Vergleichsliste BASIC-Dialekte: Befehle und Cursorsteuerungen

Befehl/CRSR-Steu.	ATARI	*1	Commodore C64	CPC 64 Schneider	ZX Spectrum
GET X$	GET X$	*5	GET X$	X$ = INKEY$	INKEY$
LET	LET	*2	LET *2	LET *2	LET
RND(X)	RND(X)	*4	RND(X) *3	RND(X) *4	RND(X)
TAB(X)	POKE 85,X	*6	TAB(X)	TAB(X)	TAB(X)
LEFT$(X$,X)	LEFT$(X$,X)		LEFT$(X$,X)	LEFT$(X$,X)	X$=X$(1 TO X)
CURSOR HOCH	PRINT CHR$(28)		PRINT CHR$(145)	PRINT CHR$(11)	PRINT CHR$(11)
CURSOR TIEF	PRINT CHR$(29)		PRINT CHR$(17)	PRINT CHR$(10)	PRINT CHR$(10)
CURSOR LINKS	PRINT CHR$(30)		PRINT CHR$(157)	PRINT CHR$(8)	PRINT CHR$(8)
CURSOR RECHTS	PRINT CHR$(31)		PRINT CHR$(29)	PRINT CHR$(9)	PRINT CHR$(9)

*1 Beim ATARI muß jede String-Variable im Programm vorher dimensioniert werden.
*2 Hier kann die Anweisung auch weggelassen werden (a=10).
*3 X=Beliebig, erzeugt immer Wert zwichen 0 und 1.
*4 X=Zahl, erzeugt Wert zwischen 0 und Zahl.
*5 Hier muß vorher ein Tastaturkanal geöffnet und die zu verwendende Variable dimensioniert werden. dim x$(1) : open#1,4,0,"k:" : get x: x$ = chr$(x)
*6 Vor dem PRINT-Befehl!

DAS BEGLEITMATERIAL ZU ALLEN 5 FOLGEN DER FERNSEHSERIE

Hallo Computerfreunde!

Im Begleitmaterial des SFB-COMPUTER-CLUB finden Sie alle Programmlistings der Sendereihe mit Erläuterungen, Tips und Hinweisen. Wenn Sie noch keine großen Erfahrungen mit Ihrem Computer gesammelt haben, dann sollten Sie mit unseren ersten Hinweisen beginnen und selbst ein bißchen am Computer herumspielen, um die ersten Befehle wirklich gut kennen zu lernen. Unter Umständen werden Sie dabei feststellen, daß bei Ihrem Gerät nicht alles so funktioniert, wie wir es beschrieben haben. Bitte schauen Sie dann in Ihrem Handbuch nach und benutzen Sie zusätzlich unsere Übersetzungstabelle auf der Rückseite des Heftes.

So, nun geht es los: wir beginnen mit den Listings und Hinweisen der ersten Sendung:

SFB-COMPUTER-CLUB, Folge 1: "Computer - wofür?"

Ihren Computer sollten Sie inzwischen angeschlossen haben, so daß der Bildschirm irgendein Zeichen zeigt, z.B. das Wort "ready". Schlagen Sie notfalls noch einmal in Ihrem Handbuch nach und bedenken Sie bitte, daß Sie, wenn Sie ein Fernsehgerät benutzen, dieses auf den Kanal des Computers einstellen müssen.

Wenn Sie nun etwas auf der Tastatur des Computers tippen, müßten Sie das auf dem Bildschirm lesen können. Manche Computer geben Ihnen dann gern auch eine Fehlermeldung als Quittung, zumindest dann, wenn Sie die Taste "return" bzw. "enter" gedrückt haben.

Wenn Ihr Computer nun bereit ist, mit Ihnen in der Programmiersprache BASIC zu reden, dann können Sie ihm die entsprechenden Befehle erteilen. Wir verwenden hier die gängigsten Befehle, die eigentlich jeder Computer verstehen müßte. Dennoch: es gibt Unterschiede. Jeder Computer verwendet seinen eigenen BASIC-Dialekt. Sollten Sie also irgendwann einmal Schwierigkeiten bekommen, dann schauen Sie bitte im Verzeichnis der BASIC-Befehle Ihres Computers nach, ob es den von uns verwendeten Befehl überhaupt bei Ihrem Computer gibt. Eine gewisse Hilfe gibt Ihnen auch unsere Übersetzungstabelle auf der Rückseite.

Das sind unsere ersten BASIC-Befehle:

PRINT

GOTO

Ein BASIC-Programm besteht aus mehreren Befehlen, die in der Regel in Zeilen untereinander stehen, wobei alle Befehlszeilen in der Reihenfolge, in der sie später bearbeitet werden sollen, durchnummeriert werden. Es hat sich als präktisch erwiesen, die Befehlszeilen in Zehnerschritten zu numerieren, dann kann man später, bei Bedarf, zusätzliche Zeilen einfügen.

Tippen Sie doch zunächst einmal das folgende kleine Programm in Ihren Computer:

```
10 PRINT "SFB-COMPUTER-CLUB"
20 GOTO 10
```

Bitte beachten Sie: nach jeder Zeile die Taste "return" (oder "enter") drücken. Damit signalisieren Sie dem Computer, daß Ihre Eingabe, der Befehl beendet ist. Gleichzeitig springt der "Cursor", die Schreibmarke, eine Zeile tiefer.

PRINT ist der Befehl für Schreiben, Drucken oder Zeigen. Alles, was in Anführungszeichen steht, wird nachher vom Computer auf den Bildschirm geschrieben bzw. ausgedruckt werden.

GOTO heißt "Gehe nach...". Es folgt dann immer eine Zahl, die Ziffer der Zeile, in die der Computer springen soll.

Hier machen wir nun etwas ziemlich Gemeines: wir schicken den Computer zurück in die erste Zeile: das ergibt eine **unendliche Schleife**, aus der wir das Gerät nur mittels der **BREAK**- oder **STOP**-Taste befreien können.

Aber schauen Sie sich erst einmal an, wie unser kleines Programm läuft. Dazu geben Sie dem Computer den entsprechenden Befehl - aber jetzt ohne Zeilennummer. Der Befehl, das Programm zu starten, gehört ja nicht unmittelbar zum Programm, sondern ist ein Befehl, den Sie Ihrem Computer immer geben können, wenn Sie irgendein Programm starten wollen. Tippen Sie also in die nächste freie Zeile den Befehl **RUN** - was soviel heißt wie "Renne los!"

Nun wollen wir dem Computer gezielt befehlen, irgendetwas mehrfach zu schreiben. Die nächsten Befehle dazu lauten:

```
FOR .... TO ...
NEXT
```

Mit diesen Befehlen bildet man die berühmte FOR-NEXT-Schleife, mit der ein bestimmter Index durchgezählt wird. **FOR I = 1 TO 20** heißt also: "Zähle von 1 bis 20. Beginne bei 1." Danach folgen in der Regel die Zeilen mit den Befehlen, die mehrfach, hier z.B. 20mal, durchgeführt werden sollen. **NEXT** befiehlt dem Computer nun, zurück in die For-Zeile zu gehen. Das geht dann so lange im Kreis herum, bis die Endzahl erreicht ist.

Bitte versuchen Sie doch unser nächstes kleines Programm:

```
10 FOR Z = 1 TO 20
20 PRINT "SFB-COMPUTER-CLUB"
30 NEXT Z
```

Wenn Sie nun wieder RUN eintippen, wird Ihnen Ihr Computer genau 20mal das hinschreiben, was Sie zwischen die Anführungszeichen gesetzt haben.

Eine andere Möglichkeit, Ihren Computer zählen zu lassen, ist eigentlich noch einfacher: Sie geben ihm in einer bestimmten Zeile die Anweisung, zu einem bestimmten Wert, den er schon im Gedächtnis hat, eins hinzu zu zählen, also eine Rechenanweisung. Zum Beispiel:

```
Z = Z + 1
```

Wer hier zunächst seine Schwierigkeiten hat, soll-

te sich folgendes klarmachen: es handelt sich nicht um eine Gleichung, sondern um eine Zuweisung. Sie müssen sich also das Gleichheitszeichen eher als einen Pfeil vorstellen, der von rechts nach links zeigt. "Z" ist für den Computer wie eine Schachtel, in die er bestimmte Werte für Z hineintun kann. Zu diesem Wert soll jetzt also 1 hinzugezählt werden. Das ist der neue Wert für Z. Der Befehl lautet also schlicht und einfach: erhöhe Z um 1!

Bitte beachten: wann immer Sie einen solchen Zählbefehl im Programm verwenden, müssen Sie zuvor angeben, von welchem Ausgangswert ausgegangen werden soll. In der Regel werden Sie in einer Zeile vorher Z auf 0 oder 1 setzen.

Um den Computer an einer bestimmten Stelle wieder zu stoppen, müssen Sie dann nur den durchgezählten Index abfragen.

IF ...THEN heißt: "Wenn ... dann", eine logische Abfrage, eine Bedingung. Nach IF folgt die Bedingung, oft ein rechnerischer Vergleich. Wenn die Bedingung erfüllt ist, dann wird die Anweisung ausgeführt, die nach dem Befehl THEN folgt. Wenn die Bedingung nicht erfüllt ist, geht der Computer einfach zur nächsten Befehlszeile weiter.

Nun werden Sie sicher unser nächstes kleines Programm verstehen:

```
10 Z = 1
20 PRINT "SFB-COMPUTER-CLUB"
30 Z = Z + 1
40 IF Z = 20 THEN GOTO 60
50 GOTO 20
60 END
```

END erklärt sich ja von selbst. Es ist übrigens nicht unbedingt nötig, jedes BASIC-Programm mit END abzuschließen. Nebenbei: Sie hätten in Zeile 40 nach THEN auch einfach END hinschreiben können, ohne umständlich in eine eigene Zeile zu springen. Aber bei anderen Anweisungen oder größeren Programmteilen, die noch folgen, ist der Sprung in eine bestimmte Zielzeile schon sinnvoll.

Hier noch eine andere Möglichkeit:

```
40 IF Z < 20 THEN GOTO 20
```

Dann können Sie sich die Zeile 50 sogar sparen.

