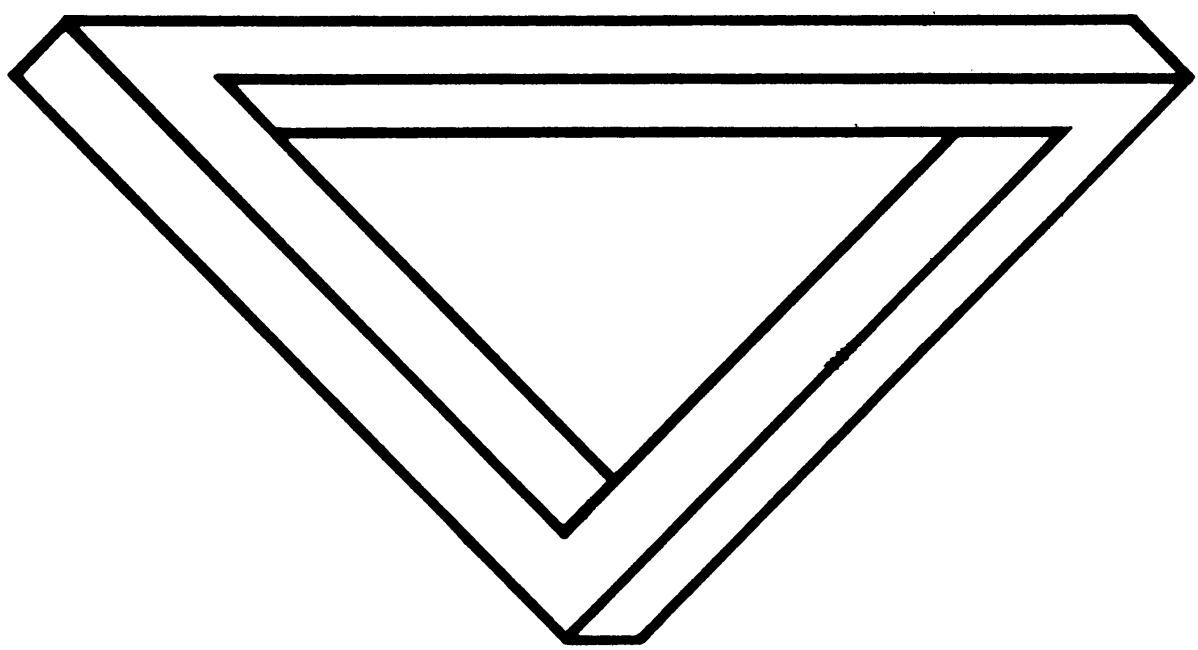
Bernd Seger

Einführung ins GENIE-BASIC



TS COMPUTERSOFT

Einführung ins GENIE-BASIC

Bernd Seger

Zur Beachtung:

TCS Computer GmbH behält sich das Recht vor, Änderungen und Verbesserungen der in diesem Handbuch beschriebenen Software zu jeder Zeit und ohne Ankündigung vorzunehmen.

Bernd Seger

Einführung ins GENIE-BASIC

VORWORT

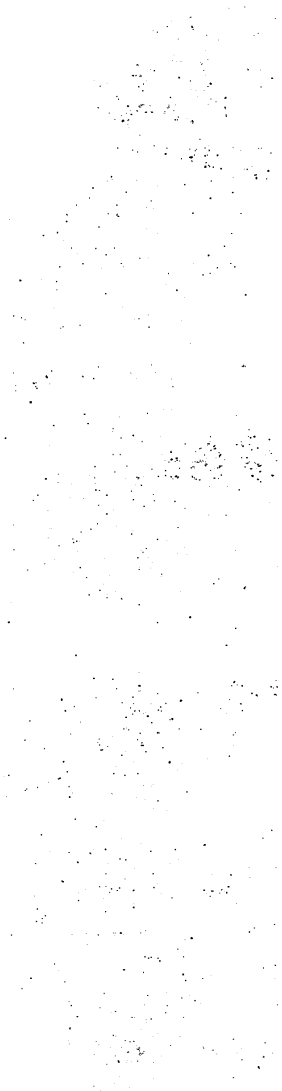
Herzlichen Glückwunsch zu Ihrem neuen GENIE-Computer. Mit dem GENIE haben Sie einen Computer erworben, der Ihnen durch seine hochmoderne Technik Möglichkeiten bietet, die noch vor wenigen Jahren nur für viele tausend Mark zu haben war.

Durch dieses Handbuch werden Sie Schritt für Schritt immer neue Fähigkeiten Ihres Computers kennen und beherrschen lernen. Wir gehen davon aus, daß der Umgang mit einem Computer eine neue Erfahrung für Sie ist. Daher ist dieses Handbuch so ausgelegt, daß Sie die BASIC-Programmiersprache von Grund auf lernen. Denjenigen von Ihnen, die bereits Computererfahrung besitzen oder von einem anderen Computer zum GENIE gewechselt haben, wird vieleicht einiges bekannt vorkommen, aber auch Sie sollten dieses Handbuch durchlesen. Im Anhang werden Sie in Form von Programmbeispielen interessante Tricks finden, die Sie evtl. bei der eigenen Programmierarbeit verwenden können.
Wir hoffen, daß Sie mit diesem Handbuch Ihren Computer beherrschen lernen, damit er ein wertvolles Werkzeug für Sie wird und Ihnen ein Menge Arbeit erspart.

Bonn, im August 1984

Bernd Seger für TCSE COMPUTER GMBH

INHALT



1. EINFÜHRUNG

1.1 Aufbau des GENIE

Bitte beachten Sie folgende Regeln:

1. Das Netzkabel (220 Volt Wechselspannung) ist sorgfältig zu verlegen und darf nicht eingeklemmt werden.

2. Der Monitor ist mit dem mitgelieferten Video-Kabel an den Computer anzuschliessen. Dieses Kabel sollte auch nicht ein geklemmt oder geknickt werden.

3. Die Tastatur ist vorsichtig mit dem Flachkabel zu verbinden. Achten Sie dabei auf den richtigen Anschluß des Kabels an das Tastaturkabel und wenden Sie keine Gewalt an. Andernfalls könnten die empfindlichen Kontaktstifte an den Steckern verbiegen oder abbrechen.

4. Schützen Ihren Computer vor Feuchtigkeit, Staub, direkter Sonneneinstrahlung oder übermäßiger Hitze.

5. Ihr Rechner (GENIE IIs, III oder IIIs) ist mit einem Lüfter ausgestattet, der eine Überhitzung ausschließt. Achten Sie jedoch darauf, daß die Luftschlitze des Gerätes frei bleiben.

6. Achten Sie darauf, daß das Gerät beim Transport keine harten Stöße abbekommt.

7. Wenn Sie den Computer reinigen wollen, benutzen Sie bitte nur handelsübliche Reinigungsmittel.

8. * Behandeln Sie die Disketten immer sorgfältig

9. * WICHTIG: Beim Ein- oder Ausschalten des Gerätes dürfen NIE die Disketten im Laufwerk sein.

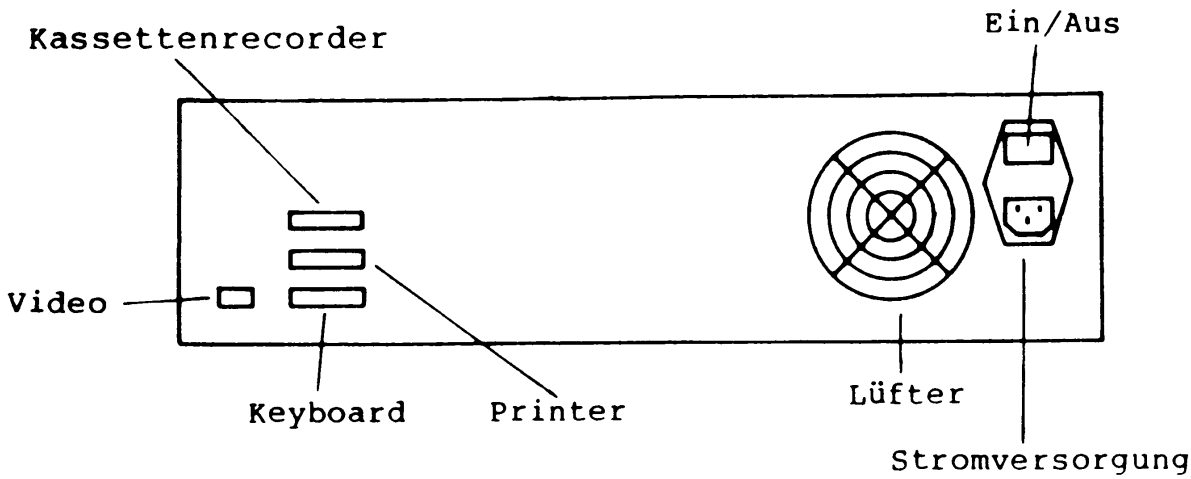
10. * Die Laufwerke sind wie folgt numeriert:

    GENIE IIs/IIIs:
    oberes Laufwerk = Drive 0
    unteres Laufwerk = Drive 1
    (externe Laufwerke = Drive 2 und 3

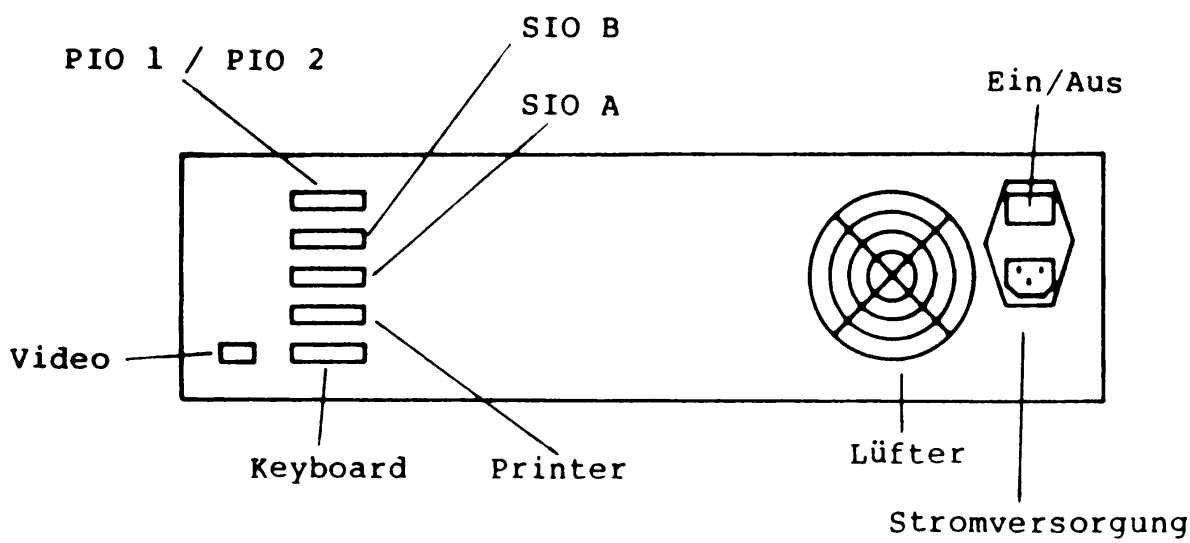
    GENIE III:
    linkes Laufwerk = Drive 0
    rechtes Laufwerk = Drive 1
    (externe Laufwerke = Drive 2 und 3

Die Angaben, die mit einem * beginnen, gelten nur für Rechner mit Diskettenlaufwerken.

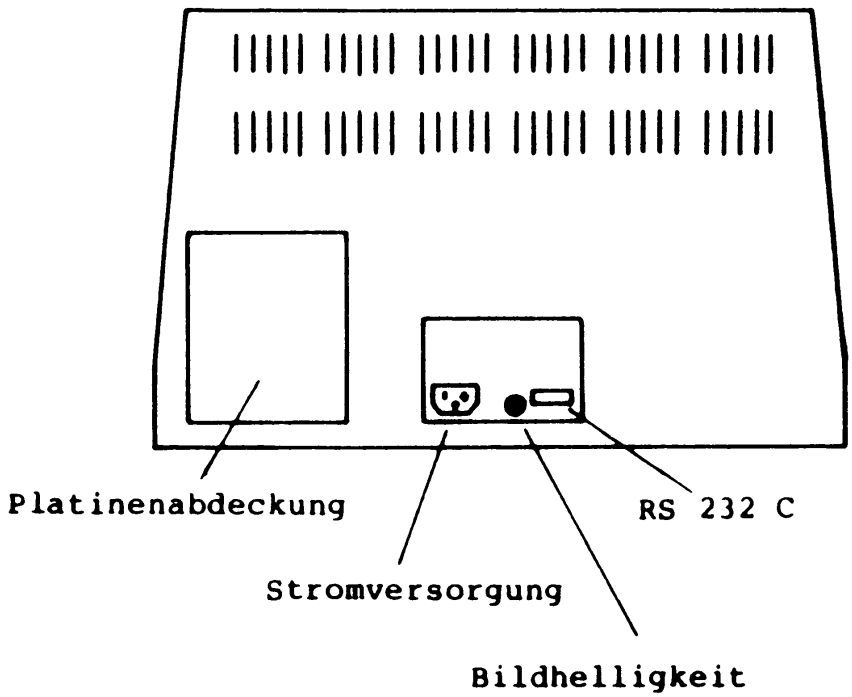
1.2 Anschlüsse des GENIE IIs

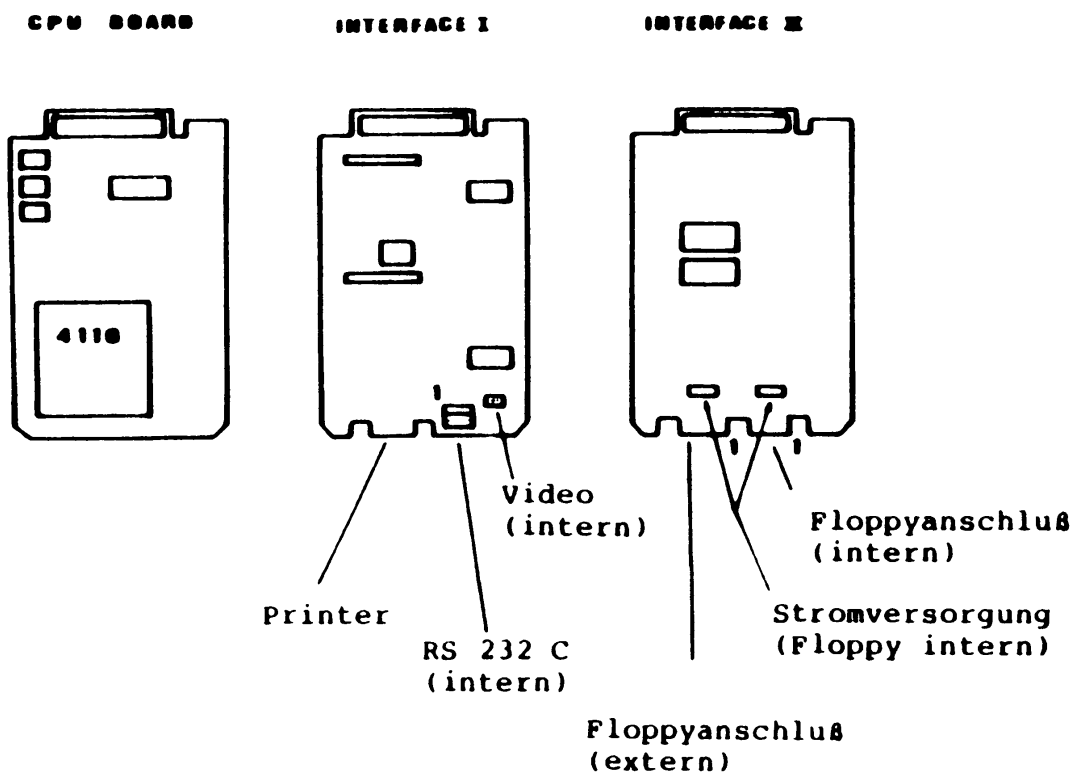


1.3 Anschlüsse des GENIE IIIs



1.4 Anschlüsse des GENIE III





2. BEHANDLUNG UND HANDHABUNG

2.1 Behandlung und Handhabung
(gilt nur für Rechner mit Diskstation)

Obwohl Disketten verhältnismäßig unempfindlich sind, sollten Sie doch einige Regeln beim Umgang mit ihnen beachten, um keine Pannen beim Betrieb zu erleben. Sie müssen wissen, daß die Aufzeichnungen sehr dicht erfolgen. So befinden sich zum Beispiel auf einer Spurlänge von nur ca. 10 mm mehrere tausend Einzelinformationen (Bits). Wenn aber auch nur ein Bit durch Verschmutzung oder Beschädigung unlesbar ist, so kann es vorkommen, daß ein ganzes Programm oder sogar die komplette Diskette nicht mehr gelesen werden kann.

Deshalb sollten Disketten immer in ihrer Papierschutzhülle bleiben, wenn sie sich nicht gerade in einem Laufwerk befinden.
Eine Diskette legen Sie in ein Laufwerk ein, indem Sie sie waagerecht, mit dem Aufkleber nach oben und dem ovalen Leseschlitz nach vorne in das Laufwerk einschieben.
Achten Sie dabei darauf, daß sie nicht geknickt wird.

Folgende Hinweise sollten Sie unbedingt befolgen:

* Die Diskette darf niemals am Leseschlitz berührt werden.

* Die Diskette sollte immer in ihrer Schutzhülle aufbewahrt werden

* Die Diskette darf niemals in die Nähe eines magnetischen Feldes, wie es z.B. bei Netzgeräten, Monitoren, Telefonen und Lautsprechern vorhanden ist, geraten.

* Die Diskette darf niemals direkter Sonneneinstrahlung oder anderen Wärmequellen ausgesetzt werden (max. 50 Grad).

* Die Diskette darf niemals geknickt, gefaltet oder gebogen werden

* Die Diskette darf nicht mit einem Kugelschreiber oder hartem Bleistift auf dem Etikett beschrieben werden

3. BASIC

3.1 Generelle Informationen über das GENIE-BASIC

Die Programmiersprache BASIC ist für das GENIE gewählt worden, da sie hochentwickelt, weitverbreitet und allgemein als leicht erlernbar gilt.

3.2 Die ersten Schritte

Wir wollen folgendermaßen vorgehen:
Zuerst werden Sie sehen, wie man mit dem GENIE ganz normal rechnen kann. Anschließend werden Sie die Funktionen von Variablen und die Grundzüge der BASIC-Programmierung kennenlernen.

Nachdem Sie sich mit den Tastenfunktionen Ihres GENIE vertraut gemacht haben, werden die fundamentalen BASIC-Befehle in aller Ausführlichkeit erklärt. Mit der Übung, die Sie dann haben werden, können Sie die restlichen Befehle und Funktionen in relativ kurzer Zeit erlernen.

Um ins BASIC des Computers zu gelangen, ist folgendes durchzuführen:

Beim GENIE IIs liegt der BASIC-Interpreter als ROM-Inhalt vor. Aus diesem Grunde können Sie beim GENIE IIs sofort nach dem Einschalten des Gerätes in BASIC programmieren.

Beim Anschluß eines Diskettenlaufwerks an den GENIE IIs oder bei den Rechnern GENIE III/IIIs und Speedmaster muß das BASIC erst von der Diskette in den Speicher des Rechners geladen werden.

Sie arbeiten in diesem Fall mit dem Betriebssystem G-DOS. Nach dem Einschalten des Computers muß die G-DOS-Diskette gebootet werden. Unter dem Wort 'booten' versteht man das erste Laden des Betriebssystems von der Systemdiskette in den Arbeitsspeicher des Computers.

Um G-DOS zu booten, schalten Sie den Computer ein und warten solange, bis er einen automatisch ablaufenden Speichertest abgeschlossen hat.
Nun können Sie die G-DOS-Diskette in Laufwerk 0 einlegen und die Klappe des Laufwerks schließen. Drücken Sie gleichzeitig die beiden 'RESET'-Tasten auf der Tastatur.
Das DOS wird nun von der System-Diskette in den Speicher des Rechners übertragen.
Abschießend erfolgt auf dem Bildschirm die Meldung:

BEFEHLSEINGABE:

Um nun in das BASIC zu gelangen, müssen Sie einfach das Wort:

BASIC

eingeben.

Der GENIE IIs meldet sich dann mit:

```
GENIE IIs - B A S I C / erweiterte Version für Diskettenbetrieb

unter G-DOS 3.0

READY
>
```

und der GENIE IIIs mit:

```
GENIE IIIs - B A S I C / erweiterte Version für  Diskettenbetrieb

unter G-DOS 2.1c

READY
>
```

Beim GENIE III und Speedmaster erfolgt eine entsprechende Meldung.

Nun sind Sie im erweiterten Disk-BASIC des Computers und der Rechner befindet sich im Eingabemodus, der durch die Meldung 'READY' angezeigt wird.

Das '>'-Zeichen, der sogenannte Prompt, bedeutet, daß hier eine Eingabe erwartet wird. Das blinkende Quadrat, der sogenannte Cursor, zeigt an, wo das nächste Zeichen, das Sie eingeben, erscheinen wird. Drücken Sie z.B. eine Buchstabentaste, so sehen Sie wie der Buchstabe auf dem Bildschirm erscheint und der Cursor eine Stelle weiterrückt. Wenn Sie nun den Buchstaben wieder löschen wollen, so drücken Sie die 'LINKSPFEIL'-Taste.

3.3 Rechnen mit dem GENIE

Geben Sie folgendes ein:

```
>PRINT 12+35
```

und drücken dann die <Return>-Taste. Die <RETURN>-Taste, (auch NEW LINE- oder ENTER-Taste genannt), ist die etwas größere Taste auf der rechten Seite der Tastatur. Die Taste kann auch mit einem Pfeil gekennzeichnet sein, der erst nach unten und dann nach links zeigt.
Wenn Sie diese Taste gedrückt haben, gibt der Rechner das Ergebnis an:

```
47
READY
>
```

Sehen wir uns nun Ihre Eingabe einmal genau an.

Zuerst das Wort 'PRINT':
Dieses Wort, genauer gesagt dieser Befehl, bedeutet, daß der Computer alles folgende berechnen und dann ausgeben soll.

Die <RETURN>-Taste besagt, daß die Eingabe nun bearbeitet werden soll. Da diese Taste hinter jeder kompletten Eingabe gedrückt werden muß, wird sie im folgenden nicht wieder erwähnt werden.

Bevor wir uns mit den anderen Rechenarten beschäftigen, wollen wir Sie noch eine auf eine Vereinfachung hinweisen:

Da der 'PRINT'-Befehl sehr oft gebraucht wird, kann man ihn auch durch ein einfaches '?' ersetzen.

