universität Dresden
Mommsenstr. 13
Dresden
DDR – 8027

Hinweise für Autoren

Herausgeber und Verlag danken den Lesern für das Interesse an den »Kleinstrechner-TIPS«, das sich in zahlreichen Zuschriften und Veröffentlichungsangeboten äußert. Beim Einsenden von Artikeln bitten sie folgende Hinweise zu beachten:

- In den »Kleinstrechner-TIPS« werden Artikel aus den auf der 4. Umschlagseite angegebenen Gebieten veröffentlicht.
- Manuskripte sind zweizeilig mit schwarzem Farbband mit Schreibmaschine zu schreiben, ä, ö und ü dürfen nicht durch ae, oe und ue ersetzt werden.
- Bilder sollten auf getrennten Blättern gezeichnet sein. Eine Bildunterschriftenliste ist beizufügen.
- Rechnerausdrucke (z. B. Programme) sollen tiefschwarz mit guter Ausnutzung des Formats (engzeilig, kein zu breites Kommentarfeld usw.) gedruckt sein.
- Die Autorenangabe soll enthalten: akadem. Grad, Vorname, Name, Tätigkeit, Arbeitsstelle, Privatanschrift.

Manuskripte mit einem Umfang von nicht mehr als 15 Schreibmaschinenseiten (einschließlich Bilder und Programme) sind in zwei Exemplaren an einen der Herausgeber zu senden.

Die Broschürenreihe

KLEINSTRECHNER-TIPS

behandelt

- Tendenzen und Theorien
- Informationen und Ideen
- Programme und Projekte
- Spaß und Spiel

und stellt sich das Ziel

- den Nutzer der Mikrorechentechnik aus allen Bereichen der Volkswirtschaft und dem Bildungswesen bei der Einarbeitung in die Informatik und Computertechnik zu unterstützen
- Entwicklungstendenzen der Informatik und Computertechnik vorzustellen und zur Erweiterung des Grundwissens beizutragen
- Anregungen für den Computereinsatz zu geben und Beispielprogramme für Kleincomputer zu veröffentlichen

um somit einem großen Kreis von Freunden der Informatik und Computertechnik zu helfen, sich moderner Hilfsmittel und Methoden zu bedienen.

ISBN 3-343-00582-7