Wichtig: experimentieren Sie unbedingt mit diesen ersten Befehlen ein wenig an Ihrem Computer herum, damit Sie mit den BASIC-Befehlen und mit Ihrem Computer Erfahrungen sammeln und sicherer werden.

Um **Spiele** selbst zu programmieren, benötigen wir jetzt unbedingt den Zufallsbefehl, die Anweisung, die dem Computer Zufallszahlen entlockt.

RND heißt eigentlich RANDOM, was soviel wie Zufall bedeutet.

```
X = RND(1)
```

Wann immer der Computer an diese Anweisung gelangt, erzeugt er für X eine Zufallszahl, die meistens zwischen 0 und 1 oder der in Klammern angegebenen Zahl liegt. Oft ist es sogar egal, was man zwischen die Klammern setzt: es ergeben sich immer Zahlen zwischen 0 und 1 - auf mehrere Stellen hinter dem Komma genau.

Mit der folgenden Anweisung erhalten Sie **ganze** Zahlen von 0 bis 36 (Roulette):

```
X = INT(RND(1)*37)
```

Mit der folgenden von 1 bis 6 (Würfel):

```
10 X = INT(RND(1)*6) + 1
```

Auch diesen Befehl (oder eine Abwandlung für Ihren speziellen Computertyp) sollten Sie unbedingt ausprobieren!

INT bedeutet übrigens INTEGER, das heißt hier ganzzahlig. Der Befehl bewirkt in der Regel eine Abrundung. Genauer: durch den Befehl INT werden einfach alle Stellen hinter dem Komma weggelassen. Bitte schauen Sie in ihrem Handbuch nach, welche anderen Befehle es für die Veränderung von Dezimalzahlen gibt - sofern Sie das brauchen ...

Oft bieten die Computer noch die folgende einfache Anweisung zur Erzeugung ganzer Zahlen:

```
X%
```

Wichtig ist hier das nachgestellte %-Zeichen, das aus einem gebrochenen Wert für X einen ganzzahli-

gen Betrag macht. Allerdings muß dann im folgenden für die Variable immer das %-Zeichen mitgeschrieben werden! Wie gesagt: diese Möglichkeit bieten nicht alle Computer - dann müssen Sie halt mit dem INT-Befehl arbeiten.

Hier ein erstes Spiel, das sich mit den bis jetzt bekannten Befehlen leicht programmieren läßt.

```
100 REM HILO - SPIEL
110 :
120 X% = RND(1) * 100
130 N = 0
140 N = N + 1
150 PRINT "WELCHE ZAHL"
160 INPUT Z%
170 IF Z% > X% THEN PRINT "ZU GROSS!" : GOTO 140
180 IF Z% < X% THEN PRINT "ZU KLEIN!" : GOTO 140
190 PRINT "TOLL,RICHTIG!!!"
200 IF N = 1 THEN
        PRINT "ALLERDINGS: REINES GLUECK!!!"
210 IF N = 2 THEN PRINT "GLUECK GEHABT!!"
220 IF N = 3 THEN
        PRINT "NUR 3 VERSUCHE, ALLE ACHTUNG !!!"
230 IF N = 4 THEN PRINT "NACH NUR 4 VERSUCHEN !!!"
240 IF N = 5 THEN PRINT
            "IMMERHIN: NUR 5 VERSUCHE !"
250 IF N = 6 THEN PRINT "ALLERDINGS: 6 VERSUCHE!"
260 IF N = 7 THEN PRINT "WAS LANGE WAEHRT ..."
270 IF N > 7 THEN PRINT "MUEHSAM, MUEHSAM ..."
```

In Zeile 170 und 180 wird die eingegebene Zahl abgefragt, ob sie größer oder kleiner als die "gedachte" Zahl ist. Sind beide Bedingungen nicht erfüllt, dann muß Z% also = X% sein. Deshalb kann in der nächsten Zeile gleich der entsprechende Ausdruck erfolgen.

Der Zähler für die verschiedenen Versuche, der in Zeile 130 auf Null gesetzt wurde, wird bei jeder Bearbeitung der Zeile 140 um eins erhöht. Ab Zeile 200 werden dann die Kommentare ausgewählt und ausgedruckt. Vielleicht lassen Sie sich ein paar neue einfallen... Viel Spaß!

Hier ein weiteres Spiel, mit dem sich Leute beglücken lassen, die schon immer gegen Computer waren:

```
100 REM VERWIRR
110 :
120 X% = RND(1) * 4
130 PRINT "ADDIEREN ODER MULTIPLIZIEREN ?"
140 PRINT "BITTE EINGEBEN A ODER M"
150 INPUT E$
160 IF E$ <> "A" AND E$ <> "M"   THEN 110
170 PRINT "BITTE ERSTE ZAHL"
180 INPUT Z1
190 PRINT "BITTE ZWEITE ZAHL"
200 INPUT Z2
210 IF E$ = "A" THEN
        PRINT "DIE SUMME IST"; Z1 + Z2 + X%
220 IF E$ = "M" THEN
        PRINT "DAS PRODUKT IST"; Z1 * Z2 + X%
```

Das ganze Geheimnis steckt in der jeweiligen Rechenanweisung in Zeile 210 und 220, wo je nach Zufall 1,2,3 oder 0 hinzugezählt wird. Wenn Sie den Bereich der Zufallszahlen weiter einengen, wird die Null häufiger erscheinen, d.h.: der Computer wird häufiger das richtige Ergebnis zeigen, also weniger oft "spinnen". Vielleicht gefällt Ihnen dann das Spiel besser. Wenn Sie noch wenig Erfahrung in der Computerei haben, versuchen Sie doch mal, das Programm aus der Erinnerung selbst zu entwerfen und zum Laufen zu bringen: Übung macht den Meister...

Das nachgestellte **$-Zeichen** signalisiert dem Computer übrigens, daß für diese Variable Buchstaben oder Zeichen verwendet werden können. Würde diese Kennzeichnung fehlen, dann würde der Computer nur Ziffern akzeptieren.

SFB-COMPUTER-CLUB
Folge 2: "Computer - was tut er?"

Die Grafikfähigkeiten der einzelnen Computer sind zu verschieden, als daß wir sie hier in allgemeiner Form vorstellen könnten. Aber es gibt eine Möglichkeit, auch mit kleinen BASIC-Programmen ziemlich witzige Grafiken auf den Bildschirm zu zaubern, mit denen man sogar Spiele programmieren kann. Wir benötigen dazu die schon bekannten PRINT- und RND-Befehle und eine neue Anweisung:

```
PRINT TAB(X) "...
```

Wie bei einer Schreibmaschine beginnt der Computer das, was innerhalb der Anführungszeichen steht, erst ab der X-ten Spalte auf den Bildschirm zu

schreiben.(Achtung ATARI-Besitzer: der TAB-Befehl funktioniert so leider nicht. Versuchen Sie andere Befehle oder die POKE-Anweisung der Übersetzungstabelle!) In die Klammern kann eine feste Zahl oder eine Variable eingesetzt werden. Hier das Programm für den schlangenförmigen Bildschirmausdruck:

```
10 FOR I = 1 TO 20
20 PRINT TAB(I) "SFB-COMPUTER-CLUB"
30 NEXT I
40 FOR I = 20 TO 1 STEP -1
50 PRINT TAB(I) "SFB-COMPUTER-CLUB"
60 NEXT I
70 GOTO 10
```

Beachten Sie in Zeile 40, daß Sie beim Rückwärtszählen unbedingt die Schrittweite angeben müssen: **STEP -1.**

Das ist unser **"Schneeflockenprogramm":**

```
10 X% = RND(1) * 40
20 PRINT TAB(X%) "*"
30 GOTO 10
```

Und hier nun das Listing für unser grafisches Spiel **"Autobahn":**

```
 100 REM AUTOBAHN
 110 B = 38
 120 A  = 15
 130 LR = B / 2 - A / 2
 140 X  = B / 2
 150 PRINT TAB(LR); ")"; TAB(X); "OO"; TAB(LR+A); "("
 160 Z = RND(1)
 170 IF Z < 0.5 AND LR > 1 THEN LR = LR - 1
 180 IF Z > 0.5 AND LR + A < B  THEN LR = LR + 1
 190 GET A$
 200 IF A$ = "1" THEN X = X - 1
 210 IF A$ = "2" THEN X = X + 1
 220 IF X <= LR THEN GOTO 250
 230 IF X >= LR + A  THEN GOTO 250
 240 GOTO 150
 250 PRINT TAB(X) "*"
 260 PRINT TAB(X) "*"
 270 PRINT TAB(X-1) "***"
 280 PRINT TAB(X-2) "BAFF!!"
 290 PRINT TAB(X-1) "* *"
 300 PRINT TAB(X) "*"
 310 PRINT TAB(X) "*"
```

Achtung: der Befehl **GET** (oder bei einigen Computern der entsprechende Befehl **INKEY**) unterscheidet sich ganz wesentlich vom Befehl **INPUT:** bei INPUT stoppt das Programm und wartet auf die entsprechende Eingabe von der Tastatur. Bei GET läuft das Programm immer durch und reagiert an der entsprechenden Stelle nur dann, wenn eine Taste auch tatsächlich gedrückt wurde. In unserem Programm fragen wir für die Rechts/Links-Steuerung des Autos die Tasten 1 und 2 ab - wir können natürlich auch andere Tasten auswählen.

Versuchen Sie doch einmal, das Spiel aus der Erinnerung selbst zu schreiben. Dazu noch ein paar Tips: so richtig professionell wird das Ganze, wenn Sie vorne noch ein Menü vorsehen. Man kann z.B. den jeweiligen Spieler den Schwierigkeitsgrad selbst wählen lassen und dann, je nach Grad, die Breite der Autobahn verändern, also die Variable A immer weiter verkleinern. Außerdem können Sie die Soundbefehle - falls vorhanden - ausnutzen und nach jedem "Zusammenstoß" ein akustisches Signal ertönen lassen.