Folgende zwei Befehle sind also vollkommen gleichbedeutend:

```
>PRINT 7+145
 152
>READY
>
```

```
>? 7+145
 152
>READY
>
```

Nun aber zu den anderen Rechenarten:

Subtrakion:

(Achtung: verwechseln Sie auf keinen Fall die Null (0) mit dem Buchstaben (O)!)
Geben Sie folgendes ein:

```
>PRINT 12-20
 -8
READY
>
```

Multiplikation:

Multipliziert wird mit dem Stern (*):

```
>PRINT 3*37
 111
READY
>
```

Division:

Dividiert wird mit den Schrägstrich (/):

```
>PRINT 30/12
 2.5
READY
>
```

Wenn Sie genau hinsehen, bemerken Sie, daß im Ergebnis der Division kein Komma, sondern ein Punkt als Trennzeichen auftritt. Dies entspricht der amerikanischen Schreibweise. Das gleiche gilt auch bei Eingabe von Kommazahlen - auch hier muß ein Punkt anstelle des Kommas gesetzt werden.

Folgendes Beispiel ist also richtig:

```
>PRINT 21.9/7.3
 3
READY
>
```

Zur Division muß noch gesagt werden, daß sich das </>-Zeichen nur auf die unmittelbar benachbarten Zahlen bezieht - es ersetzt also nicht den langen Bruchstrich.

```
>PRINT 2+3/4+5
```

entspricht also dem Term

$$2 + \frac{3}{4} + 5$$

Wenn Sie nun aber :

$$\frac{2+3}{4+5}$$

berechnen wollen, so müssen Sie Klammern setzen.

Das sieht dann so aus:

```
>PRINT (2+3)/(4+5)
```

Potenzieren:

Geben Sie folgenden Befehl ein:

```
>PRINT 2Ä6
 64
READY
>
(2Ä6=2*2*2*2*2*2=64)
```

Das Potenzierungszeichen 'Ä' erhalten Sie durch Eingabe der <PFEILAUFWÄRTS>-Taste oder durch Betätigung der Taste 'Ä'.

Die Potenz, im obigen Beispiel 6, wird nicht wie üblich als Hochzahl eingegeben, da dies auf dem Bildschirm schwer darzustellen wäre, sondern das 'Ä'-Zeichen legt die folgende Potenz fest.

Durch das Potenzieren kann man auch Wurzeln ziehen, denn es gilt:

$$X^{\frac{1}{3}} = \sqrt[3]{X} \qquad \text{z.B.:} \qquad \sqrt[3]{27} = 27^{\frac{1}{3}} = 3$$

Computerbefehl:

```
>PRINT 27Ä(1/3)
 3
READY
>
```

Für die Quadratwurzelfunktion gibt es jedoch den eigenen Befehl SQR (x).

Im letzten Beispiel mussten wieder Klammern gesetzt werden, um das richtige Ergebnis zu erhalten.

Der Befehl ohne Klammern:

```
>PRINT27Ä1/3
```

hätte nämlich

$$\frac{27}{3} = 9$$

berechnet, da Potenzieren Vorrang vor Dividieren hat. In diesem Zusammenhang spricht man von 'Rechenhierarchie', d.h. einer Vorrangsregelung, die angibt, in welcher Reihenfolge verschiedene Rechenschritte abgearbeitet werden.

Diese Rechenhierarchie sieht in BASIC wie folgt aus:

- zuerst werden in Klammern stehende Ausdrücke berechnet. Dabei wird mit der innersten Klammer begonnen.
- Bei den Rechenoperationen gilt die Rangfolge: Potenzierung, dann Multiplikation oder Division und zuletzt Addition oder Subtraktion.
- Gleichrangige Ausdrücke werden von links nach rechts abgearbeitet.

Man muß immer ebensoviele Klammern schliessen, wie man geöffnet hat, sonst gibt der Computer eine Fehlermeldung aus. Bei der Umwandlung von komplizierten Formeln in BASIC muß man sehr aufmerksam sein - eine falsch gesetzte oder fehlende Klammer kann ein völlig absurdes Ergebnis verschulden.

Geben Sie folgendes ein:

```
>PRINT999999+1
 1E+6
READY
>
```

Das Ergebnis dieser Operation sagt uns, daß alle Zahlen, die größer als 999999 oder kleiner als 0.01 sind, in Exponentialschreibweise ausgegeben werden. D.h., sie werden in Mantisse (Anteil vor dem 'E') und Exponent (Anteil nach dem 'E') aufgespalten, damit die Ausgabe nicht zu lang wird.

An dieser Stelle möchte ich noch 2 Beispiele bringen, die zeigen, wie man die Exponentialschreibweise aufzuschlüsseln hat:

3.5E+06 = 3.5*10 = 3.5*10*10*10*10*10*10 = 3500000

3.5E-04 = 3.5*10 = 3.5/(10*10*10*10) = 0.00035

Das GENIE hat einen Rechenbereich von 1.70141E-38 bis 1.70141E+38, d.h., Sie können bis zu 39stellige Zahlen verarbeiten.

Normalerweise wird mit 6 Stellen Genauigkeit gerechnet. Es gibt jedoch die Möglichkeit mit 16 Stellen Genauigkeit zu rechnen, doch davon mehr im nächsten Kapitel, das sich mit dem Thema Variablen beschäftigen wird.

Da die Rechengenauigkeit des GENIE beschränkt ist, kann es zu sogenannten Rundungsfehlern kommen.

Ein Beispiel:

```
>PRINT 3Ä4
 81.0001
READY
>
```

Richtig wäre natürlich 81 und nicht 81.0001. Sie sollten Ihr Ergebnis immer dahingehend überprüfen, auch wenn diese Fehler nicht oft auftauchen.

3.4 Variablen

Variablen spielen bei der Programmierung in der BASIC-Computersprache eine wichtige Rolle. Mit dem Wort 'Variable' wird eigentlich ein ganz bestimmter Speicherraum, der durch den Namen der Variablen gekennzeichnet ist, benannt. Dieser Speicherraum wird im Hauptspeicher des Computers in dem Augenblick reserviert, in dem dieser Variablen im Programm zum ersten Mal ein bestimmter Wert zugewiesen wird. Der Zuweisungsbefehl sieht allgemein so aus:

```
LET Variable = zuzuweisender Wert
```

(Der Befehl 'LET' kann allerdings auch weggelassen werden.)

Beispiel:

```
>LET A=5
READY
>
```

Damit ist der Variablen A der Wert 5 zugewiesen. Dies kann man folgendermaßen überprüfen:

```
>PRINT A
 5
READY
>
```

Geben Sie nun ein:

```
>PRINT B
 0
READY
>
```

Der Variablen B ist noch kein Wert zugeordnet worden, deshalb wird sie mit 0 angegeben.

Man kann einer Variablen auch den Wert einer Berechnung zuweisen.

Ein Beispiel:

```
>B=A+10*15-1
READY
>PRINT B
 154
READY
>
```

Eine Variable kann auch in einer eigenen Zuweisung benutzt werden:

```
>C=10
READY
>C=C+1
READY
>PRINT C
 11
READY
>
```

Der Befehl 'C=C+1' ist mathematisch natürlich keine richtige Gleichung, daher sagt man am besten: C wird zu C plus eins.

Geben Sie folgendes ein:

```
>TEST=0.009
READY
>PRINT TEST
 9E-03
READY
>
```

Wie Sie sehen kann man Variablen auch längere Namen geben.

Dabei gilt allgemein:

- Das erste Zeichen muß ein Buchstabe sein.

- Alle folgenden Zeichen können wahlweise ein Buchstabe oder eine Zahl sein, alle anderen Zeichen sind unzulässig.

- Zur Unterscheidung dienen nur die ersten zwei Zeichen, d.h. der Computer kann z.B. die Variablen 'TEST' und 'TEXT' nicht unterscheiden!

- In einen Variablennamem dürfen keine BASIC-Schlüsselwörter, also keine Befehlswörter vorkommen. Der Variablenname 'XPRINTY' ist z.B. unzulässig, weil er den Befehl 'PRINT' enthält.

Es gibt zwei grundsätzlich verschiedene Arten von Variablen:

1. Zahlen-Variablen (auch numerische Variablen genannt)

   Wenn der Computer eine Zahlen-Variable findet, entnimmt er deren Zahlenwert aus dem Speicher und kann dann diesen Wert in einem Rechenvorgang verarbeiten.

   Bei den Zahlenvariablen gibt es 3 verschiedene Typen:

   - Einfache Genauigkeit (6 Stellen)
   - Doppelte Genauigkeit (16 Stellen)
   - Integer-Variablen (Ganzzahl-Variablen)

   Variablen mit einfacher Genauigkeit:

   Alle Variablen, die wir bis jetzt benutzt haben waren Variablen mit einfacher (=6stelliger) Genauigkeit.

   Normalerweise sind alle Variablen, die hinter ihrem Namen keine besondere Kennzeichnung besitzen von einfacher Genauigkeit. Wie Sie jedoch später sehen werden, gibt es die Möglichkeit, Variablen per Befehl auf andere Genauigkeit zu definieren. Wenn man in diesem Falle noch mit einfacher Genauigkeit rechnen will, muß dies durch ein '!' hinter dem Variablennamen angegeben werden.

Ist dies nicht der Fall, sind Variablen mit und ohne '!' identisch.

Beispiel:

```
>A!=3.456
READY
>PRINT A
 3.456
READY
>
```

Variablen mit doppelter Genauigkeit:

Variablen mit doppelter Genauigkeit rechnen auf 16 Stellen genau. Sie werden durch ein '#' hinter dem Variablennahmen gekennzeichnet.

Geben Sie folgendes ein:

```
>A#=1/3
READY
>PRINT A#
 .3333333432674408
READY
>
```

Das Ergebnis ist zwar 16stellig, aber nach der 7. Stelle falsch!
Dies liegt an einer Eigenart des GENIE-BASIC!

Wenn ein Ergebnis mit 16stelliger Genauigkeit erreicht werden soll, muß mindestens ein Teil der Zuweisung ebenfalls 16stellig genau definiert sein. Man kann nun auch eine Zahl als als 16stellig genau definieren, indem man ein '#' anfügt.

So bekommen wir das Ergebnis der obigen Rechnung:

```
>A#=1#/3#
READY
>PRINT A#
 .3333333333333333
READY
>
```

Dasselbe geht auch ohne vorherige Variablenzuweisung:

```
>PRINT 1#/3#
 .3333333333333333
READY
>
```

Der Nachteil von Variablen mit doppelter Genauigkeit ist, daß sie mehr Speicher benötigen und relativ langsam berechnet werden.

Noch ein Hinweis:

Bei Exponentialdarstellung erfolgt bei Variablen mit doppelter Genauigkeit die Ausgabe 'D' statt 'E'.

Integer-Variablen:

Integer-Variablen, auch Ganzzahlvariablen genannt, werden durch ein '%'Zeichen hinter dem Namen gekennzeichnet. Der Bereich von Integer-Variablen reicht von -32768 bis +32767.
Nachkommastellen sind nicht möglich. dafür haben Integer-Variablen den Vorteil, daß sie erheblich schneller als die anderen beiden Zahlenvariablentypen berechnet werden und außerdem weniger Speicher benutzten.

Ein Beispiel für Integer-Variablen:

```
>A%=3.9
READY
>PRINT A%
 3
READY
>
```

Sie sehen, daß der Nachkommaanteil einfach weggelassen wird, d.h., es wird nicht gerundet!

Hier noch einmal die 3 verschiedenen numerischen Variablen auf einen Blick:

A , A!	6 Stellen
A#	16 Stellen
A%	Ganzzahlen

Dabei sind die Variablen A, A#, A% drei völlig voneinander unabhängige Variablen:

```
>A=15.6
READY
>A#=2.34567890123
READY
>A%=200
READY
>PRINT A
 15.6
READY
>PRINT A#
 2.34567890123
READY
>PRINT A%
 200
READY
>
```

Zeichenketten-Variablen:

Zeichenketten-Variablen, auch String-Variablen genannt dienen zur Abspeicherung beliebiger Texte bis zu 255 Zeichen Länge. Rechnen kann man mit ihnen nicht.

Eine Variable wird als Zeichenketten-Variable festgelegt, indem man direkt hinter den Namen ein '$'-Zeichen setzt.

Beispiel:

```
>A$="Text"
READY
>PRINT A$
Text
READY
>
```

Wie Sie sehen, wird der Text bei der Zuweisung in Anführungszeichen gesetzt. Deshalb darf im Text selbst auch kein Anführungszeichen vorkommen!

Die maximale Länge des Textes, der in einer Zeichenketten-Variablen gespeichert werden kann, beträgt 255 Zeichen. Jedoch reserviert der Computer beim Einschalten nur für 50 Zeichen Speicherplatz.
Es gibt allerdings die Möglichkeit, diesen Speicherplatz individuell für das Programm neu zu definieren.

3.5 Programmierung

Bis jetzt haben Sie alle Befehle so eingegeben, daß sie vom Computer direkt ausgeführt wurden. Diese Betriebsart wird man immer dann wählen, wenn man schnell irgendeine Rechnung ausführen will. Die weitaus wichtigeren Betriebsarten sind aber die Programmeingabe und der Programmablauf.

Im Prinzip besteht ein BASIC-Programm aus einer oder mehreren Programmzeilen, von denen jede einen oder mehrere Befehle enthält. Am Anfang jeder Programmzeile muß eine Zeilennummer stehen. Diese Programmzeilen werden dann von dem Computer der Reihe nach automatisch abgearbeitet.

Dazu ein Beispiel:

```
>10 A=10
>20 A$="TEXT"
>30 PRINT A$
>40 PRINT A
```

Das Programm ist damit eingegeben.
Geben Sie nun folgenden Befehl ein:

```
>LIST
10 A=10
20 A$="TEXT"
30 PRINT A$
40 PRINT A
READY
>
```

Wie Sie sehen, wird durch den 'LIST'-Befehl das eingegebene Programm aufgelistet.

Gestartet wird das Programm mit dem 'RUN'-Befehl:

```
>RUN
TEXT
 10
READY
>
```

Der Computer hat damit alle 4 Befehle hintereinander ausgeführt.
In Zeile 10 hat er A den Wert 10 zugewiesen und in Zeile 20 A$ die Zeichenkette 'TEXT'. Zeile 30 gibt er dann den Text von A$ aus und in Zeile 40 den Wert von A.

Wenn Sie bei der Programmeingabe eine Zeile falsch eingegeben haben, so geben Sie die entsprechende Zeile mit der gleichen Zeilennummer einfach neu ein. Die alte Zeile wird dabei automatisch gelöscht.

Das vorangegangene Beispielprogramm hat die Zeilennummern 10, 20, 30 und 40.

Im GENIE-BASIC kann man Zeilennummern von 0 bis 65529 benutzen. Bei der Wahl von Zeilennummern sollte man auf jeden Fall zwischen den einzelnen Zeilennummern einen gewissen Abstand lassen, im obigen Programm ist z.B. ein Abstand von 10 zwischen zwei Zeilennummern.

Die Programmierweise hat folgenden Vorteil: Wenn zwischen zwei Zeilennummern noch Platz ist, kann man dort jederzeit neue Zeilen einfügen.
Das können Sie mit dem Programm ausprobieren, daß Sie zuletzt eingegeben haben. Angenommen Sie wollen noch zwei Zeilen einfügen, nämlich Zeile 15, die der Variablen B1 den Wert 2.65 zuweist, und Zeile 25, die diesen dann ausgibt. Dann geben Sie folgendes ein:

```
>15 B1=2.65
>25 PRINT B1
>LIST
10 A=10
15 B1=2.65
20 A$="TEXT"
25 PRINT B1
30 PRINT A$
40 PRINT A
READY
>
```

Mit dem 'LIST'-Befehl sieht man, daß beide Zeilen eingefügt worden sind.
Wenn Sie das Programm starten, werden alle drei Werte ausgegeben.

```
>RUN
2.65
TEXT
10
READY
>
```

Wenn Sie eine Zeile löschen wollen, geben Sie einfach DELETE (für Löschen) und dann die Zeilennummer ein.
Wenn Sie z.B. die Zeile 40 löschen wollen, geben Sie einfach DELETE 40 ein:

```
>DELETE 40
READY
>LIST
10 A=10
15 B1=2.65
20 A$="TEXT"
25 PRINTB1
30 PRINTA$
READY
>
```

Der 'LIST'-Befehl zeigt, daß die Zeile 40 gelöscht wurde.

Um das ganze Programm zu löschen, gibt man den Befehl 'NEW' ein:

```
>NEW
(Der Bildschrim wird gelöscht)
READY
>LIST
READY
>
```

Dabei werden sowohl das Programm und auch alle Variablen gelöscht.

Wie schon erwähnt, lassen sich auch mehrere Befehle in eine Zeile bringen. Dies geschieht einfach dadurch, daß man die einzelnen Befehle durch einen Doppelpunkt trennt. Der Vorteil der Aneinanderreihung von Befehlen in einer Zeile liegt darin, daß Speicherplatz eingespart wird, da der Computer weniger Zeilennummern speichern muß.

Ein Beispiel:

Geben Sie vor der Programmeingabe den Befehl 'NEW' ein, um ein eventuell noch vorhandenes Programm zu löschen.

```
>10 A=10 : B1=2.65 : A$="TEXT" : PRINT B1 :
PRINT A$ : PRINT A
>RUN
 2.65
TEXT
 10
READY
>
```

Dieses Programm läuft genauso wie das Programm, bei dem Sie alle Befehle in einzelne Zeilen geschrieben haben.

Noch ein Wort zu den Leerzeichen in Programmzeilen:
Prinzipiell sind zwischen Befehlen, Variablen etc. keine Leerzeichen erforderlich.

Folgende Zeile führt also genau die Befehle aus, wie im letzten Beispiel mit Leerzeichen:

```
>10A=10:B1=2.65:A$="TEXT:PRINTB1:PRINTA$:
PRINTA
```

Ein Programm bleibt jedoch übersichtlicher, wenn man Leerzeichen einfügt. Der Nachteil ist, daß auch Leerzeichen Speicherplatz belegen. Deshalb wird man sie in langen Programmen eher weglassen, um Speicher zu sparen. In Texten darf man die Leerzeichen selbstversändlich nicht weglassen.

Wenn Sie dieses Handbuch bis hierhin aufmerksam gelesen haben, sind Sie nun mit den Grundprinzipien der BASIC-Programmierung gut vertraut.
Was Ihnen aber jetzt noch fehlt, sind die Kenntnisse neuer Befehle. Deshalb sind die nächsten Kapitel nicht mehr ganz so ausführlich.

3.6 Der PRINT-Befehl

Der 'PRINT'-Befehl ist der Ausgabebefehl für den Computer. Er weist den Computer an, das, was hinter dem 'PRINT'-Befehl steht, als Anzeige auf dem Bildschirm auszugeben.

Dabei gibt es grundsätzlich zwei Möglichkeiten. Folgt nach dem 'PRINT'-Befehl ein Anführungszeichen, so bedeutet dies, daß anschließend eine Zeichenkette kommt. In diesem Fall wird also die komplette Zeichenkette bis zu den abschließenden Anführungszeichen zur Anzeige gebracht.

Steht nach dem 'PRINT'-Befehl kein Anführungszeichen, betrachtet der Computer alles folgende als Rechenausdruck. Er berechnet das dazugehörige Ergebnis und bringt dies zur Anzeige.

Steht nach dem 'PRINT'-Befehl eine Variable, wird der unter ihrem Namen abgespeicherte Wert ausgegeben oder in die Rechnung mit einbezogen.

Eine besondere Rolle nach einem 'PRINT'-Befehl spielen die Satzzeichen Semikolon ';' und Komma ',', denn mit ihrer Hilfe können mehrere Ausgaben durch einen Befehl veranlaßt werden.

- wird ein Semikolon zur Trennung verwendet, so wird dicht hintereinander ausgegeben.
- bei der Trennung durch ein Komma wird in Kolonnen mit jeweils 10 Zeichen Abstand angezeigt.

Der 'PRINT'-Befehl ohne nachfolgendes Argument erzeugt eine Leerzeile. Da der Doppelpunkt verwendet werden kann, um mehrere Befehle in eine Zeile zu schreiben, erzeugen mehrere 'PRINT'Befehle durch Doppelpunkt getrennt mehrere Leerzeilen.

Der Zeilentransport nach einem 'PRINT'-Befehl kann durch ein Semikolon am Ende der Zeile unterdrückt werden.

```
>10 A=25 : B=10
>20 PRINT A
>30 PRINT A*5
>40 PRINT A/B
>50 PRINT "A/B"
>60 PRINT "A/B
>RUN
 25
 125
 2.5
A/B
A/B
READY
>
```

```
>10 A=5 : B=8
>20 PRINT A;B
>30 PRINT
>40 PRINT A,B
>50 PRINT A;
>60 PRINT B
>RUN
 5 8

 5          8
 5 8
READY
>
```

Die Möglichkeiten des 'PRINT'-Befehls

Bedeutung der Satzzeichen

Nun müssen Sie noch einige Besonderheiten bei den Trennungszeichen innerhalb von 'PRINT'-Befehlen beachten.

Bei der Ausgabe von positiven Zahlen wird vor die Zahl eine Leerstelle gesetzt, in die bei einer negativen Zahl das Minuszeichen kommt. Auch nach jeder Zahl erfolgt die Ausgabe einer Leerstelle.

Textstellen, die als Zeichenketten behandelt werden sollen, müssen in Anführungszeichen stehen. Alle Leerzeichen innerhalb der Anführungszeichen werden auch als Leerzeichen ausgegeben. Von dem Computer werden vor und hinter Zeichenketten keine Leerzeichen gesetzt.

Ist die in einer Kolonne auszugebende Information länger als 10 Zeichen, so wird in die nächste Kolonne hineigeschrieben. Die nächste Ausgabe erfolgt dann beim nächstfolgenden Kolonnenanfang.

Wenn Sie eine Ausgabe in Kolonnen programmieren, sollten Sie sicher sein, daß niemals mehr als 10 Zeichen auszugeben sind.

Beispiel:

```
>10 A = 10 : B = -12
>20 A$ = "KETTE" : B$ = " MIT "
>30 PRINT A; A; B; B; A; B
>40 PRINT A$; A$; A$; B$; A$
>RUN
  10  10 -12 -12  10 -12
KETTEKETTEKETTE MIT KETTE
READY
>
```

Engausgabe.

```
>10 A$ = "123456789"
>20 PRINT A$, A$
>RUN
 123456789   123456789
READY
>
```

Kolonnenausgabe mit maximal 9 Zeichen in einer Kolonne.

Sie können aber auch wie mit dem Tabulator der Schreibmaschine die Anfänge der Tabellenspalten selbst bestimmen. Hierzu dient die 'TAB(n)'-Funktion. Wenn Sie in die Klammer eine Zahl zwischen 0 und 63 (79) setzen und diese Funktion vor den auszugebenden Ausdruck schreiben, beginnt die Ausgabe erst in der durch die Zahl genannte Spalte +1. Zwischen 'TAB' und der Klammer darf kein Zwischenraum sein!

Bei Ihrem GENIE gibt es dann noch einen weiteren sehr nützlichen Befehl, mit dem die Bildschirmausgabe an einer beliebigen Stelle programmiert werden kann. Der Bildschirm des GENIE's zeigt 16 Zeilen mit je 64 Zeichen (bzw. 24 Zeilen mit je 80 Zeichen). Es gibt also insgesamt 1024 (1920) Anzeigepositionen, die in der linken oberen Ecke mit 0 beginnend bis 1023 (1919) in der unteren rechten Ecke durchnumeriert sind. Mit dem Befehl 'PRINT @' und einer nachfolgenden Ziffer können Sie die Ausgabe in einer beliebigen Bildschirmposition beginnen lassen.

Als Besonderheit kennt Ihr GENIE dann noch den 'PRINTUSING'-Befehl, mit dem die Dezimalstellen immer mit dem Punkt für die Dezimaltrennung schön untereinander ausgegeben werden können. Für die Benutzung dieses Befehls ist es zunächst notwendig, das Format der Ausgabe als Zeichenkette zu definieren. Für jede anzuzeigende oder zu druckende Stelle setzt man man ein Nummerzeichen '#' und fügt den Dezimalpunkt dort ein, wo er hingehört. Haben die auszugebenden Zahlen mehr Nachkommastellen als gewünscht, erfolgt eine korrekte Rundung der letzten angezeigten Zahl.

Es gibt noch eine Menge weiterer Möglichkeiten, diesen Befehl für das Schreiben von Dollarbeträgen zu verwenden, die aber für deutsche Verhältnisse nicht wichtig sind.

Beachten müssen Sie aber folgendes:

- Der Befehl muß genauso geschrieben werden wie in dem Beispiel gezeigt; mit einem Semikolon vor der Zahlen-Variablen.
- Hat die Zahl mehr Vorkommastellen als formatiert, wird vor die Zahl ein Prozentzeichen '%' gesetzt. Die Rundung kann dann falsch sein.
- Mehrere formatierte Ausdrücke in einer Zeile sind nach den im Beispiel gezeigten Methoden möglich.

Tabellierung mit 'TAB'-Funktion

```
>10 A=99
>20 PRINT TAB(5)A; TAB(15)A
>RUN
       99        99
READY
>
```

Formulierung mit PRINT USING

```
>10 A=123.456
>20 B=5432.1234
>30 U$="###.##"
>40 PRINT USING U$ ; A
>50 PRINT USING U$ ; B
RUN
 123.46
 %5432.12
READY
>
```

Mehrere Formatierungen in einer Zeile

```
>10 A=123.456
>20 B=432.1267
>30 U$="###.##"
>40 PRINT USING U$ ; A ;
>50 PRINT TAB(10) USING U$ ; B
>60 PRINT USING U$ ; A ; B
>RUN
 123.46      432.13
 123.46      432.13
READY
>
```

3.7 Der 'INPUT'-Befehl

Im Gegensatz zu vielen anderen höheren Programmiersprachen gibt es im BASIC auch einen Befehl, mit dem während des Programmablaufs Daten durch den Benutzer eingegeben werden können. Verwendet wird hierfür der 'INPUT'-Befehl. Jesdesmal, wenn der 'INPUT'-Befehl im Programm erscheint, unterbricht der Computer den Programmablauf und wartet, bis über die Tastatur eine entsprechende Eingabe erfolgt ist. Zum Zeichen für den Bedienenden, daß der Computer sich in Wartestellung befindet, erscheint ein Fragezeichen '?' auf dem Bildschirm.

Mit dem 'INPUT'-Befehl wird immer eine Eingabe angefordert, die einer Variablen zugewiesen wird. Im Programm muß darum auch nach dem 'INPUT'-Befehl immer der Name der betreffenden Variable stehen. Ist dies eine Zahlenvariable, so nimmt der Computer auch nur Ziffern als Eingabe an und fordert Sie mit dem Wort REDO auf, die Eingabe zu wiederholen, wenn Sie versucht haben, einen Buchstaben einzugeben. Bei einer Zeichenkettenvariablen im 'INPUT'-Befehl können Sie alle Zeichen der Tastatur eingeben.

In einem 'INPUT'-Befehl können auch mehrere Eingaben angefordert werden. Die Variablen, denen die Eingaben zugewiesen werden sollen, müssen dann durch Kommas getrennt werden.

Wird z.B. in einer Zahlenfolge ein Komma oder Doppelpunkt statt des Dezimalpunktes verwendet, übernimmt der Computer die Ziffer nach dem falschen Zeichen nicht. Folgt im 'INPUT'-Befehl eine weitere Variable, wird der Wert nach dem Komma dieser zugewiesen.

Ein nach einer 'INPUT'-Aufforderung eigegebener Doppelpunkt bewirkt in jedem Fall, daß alles, was danach kommt, nicht übernommen wird.

```
>10 INPUT A , B$
>20 PRINT A ; B$
>RUN
? 10.65
?? DM
 10.65 DM
READY
>
```

Korrekte Eingabe

```
>RUN
? 10,65
  10 65
READY
>
```

Zahl nach dem Komma wurde der 2. Eingabe zugeordnet.

```
>RUN
? DM
REDO
? 123.50
?? DM
123.5 DM
READY
>
```

Fehler bei der Eingabe. (Es wurde versucht, als erstes einen String einzugeben.)

```
>RUN
? 3 : 7
?? TORE
3 TORE
READY
>
```

Zahl nach dem Doppelpunkt wurde nicht übernommen

Anführungsstriche können bei Zeichenketten vorne und hinten eingegeben werden, sie werden dann aber nicht mit gespeichert. Stehen Anführungszeichen inmitten einer Kette, werden sie wie andere Zeichen behandelt.

Wenn beim Lauf eines längeren Programms auf dem Bildschirm ein Fragezeichen erscheint, weiß der Bediener möglicherweise gar nicht, was er eingeben soll, und wenn etwas Falsches eingegeben wird, so kann das ganze Programm durcheinander geraten.

Bei älteren BASIC-Versionen mußte man darum vor jedem 'INPUT'Befehl noch einen 'PRINT'-Befehl zur Erläuterung der angeforderten Eingabe schreiben. Ihr GENIE erlaubt die Zusammenfassung einer Ausgabe auf dem Bildschirm mit einer Eingabe.

Hierzu ist es lediglich notwendig, die Worte, die zur Erläuterung der Eingabe auf den Bildschirm geschrieben werden sollen, in Anführungsstrichen sofort hinter das Befehlswort 'INPUT' zu setzen. Danach muß vor der ersten Variable ein Semikolon stehen. Ein anderes Satzzeichen ist nicht erlaubt.

Auf dem Bildschirm erscheint nach den erläuternden Worten zunächst ein Fragezeichen und dann die eingegebenen Zeichen.

Der Computer setzt in jedem Fall, auch wenn keine Eingabe erfolgt ist, den Programmablauf fort, wenn die <RETURN>-Taste gedrückt wird. Mit Hilfe einer "leeren Eingabe", bei der eine beliebige String-Variable, die sonst nicht gebraucht wird, verwendet werden kann, ist es möglich, das Programm in Wartestellung zu bringen.

```
>10 INPUT A , B$
>20 PRINT A ; B$
>RUN
? 333
?? "Bei Issus Keilerei"
 333 Bei Issus Keilerei
READY
>RUN
? 333
?? Das ist eine "Schnapszahl"
 333 Das ist eine "Schnapszahl"
READY
>
```

Anführungsstriche werden nur dann in die Zeichenkette übernommen, wenn sie im Inneren stehen.

```
>10 INPUT "BETRAG EINGEBEN" ; A
>20 INPUT "2.BETRAG EINGEBEN" ; B
>30 S = A + B
>40 PRINT "SUMME =" ; S
>RUN
BETRAG EINGEBEN ? 342.67
2.BETRAG EINGEBEN ? 186.45
SUMME = 529.12
READY
>
```

Zwischen "Erläuterung" und Variable muß ein Semikolon stehen.

3.8 Der 'GOTO'-Befehl

Bei normalen Programmablauf wird ein Befehl nach dem anderen vom Computer abgearbeitet, bis die höchste im Programm stehende Zeilennummer erreicht ist. Von dieser rein sequentiellen Befehlsverarbeitung kann man mit dem Sprung-Befehl 'GOTO' abweichen. Dieser Befehl ist sehr einfach aufgebaut, denn hinter das Befehlswort ist nur die Zeilennummer zu setzten, in der das Programm fortgesetzt werden soll.

Mit Hilfe des 'GOTO'-Befehls kann man so zum Beispiel dafür sorgen, daß ein Programmteil mehrfach durchlaufen wird. Das Programm läuft dann in der Schleife solange, bis es durch die Betätigung der <BREAK>-Taste abgebrochen wird.

Bei einer Schleife wird ein Rücksprung programmiert. Man kann mit dem 'GOTO'-Befehl aber natürlich auch vorwärts, also in Richtung höherer Zeilennummern springen. Voraussetzung für die Anwendung dieses Befehls ist aber, daß die nach dem 'GOTO' folgende Zeilennummer auch vorhanden ist, sonst meldet der Computer einen Fehler.

Es sei bereits hier darauf hingewiesen, daß man 'GOTO'-Befehle möglichst sparsam im Programm verwenden sollte, da besonders längere Programme sonst recht unübersichtlich werden.

```
>10 INPUT A
>20 E = E + A
>30 PRINT , E
>40 GOTO 10
>RUN
?43
          43
?84
          127
?57
          184
?<RETURN>
          241
usw.
```

Diese Programm addiert unendlich die eingegebenen Zahlen.

Achtung! Wenn Sie die <NEW LINE>-Taste betätigen wird die vorangegangene Zahl wiederholt addiert.

3.9 Der Entscheidungsbefehl

Die wohl wichtigsten Befehle in der Computertechnik, sind die Befehle, durch die bedingte Sprünge veranlaßt werden. Das bedeutet, daß die Sprünge im Programm nur dann ausgeführt werden, wenn die vom Programmierer vorher definierte Bedingung vom Computer geprüft und als erfüllt anerkannt wird.

Für Entscheidungen kennt Ihr GENIE den vielseitigen Befehl:

IF...THEN...ELSE...

dessen englischen Worte sich leicht ins Deutsche übersetzten lassen. Sie beschreiben gut die Wirkung dieses Befehls

WENN	die nachfolgende Bedingung erfüllt ist,
DANN	führe die nachfolgende Anweisung aus.
SONST	arbeite mit der folgenden Anweisung weiter.

Hinter dem Befehl 'IF' muß also immer etwas stehen, was der Computer prüfen kann. Es dürfen aber nur Bedingungen formuliert werden, auf die eindeutig mit JA oder NEIN geantwortet werden kann. Dies ist der Fall, wenn zwei Ausdrücke miteinander verglichen werden und hierzu dienen die Vergleichsoperatoren.

Grundsätzlich kann hinter dem Vergleichsoperator entweder eine Variable, eine Konstante oder auch ein Rechenausdruck stehen. Es hat natürlich wenig Sinn vor und hinter dem Operator jeweils eine Konstante zu setzen, denn das Ergebnis eines derartigen Vergleiches wäre von vorneherein bekannt. An einer Seite sollte also immer eine Variable, deren Wert noch nicht bekannt ist, stehen, die dann vom Computer überprüft werden soll.

Nach dem Wort 'THEN' können ein oder auch mehrere Befehle stehen, die der Computer ausführt, wenn der Vergleich als richtig oder, wie man auch sagt, "wahr" erkannt wurde. Sehr oft ist es üblich, an dieser Stelle einen unbedingten Sprungbefehl einzusetzen, weil man dann von der Zeilennummer an, in die gesprungen wurde, beliebig viele Befehle im Programm anordnen kann.

Die BASIC-Vergleichsoperatoren:

```
IF A (wenn A)  =  B  (gleich B)
               <  B  (kleiner B)
               <> B  (ungleich B)
               <= B  (kleiner oder gleich B)
               >  B  (größer B)
               >= B  (größer oder gleich B)
```

Ergebnisse von Vergleichen (,wenn A=5 und B=6 ist):

```
A  =  B unwahr        A  <  B wahr
A  <> B wahr          A  >= B unwahr
A  >  B unwahr        A  <= B wahr
```

Programm zum Vergleich zweier Zahlen:

```
>10 INPUT "2 ZAHLEN" ; A , B
>20 IF A = B THEN PRINT "BEIDE GLEICH" : END
>30 IF A < B THEN 50
>40 PRINT "1. ZAHL GRÖSSER" : GOTO 10
>50 PRINT "2. ZAHL GRÖSSER" : GOTO 10
>RUN
2 ZAHLEN? 6,2
1. ZAHL GRÖSSER
2 ZAHLEN? 3,8
2. ZAHL GRÖSSER
2 ZAHLEN? 0.001,-10
1. ZAHL GRÖSSER
2 ZAHLEN? 33,33
BEIDE GLEICH
READY
>
```

Die Verwendung des Befehlswortes 'ELSE' ist nicht zwingend vorgeschrieben. Sie können Entscheidungsbefehle auf Ihrem Rechner also auf zwei Arten programmieren.

- Wenn das Wort 'ELSE' im Befehl steht, wird bei Nichterfüllung der Vergleichsbedingungen das Programm mit dem oder den diesem Wort folgenden Befehl fortgesetzt,
- wird das Wort 'ELSE' nicht verwendet, führt der Computer bei als unwahr erkannter Bedingung die im Programm nächstfolgende Zeile aus.

Bei der Verwendung des 'IF...THEN'-Befehls müssen die beiden Zweige des Programms immer wieder zusammenlaufen.

Die Beispiele zeigen die gleiche Aufgabe auf dreierlei Weise programmiert. Eine eingegebene Zahl wird dahingehen überprüft, ob sie positv oder negativ ist. Im letzteren Fall wird sie mit -1 multipliziert, damit immer mit einem positiven Wert weitergearbeitet werden kann.

Die Erfüllung dieser Forderung ist einfach, wenn in den beiden Zweigen, die je nach der Entscheidung zu durchlaufen sind, nur wenige Befehle nötig sind. Sie können diese dann in den Entscheidungsbefehl mit einbauen und das Programm läuft auf jeden Fall mit der nachfolgenden Zahl weiter. (Beispiel 1)

Bei allen Entscheidungsbefehlen bietet Ihr GENIE noch die Erleichterung, daß das Wort 'GOTO' unmittelbar nach 'THEN' oder 'ELSE' auch weggelassen werden kann. Es genügt die Angabe der Zeilennummer, zu der gesprungen werden soll.

Wenn der Entscheidungsbefehl ohne 'ELSE' verwendet wird, muß als letzter Befehl hinter 'THEN' ein Sprung-Befehl stehen, mit dem die Befehle der NEIN-Entscheidung übersprungen werden. (Beispiel 3)

Beispiel 1:

```
>10 INPUT "BELIEBIGE ZAHL" ; A
>20 IF A > 0 THEN PRINT "+" A ELSE PRINT A :
A = A * -1
>30 PRINT "ZAHL IST JETZT IMMER POSITIV" ; A
>RUN
 BELIEBIGE ZAHL? 5
+5
ZAHL IST JETZT IMMER POSITIV 5
READY
>RUN
 BELIEBIGE ZAHL? -7
-7
ZAHL IST JETZT IMMER POSITIV 7
READY
>
```

Beispiel 2:

```
>10 INPUT "BELIEBIGE ZAHL" ; A
>20 IF A > 0 THEN PRINT "+" ; A : GOTO 40
>30 PRINT A : A = A * -1
>40 PRINT "ZAHL IST JETZT IMMER POSITIV" ; A
>RUN
BELIEBIGE ZAHL? -7
-7
ZAHL IST JETZT IMMER POSITIV 7
READY
>
```

Mit Hilfe der Vergleichsoperatoren "größer" oder "kleiner" lassen sich Zeichenketten sogar nach dem Alphabet sortieren. Man benutzt hier die Tatsache, daß auch die Buchstaben intern im Computer als Zahlen abgespeichert werden. Für die Speicherung wird wird der ASCII-Code verwendet, den sie in der GENIE-Befehlstabelle finden. Da in diesem Code dem Buchstaben A eine kleinere Zahl zugeordnet ist, als dem im Alphabet nachfolgenden Buchstaben, bedeutet der Operator "kleiner" in diesem Fall "im Alphabet vorher stehend".

```
>10 INPUT "1. WORT" ; X$
>20 INPUT "2. WORT" ; Y$
>30 IF X$ < Y$ THEN A$ = X$ :B$ = Y$ :GOTO50
>40 A$ = Y$ : B$ = X$
>50 PRINT A$ ; " IST IM ALPHABET VOR " ; B$
>RUN
1. WORT? HAUS
2. WORT? HANS
HANS IST IM ALPHABET VOR HAUS
READY
>RUN
1. WORT? TCS
2. WORT? TROMMESCHLÄGER
TCS IST IM ALPHABET VOR TROMMESCHLÄGER
READY
>
```

Im Beispiel werden die beiden String-Variablen X$ und Y$ verglichen. Die Variable mit dem kleineren Wert wird A$ und diejenige mit dem größeren Wert B$ zugewiesen. Beides wird dann mit Erläuterung geschrieben.

3.10 Aufbau von Schleifen

Schleifen, als gewollte Wiederholungen einzelner Programmteile, sind ein besonders wichtiges Hilfsmittel für eine effektive Programmierung. Wenn Sie BASIC lernen wollen, ist es darum sehr wichtig, daß Sie das Prinzip beim Aufbau von Schleifen richtig verstehen.

Es wurde schon von der ewigen Schleife, die nur durch die Betätigung der <BREAK>-Taste abgebrochen werden kann, gesprochen, aber diese wird man im Programm wohl nur selten verwenden. Im allgemeinen muß ein gezielter Abbruch der Schleife programmiert werden.

Eine Schleife muß immer die folgenden Befehle enthalten:

- Einen Befehl zum Rücksprung an den Anfang der Schleife,
- einen Befehl, der entscheidet, ob die Bedingung für den Abbruch erfüllt ist,
- einen Befehl, der den Wert für die Abbruchbedingung bei jedem Schleifendurchgang ändert
- und natürlich die Befehle, die in der Schleife verarbeitet werden sollen.

Im Grundsatz kann man Schleifen auf zwei verschiedene Weisen aufbauen:

- Man kann den Veränderungsbefehl an den Anfang setzten, danach kommen die eigentlichen Schleifenbefehle und am Schluß wird geprüft, ob die Abbruchsbedingung erfüllt ist und an den Schleifenanfang zurückgesprungen, wenn dies nicht zutrifft.
- Man kann aber auch die Schleifenbefehle an den Anfang schreiben, dann den Entscheidungsbefehl setzen und danach erst den Veränderungsbefehl und den Sprungbefehl setzen. Meist braucht man noch einen Befehl vor der Schleife zur Festlegung der Anfangsbedingungen.

In den Beispielen sind sind Schleifen programmiert, in denen die Zahl 7 mit den Zahlen 1 bis 7 multipliziert wird.

Sie erkennen, daß in den Beispielen die Variable A bei jedem Schleifendurchlauf verändert wird und daß die Entscheidung, ob die Schleife abgebrochen wird, von dem Wert der Variablen A abhängt. Man bezeichnet A darum auch als Lauf- oder Zählvariable.

Beim Aufbau von Schleifen müssen Sie den Wert der Laufvariable immer im Auge behalten. Wenn die Schleifen nicht genauso wie in den Beispielen aufgebaut sind, können zu viel oder zu wenig Schleifendurchgänge stattfinden.

Beispiel 1

```
>10 A = A + 1
>20 B = A * 7
>30 PRINT B;
>40 IF A = 7 THEN END
>50 GOTO 10
>RUN
 7 14 21 28 35 42 49
READY
>
```

Beispiel 2

```
>10 A = 1
>20 B = A * 7
>30 PRINT B;
>40 IF A = 7 THEN END
>50 A = A + 1
>60 GOTO 20
>RUN
 7 14 21 28 35 42 49
READY
>
```

Es muß auch noch darauf hingewiesen werden, daß es manchmal zweckmäßig ist, im Entscheidungsbefehl eine Ungleichbedingung (<>) zu formulieren und dann solange zum Schleifenanfang zu springen, wie diese Bedingung noch erfüllt ist.

Bei Ihrem GENIE gibt es aber noch eine besondere Kombination von Befehlsworten, mit denen Schleifen programmiert werden können. Es ist die 'FOR...NEXT'-Anweisungen, die vollständig lautet:

Am Beginn der Schleifen:

FOR V=X TO Y STEP Z

Am Ende der Schleife:

NEXT V

Die Anwendung dieser Befehle ist ganz einfach. Zu Beginn der Schleife muß bestimmt werden, welche Variable als Laufvariable dienen soll. Diese wird bei jedem Schleifendurchgang verändert. Die Laufvariable steht hinter dem ersten Befehlswort 'FOR' vor dem Gleichheitszeichen. Dann kommt der Anfangswert der Laufvariablen, das heißt also der Wert, mit dem die Schleife beginnen soll, und nach dem Befehlswort 'TO' der Endwert, also der Wert nach dessen Erreichen die Schleife abzubrechen ist. Soll der Schritt um den die Laufvariable bei jedem Schleifendurchgang zu verändern ist, nicht +1 betragen, so kann die gewünschte Schrittweite nach dem Befehl 'STEP' angegeben werden.

```
>10 B=6
>20 FOR A = B TO 2*B+B/2-1
>30 PRINT A; : NEXT
>RUN
 6  7  8  9  10  11  12  13  14
READY
>
```

Im folgenden Beispiel wird die Schleife nur 6 mal durchlaufen, da der Abbruchbefehl vor den Schleifenbefehlen liegt.

```
>10 A = 1
>20 IF A = 7 THEN END
>30 B = A * 7
>40 PRINT B;
>50 A = A + 1
>60 GOTO 20
>RUN
 7  14  21  28  35  42
READY
>
```

Einfache 'FOR...NEXT'Schleife mit Ausdruck der Laufvariablen:

```
>10 FOR A = 1 TO 10
>20 PRINT A;
>30 NEXT A
>40 END
>RUN
 1  2  3  4  5  6  7  8  9  10
READY
>
```

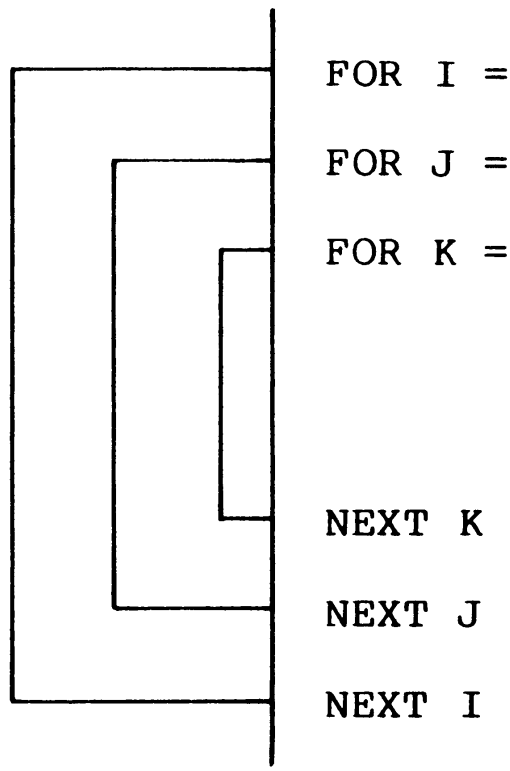
Schleife mit Ungleich-Abfrage:

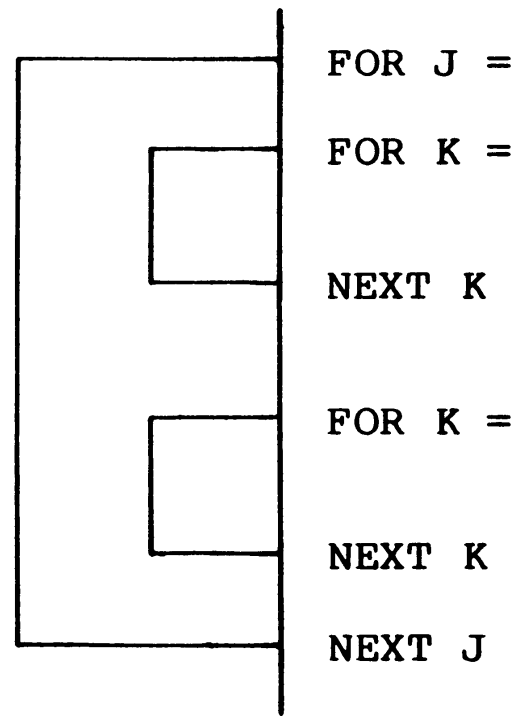
```
>10 A = A + 1
>20 B = A * 7
>30 PRINT B;
>40 IF A <> 7 THEN 10
>50 END
>RUN
 7 14 21 28 35 42 49
READY
>
```

Als Anfangs- und Endwerte für die Laufvariable und ebenso auch für den Schritt können beliebige positive und negative Zahlen aber auch Rechenausdrücke eingesetzt werden. Das obenstehende Beispiel zeigt einige hintereinanderstehende 'FOR...NEXT'Schleifen, bei denen immer nur die Laufvariable ausgegeben wird. Beachten Sie das Semikolon nach dem 'PRINT'-Befehl, das dafür sorgt, daß nach kein Zeilenvorschub erfolgt. Damit aber vor dem Ausdruck der nächsten Schleife ein Zeilenvorschub stattfindet, muß jedem 'NEXT'- ein leerer 'PRINT'-Befehl folgen.