Hier unser kleines **Statistikprogramm:**

```
 100 REM STATISTIK
 110 DATA 75, 60, 40, 30, 20, 18, 17, 25
 120 DATA 43, 41, 101, 14, 13, 102, 12, 7
 130 DATA 33, 12, 15, 103, 999
 140 J = 0
 150 H = 0
 160 MI = 120
 170 MA = 0
 180 I = 0
 190 S = 0
 200 READ X
 210 IF X = 999 THEN GOTO 290
 220 I = I + 1
 230 S = S + X
 240 IF X < 18 THEN J = J + 1
 250 IF X > 100 THEN H = H + 1
 260 IF X < MI THEN MI = X
 270 IF X > MA THEN MA = X
 280 GOTO 200
 290 PRINT "DATEN: "; I
 300 PRINT "SUMME: "; S
 310 PRINT "DER MITTELWERT IST: "; S / I
 320 PRINT "KLEINSTER WERT :"; MI
 330 PRINT "GROESSTER WERT: "; MA
 340 PRINT "JUGENDLICHE UNTER 18: "; J
 350 PRINT "UEBER HUNDERTJAEHRIGE: "; H
```

Bitte beachten Sie: zu den DATA-Zeilen gehört sinnvollerweise immer eine READ-Anweisung. Für den entsprechenden Wert, den wir X genannt haben, liest der Computer dann alle Daten ein. Um den Einlesevorgang abzuschließen, muß zum Schluß ein Wert genannt werden, der nicht zu den Daten gehören **kann**. Wir haben dafür die 999 gewählt, man könnte auch die 0 nehmen oder eine negative Zahl - eben weil diese bei Altersangaben nicht vorkommen können.

Um das Minimum zu erkennen, fängt man mit einer so großen Zahl an, daß das zu erwartende Minimum sicher darunter liegt - entsprechend wird der Anfang für das Erkennen eines Maximums auf einen extrem kleinen Wert, z.B. Null gesetzt.

Auch hier empfehlen wir Ihnen, mit dem Programm selbst etwas herumzuexperimentieren: z.B. können Sie die Altersangaben der Jugendlichen oder der Überhundertjährigen in einer kleinen Tabelle ausdrucken lassen. Oder Sie haben überhaupt eine andere Anwendung im Sinn. Vielleicht läßt sich auch eine Menüsteuerung einbauen. Sie werden sehen, das Selberprogrammieren macht am meisten Spaß!

Unsere Formulierungshilfe, für alle, die gern und viel reden (müssen):

Es können hier kurze, prägnante Formulierungen wie **synchron punktuelle Steuerrechtsproblematik** oder tiefsinnige Aussagen wie **typisch deutsch qualifizierte Aufgabenverteilungskomponenten** entstehen. Sie sehen schon, es ist eine echte Bereicherung für jede Bundestagsdebatte.

Wie arbeitet das Programm? In der Zeile 100 werden vier String-Tabellen dimensioniert, die dann unsere Wortschöpfungen aufnehmen sollen. In den Zeilen 120 - 540 werden dann den vier Tabellen die Worte zugeordnet, wobei jeweils 10 Worte in eine Tabelle kommen.

In den Zeilen 580 - 610 werden dann vier Zufallszahlen zwischen 1 und 10 erzeugt, die jeweils ein Wort aus einer Tabelle auswählen und in den Zeilen 620 bis 622 auf dem Schirm ausgegeben. Das Ganze wird, gesteuert durch eine For-Next-Schleife, fünfmal durchlaufen, sodaß jeweils fünf markige Sprüche pro Programmlauf auf dem Schirm erscheinen.

Auch in diesem Programm können Sie ihrer Phantasie freien Lauf lassen und durch Hinzufügen weiterer Worte die Vielfalt der deutschen Sprache beleben.

Die Zufallszahlen-Erzeugung in den Zeilen 580 - 610 muß notfalls unter Anwendung unserer Vergleichsliste auf Ihren Rechner angepaßt werden.

Und hier nun das Listing, ein starkes Programm für starke Sprüche:

```
100 DIM A$(10), B$(10), C$(10), D$(10)
110 :
120 A$(1) = "SYNCHRON"
130 A$(2) = "AKTIV"
140 A$(3) = "KONJUNKTUR"
150 A$(4) = "ORGANISATORISCH"
160 A$(5) = "AMBIVALENT"
170 A$(6) = "TYPISCH DEUTSCH"
180 A$(7) = "AMTSINTERN"
190 A$(8) = "WISSENSCHAFTLICH"
200 A$(9) = "BUERGERFREUNDLICH"
210 A$(10) = "MODIFIZIERT"
220 :
230 B$(1) = "PUNKTUELLE"
240 B$(2) = "ORIENTIERTE"
250 B$(3) = "QUALIFIZIERTE"
260 B$(4) = "PROGRESSIVE"
270 B$(5) = "BEDINGTE"
280 B$(6) = "MAXIMIERTE"
290 B$(7) = "KONZENTRIERTE"
300 B$(8) = "INTEGRIERTE"
310 B$(9) = "FLEXIBLE"
320 B$(10) = "STRUKTURIERTE"
330 :
340 C$(1) = "STEUERRECHTS"
350 C$(2) = "REFORM"
360 C$(3) = "GEWERKSCHAFTS"
370 C$(4) = "VERANLAGUNGS"
380 C$(5) = "AUFGABENVERTEILUNGS"
390 C$(6) = "PERSONAL"
400 C$(7) = "BEHOERDEN"
410 C$(8) = "VORGESETZTEN"
420 C$(9) = "KOLLEGIALITAETS"
430 C$(10) = "ARBEITSPLATZ"
440 :
450 D$(1) = "IDIOTIE"
460 D$(2) = "PROBLEMATIK"
470 D$(3) = "EFFIZIENZ"
480 D$(4) = "PRAKTIKEN"
490 D$(5) = "PROGRAMMIERUNG"
```

```
500 D$(6) = "TENDENZ"
510 D$(7) = "KONZEPTE"
520 D$(8) = "PRINZIPIEN"
530 D$(9) = "KOMPONENTEN"
540 D$(10) = "SCHWIERIGKEITEN"
550 :
560 PRINT
570 FOR J = 1 TO 5
580 :   A = INT(RND(1)*10+1)
590 :   B = INT(RND(1)*10+1)
600 :   C = INT(RND(1)*10+1)
610 :   D = INT(RND(1)*10+1)
620 :   PRINT A$(A);" "; B$(B)
621 :   PRINT C$(C); D$(D)
622 :   PRINT
630 NEXT J
640 END
```

SFB-COMPUTER-CLUB
Folge 3: "Computer - doch welcher?"

Das ist unsere **Kriterienliste** für die Auswahl eines Computers. Wichtig ist, daß Sie sich klar darüber werden, was Sie hauptsächlich mit dem Computer machen wollen. Wenn Sie die Computerei und das Programmieren erst einmal kennenlernen wollen, können Sie fast jedes beliebige Modell wählen und sich getrost am Preis orientieren.

Kriterienliste für die Auswahl eines Home-Computers

Vorab eine Bemerkung: Es gibt keinen "idealen" Heimcomputer. Jeder Rechner hat Stärken und Schwächen und man muß schon ziemlich genau wissen, wofür man den Rechner später einsetzen will und entsprechend auf die hierfür notwendige Ausstattung achten.

Nun zu den einzelnen zu beachtenden Anforderungen:

Gehäuse

Das Gerät sollte über ein stabiles Metall- oder Plastikgehäuse verfügen um auch einen öfteren Transport sicher zu überstehen. Ein eingebautes Netzteil, auch für die gleichzeitige Versorgung eventueller Zusatzgeräte wie z.B. Kassettenrekorder, senkt den Verkabelungsaufwand und trägt zur Sicherheit entscheidend bei.

Alle Anschlußkabel sollten ausreichend lang sein; gerade das Kabel zum Monitor oder Fernseher ist in der Regel zu kurz um einen vernünftigen Abstand zur Bildröhre zu erreichen.

Speicher

Notwendig sind hier mindestens 32 KB RAM; d.h. für den Benutzer frei verwendbaren Speicher zur Aufnahme des Programms oder der Daten. Der Umfang des ROM-Bereiches; also des Teils, der das notwendige Betriebsprogramm enthält, ist ein Maß für die Qualität des verwendeten Basic-Interpreters. Wichtig ist hier, daß das Basic bereits eingebaut ist und nicht erst zeitraubend von Kassette geladen werden muß und damit dann auch einen nicht unerheblichen Teil des freien Speichers einnimmt.

Tastatur

Wünschenswert ist eine Schreibmaschinen-Tastatur in "QWERTZ-Anordung" mit deutschen Umlauten und einem separaten Zehnerblock für die Zifferneingabe. Die Tastatur soll über normale Schreibmaschinen-Tasten mit deutlichem Druckpunkt verfügen.

Für die alleinige Verwendung von fertiger Spielsoftware ist notfalls auch eine sogenannte Gummitastatur verwendbar, für umfangreiche Text- oder Programmeingabe ist sie aber ungeeignet.

Bildschirm

Als Bildschirm kann man in der Regel jeden normalen Farb- oder Schwarzweißfernseher verwenden; ein Computermonitor ist aber auf jeden Fall vorzuziehen, da die Auflösung eines FS nicht sonderlich hoch ist.

Gerade für die Textverarbeitung ist ein FS nicht sinnvoll, da ein längeres Arbeiten, bedingt durch das Flimmern, sehr anstrengend für die Augen ist.

Dokumentation

Dem Rechner sollte ein aussagefähiges Handbuch über Hardware-Eigenschaften und Programmierung beiliegen. Für den Anfänger ist außerdem ein BASIC-Lehrgang in Form eines Buches oder - noch besser - eines rechnergesteuerten Dialogs mit schriftlichem Begleitmaterial zu empfehlen.

Bildschirmdarstellung

Der Rechner sollte in der Lage sein, mindestens 24 Zeilen mit jeweils 40 Zeichen auf dem Bildschirm darzustellen. Für die Textverarbeitung ist allerdings eine Darstellung von 80 Zeichen pro Zeile unabdingbar; man hätte sonst keine Möglichkeit, einen eingegebenen DIN-A4 Text jemals im Originalformat zu sehen. Bei 40-Zeichen würde der Text einer Zeile immer über zwei Bildschirmzeilen dargestellt werden, was eine Kontrolle erschwert.