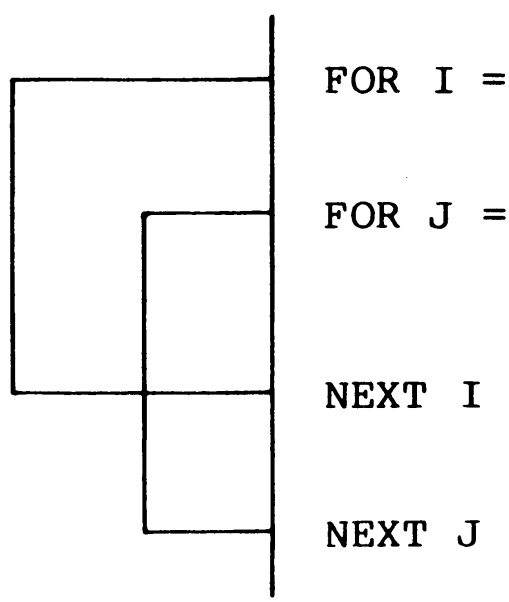
Beim GENIE können Sie die Laufvariable nach 'NEXT' weglassen, wenn nur eine 'FOR...NEXT'-Schleife geöffnet ist. Dies bedeutet, daß Sie auch mit ineinander verschachtelten Schleifen arbeiten können, dann aber die gültige Laufvariable hinter 'NEXT' einsetzen müssen. Dabei darf die Verschachtelung nur so aufgebaut werden, wie es das Bild zeigt.

Zulässig verschachtelte Schleifen:





Unzulässig verschachtelte Schleife:



Die Abfrage, ob die Schleifenbedingung erfüllt ist, erfolgt erst am Schluß. Jede Schleife wird also mindestens einmal durchlaufen.

Einsprünge in 'FOR...NEXT'-Schleifen sind grundsätzlich nicht erlaubt. Wenn man dies versucht, meldet der Computer, wenn er auf ein 'NEXT' ohne vorausgegangenes 'FOR' trifft, 'NF Error' (NEXT WITHOUT FOR Error) und bricht das Programm ab.

Bedingte Aussprünge aus Schleifen sind zulässig, können aber leicht zu Fehlern führen, wenn man den Wert der Laufvariablen nicht immer im Auge behält.

Verschiedene 'FOR...NEXT'-Schleifen:

```
>10 B=6
>20 FOR A = 20 TO 8 STEP -2
>30 PRINT A; : NEXT : PRINT
>40 FOR A = .1 TO 1.4 STEP .2
>50 PRINT A; : NEXT : PRINT
>60 FOR A = B TO 2*B+B/2-1
>70 PRINT A; : NEXT : PRINT
>RUN
 20  18  16  14  12  10  8

 .1  .3  .5  .7  .9  1.1  1.3

 6  7  8  9  10  11  12  13  14

 READY
 >
```

```
>10 FOR A = 0 TO 0
>20 PRINT "SCHLEIFE DURCHLAUFEN"
>30 NEXT
>RUN
SCHLEIFE DURCHLAUFEN
READY
>
```

Jede 'FOR...NEXT'-Schleife wird mindestens einmal durchlaufen.

3.11 Ein Programm entsteht

Nun haben Sie die fünf wichtigsten BASIC-Befehle kennengelernt und sich mit Hilfe der kleinen Übungsprogramme mit Ihrem GENIE vertraut gemacht. Jetzt sollen Sie lernen, wie man bei der Entwicklung eines für die Praxis nützlichen Programms vorgehen kann, denn diese 5 Befehle genügen wirklich, um viele Aufgaben zu lösen.

Nehmen wir an, Sie müssen das Gewicht verschiedener Eisenrohre berechnen. Das geht zwar auch mit Hilfe eines Taschenrechners, aber Ihr Computer macht das viel komfortabler.

Nun kommt noch der 'PRINT'-Befehl für den Ausdruck des Gewichts hinzu und dann können Sie mit diesem Programm schon beliebige Rohrgewichte berechnen.

Damit ist das Programm aber noch nicht fertig. Wir wollen auch schon bei dieser kleinen Aufgabe alle Vorteile, die ein Computer gegenüber einem Taschenrechner bietet, ausnutzen.

In erster Linie muß sich ein Programm selbst erklären, denn sonst weiß man bald nicht mehr, was beim Erscheinen eines Fragezeichens einzugeben ist. Die 'INPUT'-Befehle erhalten deshalb eine Erläuterung. Dann muß klar gezeigt werden, was der Computer ausgerechnet hat. Hierfür wird der 'PRINT'-Befehl entsprechend ergänzt. Da die Zahl Pi mehrfach benutzt wird, ist es ratsam, den Zahlenwert von Pi einer Variablen genannt PI zuzuweisen und einen entsprechenden Befehl dem bisherigen Programm voranzustellen.

```
>90 PI = 3.14159
>100 INPUT"ROHRLAENGE IN CM";LG
>110 INPUT"AUSSEN- UND INNENDURCHMESSER";
DA,DI
>120 FL = DA * DA * PI / 4 - DI * DI * PI/4
>130 GE = FL * LG * 7.8
>140 PRINT " DAS ROHR WIEGT";GE;"GRAMM"
```

Diese Änderungen bewirken folgenden Programmablauf:

```
>RUN
 ROHRLAENGE IN CM ?  100
 AUSSEN- UND INNENDURCHMESSER ? 2.54,1.97
 DAS ROHR WIEGT 1574.84 GRAMM
READY
>
```

Einer der Vorteile von BASIC liegt darin, daß es verhältnismaßig einfach ist, ein Programm zu ändern und zu ergänzen. So soll auch hier gezeigt werden, wie dieses Programm noch weiter ausgebaut werden kann. Es soll so weit erweitert werden, daß auf Wunsch auch eine ganze Tabelle von Rohrgewichten erstellt werden kann.

Hierfür ist zunächst notwendig, dem Benutzer eine Auswahlmöglichkeit für die Einzelberechnung oder die Tabelle zu geben. Zu diesem Zweck werden die Befehle in den Zeilen 20 - 50 hinzugefügt. Dabei sorgt der Entscheidungsbefehl in Zeile 50 für eine erneute Abfrage, wenn weder E noch T eingegeben wurden.

Entsprechend dem Befehl in Zeile 40 beginnt das Tabellenprogramm in Zeile 200 mit 'INPUT'-Befehlen. Für die Berechnung und die Ausgabe der Tabellenwerte wird eine 'FOR...NEXT'-Schleife verwendet, die die eingegebenen Variablen verarbeitet.

Wenn man etwas berechnen will, muß man die dafür anzuwendende Formel kennen. Das Gewicht eines Körpers ist immer:

Gewicht = Grundfl. mal Länge mal spez. Gew.

Beim Rohr entspricht die Grundfläche der Fläche eines Kreisringes, für die folgende Formel gilt:

$$\text{Kreisringfläche} = \frac{D^2 \cdot \pi}{4} - \frac{d^2 \cdot \pi}{4}$$

Nun können Sie mit der Programmierung beginnen. Es ist jedoch zweckmäßig, sich vorher Gedanken über die zu benutzenden Variablennamen zu machen und diese aufzuschreiben.

Hier werden zunächst gebraucht:

GE für das auszurechnenden Gewicht
FL für die Kreisringfläche
DA für den Aussendurchmesser des Rohrs
DI für den Innendurchmesser des Rohrs
LG für die Rohrlänge

Wir beginnen mit den ersten Befehlen bei einer etwas größeren Zeilennummer, damit später noch Befehle davor gesetzt werden können. Für diese Aufgabe werden 3 Eingaben gefordert, für die 'INPUT'-Befehle in den Zeilen 100 und 110 geschrieben werden.

```
>100 INPUT LG
>110 INPUT DA,DI
>120 FL = DA * DA * 3.1415 / 4 - DI * DI *
3.1415 / 4
>130 GE = FL * LG * 7.8
>140 PRINT GE
>RUN
? 100
? 2.54
?? 1.97
 1574.79
READY
>
```

Die beiden Teile der Rechenformel können zwar in einer Programmzeile untergebracht werden, wenn man die Formeln zusammenfaßt. Einfacher und besser verständlich wird das Programm aber, wenn zuerst die Fläche und dann das Gewicht berechnet wird.

In Zeile 120 wird der Variablen FL das Ergebnis des rechts neben dem Gleichheitszeichen stehenden Rechenausdrucks zugewiesen. Der Ausdruck entspricht der Formel, die nur nach den BASIC-Regeln geschrieben wurde. Die Zahl Pi ist als Konstante eingesetzt. Das gleiche gilt für die Zeile 130, wo der Rechenausdruck für das Gewicht mit der Konstanten 7.8 für das spezifische Gewicht steht.

Hierbei ist darauf zu achten, daß der kleinste Durchmesser vor dem TO steht, da mit positivem Schritt gearbeitet wird. Zeile 260 muß etwas geändert werden, da als Länge konstant 100 cm angenommen werden und das Gewicht in kg ausgegeben werden soll. Durch den 'PRINT'-Befehl werden die Durchmesser und das Gewicht in zwei Spalten gedruckt.

Das Programm erhält nun noch eine Überschrift. Die Wertzuweisung an Pi, die für beide Programmteile gilt, muß an den Anfang gesetzt werden, und in Zeile 150 muß ein 'END'-Befehl kommen.

Jetzt kann das Programm ausprobiert werden, und dabei entdeckt man noch zwei kleine Fehler. Es fehlt eine Tabellenüberschrift, die vor die Schleife in Zeile 225 programmiert wird. Außerdem kann es vorkommen, daß der letzte Schleifendurchgang wegfällt, weil der Computer GD beim Rechnen abgerundet hat. Dies wird durch hinzufügen einer Addition von 0.0001 zu GD in Zeile 230 ausgeglichen (Dies hängt damit zusammen, daß der Computer intern binär rechnet. Nicht jeder Dezimalbruch hat ein genaues binäres Äquivalent, ebensowenig wie z.B. 0.333333 genau 1/3 ist.

Das fertige Programm sieht dann wie folgt aus:

```
5 CLS
10 PRINT"ROHRGEWICHT BERECHNEN"
15 PI = 3.14159
20 PRINT"EINZELROHR (E) ODER TABELLE (T)"
30 INPUT"KENNBUCHSTABE";K$
40 IFK$="T"THEN 200
50 IFK$<>"E"THEN 20
100 INPUT"ROHRLAENGE IN CM";LG
110 INPUT"AUSSEN- UND INNENDURCHMESSER";
DA,DI
120 FL = (DA * DA - DI * DI) * PI / 4
130 GE = FL * LG * 7.8
140 PRINT"DAS ROHR WIEGT";GE;"GRAMM"
150 END
200 INPUT"WANDSTÄRKE";WA
210 INPUT"GROESSTER UND KLEINSTER
AUSSENDURCHMESSER";GD,KD
220 INPUT"STUFUNG";ST
225 PRINT"DI/DA (CM)","GEWICHT / METER (KG)"
230 FOR DA = KD TO GD STEP ST
240 FI = DA - WA * 2
250 FL = (DA * DA - DI * DI) * PI / 4
260 GE = FL * 100 * 7.8 + .0001
270 PRINT DA ; "/" ; DI , GE
280 NEXT
```

Lauf des Programms:

```
ROHRGEWICHT BERECHNEN
EINZELROHR (E) ODER TABELLE (T)
KENNBUCHSTABE? T
WANDSTÄRKE? .4
GROESSTER UND KLEINSTER
AUSSENDURCHMESSER? 4.2,3.2
STUFUNG? .2


DA/DI (CM)      GEWICHT / METER (KG)
 3.2/2.4        2.74449
 3.4/2.6        2.94053
 3.6/2.8        3.13656
 3.8/3          3.3326
 4  /3.2        3.52863
 4.2/3.4        3.72467
READY
>
```

3.12 Unterprogramme

Auch im BASIC kann man - wie in allen anderen Programmiersprachen - die Unterprogrammtechnik anwenden. Man versteht hierunter eine Programmierung, bei der gewisse, öfter gebrauchte Teile eines Programmes nur einmal als sogenanntes Unterprogramm geschrieben werden, das dann beliebig oft innerhalb des Gesamtprogrammes aufgerufen werden kann.

Unterprogramme werden mit dem Befehlswort 'GOSUB' mit nachfolgender Zeilennummer aufgerufen. Der Computer merkt sich die nach dem 'GOSUB'-Befehl Zeilennummer und setzt dort das Programm fort, wenn er im Unterprogramm den Beendigungsbefehl 'RETURN' findet.

Meist ist es zweckmäßig die Unterprogramme an den Schluß des Programmes zu legen. Man muß dann allerdings dafür sorgen, daß das Hauptprogramm durch einen 'END'-Befehl abgeschlossen wird, weil sonst das erste Unterprogramm nach dem Hauptprogramm anläuft und sich der Computer bei dem 'RETURN'-Befehl mit einem RG Error (RETURN WITHOUT GOSUB) meldet und den Programmablauf abbricht.

Unterprogramme können viel dazu beitragen, daß das Programm übersichtlich bleibt. Wenn mehrere Blöcke in einem übergeordneten Hauptprogramm zusammengefaßt werden sollen, so ist hierfür der 'GOSUB'-Befehl besser als der 'GOTO'-Befehl.

Im BASIC werden in den Unterprogrammen die gleichen Variablen verwendet, wie im Hauptprogramm, so daß die Übernahme der Daten aus Unterprogrammen in das Hauptprogramm problemlos ist.

```
>10 PRINT"HAUPTPROGRAMM"
>20 GOSUB 50
>30 PRINT"ENDE DES PROGRAMMS"
>40 END
>50 PRINT"UNTERPROGRAMM"
>60 RETURN
>RUN
HAUPTPROGRAMM
UNTERPROGRAMM
ENDE DES PROGRAMMS
READY
>
```

3.13 Logische Verknüpfungen

In der Schaltungstechnik spielen logische Verknüpfungen eine große Rolle, und auch im BASIC gibt es drei Befehle durch die jeweils zwei oder auch mehrere Bedingungen miteinander verknüpft werden können. Es gelten hierfür die Befehlsworte:

AND Bedingung erfüllt, wenn alle Teile "wahr" sind.

OR Bedingung erfüllt, wenn eins der Teile "wahr" ist.

NOT Komplementierung des folgenden Ausdrucks auf binärer Ebene.

Mit Hilfe dieser drei Befehlsworte können in einem 'IF...THEN'Befehl mehrere Bedingungen formuliert werden, zwischen denen dann eins der Befehlsworte stehen muß. Im Programm werden dann alle Bedingungen geprüft. Ist die Verknüpfung mit 'AND' erfolgt, wird der Befehl nach 'THEN' nur ausgeführt, wenn die vor und hinter dem 'AND' stehende Bedingung erfüllt ist. Bei 'OR' genügt es, wenn eine der Bedingungen erfüllt ist.

Bei der Anwendung dieser Befehlsworte muß man beachten, daß vor und hinter ihnen jeweils ein vollständiger Vergleich stehen muß. Das folgende Statement ist nicht zulässig:

```
IF A > 10 AND < 100 THEN 250      (falsch)
```

Dafür muß geschrieben werden:

```
IF A > 10 AND A < 100 THEN 250    (richtig)
```

```
>10 INPUT A,B
>20 PRINT "MIT";A;"UND";B;
>30 IFA>B AND 2*A>B THEN 70
>40 IFA>B OR 2*A>B THEN 80
>50 PRINT "IST KEINE BEDINGUNG ERFÜLLT"
>60 END
>70 PRINT"SIND BEIDE BEDINGUNGEN ERFÜLLT":
GOTO90
>80 PRINT "IST EINE BEDINGUNG ERFÜLLT"
>90 GOTO 10
```

Der Programmablauf zeigt

```
>RUN
?10,12
MIT 10 UND 12 IST EINE BEDINGUNG ERFÜLLT
?10,4
MIT 10 UND 4 SIND BEIDE BEDINGUNGEN ERFÜLLT
?10,30
MIT 10 UND 30 IST KEINE BEDINGUNG ERFÜLLT
READY
>
```

Für gewisse Programmieraufgaben ist es noch wichtig, zu wissen, daß Ihr GENIE intern eine erfüllte Bedingung mit -1 und eine nicht erfüllte mit 0 registriert.
Man kann also statt

```
IF A = B AND C = D AND E = F THEN ...
```

auch schreiben

```
IF (A = B) + (C = D) + (E = F) = -3 THEN ...
```

Wenn nur zwei Bedingungen erfüllt sein brauchen, wird statt -3 einfach -2 eingesetzt.

```
>10 INPUT A,B,C
>20 IF(A = 3)+(B = 4)+(C = 5)=-3 THEN 60
>30 IF(A = 3)+(B = 4)+(C = 5)=-2 THEN 70
>40 IF(A = 3)+(B = 4)+(C = 5)=-1 THEN 80
>50 PRINT"KEINE BEDINGUNG ERFÜLLT" : END
>60 PRINT"ALLE BEDINGUNGEN ERFÜLLT":PRINT:
GOTO10
>70 PRINT"ZWEI BEDINGUNGEN ERFÜLLT" : PRINT:
GOTO 10
>80 PRINT"EINE BEDINGUNG ERFÜLLT" : PRINT:
GOTO 10
>RUN
? 3,4,5
ALLE BEDINGUNGEN ERFÜLLT

? 3,4,8
ZWEI BEDINGUNGEN ERFÜLLT

? 3,7,8
EINE BEDINGUNG ERFÜLLT

? 6,7,8
KEINE BEDINGUNG ERFÜLLT
READY
>
```

Der logische Opererator 'NOT':

```
>10 INPUT A
>20 IF A=5 THEN PRINT"5 IST GLEICH 5" : END
>20 IF NOT A < 5 THEN PRINT A;
" IST GROESSER ALS 5" : GOTO 10
>30 PRINT"A IST KLEINER ALS 5" : GOTO 10
```

Programmlauf:

```
>RUN
? 7
7 IST GRÖSSER ALS 5
? 3
3 IST KLEINER ALS 5
? 5
5 IST GLEICH 5
READY
>
```

3.14 'ON...GOTO'- oder 'GOSUB'-Befehle

Der 'IF...THEN'-Befehl läßt, wie erklärt wurde, immer nur zwei Möglichgkeiten für die Fortführung des Programms zu. Wenn es gewünscht ist, eine ganze Reihe von Möglichkeiten für die Fortsetzung eines Programmes vorzusehen, können Sie hierfür den 'ON...GOTO'-Befehl verwenden. Die vollständige Form dieses Befehls lautet:

ON Variable GOTO Zeilennummer, Zeilennr, ...

Findet der Computer diesen Befehl, so wandelt er zunächst die Variable nach dem Wort 'ON' in eine Ganzzahl um. Bei Ihrem GENIE kann darum auch hier ein Rechenausdruck stehen. Ist das Ergebnis eine 1, so wird die erste nach dem Wort 'GOTO' stehende Zeilennummer angesprungen, bei 2 erfolgt der Sprung zur zweiten Zeilennummer etc. Man kann also mit einem 'ON...GOTO'-Befehl mehrfache Verzweigungen programmieren.

Statt 'GOTO' kann in den Befehlen auch das Wort 'GOSUB' eingesetzt werden. Es erfolgt dann der Einsprung in das Unterprogramm.

Mit diesem Befehl können Sie leicht ein Menü, wie man ein Verzeichnis für mehrere Programmverzweigungen nennt, abfragen. Nach der Anzeige des Menüs folgt ein 'INPUT'-Befehl. Die Zahl, die eingegeben wurde, wird der Variablen zugewiesen, die nach 'ON' im Verzweigungbefehl steht.

Wenn Sie diesen Befehl anwenden, müssen Sie aber immer dafür sorgen, daß das Hauptprogramm nach Beendigung der Unterprogramme erneut angesprungen wird. Im Beispiel sorgt der Sprung-Befehl in Zeile 140 dafür, daß das Menü neu erzeugt wird.

```
>10 PRINT"                SPIELAUSWAHL"
>20 INPUT"1=MÜHLE 2=DAME 3=AUTORENNEN ";A
>30 ON A GOSUB 50,150,200
>40 PRINT"EIN ANDERES SPIEL ? " : GOTO 10
>50 PRINT"PROGRAMM FÜR MÜHLE" : RETURN
>150 PRINT"PROGRAMM FÜR DAME" : RETURN
>200 PRINT"PROGRAMM FÜR AUTORENNEN":RETURN
>RUN
              SPIELAUSWAHL
1=MÜHLE 2=DAME 3=AUTORENNEN ? 2
PROGRAMM FÜR DAME EIN ANDERES SPIEL ?
              SPIELAUSWAHL
```

3.15 Aktive Befehle

Wenn das System zusammengestellt und der Computer eingeschaltet wird, so befindet sich der Benutzer in der aktiven Befehlsebene. Man erkennt dies daran, daß das Wort 'READY', gefolgt von einem '>'-Zeichen in der nächsten Zeile in der linken, oberen Ecke des Bildschirms erscheint. Man sagt auch der Computer ist "ready".

Nun können Sie einen der folgenden Befehle eingeben:

AUTO	EDIT	(SYSTEM)
CLEAR	LIST	TROFF
CONT	NEW	TRON
DELETE	RUN	LLIST

Wir werden diese Kommandos im einzelnen erläutern. Bitte beachten Sie, daß alle Angaben in Klammern optional sind.

Beispiel: 'AUTO' (Zeilennummer, Inkrement).

Das konkrete Kommando sieht so aus:

AUTO 10,5 oder AUTO oder AUTO 10

Ein Kommando wird mit der <NEW LINE>-Taste an den Computer abgeschickt, der es dann ausgeführt.

AUTO (Zeilennummer, Inkrement)

Dieses Kommando erzeugt automatisch Zeilennummern vor jeder Programmzeile, die Sie eingeben wollen. Die Optionen erlauben Ihnen, die Anfangszeilennummer und die Schrittweite zwischen den Zeilen anzugeben. Wird nun 'AUTO', wie bei allen Kommandos gefolgt von der <NEW LINE>-Taste, eingegeben, so nimmt der Computer als erste Zeilennummer die Zahl 10 und als Schrittweite auch die 10 an. Sie können den Programmbefehl direkt nach der Zeilennummer eingeben. Nach jeder Eingabe einer Programmzeile erhöht der Computer automatisch die Zeilennummer um den im Inkrement angegebenen Betrag.

Das 'AUTO'Kommando bleibt so lange aktiv, bis die <BREAK>-Taste gedrückt wird. Stößt der Computer im 'AUTO'-Modus auf eine schon benutzte Zeile, so erscheint ein ein Stern '*' rechts neben der Zeilennummer. Wollen Sie diese Zeile nicht neu schreiben, so betätigen Sie die 'BREAK'-Taste und sind damit wieder auf der Kommando-Ebene.

Beispiel:

```
READY
>AUTO 1,2
1 PRINT"ZEILE 1"
3 PRINT"ZEILE 3"
5 PRINT"ZEILE 5"
7 PRINT"ZEILE 7"
9 <BREAK>-Taste
READY
>AUTO 2,2
2 PRINT"ZEILE 2"
4 PRINT"ZEILE 4"
6 PRINT"ZEILE 6"
8 <BREAK>-Taste
READY
>AUTO
10 PRINT"ZEILE 10"
20 PRINT"ZEILE 20"
30 PRINT"ZEILE 30"
40 <BREAK>-Taste
READY
>AUTO 1,1
1*
2*
3*
```

CLEAR (Anzahl der Bytes)

Das 'CLEAR'-Kommando löscht eine gewisse Zahl von Bytes und stellt sie zum Abspeichern von Zeichenketten zur Verfügung. Außerdem werden alle numerischen Variablen auf Null gesetzt und leere Zeichenketten in die Zeichenkettenvariablen geschrieben. CLEAR 100 ordnet den Zeichenkettenvariablen 100 Bytes zu. Wenn die Fehlermeldung 'OS' (OUT of STRINGSPACE) auftritt, so kann mit dem 'CLEAR'-Kommando mehr Platz für die Zeichenkettenvariablen freigegeben werden.

CONT

Wird ein Programm durch die <BREAK>-Taste unterbrochen oder hat das Programm einen 'STOP'-Befehl ausgeführt, so kann die Weiterführung des Programms mit 'CONT' an der Stelle, an der es vorher unterbrochen wurde, wieder aufgenommen werden.

DELETE Zeilennummer (- Zeilennummer)

Dieses Kommando löscht die angegebene(n) Zeile(n). Wenn eine der angegebenen Zeilennummern nicht existiert, wird der Befehl nicht ausgeführt und 'FC-ERROR' ausgegeben.

Beispiele:

DELETE 5	Löscht die Zeile 5
DELETE 7-10	Löscht die Zeile 7,10 und alle dazwischen
DELETE .	Löscht die gerade eingegebene (editierte) Zeile
DELETE -12	Löscht alle Zeilen vom Programmanfang bis einschließlich Zeile 12

EDIT Zeilennummer

Dieses Kommando veranlaßt den Computer, von der aktiven Kommandoebene in den Editierungsmodus überzugehen. Die Editierungsebene erlaubt es, Zeilen im Speicher zu verändern, ohne sie neu eintippen zu müssen. Das 'EDIT'-Kommando hat eine Reihe von Unterbefehlen, die auf der Editierungsebene ausgeführt werden können. Dem 'EDIT'-Kommando muß eine gültige Zeilennummer folgen.

Beispiel:

EDIT 20 Schaltet um auf die Editierungsebene, um Zeile 20 zu bearbeiten.

LIST (Zeilennummer - Zeilennummer)

Mit diesem Kommando veranlassen Sie den Computer, eine oder mehrere Programmzeilen auf dem Bildschirm darzustellen. 'LIST' ohne Optionen schreibt das gesamte Programm, das sich momentan im Speicher befindet, heraus. Um den schnellen Durchlauf der Programmzeilen auf dem Schirm anzuhalten, betätigen Sie die <SHIFT>- und @-Taste (Klammeraffe) gleichzeitig. Das Durchlaufen der Zeilen wird durch Drücken einer beliebigen Taste wieder gestartet.

Beispiele:

LIST 3	Zeile 3 wird gelistet
LIST 10 - 20	Zeile 10, 20 und alle dazwischen werden gelistet
LIST - 50	alle Zeilen bis Zeile 50 werden gelistet
LIST 20 -	alle Zeilen ab Zeile 20 werden gelistet
LIST .	die Zeile, die gerade editiert wurde wird gelistet
LIST	alle Zeilen werden gelistet (das ganze Programm)

NEW

Dieses Kommando löscht alle Programmzeilen, setzt alle numerischen Variablen auf Null und schreibt in alle Zeichenkettenvariablen die leere Zeichenkette. Der Speicherplatz für die Zeichenkettenvariablen, der mit 'CLEAR' gesetzt wurde, wird nicht verändert.

RUN (Zeilennummer)

Dieses Kommando startet ein Benutzerprogramm im Speicher. Wird die Zeilennummer nicht angegeben, so wird das Programm in der ersten Zeile gestartet. Wird jedoch die Zeilennummer angegeben, startet das Programm mit dieser Zeile. Die Eingabe einer ungültigen Zeilennummer führt zu einer Fehlermeldung.

Mit jedem 'RUN'-Kommando wird automatisch auch ein 'CLEAR' ausgeführt.

Beispiele:

RUN 50	Startet das Programm in Zeile 50
RUN	Startet das Programm in der ersten Zeile

SYSTEM

Dieses Kommando wird benutzt, wenn Sie ein Programm von Kassette laden wollen. Allgemein bringt dieser Befehl den Computer in den Monitor-Modus. In diesem Modus können Maschienenprogramme von der Kassette geladen und ausgeführt werden. Nach der Eingabe des 'SYSTEM'-Kommandos erscheint ein '*?' auf dem Bildschirm.

Der Computer erwartet dann die Eingabe des Dateinamens der Maschienencodedatei (Objekt code), die er von der Kassette laden soll. Ist das Programm vollständig geladen, so erscheint ein weiteres '*?'. Um das Programm nun zu starten, tippen Sie einen '/', gefolgt von von der Startadresse des Maschinenprogramms (in dezimaler Schreibweise) ein. Wird nur ein '/' eingegeben, so startet das Programm an der Adresse, die in der Datei spezifiziert wurde.

Beim Laden erscheinen - wie bei 'CLOAD' - zwei Sternchen; wird das linke zu einem 'C', liegt ein Ladefehler (CHECKSUM ERROR) vor.

TRON

Dieses Kommando schaltet die TRACE-Funktion ein. Sie erlaubt es, den Ablauf eines Programms zur Fehlersuche zu verfolgen. Sobald der Computer eine neue Programmzeile ausführt, wird ihre Zeilennummer in Klammern ausgegeben.

Beispiel:

```
>10 PRINT"  ** PROGRAMM  **"
>20 A = 1
>30 IF A = 3 THEN 70
>40 PRINT A
>50 A = A + 1
>60 GOTO 30
>70 PRINT" ** ENDE DES PROGRAMMS **"
>80 END
```

Geben Sie ein:

```
>TRON
>RUN
(10)  ** PROGRAMM  **
(20)(30)(40) 1
(50)(60)(30) 2 (70) ** ENDE DES PROGAMMS **
(80)
```

Soll die Programmausführung vor dem Programmende kurz unterbrochen werden, so kann das durch gleichzeitiges Betätigen der <SHIFT>- und @-Taste erreicht werden. Um wieder fortzufahren müssen Sie lediglich irgendeine Taste betätigen. 'TRON' und der dazugehörige 'TROFF'-Befehl können auch innerhalb eines Programms verwendet werden.

Beispiel:

```
 :
 :
 :
90 IF A = B THEN 160
100 TRON
110 A = B + C
120 TROFF
:
:
:
```

Ist bei der Ausführung dieses Programmteils A nicht gleich B, werden Zeile 100 bis 120 ausgeführt. Die TRACE-Funktion wird eingeschaltet und das Durchlaufen des Befehls in Zeile 120 wird angezeigt. Danach wird die TRACE-funktion wieder abgeschaltet. 'TRON' und 'TROFF' können nach der Fehlersuche wieder aus dem Programm entfernt werden.

TROFF

Wie bereits erwähnt schaltet dieses Kommando das 'TRON'-Kommando wieder aus.

LLIST

Dieses Kommando listet ein Programm auf dem Drucker. Dieses Kommando arbeitet genauso wie das 'LIST'-Kommando.

3.16 Fehler und Fehlersuche

Einen nicht unerheblichen Teil der Arbeitszeit beim Programmieren muß man für die Fehlerbeseitigung einkalkulieren. Vorher kommt aber die Fehlersuche, und hierfür bietet Ihr GENIE eine Reihe von Hilfen an.

Eine Fehlermeldung wird entweder als Fehlercode oder als englischer Text bei Diskettenbetrieb auf dem Bildschirm ausgegeben, wenn der Computer bei der Abarbeitung eines Befehls nicht weiterkommt, weil er einen Fehler entdeckt hat. Im wesentlichen geht es hier um zwei Fehlerarten.

Als Syntaxfehler werden diejenigen Fehler bezeichnet, bei denen der Computer ein in BASIC nicht zulassiges Sprachelement feststellt oder bei denen eine Regel dieser Sprache nicht eingehalten wurde.

Findet Ihr GENIE einen Syntaxfehler, bricht er den Programmablauf ab und meldet, in welcher Zeile der Fehler gefunden wurde. Zur Erleichterung der Korrektur wird schon das Editor-Programm aufgerufen, und es erscheint nach der Fehlermeldung die Zeilennummer der fehlerhaften Zeile. Sie können diese nun sofort mit Hilfe der Editor-Befehle berichtigen.

Es gibt dann noch eine ganze Reihe von anderen Fehlern, die der Computer feststellen und melden kann. Diese Fehler können aber nicht durch Korrektur in einer Zeile behoben werden. Zum Teil müssen neue Befehle eingefügt werden oder es ist auf andere Weise in das Programm einzugreifen. Der Computer beendet darum bei diesen Fehlern nur den Programmablauf und meldet nach der Fehlerbeschreibung mit READY, daß er auf direkte Eingaben wartet.

3.17 Beschreibung der Fehlermeldungen

NF NEXT ohne FOR. Ein NEXT ohne entsprechendes FOR wurde benutzt. Dieser Fehler kann auch auftreten, wenn die NEXT-Variable in verschachtelten Schleifen vertauscht wurde.

SN Syntax-Fehler: Ist normalerweise das Ergebnis von falscher Interpunktion, offenen Klammern, ungültigen Zeichen oder falsch geschriebenen Statements.

RG RETURN ohne GOSUB: Ein RETURN-Statement wurde gefunden, ohne daß ein entsprechendes GOSUB existierte.

OD Keine Daten mehr: Einem READ oder INPUT #-Statement stehen nicht mehr genug Daten zur Verfügung. Ein DATA-Statement kann vergessen worden sein oder auf der Diskette (Kassette) sind bereits alle Daten gelesen worden.

FC Illegaler Funktionsaufruf: Es wurde der Versuch gemacht, eine Operation mit ungültigen Parametern auszuführen.

Beispiel: Quadratwurzel mit negativem Argument, negative Matrixdimensionen, Negativ- oder Null-Argumente für LOG, USR-Aufruf, ohne die Startadresse zu POKEN.

OV Überlauf: Ein berechneter Wert oder eine Eingabe ist zu groß oder zu klein, der Computer kann ihn nicht mehr handhaben.

OM Kein Speicherplatz mehr: Der gesamte Speicherplatz ist entweder benutzt oder reserviert worden. Die Ursache können zu große Felddimensionen oder verschachtelte Sprungbfehle wie GOTO, GOSUB und FOR-NEXT sein. Es kann aber auch schlicht das Programm zu lang sein.

UL Undefinierte Zeile: Es wurde versucht zu einer nicht vorhandenen Zeile zu springen.

BS Dimensionen aus ihrem Bereich: Es wude versucht, ein Feldelement anzusprechen, dessen Dimensionen über der im DIM-Statement angegebenen liegt.

DD Redimensionierung: Es wurde versucht ein Feld zu dimensionieren, das schon dimensioniert ist (mit DIM oder implizit).

/0 Division durch Null: Es wurde versucht, Null als Divisor zu benutzen.

ID Illegaler,direkter Einsatz: Es wurde versucht, das INPUT Statement auf der Kommandoebene auszuführen.

TM Typ stimmt nicht überein: Es wurde versucht, in eine Zahlenvariable eine Zeichenkette zu schreiben oder umgekehrt.

OS Kein Platz mehr für Zeichenkette: Siehe CLEAR

LS Zeichenkette zu lang: Eine Zeichenkette darf nur 255 Zeichen lang sein.

ST Stringformel zu komplex: Zeichenkettenformel ist zu komplex, um sie zu handhaben.

CN Kann CONT nicht ausführen: Ein CONT wurde eingegeben, obwohl kein Programm zur Fortführung im Speicher vorhanden ist.
Z.B. wenn das Programm mit END abgeschlossen wurde oder nach EDIT.

NR kein RESUME: Das Programmende wurde im Fehlererkennungsmodus erreicht.

RW RESUME ohne ERROR: Ein RESUME wurde vor einer ON ERROR GOTO-Anweisung aufgefunden.

UE Nicht ausgebbarer Fehler: Es wurde versucht, einen Fehler mit nicht existierendem ERROR-Code mit ERROR zu simulieren.

MO Fehlender Operand: Es wurde versucht, eine Operation auszuführen, bei der einer der Operanden fehlte.

FD Defekte Datei: Die Dateneingabe von einer externen Quelle (Kassettenrekorder) war falsch.

Weitere mögliche Fehler, die sich speziell auf die Arbeit mit Diskettenlaufwerken beziehen, können Sie im G-DOS Handbuch nachlesen. Sie treten nur im erweiterten Disk-BASIC auf.

3.18 Der EDITOR

Der Editor des GENIE-Systems erlaubt seinem Benutzer, Programmzeilen, die er geschrieben hat, zu korrigieren, ohne sie neu eintippen zu müssen. Die Notwendigkeit eines Editors wird besonders bei langen, komplexen Programmzeilen deutlich. In diesem Kapitel werden wir alle Editfunktionen des GENIE-Systems mit ihren Unterkommandos besprechen und mit vielen Beispielen verständlicher machen.

EDIT - Zeilennummer

Dieses Kommando bringt den Computer von der aktiven Kommandoebene in die Editierungsebene. Der Benutzer muß angeben, welche Zeile er editieren will. Wird keine Zeilennummer angegeben, reagiert der Computer mit einer UL-Fehlermeldung.

Beispiel:

>EDIT 100	eröffnet Editierung von Zeile 100
>EDIT .	eröffnet Editierung der gerade eingetippten Zeile.

Auf der Editierungsebene kann der Computer folgende Subkommandos ausführen:

RETURN-Taste

Wird die RETURN-Taste betätigt, wenn sich der Rechner in der Editierungsebene befindet, so speichert er alle vorher gemachten Änderungen ab und kehrt in die aktive Kommandoebene zurück.

LEERTASTE

Im Editierungsmodus wird bei jeder Betätigung der Leertaste ein weiteres Zeichen in der editierten Zeile angezeigt. Wird eine Zahl n vor der Betätigung der Leertaste eingegeben, so werden n Zeichen der gerade editierten Zeile ausgegeben.

Nehmen wir an, wir haben folgende Zeile eingegeben:

```
100 IF A = B THEN 150 : A = A + 1 : GOTO 100
```

Soll nun diese Zeile 100 editiert werden, so muß der Benutzer EDIT 100 eingeben.

```
>EDIT 100
```

Der Computer antwortet:

```
100
```

Wird die Leertaste zwölfmal betätigt, so bewegt sich der Cursor um 12 Zeichen nach rechts.

Auf dem Schirm steht:

```
100 IF A = B THE
```

Man muß nicht für jedes Zeichen einzeln die Leertaste betätigen. Geben Sie eine 8, gefolgt von der Leertaste ein, so erscheint folgende Zeile:

```
100 IF A = B THEN 150 :
```

Geben Sie eine 20, gefolgt von der Leertaste ein, so erscheint schließlich die gesamte Zeile:

```
100 IF A = B THEN 150 : A = A + 1 : GOTO 100
```

<LINKSPFEIL>-Taste

Bewegt den Cursor ein Zeichen zurück (entgegengesetzt zur Leertaste). Wird eine Zahl vor dem Betätigen der <LINKSPFEIL>-Taste eingegeben, so wird der Cursor um dem Betrag dieser Zahl nach links bewegt.

Entgegen der normalen Backspace-Funktion bleiben die so auf dem Bildschirm gelöschten Zeichen im Speicher erhalten.

Beispiel:

```
100 IF A = B THEN 150 : A = A + 1 : GOTO 100
```

Nach fünfmaligem Betätigen der <LINKSPFEIL>-Taste:

```
100 IF A = B THEN 150 : A = A + 1 : GOT
```

Geben Sie nun eine 10, gefolgt von der <LINKSPFEIL>-Taste ein, so erscheint folgende Zeile:

```
100 IF A = B THEN 150 : A = A
```

Nochmals betont, um den Editierungsmodus zu verlassen, betätigen Sie die <RETURN>-Taste. Der Computer geht zurück auf die aktive Kommandoebene und auf dem Schirm steht dann der Prompt und der Cursor. Wenn nun noch weitere Änderungen an Zeile 100 vorgenommen werden sollen, müssen Sie mit 'EDIT 100' in den Editierungsmodus zurück.

<SHIFT> und <HOCHPFEIL>

Wird die <SHIFT>-Taste und die <HOCHPFEIL>-Taste gleichzeitigg betätigt, so stellt der Computer die Ausführung eines vorher gestarteten Subkommandos ein.

Der Computer befindet sich danach weiter in der Editierungsebene und die Position des Cursors bleibt unverändert. Ein anderer Weg, um die Ausführung eines Subkommandos zu beenden, ist die <RETURN>-Taste. Wird sie betätigt, so kehrt der Computer auf die aktive Kommandoebene zurück.

<H>-Taste

<H> steht für abtrennen (engl. hack) und einsetzen. D.h. der Rest der Zeile wird gelöscht und an der momentanen Position des Cursors können Zeichen eingegeben werden.

Beispiel:

```
100 IF A = B THEN 150 : A = A + 1 : GOTO 100
```

Nehmen wir an, Sie wollten A = A + 1 durch A = A + B ersetzen und das 'GOTO 100' löschen. Zunächst gehen Sie in den Editierungsmodus mit EDIT 100.
Die Zahl 25, gefolgt von der Leertaste, bringt den Cursor auf die 25. Position der Zeile:

```
100 IF A = B THEN 150 : A = A
```

Nun betätigen Sie die <H>-Taste und tippen B ein. Dann drücken Sie die <SHIFT>- und <HOCHPFEIL>-Taste. Geben Sie nun ein <L> für List ein. Der Computer gibt einmal die gesamte Zeile aus und der Cursor geht an den Anfang der Zeile:

```
100 IF A = B THEN 150 : A = A + B
100
```

Alles, was nicht ausgegeben wurde, ist effektiv gelöscht.

<I>-Taste

<I> steht für einsetzen (engl. insert). Mit diesem Subkommando können Sie Zeichen an einer beliebigen Position einsetzen, ohne andere Teile ändern zu müssen.

Beispiel:

Nehmen wir an, wir wollen den Befehl 'PRINT A' zwischen 'A = A + 1' und 'GOTO 100' in der Zeile 100 einsetzen. Die Zeile 100 soll vorher so aussehen:

```
100 IF A = B THEN 150 : A = A + 1 : GOTO 100
```

Mit der Leertaste bringen Sie den Cursor zur Position:

```
100 IF A = B THEN 150 : A = A + 1 :
```

Nun betätigen Sie die <I>-Taste und geben 'PRINT A :' ein. Mit der <SHIFT>- und der <HOCHPFEIL>-Taste verlassen Sie den Einsetzmodus (insertmode). Nun können Sie mit <L> die gesamte Zeile listen:

```
100 IF A = B THEN 150 : A = A + 1 : PRINT A
: GOTO 100
100
```

Mit <RETURN> können Sie dann auf die aktive Kommandoebene zurück.

<X>-Taste

<X> meint: setze am Ende der Zeile ein. Der Cursor wird an das Ende der Zeile bewegt und der Computer geht in den Einsetzmodus. Man kann nun am Ende der Zeile hinzufügen oder mit der <LINKSPFEIL>-Taste am Ende der Zeile Zeichen löschen.

Beispiel:

In den Editierungsmodus:

```
>EDIT 100
100
```

Geben Sie nun <X> (ohne <RETURN>-Taste) ein:

```
100 IF A = B THEN 150 : A = A + 1 : PRINT A
: GOTO 100
```

An dieser Stelle können Sie nun Zeichen eingeben, schon existierende löschen und dieses Subkommando mit <SHIFT> und <HOCHPFEIL> verlassen.

<L>-Taste

<L> steht für Zeilenlisten. Ist der Computer im Editierungsmodus, führt aber im Moment keine der Subkommandos H, I oder X aus, so kann man mit dem Subkommando <L> den verbliebenen Teil der Zeile auf dem Bildschirm ausgeben.

```
>EDIT 100
100
```

Betätigen Sie die <L>-Taste:

```
100 IF A = B THEN 150 : A = A + 1 : PRINT A
: GOTO 100
100
```

In der zweiten Zeile können Sie nun editieren und dabei den Ursprungstext im Auge behalten.

<A>-Taste

<A> meint: beginne neu (cancel and restart). Der Cursor wird wieder zum Zeilenanfang gebracht und alle vorher gemachten Änderungen gelöscht. Die Zeile bleibt in ihrem ursprünglichen Zustand.