Für den Einsatz von Computerspielen und deren eigene Programmierung sollte der Rechner über Farbmöglichkeiten verfügen, wobei hier auch eine gute Qualität des Bildschirms notwendig ist um die verschiedenen Farben auch sauber trennen zu können.

Wünschenswert ist die Möglichkeit, hochauflösende Grafik darzustellen. Die Auflösung sollte hier mindestens 320 mal 200 Punkte betragen. Höhere Auflösungen sind selbstverständlich besser, bedingen aber den Einsatz eines entsprechend guten Monitors.

Tonerzeugung

Wünschenswert sind mindestens 3 unabhängige Tonstimmen, die sowohl in der Tonhöhe als auch im Klangcharakter veränderbar sind. Ideal ist die zusätzliche Beeinflußung des zeitlichen Klang- bzw. Lautstärkeverlaufs, wie es von normalen Synthesizern bekannt ist.

Datenspeicher

Es sollte wenigstens ein Kassettenrekorder vorhanden sein, um fertige Software einzuladen bzw. selbsterstellte Programme sichern zu können. Ein Diskettenlaufwerk sollte für die spätere Anwendung erhältlich sein.

Schnittstellen

Der Rechner sollte über wenigstens eine der gebräuchlichen Schnittstellen wie z.B. Parallel-, V24-Schnittstelle oder einen IEEE-Bus verfügen, um Peripheriegeräte, auch anderer Hersteller, anschließen zu können.

Software

Es sollte eine ausreichende Anzahl fertiger Spiel- oder Anwendungsprogramme existieren um gerade dem Anfänger den Einstieg zu erleichtern. Für den Selbstprogrammierer ist eine ausreichende Verbreitung des Rechners notwendig um eigene Software auch auf den Markt bringen zu können.

Kompatibilität, Erweiterungsmöglichkeiten

Der Rechner sollte zu anderen Homecomputern zumindest im BASIC-Bereich kompatibel sein; d.h. BASIC-Programme anderer Rechner sollten ohne großen Änderungsaufwand einsetzbar sein.

Für den späteren Einstieg ins "PROFI"-Lager sollte durch eine entsprechende Hardware-Erweiterung der Einsatz eines gängigen Betriebssystems wie z.B. CP/M möglich sein.

** -- *** -- **

Das folgende Programm ist eigentlich nicht das, was man unter einem Textverarbeitungsprogramm versteht. Aber man kann damit Standardtexte bzw. Textmodule zu Briefen verarbeiten: also ein "**Brieftextverarbeitungsprogramm**". Selbstverständlich können Sie das Programm nach Ihren Wünschen beliebig bzw. sinnvoll verändern. Sollten Sie allerdings tatsächlich mit unserem Programm Mahnschreiben versenden wollen, dann raten wir Ihnen, in einer zusätzlichen Zeile 245 noch abzufragen, ob KO = 0 ist und dann an's Ende zu springen - oder einen anderen Text vorzusehen.

```
100 INPUT "NAME"; N$
110 INPUT "STRASSE"; S$
120 INPUT "ORT"; O$
130 INPUT "KONTOSTAND"; KO
135 INPUT "STANDARDTEXT NR.";T
136 IF T < 1 OR T > 3 THEN 135
140 PRINT "----------------------------------------"
150 PRINT N$
160 PRINT S$
170 PRINT
180 PRINT O$
190 FOR I = 1 TO 3
200 PRINT
210 NEXT
220 PRINT "SEHR GEEHRTE DAMEN UND HERREN,"
```

```
230 PRINT
240 IF KO > 0 THEN 310
250 KO = ABS(KO)
260 PRINT "IHR KONTO BEI UNS WEIST EINEN FEHLBE-"
270 PRINT "TRAG VON DM"; KO; "AUF. WIR BITTEN"
280 PRINT "SIE HIERMIT, DAS KONTO INNERHALB VON"
290 PRINT "14 TAGEN AUSZUGLEICHEN."
300 GOTO 350
310 PRINT "WIR FREUEN UNS IHNEN MITTEILEN ZU"
320 PRINT "KOENNEN, DASS IHR KONTO BEI UNS EIN"
330 PRINT "GUTHABEN VON DM";KO;"AUFWEIST,"
340 PRINT "UEBER DAS SIE FREI VERFUEGEN KOENNEN"
350 PRINT
355 ON T GOSUB 1000, 2000, 3000
356 PRINT
360 PRINT "MIT FREUNDLICHEN GRUESSEN"
370 PRINT
380 PRINT "-----------------------------------"
390 END
1000 PRINT "DIESE ANSAGE WAR GEBUEHRENFREI"
1010 RETURN
2000 PRINT "DARUEBER HINAUS TEILEN WIR IHNEN MIT,"
2010 PRINT "DASS WIR DIE GESCHAEFTSVERBINDUNG"
2020 PRINT "ALS BEENDET ANSEHEN."
2030 RETURN
3000 PRINT "WIR WUERDEN UNS FREUEN, WEITER MIT"
3010 PRINT "IHNEN ZUSAMMENARBEITEN ZU KOENNEN."
3020 RETURN
```

SFB-COMPUTER-CLUB
Folge 4: "Computer - was nun?"

Eine wichtige Anwendung für Computer sind Sortierungsaufgaben, besonders dann, wenn viele Daten verwaltet werden müssen. Wir verwenden hier ein ziemlich einfaches Sortierungsverfahren, den sogenannten "bubble-sort" - es gibt auch aufwendigere Verfahren, die vor allem schneller sind. Die Sortierung nach dem Alphabet funktioniert übrigens, weil intern Buchstaben als Ziffern dargestellt werden. Daher können wir mit den Abfragen < und > arbeiten, obwohl ja eigentlich ein Z nicht wirklich "größer" ist als ein A oder ein Y.

Als Beispiel haben wir uns einmal vorgenommen, einige Namen in die richtige alphabetische Reihenfolge zu bringen. Werfen wir zunächst einen Blick in das Programm:

```
100 N$(1) = "WINFRIED"
110 N$(2) = "NORBERT"
120 N$(3) = "RENATO"
130 N$(4) = "JUERGEN"
140 N$(5) = "SUSANNE"
150 N$(6) = "HOLLY"
160 N$(7) = "STEFAN"
170 N$(8) = "MONA"
180 N$(9) = "INGO"
190 N$(10)= "KARIN"
200 VT = 0
210 FOR I = 1 TO 9
220 IF N$(I) <= N$(I+1) THEN 270
230 G$ = N$(I)
240 N$(I)  = N$(I+1)
250 N$(I+1) = G$
260 VT = 1
270 NEXT I
280 IF VT = 1 THEN GOTO 200
290 PRINT
300 FOR I = 1 TO 10
310 PRINT N$(I)
320 NEXT I
```

In den Zeilen 100 bis 190 weisen wir zunächst einer Tabelle die zu sortierenden Namen zu. Wir haben hier nur 10 Namen genommen um das Programm übersichtlich zu halten. In der Zeile 200 setzen wir dann einen sogenannten "Schalter" auf den Wert

Null. Mit Schalter bezeichnet man Variablen, die nicht unbedingt einen Zahlenwert enthalten müssen, sondern nur als "Merker" für einen bestimmten Programmzustand dienen. In unserem Beispiel vermerkt der Schalter **VT**, ob während unseres Sortiervorganges noch Vertauschungen vorgenommen wurden oder nicht.

In der Zeile 210 beginnt dann der eigentliche Sortiervorgang. Wir haben 10 Elemente in unserer Tabelle und müssen jedes Element mit dem nächsten vergleichen, um die Sortierfolge herzustellen. Wir bedienen uns hier einer FOR-NEXT-Schleife um die Elemente der Reihe nach durchzugehen.

In der Zeile 220 vergleichen wir das aktuelle Element mit dem Nächsten. Wenn es kleiner als das nächste oder gleich ist, ist die Reihenfolge in Ordnung und wir können zum nächsten Element übergehen. Andernfalls müssen die beiden Elemente vertauscht werden.

Es gibt nun im BASIC leider keinen Befehl um zwei Variablen direkt vertauschen zu können, also müssen wir uns eines kleinen Tricks bedienen. Wir nehmen zunächst das größere Element **N$(i)** und legen es in einer Variablen mit dem Namen **G$** ab. Anschließend bringen wir das nächste Element **N$(I+1)** in das zuvor ausgelagerte. Nun müssen wir nur noch den Inhalt von **G$** in das Folgeelement bringen und die Vertauschung ist komplett und zumindest diese beiden Elemente sind in der richtigen Reihenfolge.

Das folgende Listing zeigt noch eine Abwandlung, bei der die Namen nicht bereits im Programm vorgegeben sind sondern mit **INPUT** eingegeben werden. Der eigentliche Sortiervorgang ist identisch. Nehmen Sie es ruhig einmal als Anregung für eigene Programme; es gibt viel zu sortieren ....

```
100 DIM N$(100)
110 PRINT "BITTE NAMEN EINGEBEN"
115 PRINT "ENDE MIT '*'"
120 IN = 1
130 INPUT Z$
140 IF Z$ = "*" THEN 180
150 N$(IN) = Z$
160 IN = IN + 1
170 GOTO 130
180 IN = IN - 1
190 VT = 0
200 FOR I = 1 TO IN - 1
210 IF N$(I) <= N$(I+1) THEN 260
220 N$ = N$(I)
230 N$(I) = N$(I+1)
240 N$(I+1) = N$
250 VT = 1
260 NEXT I
270 IF VT > 0 THEN 190
280 FOR I = 1 TO IN
290 PRINT N$(I)
300 NEXT
```

Fehler, Tips und Tricks

Im folgenden stellen wir einmal eine Reihe von Tips und Tricks, aber auch von beliebten Fehlern vor.

Ein gerade bei Anfängern häufiger Fehler beim Arbeiten mit Tabellen ist eine Automatik, die BASIC mitgegeben wurde. Tabellen dimensionieren sich alleine, wenn nicht mehr als 10 Elemente angesprochen werden. Wann immer in Ihrem Programm ein Befehl eine Variable mit Index anspricht, z.B. **a(3)** richtet BASIC automatisch 10 Elemente ein; genaugenommen sogar 11 (von 0 - 10).

Wenn nun im Programm der Index über 10 hinausgeht tritt eine Fehlermeldung auf, die z.B. **bad subscript error** heißen kann oder **index out of range** usw. Alle diese Fehlermeldungen besagen, daß hier eine Tabelle mit einem unerlaubten - weil nicht dimensionierten - Index angesprochen wird.

Machen Sie es sich daher zur Regel, Tabellen immer in der erwarteten Größe **vor** dem ersten Ansprechen im Programm zu dimensionieren. Gerade diese Fehler treten meistens erst später auf und dann ist der Ärger groß.

... ...

Ein weiterer beliebter Fehler ist das unzulässige Verlassen von Unterprogrammen mit einem GOTO-Befehl bzw. das wiederholte Aufrufen eines Unterprogramms aus sich selbst heraus:

```
100 A = 1
110 GOSUB 1000
```

```
120 A = A + 1
130 GOTO 110
.
.
1000 PRINT "A IST GLEICH "; A
1010 GOTO 120
```

Der Fehler ist hier natürlich sofort zu erkennen aber es ist ja auch ein stark vereinfachtes Beispiel.

Was passiert nun in unserem kleinen Programm? In Zeile 110 wird das Unterprogramm mit Zeilennummer 1000 aufgerufen. Dieses gibt dann etwas auf dem Bildschirm aus und springt nun fälschlicherweise nicht mit **RETURN** sondern mit **GOTO** zurück. Bei jedem GOSUB muß sich der Rechner jeweils merken, woher der Aufruf kam um beim RETURN wieder ins Hauptprogramm zurückzufinden. Je nach Rechner werden hier schon bis zu 10 Bytes benötigt. Nun springt das Programm wieder in die Zeile 120, von dort nach 110 und schon wird das Unterprogramm wieder aufgerufen. Wieder werden 10 Bytes belegt und nicht durch ein RETURN freigegeben. Man kann schon absehen, daß das irgendwann zuviel wird. Je nach Rechner geht das Ganze 10 - 30 mal gut aber dann "knallts" erheblich. Meistens kommen dann auch noch verwirrende Fehlermeldungen wie **out of memory error**, obwohl bei einer Überprüfung noch reichlich Speicher vorhanden ist. Nur der Speicher für die Rücksprungadressen ist übergelaufen weil er in diesem Programm nie wieder geleert wird.

... ...

Und nun zur Abwechslung mal ein kleiner Tip. In unserem "Verwirrspiel" hatten wir eine Zeile, die lautete:

IF E$ <> "A" AND E$ <> "M" THEN ....

Erreicht werden soll hier z.B. ein Rücksprung, wenn **beide** Bedingungen zutreffen. Man kann den Befehl auch etwas anders schreiben; nicht unbedingt kürzer aber erheblich zeitsparender:

IF E$ <> "A" THEN IF E$ <> "M" THEN ...

Im ersten Beispiel muß der Rechner durch die **Und**-Verknüpfung immer beide Bedingungen überprüfen um eine Entscheidung zu treffen. Im zweiten Beispiel wird der Rest der Zeile bereits ignoriert, wenn die erste Bedingung nicht zutrifft - und das spart erheblich Rechenzeit.

... ...

Wir hatten in einigen Programmbeispielen mit den **%**-Variablen gearbeitet. Es gibt hier eine Besonderheit, die leicht zu Fehlern führen kann. Diese Variablen sind in ihrem Wertebereich eingeschränkt, d.h. man kann hier nicht beliebige Zahlenwerte eingeben. In der Regel reicht dieser Bereich von - 32768 bis + 32767. Alle Werte darunter oder darüber führen unweigerlich zu einem Fehler. Und, wie es Fehler so an sich haben, treten sie natürlich nicht beim Austesten eines Programmes sondern erst bei der Anwendung auf. Wenn man sich in seinem Programm also nicht ganz über die Größe der Zahlen klar ist, dann sollte man lieber mit den normalen Zahlenvariablen arbeiten und die Rundung mit dem **INTEGER**-Befehl vornehmen.

... ...

Ein weiteres Problem, mit dem Anwender von Homecomputern zu kämpfen haben, ist die Rechenungenauigkeit des BASICs. Sehen wir uns einmal folgendes Programmbeispiel an:

```
100 FOR I = 1 TO 10 STEP 0.1
110 IF I = 6 THEN PRINT "HIER"
120 NEXT I
```

In der Zeile 100 setzen wir eine FOR-NEXT-Schleife mit dem Anfangswert 1, dem Endwert 10 und einer Schrittweite (STEP) von 0,1. Die FOR-NEXT-Schleife soll also alle Werte in 0,1-Schritten durchlaufen, also 1.0, 1.1, 1.2, 1.3 usw. bis einschließlich 10.

Wenn man das Programm startet wäre eigentlich zu erwarten, daß zumindestens einmal "HIER" auf dem Bildschirm erscheint.

Probieren Sie es ruhig einmal aus; es wird nicht einmal etwas auf dem Bildschirm erscheinen. Das Problem liegt in der Art und Weise, mit der BASIC intern Zahlen darstellt. Alle ganzen Zahlen stellen überhaupt kein Problem für BASIC dar, auch Kommastellen, deren Wert glatt durch zwei teilbar sind, sind noch problemlos. Alle anderen Zahlen können aber nur näherungsweise intern dargestellt werden. Die Schrittweite 0.1 ist so ein Wert, und

die wiederholte Addition kumuliert auch den Fehler und irgendwann kommen dann sehr krumme Werte zum Vorschein.

In unserem Beispiel geht das bis einschließlich 3.6 noch gut; statt der 3.7 ermittelt der Rechner dann aber schon 3.69999999 und statt unserer gesuchten 6 findet er 5.99999999, und das ist nun mal nicht 6 und unsere Bedingung wird somit niemals erfüllt.

Man sollte sich daher unbedingt zur Regel machen, niemals auf direkte Zahlenwerte abzufragen, sofern man mit Kommastellen arbeitet. In unserem Beispiel könnte man folgendes programmieren:

```
100 FOR I = 1 TO 10 STEP 0.1
110 IF  I > 5.99 THEN IF I < 6.01
                 THEN PRINT "HIER"
120 NEXT I
```

Durch diese Bereichsabfrage ist sichergestellt, daß wir zumindest den Näherungswert für 6 treffen.

Man muß hier der Gerechtigkeit halber noch erwähnen, daß auch die größeren Brüder der Homecomputer, die Personal Computer, mit diesen Zahlen so ihre Probleme haben. Es gibt nur wenige Rechner, die mit einer anderen Zahlendarstellung arbeiten und damit solche Fehler vermeiden.

Mini-Nim, ein Strategie-Spiel

Unser kleines Strategie-Spiel gibt Ihnen die Möglichkeit, einmal direkt gegen den Rechner zu spielen. Zunächst 'denkt' sich der Computer eine Zahl zwischen 50 und 99. Anschließend können beide Spieler, Sie und der Computer, abwechselnd eine Zahl zwischen 1 und 10 hiervon abziehen. Das jeweils neue Zwischenergebnis wird angezeigt, der Gegner ist am Zug. Das Spiel ist zu Ende, wenn es einem der beiden Spieler gelungen ist, mit dem letzten Zug auf Null zu kommen.

Der Rechner spielt natürlich streng nach einer bestimmten Regel und man kann ihn nur besiegen, wenn man ebenfalls diese Regel beherrscht. Untersuchen Sie doch einmal das folgende Listing und versuchen Sie, die Regel zu erkennen.

In Zeile 120 wird eine Zufallszahl zwischen 50 und 99 erzeugt; hier ist notfalls das Programm auf Ihren Rechner anzupassen (siehe unsere Vergleichsliste auf der Rückseite des Heftes).

```
100 REM   MINI-NIM
110 :
120 ZA = INT(RND(1)*50) + 50
130 GOSUB 370
140 PRINT
150 PRINT "WIR HABEN NOCH"; ZA; "HOELZER"
160 PRINT "WIEVIEL NEHMEN SIE"
170 INPUT W
180 IF W < 1 OR W > 10 THEN 170
190 ZA = ZA - W
200 IF ZA > 0 THEN 250
210 PRINT
220 PRINT "GRATULIERE, SIE HABEN"
230 PRINT "G E W O N N E N"
240 END
250 PRINT
260 PRINT "OK, ES SIND JETZT NOCH"; ZA; "HOELZER"
270 ZW = INT(ZA/11)
275 W = ZA - ZW * 11
280 IF W = 0 THEN W = 1
290 IF W > 1 THEN PRINT "ICH NEHME"; W; "HOELZER"
300 IF W = 1 THEN PRINT "ICH NEHME 1 HOLZ"
310 ZA = ZA - W
320 IF ZA > 0 THEN 140
330 PRINT
340 PRINT "TUT MIR LEID, SIE HABEN"
350 PRINT "V E R L O R E N"
360 END
370 PRINT
380 PRINT "HALLO, WIR WOLLEN JETZT GEGENEINANDER"
390 PRINT "MININIM SPIELEN. ICH DENKE MIR EINE"
400 PRINT "ZAHL ZWISCHEN 50 UND 99. DANN KOENNEN"
410 PRINT "WIR ABWECHSELND EINE ZAHL VON 1 - 10"
420 PRINT "DAVON ABZIEHEN. WER MIT SEINEM ZUG"
430 PRINT "AUF NULL KOMMT HAT GEWONNEN."
440 PRINT
450 PRINT "ICH WUENSCHE IHNEN VIEL GLUECK ..."
460 RETURN
```

Wir haben dieses Programm zum Anlaß genommen, einmal den Unterschied in der Programmierung zwischen BASIC und PASCAL zu zeigen.

Das BASIC-Programm enthält etliche GOTO-Befehle, Sprünge also, die ein Programm schnell unleserlich machen. Das folgende PASCAL-Programm mit der glei-

chen Funktion hingegen kommt ohne einen einzigen GOTO-Befehl aus und ist auch insgesamt viel übersichtlicher. Zusammenhängende Befehle können durch Einrücken auch als zusammenhängend dargestellt werden.