<E>-Taste

Das Kommando <E> (exit) bringt den Computer zurück auf die aktive Kommandoebene und trägt alle vorher gemachten Änderungen in der editierten Zeile ein. Der Computer darf kein Subkommando ausführen (H, I und X), wenn <E> eingegeben wird.

<Q>-Taste

Das Kommando <Q> (quit) bringt den Computer von der Editierungsebene zurück auf die aktive Kommandoebene, aber löscht alle vorher gemachten Änderungen. Die Zeile bleibt in ihrem ursrünglichen Zustand.

<D>-Taste

<D> steht für löschen (delete). Wird vor <D> eine Zahl n eingegeben, so werden n Zeichen rechts von der momentanen Cursorposition gelöscht. Die gelöschten Zeichen werden in Ausrufezeichen eingeschlossen, um zu zeigen, daß sie von der Operation betroffen sind.

Beispiel:

```
100 IF A = B THEN 150 : A = A + 1 : PRINT A
: GOTO 100
```

Schalten Sie auf die Editierungebene und bewegen Sie den Cursor mit der Leertaste in folgende Position:

```
100 IF A = B THEN 150 : A = A + 1
```

Nun geben Sie 15D ein (lösche 15 Zeichen):

```
100 IF A = B THEN 150 : A = A + 1! : PRINT A
: GO!
```

Benutzen Sie <L>, um die gesamte Zeile zu listen:

```
100 IF A = B THEN 150 : A = A + 1! : PRINT A
: GO!TO 100
100
```

Ein weiteres List bringt folgendes auf den Schirm:

```
100 IF A = B THEN 150 : A = A + 1 TO 100
100
```

Um das noch übriggebliebene 'TO 100' zu löschen, können Sie das Kommando <D> ein zweites Mal benutzen.

<C>-Taste

<C> steht für verändern (change). Wird eine Zahl n vor dem <C> eingegeben, so werden n Zeichen rechts von der momentanen Cursorposition vereändert. Wird die Zahl nicht angegeben, so wird nur ein Zeichen verändert.

Beispiel:

```
100 IF A = B THEN 150 : A = A + 1
```

Wollen Sie die 150 in eine 230 ändern, so schalten Sie den Editierungsmodus (EDIT 100) ein und bewegen den Cursor mit der Leertaste an folgende Position:

```
100 IF A = B THEN
```

Geben Sie nun ein 2C (ändere 2 Zeichen), gefolgt von 23 (neue Daten).
Listen Sie nun die Zeile mit <L>.

```
100 IF A = B THEN 230 : A = A + 1
100
```

n <S> c

Dieses Kommando sucht (search) nach dem n-ten Vorkommen des Zeichens c in der gerade editierten Zeile und setzt den Cursor an dessen Position. Wird n nicht angegeben, sucht der Rechner nach dem ersten Auftreten des Zeichens c. Wird das Zeichen c nicht gefunden, so steht der Cursor am Ende der Zeile. Wie bei den anderen Subkommandos, beginnt der Rechner mit der Suche an der momentanen Cursorposition.

Beispiel:

```
100 IF A = B THEN 230 : A = A + 1
```

Wenn Sie nun im Editierungsmodus sind, so geben Sie 2S= ein, um dem Computer mitzuteilen, er soll nach dem zweiten Auftreten des '=' suchen:

```
100 IF A = B THEN 230 : A
```

An der Stelle, wo die gesuchte Date gefunden wurde, kann nun ein anderes Subkommando aktiviert werden, um zum Beispiel dieses Zeichen zu ändern.

```
n <K> c
```

Dieses Kommando löscht alle Zeichen bis zum n-ten Vorkommen des Zeichens c und setzt den Cursor an diese Stelle.

Folgendes Beispiel:

```
100 IF A = B THEN 230 : A = A + 1
```

Schalten Sie nun auf den Editierungsmodus (EDIT 100)

```
100
```

Nun geben Sie ein 1K: , um dem Computer mitzuteilen, er solle nach dem ersten Auftreten des Zeichens ':' suchen und solle alles löschen, was vor ihm in der Zeile steht:

```
100 !IF A = B THEN 230!
```

Soll der Doppelpunkt noch dazu gelöscht werden, geben Sie <D> ein:

```
100 !IF A = B THEN 230!!:!
```

Um das Ergebnis zu betrachten, geben Sie <L> ein (2 mal):

```
100 A = A + 1
100
```

3.19 BASIC Programm Statements

Wir wollen in diesem Kapitel die Programmbefehle (Statements) unseres BASIC erläutern. Im ersten Teil setzen wir uns mit den Ein- und Ausgabebefehlen, die dem Computer zur Verfügung stehen, auseinander. Besonders behandelt werden solche, die mit dem Bildschirm und der Tastatur arbeiten und die Datenspeicherung auf der Kassette ermöglichen.

Der zweite Teil des Kapitels handelt von verschiedenen Funktionen der Programmbefehle, die vom GENIE interpretiert werden. Da dies eine recht große Anzahl von Statements ist und jedes seine eigenen charakteristischen Eigenschaften hat, wird dem Benutzer nahe gelegt, jedes Statement mit Hilfe der angegebenen Beispiele genau zu studieren.

Ein- und Ausgabestatements:

PRINT ITEMLISTE

Schreibt ein ITEM oder eine ITEMLISTE auf den Bildschirm. Als Item wird folgendes aufgefaßt:

a) Numerische Konstante (Zahlen wie 0, 32455, 0.2, -32)

b) Zeichenkonstanten (Zeichen in Anführungszeichen wie etwa "GENIE", "123=%?" usw.)

c) Zeichenvariablen (Namen, die eine Zeichenkonstante repräsentieren wie A$, X$ usw.)

d) Ausdrücke (Verknüpfungen der obigen Items, wie X+10/Y oder "COMP"+"UTER" usw.)

Die Items in der Itemliste können durch Komma oder Semikolon getrennt werden. Wird ein Komma benutzt, geht der Cursor automatisch zur nächsten Schreibzone, bevor er das Item ausgibt. Wird ein Semikolon benutzt, so wird kein Platz zwischen den Items gelassen; eines wird an das andere gehängt. Bei numerischen Items wird allerdings eine Leerstelle gelassen.

Beispiele:

```
>10 N = 25 + 7
>20 PRINT" 25 + 7 IST GLEICH ";N
>30 END
>RUN
 25 + 7 IST GLEICH 32
READY

>10 H$ = "PERSONAL"
>20 C$ = "COMPUTER"
>30 PRINT "TESTEN SIE UNSEREN "; H$; C$
>RUN
TESTEN SIE UNSEREN PERSONALCOMPUTER
READY
>
```

Werden Kommata benutzt, um die Items zu trennen, so erzeugt der Computer mehrere Spalten pro Zeile. Jede Spalte kann maximal 9 Zeichen enthalten. Zeichen, die darüber hinausgehen, werden in die nächste Zeile geschrieben. Werden zwei oder mehr Kommata angegeben, so erzeugt jedes 9 Leerstellen.

Beispiel:

```
>10 PRINT "SPALTE 1",,"SPALTE 2"
>20 END
>RUN
 SPALTE 1          SPALTE 2
READY
>
```

Beachten sie folgendes Beispiel:

```
>10 PRINT "ZEILE 1"
>20 PRINT "ZEILE 2"
>30 END
>RUN

ZEILE 1
ZEILE 2
```

```
>10 PRINT "ZEILE 1",
>20 PRINT "ZEILE 2"
>30 END
>RUN

ZEILE 1   ZEILE 2
```

PRINT@ Stelle, Itemliste

Dieses Statement gibt die Items der Itemliste an der angegebenen Stelle auf dem Schirm aus. Das @-Zeichen muß unmittelbar auf das 'PRINT' folgen und Stelle darf Werte zwischen 0 und 1023 (64 Zeichen/ZeileModus) oder zwischen 0 und 1919 (80 Zeichen/Zeile-Modus) annehmen.

Beispiel:

```
>20 PRINT@ 100, "STELLE 100"
```

Wenn der Benutzer ein 'PRINT'-Statement eingibt, das auf die letzte Zeile des Schirms schreibt, wird automatisch ein Zeilenvorschub erzeugt, der alle Zeilen um eine Position nach oben wandern läßt. Um dies zu unterbinden, setzen Sie ein Semikolon an das Ende des Statements.

PRINT TAB (Ausdruck)

Erlaubt es, den Cursor an jede beliebige Stelle in der Zeile zu setzen. Man kann mehr als ein 'TAB' in einem 'PRINT'-Statement benutzen. Doch der Wert von Ausdruck muß zwischen 0 und einschließlich 255 liegen.

Beispiele:

```
>10 PRINT TAB (10)"POSITION 10" TAB (30)
"POSITION 30"

>RUN
          POSITION 10          POSITION 30
READY
>

>10 N = 4
>20 PRINT TAB (N)"POS"; N TAB (N+10)"POS"; N
+ 10
>30 END
>RUN
   POS 4      POS 14
READY
>
```

PRINT USING Format, Itemliste

Dieses Statement erlaubt es, Daten in einem vorher festgelegten Format auszugeben. Bei den Daten kann es sich um numerische Werte oder Zeichenketten handeln.

Die Format- und Item-Liste im 'PRINT USING'-Statement kann Variablen oder Konstanten enthalten. Das Statement schreibt die Items in der Form, wie es in der Formatangabe vorgegeben wird.

Die folgenden Deskriptoren können in dem Formatfeld benutzt werden:

\# Dieses Zeichen repräsentiert die richtige Stellung jeder Dezimalstelle in einer Zahl der Itemliste. Die Anzahl der #-Zeichen bildet das erwünschte Format. Ist das Formatfeld größer als die Anzahl der Stellen in der Zahl, so werden die unbenutzten Feldpositionen links von der Zahl als Leerzeichen und die rechts vom Dezimalpunkt als Nullen ausgegeben. Der Dezimalpunkt kann irgendwo in das durch die #-Zeichen gebildete Formatfeld gesetzt werden. Werden Nachkommastellen unterdrückt, so wird gerundet. Wird ein Komma in eine Position zwischen der ersten Ziffer und dem Dezimalpunkt gesetzt, so erscheint in der Ausgabe nach je drei Vorkommaziffern ein Komma (beachte: der Computer benutzt die angloamerikanische Dezimalschreibweise, das deutsche Dezimalkomma entspricht hier dem Punkt).

Beispiel:

```
>10 INPUT "FORMAT EINGEBEN"; F$
>20 IF F$ = "STOP" END
>30 INPUT "ZAHL EINGEBEN"; N
>40 PRINT USING F$; N
>50 GOTO 10
```

Dieses Programm fragt nach einer Formatliste und nach einem Item (in diesem Fall ein numerischer Wert). Es stoppt erst, wenn "STOP" eingegeben wird.

Nun Probieren Sie das Programm aus:

```
>RUN
FORMAT EINGEBEN? ##.##
ZAHL EINGEBEN? 12.34
12.34
FORMAT EINGEBEN? ###.##
ZAHL EINGEBEN? 12.34
12.34
FORMAT EINGEBEN? ##.##
ZAHL EINGEBEN? 123.45
%123.45
FORMAT EINGBEN? STOP
READY
>
```

Das %-Zeichen wird ausgegeben, wenn das angegebene Feld für die Zahl zu klein ist. Die letzte Zahl links vom Dezimalpunkt wird nach dem %-Zeichen ausgegeben.

Nun starten wir das obige Programm wieder:

```
>RUN
FORMAT EINGEBEN ? ##.##
ZAHL EINGEBEN ? 12.345
12.35
FORMAT EINGEBEN ? STOP
```

Weil im Format nur zwei Nachkomma- bzw. Punkt-Stellen angegeben sind, wird aufgerundet.

** Zwei Sternchen am Anfang des Formatfeldeldes bewirken, daß die unbenutzten Positionen links vom Dezimalpunkt mit Sternchen aufgefüllt werden. Die zwei Sternchen erzeugen zwei Feldpositionen mehr.

$$ Zwei Dollarzeichen am Anfang des Formatfeldes erzeugen ein gleitendes Dollarzeichen. D.h. ein Dollarzeichen wird von der höchsten Stelle der Anzahl ausgegeben.

**$ Kombiniert den Effekt von ** und $$. Jede leere Position links von der Zahl wird mit Sternchen aufgefüllt und das Dollarzeichen wird vor der höchsten Stelle der Zahl ausgegeben.

Betrachten sie das Beispielprogramm von vorhin:

```
>RUN
FORMAT EINGEBEN ? ** ##.##
ZAHL EINGEBEN ? 12.3
** 12.3
FORMAT EINGEBEN ? $$ ##.##
ZAHL EINGEBEN ? 12.34
$ 12.34
FORMAT EINGEBEN ? ** $ ###.##
ZAHL EINGEBEN ? 12.34
*** $ 12.34
FORMAT EINGEBEN ? STOP
```

+ Wird ein Pluszeichen am Anfang oder am Ende des Formatfeldes angegeben, so schreibt der Computer ein Pluszeichen für positive Zahlen und ein Minuszeichen für negative Zahlen am Anfang bzw. Ende der Zahl.

- Wird ein Minuszeichen an das Ende des Formatfeldes gesetzt, so wird ein Minuszeichen nach jeder negativen Zahl ausgegeben und ein Leerzeichen für positive Zahlen.

Beispiel:

```
>RUN
FORMAT EINGEBEN? "####.#
ZAHL EINGEBEN? 12345.6
12,346
FORMAT EINGEBEN? + ##.##
ZAHL EINGEBEN? -12.34
-12.34
FORMAT EINGEBEN? ##.## +
ZAHL EINGEBEN? -12.34
12.34-
FORMAT EINGEBEN? ##.##
ZAHL EINGEBEN? 12.34
12.34
FORMAT EINGEBEN? ##.###
ZAHL EINGEBEN? 123456
%123456.000
FORMAT EINGEBEN? STOP
```

Wird bei 'INPUT' ein Komma verwendet, muß der String mit einer Anführung <"> beginnen.

% LEERZEICHEN %

Dient zum Definieren eines Zeichenfeldes, daß mehr als ein Zeichen enthält. Die Länge des so formatierten Feldes ist die Anzahl der Leerzeichen zwischen den Prozentzeichen plus Zwei. Ein Ausrufezeichen bringt den Computer dazu, nur das erste Zeichen einer aktuellen Zeichenkette zu benutzen.

Beispiel:

```
>10 INPUT "FORMAT EINGEBEN"; F$
>20 IF F$ = "STOP" END
>30 INPUT "ZEICHENKETTE EINGEBEN"; C$
>40 PRINT USING F$; C$
>50 GOTO 10
```

Dieses Programm arbeitet ähnlich wie das Vorherige, benutzt jedoch keine numerische Variable, sondern eine Stringvariable.

Nun starten wir das Programm:

```
>RUN
FORMAT EINGEBEN? !
ZEICHENKETTE EINGEBEN? ABCDE
A
FORMAT EINGEBEN? %%
ZEICHENKETTE EINGEBEN? ABCDE
AB
FORMAT EINGEBEN? %    %
ZEICHENKETTE EINGEBEN? ABCDEF
ABCDE
FORMAT EINGEBEN? STOP
```

```
! Mit dem Ausrufezeichen können Sie
  Zeichenketten hintereinander hängen.

Beispiel:

>10 PRINT "GIB 3 ZEICHENKETTEN EIN"
>20 INPUT A$, B$, C$
>30 PRINT "DAS ERGEBNIS IST:"
>40 PRINT USING "!"; A$; B$; C$
>50 END
>RUN
GIB 3 ZEICHENKETTEN EIN
? ABC,XYZ,IJK
DAS ERGEBNIS IST:
AXI
READY
>

oder

>RUN
GIB 3 ZEICHENKETTEN EIN
? TROMMESCHLÄGER,COMPUTER,SYSTEME
DAS ERGEBNIS IST:
TCS
READY
>
```

Benutzt man mehr als das eine Ausrufezeichen, wird der erste Buchstabe der Zeichenkette mit sovielen Leerzeichen dahinter ausgedruckt, wie Leerzeichen zwischen den Ausrufezeichen eingesetzt wurden.

Beispiel:

```
>10 PRINT "GIB 3 ZEICHENKETTEN EIN"
>20 INPUT A$, B$, C$
>30 PRINT "DAS ERGEBNIS IST:"
>40 PRINT USING "!!!"; A$; B$; C$
>50 END
>RUN
GIB 3 ZEICHENKETTEN EIN
? XYZ,FGH,ABC
DAS ERGEBNIS IST:
X  F  A
READY
>
```

oder

```
>RUN
GIB 3 ZEICHENKETTEN EIN
? TROMMESCHLÄGER,COMPUTER,SYSTEME
DAS ERGEBNIS IST:
T  C  S
READY
>
```

INPUT Itemliste

Dieses Statement veranlaßt den Rechner, das Programm zu unterbrechen und zu warten, bis der Benutzer Daten eines spezifizierten Typs und einer spezifizierten Anzahl auf der Tastatur eingegeben hat.

Jedes Item (wenn mehr als eins eingegeben wird) muß durch ein Komma vom anderen getrennt werden.

Beispiel:

```
>10 INPUT A$, B$, A, B
```

Bei Ausführung dieses Statements muß der Benutzer zwei Zeichenketten und zwei numerische Werte eingeben. Die Reihenfolge der Eingaben muß konsistent sein. Führt der Computer ein 'INPUT'-Statement aus, so schickt er ein Signal zum Bildschirm. Und wartet auf die Eingabe(n). Man kann alle vier Werte in einer Zeile eingeben (durch Kommata getrennt).

In diesem Fall könnte die Eingabe wie folgt aussehen:

```
BANANE , ZITRONE , 59 , 3.14  <RETURN>-Taste
```

Der Computer ordnet die Werte wie folgt zu:

```
A$ = "BANANE"
B$ = "ZITRONE"
A = 59
B = 3.14
```

Eine andere Methode, diese Daten einzugeben, wäre es, sie in seperate Zeilen aufzuteilen. Nach jeder Betätigung der <RETURN>-Taste erinnert Sie der Computer, daß er noch Daten erwartet indem er zwei Fragezeichen und den Cursor ausgibt, bis er Werte für alle Variablen erhalten hat. Dann geht er zum nächsten Statement über. Die Eingaben müssen mit den Variablentypen kompatibel sein. Man darf also für eine numerische Variable keine Zeichenkette eingeben. Wenn solches versucht wird, reagiert der Computer mit:

```
? REDO
?
```

und erwartet neue Daten für alle Variablen der Itemliste.

Beispiel:

```
>10 INPUT A$, A
>20 PRINT A$, A
>30 PRINT
>40 GOTO 10
```

Programmstart:

```
>RUN
? STRING,10
STRING      10

? DIES IST EINE ZEICHENKETTE,13.5
DIES IST EINE ZEICHENKETTE      13.5

? ABCD,IJK
?REDO
? ABCDE
?? 25
ABCDE      25
<BREAK>-Taste
```

Besteht eine Eingabezeichenkette nur aus Leerzeichen, so muß sie in Anführungszeichen eingeschlossen werden.

Um klarer zu machen, welche Eingabe nun erwartet wird, kann man jedem 'INPUT'-Statement eine Nachricht mitgeben, die ausgegeben wird, bevor der Computer mit dem '?'-Zeichen Daten erwartet. Diese Nachricht muß dem 'INPUT'-Statement unmittelbar folgen, in Anführungszeichen eingeschlossen und von einem Semikolon gefolgt werden.

Beispiel:

```
>10 INPUT"ART DER WARE UND STÜCKZAHL"; N$, S
>RUN
ART DER WARE UND STÜCKZAHL?
```

DATA Itemliste

Dieses Statement erlaubt es dem Benutzer, Daten im Programm anzugeben und mit dem 'READ'-Statement auf sie zuzugreifen. Auf die Daten greift der Computer sequentiell, beginnend mit dem ersten Item und endend mit dem letzten, zu. Jedes Item darf eine Zeichenkette oder auch eine numerische Variable sein. Genau wie bei der Eingabe über die Tastatur, muß jeder Zeichenkette, die entweder ein Leerzeichen, Semikolon oder Komma enthält, in Anführungszeichen eingeschlossen werden.

Die Anordnung der Items im 'DATA'-Statement muß mit dem Typ der Variablen im dazugehörigen 'READ'-Statement übereinstimmen (es darf nicht versucht werden, Zeichenketten in numerische Variablen zu lesen.)
Das 'DATA'-Statement darf an beliebiger Stelle im Programm stehen.

Beispiel:

```
>10 READ A$, B$, C, D
>20 PRINT A$, B$, C, D
>30 DATA"ZEICHEN", "EIN LANGER SATZ"
>40 DATA 20, 137.54
>50 END
>RUN
ZEICHEN  EIN LANGER SATZ     20
    137.54
READY
>
```

READ Itemliste

Dieses Statement erlaubt es, Daten vom 'DATA'-System zu lesen und Variablen zuzuordnen. Die Werte im 'DATA'-Statement werden sequentiell vom 'READ'-Statement gelesen. Sind alle Daten im ersten 'DATA'-Statement, liest das nächste vorkommende 'READ'-Statement aus dem nächsten 'DATA'-Statement. Sind alle Werte in allen 'DATA'-Statements einmal gelesen und versucht danach ein 'READ'-Statement, weitere Werte zu bekommen, tritt ein "keine Daten mehr"-Fehler auf (OUT of DATA error) mit dem Code OD.

Beispiel:

```
>10 READ C$
>20 IF C$ = "EOF" GOTO 60
>30 READ Q
>40 PRINT C$,,Q
>50 GOTO 10
>60 PRINT : PRINT "ENDE DER LISTE" : END
>70 DATA BÜCHER,4,BLEISTIFTE,5,FÜLLER,6
>80 DATA KUGELSCHREIBER,6,TIPPEX,7,EOF
```

Geben Sie nun 'RUN' ein:

```
BÜCHER              4
BLEISTIFTE          5
FÜLLER              6
KUGELSCHREIBER      6
TIPPEX              7

ENDE DER LISTE
```

RESTORE

Dieses Statement erlaubt es dem nächsten 'READ'-Statement, das erste Item der ersten 'DATA'-Liste zu lesen, auch wenn zuvor schon andere 'READ's ausgeführt wurden.

Beispiel:

```
>10 READ A$ , A
>20 PRINT A$ , A
>30 RESTORE
>40 READ B$ , B
>50 PRINT B$ , B
>60 DATA"JUPP SCHMITT",25,"FRANZ SCHULZ",32
>70 DATA"BERND SEGER",18
>80 END
>RUN
JUPP SCHMITT  25
JUPP SCHMITT  25
READY
>
```

PRINT #-1, Itemliste

Dieses Statement gibt die Werte der angegebenen Variablen in Itemliste auf dem Kassettenrecorder aus. Der Recorder muß geeignet vorbereitet sein, bevor dieses Statement ausgeführt wird.