Als Beispiel soll einmal die Eingabe-Routine für den Zug des menschlichen Gegners dienen. In BASIC lautet sie:

```
170 INPUT W              hole einen wert
180 IF W < 1 OR          Wenn die Eingabe kleiner
    W > 10 THEN 170      als 1 oder größer als
                         10 ist wieder zur
                         Eingabe verzweigen.
```

In PASCAL hingegen wird das so programmiert:

```
REPEAT                    wiederhole
  READ (WERT)             hole 'wert'
UNTIL (WERT > 0)          bis der Wert im korrekten
      AND (WERT < 11)     Bereich von 1 - 10 liegt.
```

Alles zwischen 'repeat' und 'until' wird solange wiederholt, bis die hinter 'until' genannte Bedingung erfüllt ist. Durch diese Form der Programmierung ist ein Programm auf Anhieb lesbar und Änderungen sind schnell und einfach durchführbar, ohne durch eingefügte Zeilennummern die Sprungziele der GOTO-Befehle zu verändern.

Hier nun das gesamte Mini-Nim in der PASCAL-Fassung:

```
program minimin (input,output);
var zahl,wert     : integer;
(* unterprogram zur einleitung *)
procedure intro;
begin
   writeln;
   writeln ('hallo, wir wollen jetzt gegeneinander');
   writeln ('mininim spielen. ich denke mir eine');
   writeln ('zahl zwischen 50 und 99. dann koennen');
   writeln ('wir abwechselnd eine zahl von 1 - 10');
   writeln ('davon abziehen. wer mit seinem zug');
   writeln ('auf null kommt hat gewonnen.');
   writeln;
   writeln ('ich wuensche ihnen viel glueck ...');
end;
(* hauptprogramm *)
begin
   zahl    := random mod 50 + 50;
   intro; (* einleitung aufrufen *)
repeat (* wiederholen bis spielende *)
      writeln;
      writeln ('wir haben noch ', zahl : 2,' hoelzer');
      writeln ('wieviel nehmen sie');
         repeat
            read (wert);
      until (wert > 0) and (wert < 11);
      zahl := zahl - wert;
      if zahl = 0 then
         begin
            writeln;
```

```
    writeln ('gratuliere, sie haben');
    writeln ('g e w o n n e n');
  end
else
  begin
    writeln;
    writeln ('ok, es sind jetzt noch ', zahl : 2,' hoelzer');
    wert := zahl mod 11;
    if wert = 0 then wert := 1;
    zahl := zahl - wert;
    if wert > 1 then
      writeln ('ich nehme ', wert : 2,' hoelzer')
    else
      writeln ('ich nehme 1 holz');
    if zahl = 0 then
      begin
        writeln;
        writeln ('tut mir leid, sie haben');
        writeln ('v e r l o r e n');
      end;
  end;
until zahl = 0;
end.
```

Unser Vokabel-Lernprogramm

Für jeden der Fremdsprachen erlernen will (oder muß) ist das Lernen und Üben von Vokabeln eine meist lästige Pflicht. Hier übernimmt nun der Computer die Rolle des geplagten "Abfragers". Sie können hier Vokabeln üben oder sich auch ein Wort aus dem Deutschen in die andere Sprache übersetzen lassen und umgekehrt.

Hier zunächst einmal das Listing:

```
100 PRINT
110 PRINT "VOKABEL-LERNPROGRAMM
120 PRINT
130 PRINT "BITTE AUSWAEHLEN:
140 PRINT
150 PRINT "1 = VOKABELN UEBEN"
160 PRINT "2 = SUCHEN DEUTSCH / ENGLISCH"
170 PRINT "3 = SUCHEN ENGLISCH / DEUTSCH"
180 INPUT AW
190 IF AW < 0 OR AW > 3 THEN 180
200 ON AW GOTO330, 220, 220
205 END
210 :
220 PRINT
230 PRINT "WELCHES WORT SUCHEN SIE"
240 INPUT Z$
245 IF Z$ = "*" THEN 100
250 RESTORE
260 READ D$, E$
270 IF D$ = "*" THEN PRINT
        "DIESES WORT KENNE ICH NICHT" : GOTO 220
280 IF AW = 2 THEN IF D$ <> Z$ THEN 260
290 IF AW = 3 THEN IF E$ <> Z$ THEN 260
300 IF AW = 2 THEN PRINT
          "OK, DAS HEISST AUF ENGLISCH : "; E$
310 IF AW = 3 THEN PRINT
          "OK, DAS HEISST AUF DEUTSCH : "; D$
320 GOTO 220
330 RESTORE
340 AN=0
350 READ D$, E$
360 IF D$ <> "*" THEN AN = AN + 1 : GOTO 350
370 DIM GF(AN)
380 FF = 0
390 NG = 0
400 WA = 0
410 WO = INT(RND(1) * AN) + 1
420 IF GF(WO) > RU THEN 410
430 RESTORE
440 GF(WO) = GF(WO) + 1
```

```
450 WA = WA + 1
460 FOR I=1 TO WO
470 READ D$, E$
480 NEXT
490 PRINT
500 PRINT "RUNDE"; RU + 1; ", WORT"; WA
510 PRINT "WIE LAUTET DAS ENGLISCHE"
520 PRINT "WORT FUER "; D$
530 FE = 0
540 INPUT Z$
545 IF Z$ = "*" THEN 670
550 PRINT
560 IF Z$ = E$ THEN PRINT "RICHTIG" : GOTO 650
570 PRINT "DAS IST LEIDER FALSCH"
580 FE = FE + 1
590 FF = FF + 1
600 IF FE < 3 THEN 540
610 NG = NG + 1
620 FF = FF - 3
630 PRINT "DIE RICHTIGE LOESUNG WAERE : "; E$
640 :
650 IF WA < AN THEN 410
660 PRINT "ENDE DER"; RU + 1; ". RUNDE"
670 PRINT "SIE HATTEN"; FF; "FEHLVERSUCH(E)"
680 PRINT "UND"; NG; "WORT(E) NICHT GEWUSST"
690 RU = RU + 1
700 PRINT
710 PRINT "WEITER (J/N)"
720 INPUT Z$
730 IF Z$ = "J" THEN 380
740 END
750 :
760 DATA ABEND,EVENING
770 DATA MORGEN,MORNING
780 DATA DU,YOU
790 DATA ER,HE
800 DATA SIE,SHE
810 DATA IST,IS
820 DATA HEUTE,TODAY
830 DATA MANN,MAN
840 DATA FRAU,WOMAN
850 DATA KIND,CHILD
860 DATA HAUS,HOUSE
870 DATA *,*
```

In den Zeilen ab 760 sieht man zunächst die Tabelle der gespeicherten Vokabeln. Hier können Sie natürlich beliebig erweitern, solange der Speicher reicht ...

Nach dem Starten des Programms erscheint ein kleines "Menü", in dem Sie die Art Ihrer Arbeit wählen können. Sie können hier zwischen dem Üben von Vokabeln oder dem Übersetzen wählen.

Das "Übersetzungsprogramm" beginnt dann ab Zeile 230. Geben Sie das gesuchte Wort ein und der Rechner durchsucht dann die gesamte Tabelle, ob das gesuchte Wort darin enthalten ist. Je nach vorheriger Anwahl sucht er das Wort entweder im deutschen (**aw = 2**) oder im englischen (**aw = 3**) Teil. Ist das Wort gespeichert, wird es angezeigt. Andernfalls erscheint eine entsprechende Fehlermeldung.

Das Übungsprogramm ist schon etwas komplizierter: Zunächst wird in den Zeilen 330 bis 360 festgestellt, wieviel Worte gespeichert sind und anschließend eine Tabelle mit dieser Größe angelegt. Der Sinn ist hier, jedes Wort nur einmal abzufragen um das Üben nicht durch ständige Wiederholungen langweilig zu gestalten. Auch wenn wir - wie Sie noch sehen werden - hier wieder mit Zufallszahlen arbeiten, ist die "Wiederholungsrate" ansonsten recht hoch.

In den Zeilen 380 bis 400 werden dann einige Zähler auf Null gesetzt. In Zeile 410 wird eine Zufallszahl zwischen 1 und der Anzahl der Wörter erzeugt und in Zeile 420 überprüft, ob sie in dieser Runde schon mal gezogen wurde. Hierzu dient die zuvor dimensionierte Tabelle, in der jedes bereits bearbeitete Wort vermerkt wird.

Nun wird ihnen ein zufälliges deutsches Wort angezeigt und Sie haben drei Versuche, die richtige Lösung einzugeben. Jeder Fehlversuch wird mitgezählt, desgleichen auch die Anzahl überhaupt nicht erkannter Worte. Das Ganze wiederholt sich, bis alle Worte einmal bearbeitet wurden und endet mit der Ausgabe einer kleinen Statistik, in der Ihnen die Anzahl der Fehlversuche mitgeteilt wird. Anschließend können Sie eine neue Runde beginnen, wobei die Worte dann natürlich in anderer Reihenfolge erscheinen.

Auch hier sind Ihrer Phantasie keine Grenzen gesetzt; probieren Sie doch mal aus, das Programm noch für eine dritte Sprache anzupassen und langsam entsteht dann ein komfortables Programm das keine Wünsche mehr offen lassen dürfte.

SFB-COMPUTER-CLUB
Folge 5: "Computer - und weiter?"

Unser Karteiprogramm:

```
100 DIM T$(100), I$(100), Z$(100)
110 RESTORE
120 IN = 1
130 READ T$(IN), I$(IN), Z$(IN)
140 IF T$(IN) <> "*" THEN IN = IN + 1 : GOTO 130
150 IN = IN - 1
160 PRINT "AUSWAHL : 1 = SUCHEN"
170 PRINT "          2 = SORTIEREN"
180 PRINT "          3 = AUSDRUCKEN"
190 PRINT "          0 = ENDE"
200 INPUT M
210 IF M < 0 OR M > 3 THEN 200
215 IF M = 0 THEN END
220 ON M GOSUB 590, 250, 510
230 GOTO 160
240 :
250 PRINT "WONACH SORTIEREN: 1 = TITEL"
260 PRINT "                  2 = INTERPRET"
270 PRINT "                  3 = SPIELDAUER"
280 INPUT AW
290 FOR I = IN - 1 TO 1 STEP - 1
300 VT = 0
310 FOR J = 1 TO I
320 ON AW GOTO 330, 350, 370
330 IF T$(J) < T$(J+1) THEN 480
340 GOTO 380
350 IF I$(J) < I$(J+1) THEN 480
360 GOTO 380
370 IF Z$(J) < Z$(J+1) THEN 480
380 T$ = T$(J)
390 I$ = I$(J)
400 Z$ = Z$(J)
410 T$(J) = T$(J+1)
420 I$(J) = I$(J+1)
430 Z$(J) = Z$(J+1)
440 T$(J+1) = T$
450 I$(J+1) = I$
460 Z$(J+1) = Z$
470 VT = 1
480 NEXT J
490 IF VT > 0 THEN NEXT I
500 RETURN
510 FOR I = 1 TO IN
520 GOSUB 550
530 NEXT
540 RETURN
550 PRINT T$(I)
560 PRINT I$(I) TAB(20) Z$(I)
570 PRINT
580 RETURN
590 PRINT "WONACH SUCHEN : 1 = TITEL"
600 PRINT "                2 = INTERPRET"
610 PRINT "                3 = SPIELZEIT"
620 INPUT AW
630 ON AW GOSUB 650, 710, 770
640 RETURN
650 INPUT "TITEL"; T$
660 T = LEN(T$)
670 FOR I = 1 TO IN
680 IF LEFT$(T$(I), T) = T$ THEN GOSUB 550
690 NEXT
700 RETURN
710 INPUT "INTERPRET"; I$
720 J = LEN(I$)
730 FOR I = 1 TO IN
740 IF LEFT$(I$(I), J) = I$ THEN GOSUB 550
750 NEXT
760 RETURN
770 INPUT "SPIELZEIT"; SP
775 S1 = INT(SP) : S2 = (SP - S1) * 100 / 60
780 SP = S1 + S2 : DI = 99999
790 IR = 0
800 FOR I = 1 TO IN
810 Z = VAL(Z$(I))
813 Z1 = INT(Z) : Z2 = (Z - Z1) * 100 / 60
816 Z = Z1 + Z2 - SP
820 Z = ABS(Z)
830 IFZ < DI THEN DI = Z : IR = I
840 NEXT
850 I = IR
860 GOSUB 550
870 RETURN
880 DATA "PURPLE RAIN" , "PRINCE" , 8.40
890 DATA "ANARCHY IN UK","THE SEX PISTOLS",2.56
900 DATA "JUST LIKE A VIRGIN" , "MADONNA" , 3.33
910 DATA "DER GUGELHUPF" , "HEINO" , 2.25
911 DATA "MAENNER" , "HERBERT GROENEMEYER",4.18
912 DATA "99 RED BALLOONS","NENA",4.34
913 DATA "TIME AFTER TIME","CYNDI LAUPER",3.50
914 DATA "ONCE IN A LIFETIME","THE TALKING HEADS",4.18
920 DATA "AN DER BLAUEN DONAU" , "ORCH. QUACK" , 2.35
921 DATA "THE ELECTRIC BIRD" , "VIVALDI/WILBRANDT " , 2.57
930 DATA *,*,*
```

Dieses Programm ist schon ein wenig aufwendiger als unsere bisherigen Programmbeispiele. Hier dient es zur Verwaltung einer Plattenkartei; es kann aber auch auf viele andere Anwendungen angepaßt werden, z.B. eine Videokartei oder auch eine

Diasammlung. Auch hier soll es als Anregung dienen, eigene Programme zu erstellen oder auch die vorgestellten Beispiele nach eigenen Wünschen anzupassen.

In den Zeilen ab 880 sehen Sie zunächst die Auflistung der gespeicherten Titel; immer in der Reihenfolge Titel, Interpret und Spielzeit. Das Programm ist im Beispiel auf maximal 100 Titel ausgelegt, kann aber durch Veränderung des **DIM**-Befehls in Zeile 100 auch größere Mengen verkraften, sofern es der Speicher Ihres Rechners zulässt.

Das gesamte Programm ist menügesteuert, bietet die Funktionen Sortieren nach Titel, Interpret oder Spielzeit, Suchen nach einem bestimmten Eintrag und Ausgeben der ermittelten Daten auf dem Bildschirm.

Die Sortierroutine ist Ihnen mittlerweile schon bekannt. Da wir ja jeweils drei zusammenhängende Informationen pro Titel, nämlich Titel, Interpret und Spielzeit haben, müssen auch jeweils alle drei Informationen ihren Platz wechseln.

Das Suchen nach einem bestimmten Eintrag ist recht komfortabel gestaltet. Sie müssen nicht unbedingt den gesamten Suchbegriff eingeben, sondern können auch durch Eingabe von z.B. '**A**' alle Titel, die mit **A** beginnen, auf dem Schirm ausgeben. Hierzu dienen die neuen Befehle **LEN** und **LEFT$**. Der Befehl

```
J = LEN(I$)
```

in Zeile 720 übergibt der Variablen J die Länge des eingegebenen Suchbegriffs I$. In Zeile 740 wird nur in der Länge des Begriffes gesucht mit dem Befehl

```
IF LEFT$(I$(I), J) = I$ THEN ...
```

LEFT$ bedeutet, es wird nur der Anfang der String-Variablen in der Länge J für den Vergleich herangezogen; d.h. wenn Sie hier nur einen Buchstaben eingeben ist die Länge J gleich eins und es wird auch nur auf diesen einen Buchstaben hin abgefragt. Es erscheinen alle Titel, deren Interpreten mit diesem Buchstaben beginnen.

Bei der Suche nach der Spielzeit haben wir uns noch eine Besonderheit einfallen lassen. Es wird hier immer der Titel angegeben, der der gesuchten Spielzeit am nächsten kommt. Der entsprechende Programmteil befindet sich in den Zeilen 770 bis 870. Hier wird zunächst die gesuchte Spielzeit **SP** eingegeben. Nun gibt es ein kleines Problem. Zeiten werden ja immer in Minuten und Sekunden angegeben; die Eingabe von 3.30 bedeutet also 3 Minuten und 30 Sekunden. Der Rechner interpretiert diese Zahl aber als 3,3 - also eigentlich als 3 3/10 Minuten. Zur Korrektur dienen die Zeilen 775 und 780. In Zeile 775 wird der Variablen S1 der ganzzahlige Anteil - in unserem Beispiel die 3 Minuten - zugewiesen und der Variabeln S2 der Rest, also die 0.30. Diese wird dann mit 100 multipliziert und durch 60 - die Anzahl der Sekunden pro Minute - geteilt. Als Ergebnis erhalten wir dann 0,5, d.h. eine halbe Minute, also genau unsere 30 Sekunden.

Anschließend wird die Variable DI auf 99999 gesetzt. Sie gibt uns später die Spielzeit an, die am nächsten an die gesuchte Zeit herankommt. In den Zeilen 810 - 816 müssen wir nun unsere Sekunden-Korrektur natürlich auch für die gespeicherten Werte vornehmen; danach steht in der Variablen Z die Differenz zur gerade gesuchten Zeit. Wenn wir also nach 3.30, also 3 1/2 Minuten suchen (3,50) und der aktuelle Titel hat eine Spielzeit von 3 3/4 Minuten (3,75) enthält die Variable die Differenz 0,25.

Der wiederum neue Befehl **ABS** gibt uns den sogenannten Absolutwert einer Variablen, d.h. er ignoriert ein eventuelles Vorzeichen. Es könnte ja auch ein Titel gespeichert sein, der 3 1/4 Minuten lang ist. Die Differenz wäre dann -0,25. Da wir aber nach der besten Näherung suchen ist -0,25 genauso nahe am gesuchten Wert wie 0,25. Also schneiden wir einfach das Vorzeichen ab und behandeln den Wert wie 0,25.

In Zeile 830 vergleichen wir dann die gefundene Differenz mit der bisher kleinsten DI, am Anfang natürlich mit 99999, und ersetzen sie notfalls durch die neue, sofern sie kleiner ist. Dieses Verfahren haben wir schon einmal in unserem kleinen Statistikprogramm in der 2. Sendung gesehen. In der Variablen IR merken wir uns zusätzlich noch die laufende Nummer des gerade kleinsten Titels.

Wenn alle Titel durchgelaufen sind haben wir dann in DI die geringste Differenz und in IR die Nummer des entsprechenden Titels. In Zeile 850 setzen wir

dann den Ausgabeindex auf diesen Wert und geben in Zeile 860 den gefundenen Titel auf dem Bildschirm aus.

Terminal-Programm

Dieses Programm, das übrigens nur auf dem COMMODORE C64 läuft, erlaubt Ihnen die Kommunikation mit anderen Computern und Mailboxen, sofern Sie über einen Akustik-Koppler verfügen. Es ist wirklich sehr sparsam gehalten - aber fürs Erste reicht's - bis Sie sich mal ein komfortables Terminalprogramm besorgen. Noch ein Hinweis: im Anfangsteil sind alle notwendigen Parameter aufgeführt, die Sie bei Bedarf im Programm direkt verändern müssen; auf eine aufwendige Menüsteuerung haben wir hier verzichtet.

```
100 REM TERMINAL
110 :
120 DB = 32 :REM 7 DATENBITS = 32
                 8 DATENBITS = 0
130 BR = 6  :REM 300 BAUD = 6
                 600 BAUD = 7
                1200 BAUD = 8
140 SB = 0  :REM 1 STOPBIT = 0
                 2 STOPBIT = 128
150 PA = 0  :REM NO PARITY = 0
                 EVEN PAR. = 96
                 ODD PAR.  = 32
160 :
170 OPEN 1,2,0,CHR$(SB+DB+BR) + CHR$(PA)
180 :
190 L$ = CHR$(20)
200 C$ = "-"
210 PRINT C$;
220 GET#1,Z$
230 IF Z$ = "" THEN 310
240 Z = ASC(Z$)
250 IF Z < 65 THEN 300
260 IF Z > 90 THEN 280
270 Z = Z OR 128
280 IF Z < 97 OR Z > 122 THEN 300
290 Z = Z AND 223
300 PRINT L$; CHR$(Z); C$;
310 GET Z$
320 IF Z$ = "" THEN 220
330 Z = ASC(Z$)
340 IF Z < 65 THEN 390
350 IF Z > 90 THEN 370
360 Z = Z OR 32
370 IF Z < 193 OR Z > 218  THEN 390
380 Z = Z AND 127
390 PRINT#1,CHR$(Z);
400 GOTO 220
```

In den Zeilen 120 bis 150 werden zunächst einmal die Übertragungsparameter festgelegt. Auch hier gibt es leider keine Norm, Sie müssen also schon je nach Einsatzzweck die Parameter richtig setzen. In Zeile 190 wird die Variable L$ mit dem Steuerzeichen zum Löschen des letzten Zeichens auf dem Schirm belegt; die Variable C$ in Zeile 200 wird mit einem kleinen Cursor, hier ein Unterstreichungszeichen belegt.