Beispiel:

```
>10 A$ = "ANFANG BAND"
>20 B$ = "3.1416"
>30 C$ = "50"
>40 D$ = "DATEN"
>50 PRINT #-1,A$,B$,C$,D$,"ENDE DER DATEI"
```

Dieses Programm weist zunächst den Variablen A$, B$, C$ und D$ Werte zu und schreibt sie dann auf Band. Beachten Sie, daß die Zeichenkettenkonstante "ENDE DER DATEI" genauso auf Band geschrieben wird, wie die Variablen. Sind die Daten einmal Band gespeichert, so können Sie zurückgelesen werden. Das ist im Prinzip der gleiche Vorgang, wie beim Abspielen einer Musikkassette.

WICHTIG

Die gesamte Anzahl der einzelnen Zeichen in allen Variablen und Konstanten der Itemliste eines 'PRINT #'-Statements darf 247 nicht übersteigen. Werden mehr als 247 Zeichen angegeben, so werden die restlichen ignoriert. (Es zählen auch die Kommata, die der Trennung einzelner Items dienen.)

Beispiel:

```
>10 PRINT #-1,A$,B$,C$,D$,E$
```

Nehmen wir an, die gesamte Anzahl der Zeichen in A$, B$, C$ und D$ beträgt 243 und E$ habe die Länge 35. E$ wird nicht auf Band gespeichert, der Versuch, E$ später wieder einzulesen, erzeugt einen OD-Error.

Sollen numerische Variablen auf Band geschrieben werden, so sind sie mit 'STR$' vorher in Zeichenketten umzuwandeln.

INPUT #-1, Itemliste

Dieses Statement veranlaßt den Computer, Werte vom Recorder zu lesen und sie in die Variablen der Itemliste zu schreiben.

Beispiel:

```
>10 INPUT #-1,A$,B$,C$,D$
```

Dieses Statement liest Daten vom Recorder. Der erste Wert wird A$ zugewiesen, der zweite B$ usw. Die <PLAY>-Taste beim Recorder muß gedrückt sein.
Ein OD-Error tritt auf, wenn nicht genug Daten für alle Variablen in der Itemliste auf Band stehen.
Durch 'STR$' umgewandelte Zahlen können mit 'VAL' wieder zurückgewandelt werden.

3.20 Programm-Befehle

DEFINT Buchstabenbereich

Variablen, die mit einem Buchstaben aus dem definierten Buchstabenbereich beginnen, werden als ganze Zahlen (integers) behandelt und abgespeichert. Eine Typenangabe (mit %, $ usw.) aber überschreibt diese Typendefinition. Erklärt man eine Variable zur ganzzahligen Variable, so spart man damit nicht nur Speicherplatz sondern auch Rechenzeit. Berechnungen mit ganzzahligen Variablen sind schneller als solche mit Variablen einfacher und doppelter Genauigkeit. Beachten Sie, daß eine ganzzahlige Variable nur Werte zwischen -32768 und +32767 annehmen kann.

Beispiel:

```
10 DEFINT X, Y, Z
```

Hat der Computer Zeile 10 ausgeführt, so werden alle Variablen, die mit X, Y, oder Z anfangen, als ganzahlige behandelt. Daher sind von da an X2, X3, YA, YB, ZI, ZJ ganzzahlige Variablen. Jedoch X1#, YA#, YB#, ZI#, ZJ# bleiben Variablen doppelter Genauigkeit weil die Typenfestlegung mit "#" die mit 'DEFINT' überschreibt.

Beispiel:

```
>10 DEFINT A - D
```

Erklärt alle Variablen, die mit A, B, C oder D beginnen, zu ganzzahligen Variablen.

Beachten sie, das 'DEFINT' zwar an beliebiger Stelle im Programm eingesetzt werden kann, aber es die Bedeutung von Variablen ohne explizit Typendeklaration ($, #, %) recht unkontrolliert ändern kann. Deshalb sollte es normalerweise an den Anfang eines Programmes gestellt werden.

DEFSNG Buchstabenbereich

Variablen, die mit einem Buchstaben aus 'Buchstabenbereich' beginnen, werden als Variablen einfacher Genauigkeit behandelt. Explizite Typenfeslegung (mit #, $, %) überschreibt diese Definition. Variablen und Konstanten einfacher Geneuigkeit werden mit 7 Stellen gespeichert und mit 6 Stellen ausgegeben. Alle numerischen Variablen haben einfache Genauigkeit wenn nichts anderes angegeben wird.

Beispiel:

```
>10 DEFSNG A-D,Y
```

Macht alle Variablen, die mit A-D oder Y beginnen zu Variablen einfacher Genauigkeit. Aber A# bleibt weiterhin eine Variable doppelter Genauigkeit und Y% bleibt eine ganzzahlige Variable.

DEFDBL Buchstabenbereich

Variablen, die mit einem Buchstaben aus 'Buchstabenbereich' beginnen, werden als Variablen doppelter Genauigkeit behandelt und gespeichert. Eine explizite Typenfestlegung (mit !, $, %) kann diese Definition überschreiben. Variablen doppelter Genauigkeit ermöglichen das Rechnen mit 17 Stellen, wovon 16 ausgegeben werden.

Beispiel:

```
>10 DEFDBL M-P,G
```

Macht alle Variablen, die mit M-P oder G beginnen, zu Variablen doppelter Genauigkeit (M% oder G# bleiben unbeeinflußt).

DEFSTR Buchstabenbereich

Variablen, die mit einem Buchstaben aus 'Buchstabenbereich' beginnen, werden als Zeichenkettenvariablen behandelt und abgespeichert.

Explizite Typenfestlegung (mit #, !, %) überschreibt diese Definition. Wenn genug Platz für Zeichenketten zur Verfügung steht (siehe 'CLEAR'), kann eine Zeichenkettenvariable bis zu 255 Zeichen aufnehmen.

Beispiel:

```
>10 DEFSTR A-D
```

Macht alle Variablen, die mit einem Buchstaben von A-D beginnen zu Zeichenkettenvariablen, außer es wurde eine explizite Typenfestlegung angegeben (mit #, !, %). Nach der Ausführung von Zeile 10, wird folgender Ausdruck richtig:

```
B3 = "EINE ZEICHENKETTE"
```

CLEAR n

Dieses Statement setzt alle Variablen auf Null. Wird eine Zahl n angegeben, so stellt der Computer n Bytes zur Speicherung von Zeichenketten ab.

Wird das GENIE eingeschaltet, so werden jedes Mal automatisch 50 Bytes zur Speicherung von Zeichenketten freigesetzt.

Das 'CLEAR'-Statement wird dann kritisch, wenn der Computer während der Programmausführung auf einen 'OUT of STRINGSPACE'-Fehler (OS Error) stößt. Dieser Fehler tritt auf, wenn die Zeichenketten des Programms mehr Platz beanspruchen, als freigegeben wurde.

Beispiel:

```
>10 CLEAR 1000
```

Setzt 1000 Bytes für Zeichenketten frei und löscht alle Variablen.

DIM Name (Dim 1,...,Dim n)

Dieses Statement legt eine Variable oder eine Liste von Variablen als Feld (array) fest. Die Zahl der Elemente jeder Dimension können mit 'Dim 1, Dim 2' usw. angegeben werden. Wird 'DIM' nicht ausgeführt, so wird angenommen, daß jede Dimension 11 Elemente hat. Die Anzahl der möglichen Dimensionen ist nur durch den Speicherplatz begrenzt.

Beispiel:

```
>10 DIM A(5), B(3,4), C(2,3,3)
```

Dieses Statement definiert A als eindimensionales Feld (Vektor) mit 6 Elementen (0 - 5), B als zweidimensionales Feld mit 20 Elementen (4 mal 5) und C als dreidimensionales Feld mit 48 Elementen (3 mal 4 mal 4).

'DIM' darf an beliebiger Stelle in das Programm gesetzt werden. Die Dimensionsangabe darf ganzzahlig oder ein Ausdruck sein.

Beispiel:

```
>10 INPUT "ANZAHL DER PUNKTE", N
>20 DIM A(N+2,4)
```

Die Anzahl der Elemente von A kann, abhängig von N, verändert werden. Um ein Feld neu zu dimensionieren, muß zuvor ein 'CLEAR'Statement ohne Argument n eingegeben werden, sonst wird ein DD-Error ausgegeben.

Beispiel:

```
>10 X(2) = 13.6
>20 PRINT "DAS ZWEITE ELEMENT IST:";X(2)
>30 DIM X(15)
>40 PRINT X(2)
>50 END
>RUN
DAS ZWEITE ELEMENT IST: 13.6
? DD ERROR IN 30
```

Durch den Gebrauch von X(2) in 10, vor dem 'DIM' in 30, wurde es implizit mit 'DIM' X(10) dimensioniert.

LET Variable = Ausdruck

Dieses Statement ordnet einer Variablen einen Wert zu. Das Wort 'LET' ist in einem solchen Zuordnungsstatement beim GENIE BASIC nicht umbedingt erforderlich, wurde aber in seinen Sprachumfang aufgenommen, um Kompatibilität mit anderen Systemen zu gewährleisten.

Beispiel:

```
>10 LET A = 5.67
>20 B% = 20
>30 S$ = "ZEICHEN"
>40 LET D% = D% + 1
>50 PRINT A, B%, S$, D%
>60 END
>RUN
5.67        20        ZEICHEN        1
```

In den obigen Zuordnungen wird in die Variable rechts vom Gleichheitszeichen der Wert des Ausdrucks links vom Gleichheitszeichen geschrieben. All diese Statements sind korrekt.

END

Diese Statement bewirkt die Beendigung der Programmausführung. Das 'END'-Statement wird primär zur Beendigung von Programmen an einer anderen Stelle als am Ende ihres logischen Textes eingesetzt.

Beispiel:

```
>10 B = 3 : C = 14
>20 A = C + B
>30 GOSUB 80
>40 D = X + Y
>50 PRINT "DAS ERGEBNIS IST:";
>60 PRINT A, D
>70 END
>80 X = 50
>90 Y = A * X
>100 RETURN
>RUN

DAS ERGEBNIS IST: 17             900
```

Das 'END'-Statement in Zeile 70 hält den Computer davon ab, die Zeilen 80 bis 100 auszuführen. Damit kann das Unterprogramm, das in Zeile 80 beginnt, nur von der Zeile 30 ausgeführt werden.

STOP

Dieses Statement ist eine große Hilfe bei der Fehlersuche in Programmen. Es setzt einen Unterbrechungspunkt (break point) in der Programmausführung und erlaubt dann, Werte zu verändern oder auszugeben. Führt der Computer einen 'STOP'-Befehl aus, so gibt er die Nachricht 'BREAK IN' Zeilennummer (Unterbrechung in der Zeile...) aus.

Mit dem aktivem Kommando 'CONT' können sie die Programmausführung an dem Punkt, an dem vorher unterbrochen wurde, wieder aufnehmen.

Beispiel:

```
>5  INPUT B, C
>10 A = B + C
>20 STOP
>30 X = ( A + D) / 0.74
>40 IF X < 0 GOTO 70
>50 PRINT A, B, C
>60 PRINT X
>70 END

>RUN

>? 2, 4
>BREAK IN 20
>READY
>PRINT A
>6
>READY
>CONT

6       2       4
8.10811
```

Das 'STOP'-Statement gab uns die Möglichkeit, den Wert von A zu betrachten, bevor Zeile 30 ausgeführt wurde.

GOTO Zeilennummer

Dieses Statement übergibt die Kontrolle über das Programm an die angegebene Zeile (Sprungbefehl). Wird es unabhängig benutzt, so wird ein unbedingter Sprung ausgeführt. Ein Bedingungsstatement kann vor das 'GOTO' gesetzt werden, um einen bedingten Sprung zu erzeugen.

Beispiel:

```
>10 A = 10
>20 B = 45
>30 C = A + B
>40 C = C * 3.4
>50 GOTO 90
>60 ...
>70 ...
>80 ...
>90 PRINT "A ="; A, "B ="; B, "C ="; C >100
END
>RUN
 A = 10        B = 45        C = 187
READY
>
```

Wird Zeile 50 ausgeführt, so wird die Kontrolle an Zeile 90 übergeben.

Beispiel:

```
>10 IF A = 2 GOTO 140
```

Wird Zeile 10 ausgeführt und A = 2, dann springt der Computer zur Zeile 140. Ist A nicht gleich 2, geht er zum nächsten Statement.
Man kann das 'GOTO' auch auf der aktiven Kommandoebene als Alternative zum 'RUN' benutzen. Mit diesem Vorgehen vermeidet man, daß die Variablen gelöscht werden.

GOSUB Zeilennummer

Übergibt die Kontrolle an die Zeile, in der das angegeben Unterprogramm beginnt. Führt der Computer dann in dem Unterprogramm ein 'RETURN'-Statement aus, so kehrt er zu dem nach dem 'GOSUB' folgenden Statement im (Haupt-) Programm zurück. Wie beim 'GOTO' darf auch beim 'GOSUB' ein Bedienungsstatement vorausgehen, wie etwa:

```
>10 IF A = B THEN GOSUB 100
```

Beispiel:

```
>10 PRINT "HAUPTPROGRAMM" : PRINT
>20 GOSUB 50
>30 PRINT "HAUPTPROGRAMM ENDE" : PRINT
>40 END
>50 PRINT "UNTERPROGRAMM" : PRINT
>60 RETURN
>RUN
HAUPTPROGRAMM

UNTERPROGRAMM

HAUPTPROGRAMM ENDE

READY
>
```

RETURN

Dieses Statement beendet ein Unterprogramm und übergibt die Kontrolle an das Statement, welches dem 'GOSUB' folgt. Ein Fehler tritt auf, wenn für ein 'RETURN' kein entsprechendes 'GOSUB' vorhanden war (RG Error = RETURN without GOSUB).

ON n GOTO Zeilennummernliste

Dieses Statement erlaubt es, gleich mehrere Sprungziele anzugehen. Gesprungen wird in Abhängigkeit von n. Das generelle Format von 'ON n GOTO' ist:

ON Ausdruck GOTO 1.Zeilennr, 2.Zeilennr, ...

Der Wert von Ausdruck muß zwischen 0 und 255 einschließlich liegen.
Wird ein 'ON-GOTO'-Statement ausgeführt, so wird zunächst der ganzzahlige Anteil von Ausdruck berechnet (entspricht INT(Ausdruck)). Dieses Ergebnis sei n. Dann sucht der Computer das n'te Element der Zeilennummerliste und springt zu dieser Zeilennummer. Ist n nun größer als die Anzahl der angegebenen Zeilennummern, so wird das, auf das 'ON-GOTO'-Statement folgendes Statement ausgeführt. Ist n kleiner als Null, so tritt ein Fehler auf. Die Zeilennummernliste kann eine beliebige Anzahl von Zeilennummern enthalten.

Beispiel:

```
>10 INPUT "KOMMANDO EINGEBEN"; C
>20 ON C GOTO 100, 120, 130, 150, 130
>30 PRINT "ENDE DES PROGRAMMS": END
>100 PRINT "HIER ZEILE 100": GOTO 10
>120 PRINT "HIER ZEILE 120": GOTO 10
>130 PRINT "HIER ZEILE 130": GOTO 10
>150 PRINT "HIER ZEILE 150": GOTO 10
>RUN
KOMMANDO EINGEBEN? 5
HIER ZEILE 130
KOMMANDO EINGEBEN? 4
HIER ZEILE 150
KOMMANDO EINGEBEN? 1
HIER ZEILE 100
KOMMANDO EINGEBEN? 2
HIER ZEILE 120
KOMMANDO EINGEBEN? 3
HIER ZEILE 130
KOMMANDO EINGEBEN? 0
ENDE DES PROGRAMMS
READY
>RUN
KOMMANDO EINGEBEN? 4
HIER ZEILE 150
KOMMANDO EINGEBEN? 6
ENDE DES PROGRAMMS
READY
>
```

Das 'ON-GOTO'-Statement ist eine elegante Methode, um das gleiche zu erreichen, was folgennde 'IF...THEN GOTO'-Statements bewirken:

```
>10 IF C = 1 THEN GOTO 100
>20 IF C = 2 THEN GOTO 120
>30 IF C = 3 OR C = 5 THEN GOTO 130
>40 IF C = 4 THEN GOTO 150
>60 IF C < 1 OR C > 5 THEN GOTO 30
```

ON n GOSUB Zeilennummer-Liste

Arbeitet wie 'ON n GOTO', nur statt der Sprünge werden Unterprogramme aufgerufen.

Beispiel:

```
>10 PRINT" * FUNKTION DER UNTERPROGRAMME * "
>20 PRINT " 1. FUNKTION A"
>30 PRINT " 2. FUNKTION B"
>40 PRINT " 3. FUNKTION C"
>50 INPUT "GIB 1, 2 ODER 3 EIN"; N
>60 ON N GOSUB 150, 100, 250
>70 END
>100 PRINT "HIER FUNKTION B": RETURN
>150 PRINT "HIER FUNKTION A": RETURN
>250 PRINT "HIER FUNKTION C": RETURN
```

Programmstart:

```
>RUN

 * FUNKTION DER UNTERPROGRAMME *
 1. FUNKTION A
 2. FUNKTION B
 3. FUNKTION C
GIB 1, 2 ODER 3 EIN? 2
HIER IST FUNKTION B
READY
>
```

```
FOR Name = Ausdruck TO Ausdr. STEP Ausdr.
...
NEXT Name
```

Diese Statements bilden eine Schleife. Alle Statements, die zwischen 'FOR' und 'NEXT' stehen, werden einige Male ausgeführt.

Der Name steht für eine Zählervariable. Der Ausdruck hinter dem Gleichheitszeichen ist der Anfangswert. Er wird beim ersten Durchlauf der Schleife der Zählervariablen zugewiesen. Der zweite Ausdruck ist der Endwert der Schleife. Wird er überschritten, dann wird die Programmausführung hinter dem NEXT-Befehl fortgesetzt.

Der NEXT-Befehl bildet das Ende der Schleife. Bei seiner Ausführung wird der Wert des Ausdrucks hinter STEP (Schrittweite) zur Zählervariablen hinzugezählt. Dann folgt ein Vergleich, ob der Endwert überschritten wurde. Wenn nein, dann wird die Programmausführung beim ersten Befehl innerhalb der Schleife fortgesetzt. Ist der Endwert überschritten, dann wird die Programmausführung hinter dem NEXT-Befehl aufgenommen. Die Schleife ist damit beendet.
Der Teil 'STEP Ausdr.' im FOR-Statement kann auch entfallen. In diesem Fall wird für die Schrittweite automatisch der Wert 1 angenommen.
Es ist auch möglich die Schrittweite negativ zu wählen. Die Schleife endet dann bei Unterschreiten des Endwertes.

Beispiel:

```
>10 FOR K = 0 TO 1 STEP 0.3
>20 PRINT "WERT VON K :"; K
>30 NEXT K
>40 END
>RUN
DER WERT VON K : 0
DER WERT VON K : .3
DER WERT VON K : .6
DER WERT VON K : .9
```

Dann ist K = 1.2 und damit größer als der Endwert 1. Deshalb endet die Schleife dort und druckt 1.2 nicht mehr.

Beispiel:

```
>10 FOR N = 5 TO 0
>20 "PRINT DER WERT VON N : " ; N
>30 NEXT N
>40 END
>RUN
>DER WERT VON N : 5
READY
>
```

Die Schleife wird nur einmal durchlaufen, da der Anfangswert größer als der Endwert ist und keine negative Schrittweite angegeben ist.

Im folgenden Programm ist die negative Schrittweite enthalten und die Schleife wird dementsprechend abgearbeitet.

```
>10 FOR N = 5 TO 0 STEP -1
>20 PRINT "DER WERT VON N : " ; N
>30 NEXT N
>40 END
>RUN
DER WERT VON N : 5
DER WERT VON N : 4
DER WERT VON N : 3
DER WERT VON N : 2
DER WERT VON N : 1
DER WERT VON N : 0
READY
>
```

Beispiel:

```
>10 FOR A = 0 TO 3
>20 PRINT "DER WERT VON A:";A
>30 NEXT
>40 END
>RUN
DER WERT VON A : 0
DER WERT VON A : 1
DER WERT VON A : 2
DER WERT VON A : 3
READY
>
```

Wird kein Step angegeben, so wird automatisch 'STEP 1' angenommen und hochgezählt. Wird A = 4 und damit größer als der Endwert 3 endet die Schleife.
Statt 'NEXT A' können können wir in Zeile 30 auch einfach 'NEXT' schreiben. Bei der Programmierung von verschachtelten 'FOR-NEXT'-Schleifen ist es jedoch günstig anzugeben, welche Schleife man nun mit dem konkreten 'NEXT' abgeschlossen hat.

Hier ist ein Beispiel für geschachtelte Schleifen, daß zeigen soll, wie man den Zähler in jedem 'NEXT'-Statement identifiziert:

```
>10 I = 1
>20 J = 2
>30 K = 3
>40 FOR N = I + 1 TO J + 1
>50 PRINT "ERSTE  SCHLEIFE"
>60 FOR M = 1 TO K
>70 PRINT "ZWEITE SCHLEIFE"
>80 NEXT M
>90 NEXT N
>100 END
>RUN
ERSTE  SCHLEIFE
ZWEITE SCHLIEFE
ZWEITE SCHLEIFE
ZWEITE SCHLEIFE
ERSTE  SCHLEIFE
ZWEITE SCHLEIFE
ZWEITE SCHLEIFE
ZWEITE SCHLEIFE
READY
>
```

ERROR Code

Dieses Statement wird benutzt, um in eine 'ON ERROR GOTO'-Routine zu verzweigen. Führt der Computer ein 'ERROR Code'-Statement aus, so verhält er sich genau so, als wäre dieser Fehler aufgetreten. Er simuliert die Fehlersituation.

Beispiel:

```
>10 ERROR 1
>RUN
? NF ERROR IN 30
```

ON ERROR GOTO Zeilennummer

Dieses Statement erlaubt es, ein Abfangprogramm für Fehler (engli. error trapping routine) aufzurufen. Damit kann man, kommt es bei der Programmausführung zu einem Fehler, erreichen, daß das Programm nicht mit einer Fehlermeldung anhält, sondern in ein Programm verzweigt, das die Ursache erkennt und beseitigt. Meist hat der Benutzer einen bestimmten Fehlertyp im Auge, wenn er das 'ON ERROR GOTO'-Statement verwendet.
Wird innerhalb eines Programms eine Disvision ausgeführt bei der eine Null als Teiler nicht ausgeschlossen ist, kann diese Division mit einem Fehlerbehandlungs-Programm abgehandelt werden.

Beispiel:

```
>5 B = 15 : C = 0
>10 ON ERROR GOTO 120
>20 A = B/C
>30 PRINT A, B, C
>40 END
>120 PRINT"MAN DARF NICHT DURCH NULL TEILEN"
>130 RESUME 40
>RUN

MAN DARF NICHT DURCH NULL TEILEN
READY
>
```

In diesem Beispiel hat C den Wert 0 und in Zeile 20 kommt es zu einer Divsion durch 0, was normalerweise eine Fehlermeldung und Beendigung des Programms zur Folge hätte. Wegen Zeile 10 ignoriert der Computer aber diese Situation einfach und geht in Zeile 120 zurück, zur Fehlerbehandlugsroutine. Beachten sie, daß das 'ON ERROR GOTO'-Statement vor dem möglichen auftreten des Fehlers stehen muß, sonst hat es keine Wirkung. Außerdem muß die Fehlerbehandlungs-Routine mit 'RESUME' abgeschlossen werden.