In Zeile 210 bringen wir dann unseren Cursor auf den Schirm und holen uns in der Zeile 220 ein Zeichen vom Akustikkoppler. Wenn kein Zeichen angekommen ist überspringen wir den folgenden Programmteil, andernfalls müssen wir hier noch eine Zeichenwandlung vornehmen. Der Commodore C-64 stellt Zeichen intern auf eine Art dar, die von anderen Rechnern nicht immer verstanden wird. Alle Datenübertragung findet daher im ASCII-Code statt, einem Kode, der schon vor langer Zeit definiert wurde und einer der wenigen Standards ist. Die Zeilen 240 bis 290 wandeln also das empfangene Zeichen im ASCII-Code in Zeichen um, die der C-64 versteht. In Zeile 300 wird dann durch die Ausgabe von L$ der Cursor gelöscht, das Zeichen auf dem Schirm ausgegeben und wieder ein neuer Cursor gesetzt. Der neue Befehl **CHR$(z)** gibt das Zeichen aus, das mit dem Wert **z** intern gespeichert wird. Als Beispiel ist ein **A** intern mit dem Wert 65 gespeichert. Wenn wir also CHR$(65) ausgeben erscheint wieder ein A auf dem Schirm.

In der Zeile 310 holen wir uns dann ein Zeichen von der Tastatur. Wenn hier nichts eingegeben wurde, springen wir gleich wieder zur Eingabe vom Akustikkoppler zurück. Ansonsten muß auch dieses Zeichen wieder gewandelt werden, und zwar diesmal vom C-64 Zeichensatz in den ASCII-Code. In der Zeile 390 wird dann dieses Zeichen zum Akustikkoppler gesandt und wieder zur Zeile 220 gesprungen.

Anzumerken ist noch, daß dieses Programm im sogenannten "Vollduplex-Betrieb" arbeitet. Das bedeutet: wenn Sie ein Zeichen auf der Tastatur eingeben, wird es zunächst zum anderen Rechner geschickt und erscheint erst als Quittung von dort

auf Ihrem Schirm. Das hat den großen Vorteil, daß Sie sofort sehen, ob das Zeichen richtig angekommen ist. Wenn Sie z.B. ein "a" eingeben und auf dem Bildschirm erscheint ein "s", dann kann irgendetwas nicht geklappt haben.

Das Spiel des Lebens (Life-Game)

Hier nun das Programm zu unserem Life-Game. Die Idee zu diesem Spiel stammt übrigens von dem englischen Mathematiker Conway. Abgesehen davon, daß es hübsch aussieht, vermittelt es einen gewissen Eindruck von spielerischen Gesetzmäßigkeiten, die bei der Bevölkerungsentwicklung eine Rolle spielen.

Am Anfang wird eine Startbelegung vorgegeben und daraus errechnen sich dann jeweils weitere Generationen mit zum Teil recht unterschiedlichen Belegungen. Die Regeln, nach denen hier gerechnet wird, sind sehr einfach:

Gespielt wird auf einer Art Schachbrett von beliebiger Größe, wobei hier die Größe natürlich vom Bildschirmformat Ihres Rechners abhängt. Es kann nun eine beliebige Figur gesetzt werden, indem einzelne Felder mit Sternchen vorbelegt werden. Bei der Berechnung der nächsten Generation überlebt jedes Sternchen, wenn es zwei oder drei Nachbarn hat. Bei weniger als zwei Nachbarn stirbt es an Vereinsamung; bei mehr als drei an Übervölkerung. Wenn ein leeres Feld genau drei belegte Nachbarfelder besitzt, wird zur nächsten Generation ein Sternchen 'hinzugeboren'. Trotz dieser doch recht einfachen Regeln ist es erstaunlich, welche Formenvielfalt im Laufe der einzelnen Generationen auftritt.

Die Zeilen 100 und 110 im Listing geben die maximale Größe des Feldes an; je größer das Feld, umso länger braucht das Programm auch zur Berechnung einer neuen Generation.

In den Zeilen 130 und 140 wird dann eine Tabelle mit den Maximalwerten plus 1 definiert. Die zusätzlichen Felder dienen nur zur Begrenzung des eigentlichen Spielfeldes und um eine korrekte Berechnung zu gewährleisten.

In den Zeilen 190 bis 222 wird nun die Startbelegung eingegeben, indem jedes belegte Feld mit **1** vorbelegt wird. Hier können Sie dann beliebig Änderungen vornehmen um eigene Schöpfungen auszuprobieren.

Zeile 240 setzt den Cursor wieder auf die Home-Position; die linke, obere Ecke. Hier muß notfalls, je nach Rechner, eine Änderung vorgenommen werden (siehe auch unsere Vergleichsliste auf der Rückseite des Heftes).

Die Zeilen 280 - 340 geben dann das aktuelle Feld auf dem Bildschirm aus, wobei für jedes nicht belegte Feld ein Leerzeichen und für jedes belegte Feld ein Sternchen auf dem Schirm erscheint. Anschließend wird nun endlich gerechnet. Für jedes einzelne Feld wird zuerst die Anzahl der belegten Nachbarfelder (acht Felder) aus der Tabelle 'al' (alte Generation) ermittelt. Anschließend wird das Feld für die nächste Generation in die Tabelle 'ne' (Neue Generation) übernommen, oder es entfällt; ganz nach den oben genannten Regeln. Dieser Vorgang wiederholt sich für alle Felder, sodaß in unserem Beispiel mit 10 mal 10 Felder schon 800 Berechnungen notwendig sind! Das Programm ist also - leider - nicht sehr schnell...

Die Zeilen 550 - 590 schließlich kopieren abschließend die neue Tabelle in die alte Tabelle. Anschließend wird wieder zur Ausgabe auf dem Schirm gesprungen und das Ganze wiederholt sich, bis Sie die Stop-Taste drücken.

Hier nun das Listing für unser **Life-Game:**

```
100 ZE = 10     :REM ANZAHL ZEILEN AUF SCHIRM
110 SP = 10     :REM ANZAHL SPALTEN AUF SCHIRM
120 :
130 DIM AL(ZE,SP+1)  :REM ALTES FELD SCHIRM-
                      GROESSE ZEILEN, SPALTEN+1
140 DIM NE(ZE,SP+1)  :REM NEUES FELD SCHIRM-
                      GROESSE ZEILEN, SPALTEN+1
150 :
160 REM * VORBELEGUNG DER FELDER *
170 :
190 AL(5,4) = 1
195 AL(4,4) = 1
200 AL(5,5) = 1
210 AL(5,6) = 1
220 AL(6,5) = 1
221 AL(6,4) = 1
222 AL(7,4) = 1
230 :
240 PRINT CHR$(19); :REM CURSOR AUF HOME-POSITION
```

```
250 :
260 REM * ERRECHNETES FELD AUF SCHIRM AUSGEBEN *
270 :
280 FOR I = 1 TO ZE - 1
290 :   FOR J = 1 TO SP
300 :      IF AL(I,J) = 0 THEN
                        PRINT" "; : GOTO 320
310 :      PRINT "*";
320 :   NEXT J
330 :   PRINT
340 NEXT I
350 :
360 REM * NEUE GENERATION ERRECHNEN *
370 :
380 FOR I = 1 TO ZE - 1
390 :   FOR J = 1 TO SP
400 :      AN = 0
410 :      NE(I,J) = 0
420 :      AN = AN + AL(I-1,J-1)
430 :      AN = AN + AL(I-1,J)
440 :      AN = AN + AL(I-1,J+1)
450 :      AN = AN + AL(I,J-1)
460 :      AN = AN + AL(I,J+1)
470 :      AN = AN + AL(I+1,J-1)
480 :      AN = AN + AL(I+1,J)
490 :      AN = AN + AL(I+1,J+1)
500 :      IF AL(I,J) = 0 THEN 520
510 :      IF AN = 2 OR AN = 3 THEN
                       NE(I,J) = 1 : GOTO 530
520 :      IF AN = 3 THEN NE(I,J) = 1
530 :   NEXT J
540 NEXT I
550 FOR I = 0 TO ZE
560 :    FOR J = 1 TO SP
570 :       AL(I,J) = NE(I,J)
580 :    NEXT J
590 NEXT I
600 GOTO 240
```

Wenn Sie häufiger Programme anwenden wollen, in denen viel berechnet werden muß, dann werden Sie bald eine gewisse Schnelligkeit zu schätzen wissen. Aber: BASIC-Programme sind nun einmal "von Natur" aus langsam, weil der Interpreter bei jedem Befehlsschritt ständig die Regeln überprüft - und das hält auf. Vielleicht besorgen Sie sich dann ein COMPILER-Programm, mit dem sich BASIC-Programme compilieren lassen. Die Regeln werden dann nur noch einmal überprüft, nämlich beim Compilieren. Die BASIC-Programme laufen dann aber viel schneller ab.

Das war's, liebe Computerfreunde!

Viel Spaß an der Computerei und beim Selberprogrammieren wünschen

Norbert Demgensky
und
Winfried Göpfert

vom
SFB-COMPUTER-CLUB

** -- *** -- **

Telefon-Computer des WDR:

(0221) 37 10 76

Versandadresse des Begleitmaterials:

Deutsches Video Istitut (DVI)
Budapester Str. 44
D-1000 Berlin 30

Versandadresse der Videocassette:

VIDEAL
Alsterkamp 17
D-2000 Hamburg 13

Hergestellt vom SENDER FREIES BERLIN (mit Unterstützung des Deutschen Video Instituts, Berlin)