Mit 'ON ERROR GOTO 0' kann man das Abfangen der Fehler wieder ausschalten.

RESUME Zeilennummer

Dieses Statement beendet eine Fehlerbehandlungs-Routine und gibt an, wo die normale Programmausführung weiter gehen soll.

'RESUME 0' oder 'RESUME' ohne Zeilennummer bewirkt, daß der Computer zu dem Statement zurückkehrt, in dem es zu einem Fehler gekommen war. Ist eine Zeilennummer angegeben, so geht er zu dieser Zeile zurück.

'RESUME NEXT' veranlaßt den Computer, zu dem Befehl zurück zu gehen, der hinter dem Befehl steht, in dem der Befehl erkannt wurde.

Beispiel:

```
>10 ON ERROR GOTO 80
>20 PRINT : PRINT "EINFACHE DIVISION."
>30 INPUT "GIB ZWEI ZAHLEN EIN"; A, B
>40 IF A = 0 END
>50 C = A / B
>60 PRINT "DER QUOTIENT IST :"; C
>70 GOTO 20
>80 PRINT "VERSUCH, DURCH 0 ZU TEILEN"
>90 PRINT "VERSUCHEN SIE ES NOCH EINMAL ..."
>100 RESUME 20
```

Start des Programms:

```
>RUN

EINFACHE DIVISION
GIB ZWEI ZAHLEN EIN? 6 , 2
DER QUOTIENT IST : 3

EINFACHE DIVISION
GIB ZWEI ZAHLEN EIN? 7 , 3
DER QUOTIENT IST : 2.33333

EINFACHE DIVISION
GIB ZWEI ZAHLEN EIN? 5 , 0
VERSUCH, DURCH 0 ZU TEILEN
VERSUCHEN SIE ES NOCH EINMAL ...

EINFACHE DIVISION
GIB ZWEI ZAHLEN EIN? 9 , 4
DER QUOTIENT IST : 2.25

EINFACHE DIVISION
GIB ZWEI ZAHLEN EIN? 0 , 0
READY
>
```

REM oder '

'REM' gibt Fußnoten an. Dieses Statement informiert den Computer, daß in dieser Zeile nur noch Kommentare folgen. Diese werden im Programmablauf ignoriert. Es ermöglicht dem Benutzer, durch die Kommentare übersichtlichere Programme zu schreiben.

Beispiel:

```
>10 REM  * BEDEUTUNG DER VARIABLEN *
>20 REM  * A = AUFWAND             *
>30 REM  * B = TEILEZAHL           *
>40 REM  * C = KOSTEN DER EINHEIT  *
>50 REM-------------------------------
>60 A = B * C : ' AUFWAND = TEILE MAL KOSTEN
```

IF Ausdruck Aktion

Dieses Statement bringt den Computer dazu, einen logischen und relationalen Ausdruck zu entscheiden. Ist der Ausdruck 'wahr', so wird 'Aktion' ausgeführt, ist er falsch, so wird 'Aktion' ignoriert und mit dem nächsten Statement fortgefahren. Numerisch bedeutet 'wahr', daß der Wert von 'Ausdruck' nicht Null ist.

Beispiel:

```
>10 PRINT: INPUT "GIB ZAHL EIN (MAX. 20)"; A
>20 IF A > 20 GOTO 60
>30 A = A * 3.1416 * 2
>40 PRINT "DER KREISUMFANG IST :"; A
>50 END
>60 PRINT "ZAHL ZU GROSS, MAX. 20" : GOTO 10
>RUN

GIB ZAHL EIN (MAX. 20)? 24
ZAHL ZU GROSS, MAX. 20

GIB ZAHL EIN (MAX. 20)? 18
DER KREISUMFANG IST : 113.098
READY
>
```

In dem Beispiel wird jedesmal, wenn eine Zahl größer als 20 eingegeben wird, eine Warnung ausgegeben und eine neue Eingabe erwartet. Ist A jedoch kleiner oder gleich 20, ignoriert der Computer das 'GOTO 60' in Zeile 20 und beendet seine Berechnung ohne eine Warnung auszugeben.

THEN Statement oder Zeilennummer

Bezeichnet den Beginn der 'Aktion' in einem 'IF...THEN'-Statement. 'THEN' kann entfallen, wenn es benutzt wird, um eine Zeilennummer als Sprungziel anzugeben.

Beispiel:

```
>120 INPUT A : IF A = 10 AND A < B THEN 160
>120 INPUT A : IF A = 10 AND A < B GOTO 160
```

Beide Statements haben die gleiche Wirkung.

ELSE Statement oder Zeilennummer

Dieses Statement kann nur hinter einem 'IF'-Statement stehen und gibt eine Alternative an, wenn der Ausdruck im 'IF'-Statement 'falsch' wird.

Beispiel:

```
>10 IF A = 1 THEN 60 ELSE 40
```

In diesem Beispiel wir nach 60 verzweigt, wenn A = 1 ist, ist dies nicht der Fall, so wird nach 40 verzweigt.
'IF...THEN...ELSE'-Statements können verschachtelt werden, aber die Anzahl von 'IF'- und 'ELSE'- Statements müssen übereinstimmen.

Beispiel:

```
>10 INPUT "GIB 3 ZAHLEN EIN"; X, Y, Z
>20 PRINT "DIE GRÖSSTE ZAHL IST :";
>30 IF X < Y OR X < Z THEN IF Y < Z THEN
PRINT Z ELSE PRINT Y ELSE PRINT X
>40 END
>RUN
GIB 3 ZAHLEN EIN? 30 , 75 , 73
DIE GRÖSSTE ZAHL IST: 75
```

Das Programm erwartet drei Zahlen als Eingabe und gibt die Größte der drei aus.

CLS

Mit diesem Statement kann man den Bildschirm löschen, daß heißt, daß alles, was dann geschrieben wird wieder in der obigen linken Ecke ausgegeben wird. Diesen Befehl benutzt man dazu, daß ein Programmablauf ordentlicher aussieht, denn normalerweise wird der Text nach oben herausgeschoben, wenn der untere Rand des Bildschirms erreicht wird. Dieser Befehl kann auch in der aktiven Kommandoebene benutzt werden.

LPRINT

Dieses Statement gibt Daten auf dem Drucker aus. Es arbeitet analog zum 'PRINT'-Statement.
Ist kein Drucker angeschlossen, so gerät der Computer bei Ausführung dieses Statements in eine endlose Schleife, aus der er nur durch gleichzeitiges Drücken der <RESET>-Tasten befreit werden kann.

Beispiel:

```
>10 FOR X = 1 TO 0 STEP - 0.25
>20 LPRINT "X BETRÄGT :"; X
>30 NEXT X
>40 END
>RUN
```

Wenn nun das Programm gestartet wird, kommt es zur folgenden Druckerausgabe:

```
X BETRÄGT :  1
X BETRÄGT :  .75
X BETRÄGT :  .5
X BETRÄGT :  .25
X BETRÄGT :  .0
```

3.21 CSAVE, CLOAD?, CLOAD

Dieses Kapitel ist nur für diejenigen wichtig, die mit einem Kassettenrecorder an Ihrem Computer arbeiten. Wenn Disketten-Laufwerke angeschlossen sind, können Sie ein äquivalentes Kapitel im G-DOS Handbuch nachlesen.

Beim Ausschalten verliert das GENIE-System alle Programme und Daten. Daher wird man Programme, die man später noch braucht, auf einer Kassette sichern. Das GENIE schreibt und lädt mit einer recht hohen Aufzeichnungsrate von 500 Baud, dies entspricht etwa 62 Zeichen pro Sekunde.

Der Kassettenrecorder wird mit einem Kabel an die DIN-Buchse, die sich auf der Rückseite des Rechners befindet, angeschlossen. Als Recorder sollten Sie einen handelsüblichen Recorder benutzen, der die Buchsen (für 3.5 mm Klinkenstecker) EAR (Kopfhörer) und MIC (Mikrofon) hat.

Dabei reicht ein einfaches Monogerät ohne automatische Aussteuerung. Eine Rauschunterdrückungsschaltung u.ä. ist auf jeden Fall störend.

Ein Bandzählwerk ist beim Suchen von Programmen auf Kassette eine große Hilfe.

Als Kassetten sind gute (= Drop-out freie) Eisenoxid-Bänder am besten geeignet. Chrom- oder Reineisenbänder bringen keinen Vorteil.

CSAVE, CLOAD und CLOAD? sind die drei Befehle, mit denen Sie ein BASIC Programm auf Kassette schreiben, eine Aufnahme verifizieren und von der Kassette laden können.

CSAVE

Dieser Befehl schreibt ein Programm auf die Kassette.
Benutzen Sie ihn auf folgende Weise:

- Spulen Sie das Band an die Stelle, an der die Aufzeichnung beginnen soll.

- Geben Sie

  >CSAVE"N"

  ein, wobei N für irgendeinen Buchstaben steht, mit dem Sie der Aufzeichnung einen Namen geben.

- Drücken Sie gleichzeitig die 'PLAY'- und 'RECORD'-Tasten Ihres Recorders.

- Drücken Sie die <RETURN>-Taste. Das Programm wird nun aufgezeichnet.

Von wichtigen Programmen sollten Sie aus Sicherheitsgründen mehr als eine Aufnahme machen. (Sicherheitskopie !!)

CLOAD?

Dieser Befehl vergleicht ein Programm, das sich im Speicher befindet, mit einer Bandaufzeichnung. Man benutzt diesen Befehl dazu, ein mit 'CSAVE' aufgezeichnetes Band zu testen. Gehen Sie dabei folgendermaßen vor:

- Spulen Sie das Band zum Anfang der Aufnahme.
- Geben Sie 'CLOAD?' ein und drücken Sie die <RETURN>-Taste.
- Drücken Sie die 'PLAY'-Taste Ihres Recorders.

Wenn die Aufzeichnung gefunden ist, erscheinen in der rechten oberen Bildschirmecke zwei Sternchen, von denen das rechte zur Kontrolle des Ladevorgangs blinkt. Erfolgreiche Verifizierung endet mit der 'READY'-Meldung. Sollte ein Fehler auftreten, meldet sich der Computer mit 'BAD'. In diesem Fall probieren Sie eine andere Lautstärkeneinstellung und verifizieren die Aufnahme erneut. Sollte dies nicht helfen, nehmen Sie eine andere Kassette.

CLOAD

Mit diesem Befehl wird ein BASIC Programm von der Kassette in den Speicher geladen, wobei ein vorher eventuell im Speicher stehendes Programm gelöscht wird. Gehen Sie folgendermaßen vor:

- Spulen Sie das Band an den Anfang der Aufzeichnung.

- Geben Sie CLOAD"N" ein, wobei N für den Kennzeichnungsbuchstaben steht. Wenn Sie nur 'CLOAD' eingeben, beachtet der Computer die Kennzeichnung nicht und lädt das BASIC-Programm, welches er als nächstes auf dem Band vorfindet. Wenn das richtige Programm gefunden ist (mit Hilfe des Kennbuchstabens), erscheinen zwei Sternchen in der rechten oberen Bildschirmecke, wobei das rechte zur Kontrolle blinkt. Sollte dieses Sternchen nicht mehr blinken oder sich der Computer nach einer Zeit von ca. 2 Minuten nicht mit 'READY' melden, so liegt ein Ladefehler vor. Versuchen Sie erneut das Programm zu laden (andere Lautstärke, Höhenregelung usw.).

3.22 Verarbeitung von Feldern

Ein Feld (array) ist einfach eine geordnete Liste von Daten. Das 'VIDEO-GENIE'-System verarbeitet numerische und auch Zeichenketten-Felder. Beide Datentypen können aber nicht im gleichen Feld gemischt auftreten. Die Konzepte der Programmierung von Feldern für die Computeranwendung und deshalb sollte der Benutzer versuchen, die Beispiele dieses Kapitels zu verstehen.

Nehmen wir an, Fritz Walter studiert an der Universität. Das Gebäude hat drei Stockwerke, von denen jedes vier Klassenräume beherbergt. Jeder Raum hat 45 Plätze. Fritz hat eine Vorlesung in Geschichte belegt und es sind nur 36 Studenten mit ihm in der Klasse. Sehen wir uns die Namensliste von Fritzs Klasse an:

NAMENLISTE:

1. Maria Adam
2. Jupp Braun
3. Heinrich Cox
:
:
:
36.Fritz Walter

Um eine Person in der Liste zu finden, müssen wir die Liste lediglich von oben nach unten durchlesen. Die Methode, mit der der Name gesucht wird, ist nicht sonderlich wichtig.

Wichtig ist, wie man eine Person in der Liste finden kann, wenn man nur ihre Nummer kennt. In der obigen Liste ist Maria Adam die erste, Jupp Braun die zweite Person usw. Die Zahlen geben also einen systematischen Weg an, um eine Person zu finden.

Benutzen wir den Computer, um die Namensliste aufzuzeichnen, so können wir jedem Namen eine einheitliche Variable zuweisen:

```
>10 N0$ = "MARIA ADAM"
>20 N1$ = "JUPP BRAUN"
>30 N2$ = "HEINRICH COX"
...
>100 NZ$ = "THOMAS HAGEN"
```

Bedenkt man, daß 36 Studenten in der Klasse sind, so sieht man, daß dies eine uneffektive und zeitraubende Methode ist. Sehen wir das Programm genauer an, ao stellen wir fest, daß alle Variablen laufende Nummern angehängt bekommen haben. Das haben wir einfach intuitiv und von Hand gemacht. Erzeugt der Computer nun diese laufenden Nummern selbst, so ist das ein Feld (Array). Wir definieren das Feld AR$ mit 45 Elementen (45 Plätze) und ordnen jedem Element einen Namen zu.

Beispiel:

```
>5 CLEAR 1000
>10 REM SETZE 1200 BYTES FÜR STRING FREI
>20 DIM AR$ (44)
>30 REM FELD AR$ HAT 45 ELEMENTE BEKOMMEN
>40 FOR N = 0 TO 44
>50 REM 45 SCHLEIFENDURCHGÄNGE
>60 INPUT "NAMEN DES STUDENTEN ", AR$ 7(N)
>70 REM ORDNE DEN FELDELMENTEN DIE NAMEN ZU
>80 NEXT N
>90 END
```

Diese Programm liest 45 Namen ein und speichert sie im Feld AR$. Nach der Programmausführung haben wir folgende Variablenbelegung:

```
Element AR$ (0) hat den Wert 'MARIA ADAM'
Element AR$ (1) hat den Wert 'JUPP BRAUN'
Element AR$ (2) hat den Wert 'HEINRICH COX'
...
Element AR$ (35) hat den Wert 'FRITZ WALTER'
```

Wollen wir die Liste nun ausdrucken, können wir folgendes Programm an das eben geschriebene Programm anhängen:

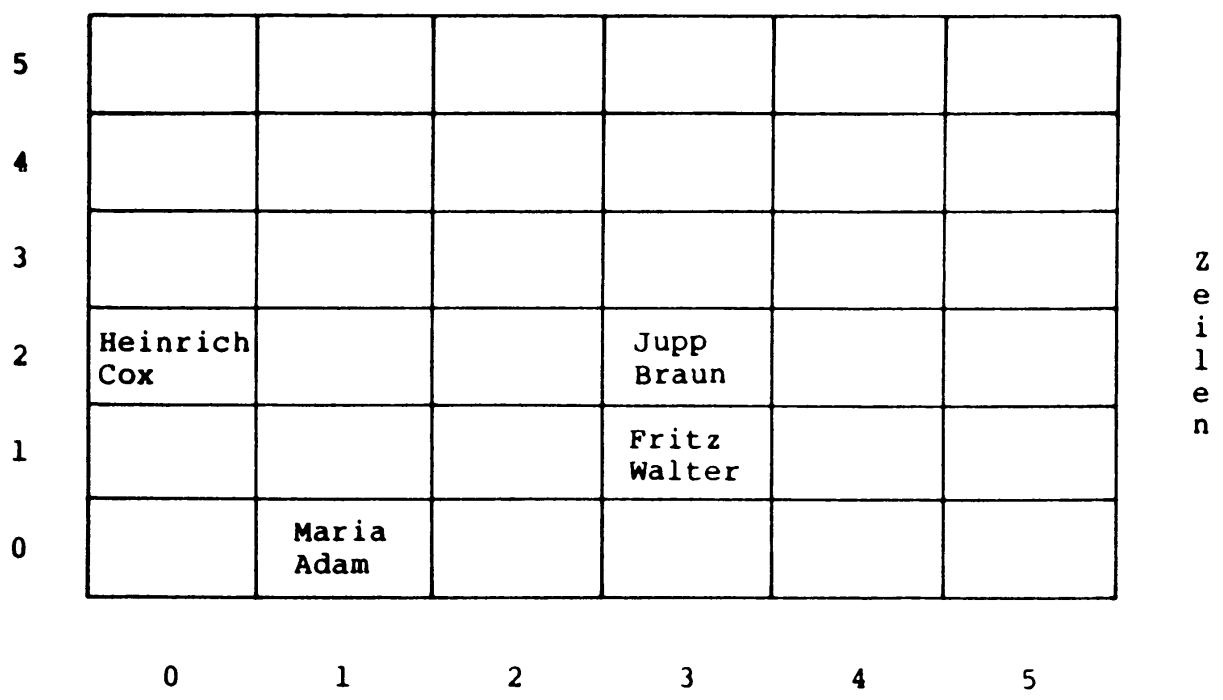
```
>90 REM AUSDRUCKEN DER FELDERELEMENTE
>100 FOR N = 0 TO 44
>110 REM WIEDER 45 MAL DIE SCHLEIFE
>120 PRINT AR$ (N)
>130 REM DRUCKT DIE FELDERELEMENTE (NAMEN)
>140 NEXT N
>150 END
```

Dieser Ausgabeteil ersetzt folgende 45 'PRINT'-Statements in der Art

```
>10 PRINT N0$
>20 PRINT N1$
>...
>70 PRINT NS$
>...
>100 PRINT NZ$
```

Nun sollten Sie ein Gefühl für die Nützlichkeit der Felder bekommen haben.

Angenommen, ein Lehrer dieser Klasse möchte nun einen Sitzplan aufstellen. Es sind 6 Zeilen und 6 Spalten von Sitzplätzen vorhanden:



Die Sitzpositionen unserer vier Beispiel-Studenten sind im Plan eingezeichnet. Da sie sich aber nicht der Namensliste folgend hingesetzt haben, müssen wir uns eine andere Methode ausdenken, um auf den Sitzplan zuzugreifen. Will der Lehrer z.B. feststellen, ob Fritz Walter anwesend ist, so muß er nachsehen, ob der Stuhl in Zeile 1, Spalte 3 leer ist oder nicht. Er kann auch in Zeile 2, Spalte 0 nach Heinrich Cox suchen. Der Computer tut genau das gleiche, wie dieser Lehrer. Wir können diese Sitzplan in ein zweidimensionales Feld SP$ (5,5) abbilden. Die erste 5 steht für die Zeilen, die zweite für die Spalten. Wollen wir nun Jupp Braun aufrufen, so müssen wir in SP$ (2,3) nachschauen.

Nun wollen wir unseren Sitzplan ausgeben:

```
>10 CLEAR 1000
>20 FOR R = 5 TO 0 STEP -1
>30 FOR C = 0 TO 5
>40 PRINTSP$ (R,C);
>50 NEXT C
>60 PRINT
>70 NEXT R
>80 END
```

Das Programm gibt den Sitzplan in Tabellenform aus, beginnend bei der letzten Sitzreihe der Klasse und endend mit der ersten. Es initialisiert zuerst R = 5 und C = 0 und und gibt dann die Werte der Elemente aus: SP$(5,0); SP$(5,1); SP$(5,2),;... SP$(5,5)

An diesem Punkt wird der Wert von C = 5. Der Computer springt von der inneren 'C'-Schleife zurück in die äußere 'R'-Schleife. Subtrahiert 1 von R und R wird damit zu 4. C wird wieder zu 0 zurückgesetzt und der Vorgang beginnt mit R = 4 neu:

```
SP$ (5,0); SP$ (5,1); SP$ (5,2); SP$ (5,3); ... SP$ (5,5)
SP$ (4,0); SP$ (4,1); SP$ (4,2); SP$ (4,3); ... SP$ (4,5)
SP$ (3,0); SP$ (3,1); SP$ (3,2); SP$ (3,3); ... SP$ (3,5)
  .
  .
  .
SP$ (0,0); SP$ (0,1); SP$ (0,2); SP$ (0,3); ... SP$ (0,5)
```

Mit einem solchen, zweidimensionalen Feld können wir jeden Studenten in der Klasse lokalisieren. Aber wie könnte man einen Studenten lokalisieren, der auf dem gleichen Platz wie Fritz Walter sitzt, aber in der nächsten Klasse, sitzt? Natürlich müssen wir dann neben Reihe und Spalte der Sitze noch die Nummer der Klasse angeben. Damit bekommt unser Feld noch eine weitere Dimension. Es gibt nämlich 12 Klassenräume im Schulgebäude. Es stehen verschiedene Wege zur Verfügung, um dieses Problem zu lösen. Der zweite ist, die Räume nach Stockwerken anzuordnen. 1. Raum im 1. Stock, 1. Raum im 2. Stock usw. Hier brauchen wir dann zwei weitere Dimensionen (insgesamt: Reihe, Spalte, Raum, Stock).

Angenommen, Fritz's Klassenraum wäre der 3. Raum im 2. Stock. Nach der ersten Methode könnten wir Fritz's Platz mit SP$ (N,R,C) angeben. Dabei ist: N die Zahl der Räume, R die Reihe und C die Spalte. Jupp sitzt dann z.B. im SP$ (7,1,3) d.h. in Raum 7, Reihe 1, Spalte 3.

Benutzen wir die zweite Methode, bekommen wir SP$ (F,N,R,C). Dabei ist: F die Nummer des Stockwerks, N die Nummer des Raumes in diesem Stock, R die Reihe und C die Spalte. Um Fritz nun anzusprechen, müßten wir sagen, SP$ (2,3,1,3) d.h.: 2. Stock, 3. Raum, 1. Reihe, 3.Salte.

Die Anzahl der Dimensionen steigt an, wenn wir noch mehr Studenten in unserer Systematik aufnehmen wollen. Wir können noch weitere Dimensionen einführen für die Gebäude, die Universitäten, Städte, Länder usw.

In jedem GENIE-System wird die Anzahl der möglichen Dimensionem nur durch den vorhandenen Speicherplatz begrenzt.

3.23 Behandlung von Zeichenketten

Zeichenketten haben eine große Bedeutung in der Datenverarbeitung. Ein Computer, der keine Zeichenketten verarbeiten kann, ist nichts weiter als ein sehr leistungsfähiger Taschenrechner. Dieser Erkenntnis folgend, stehen ihnen im 'GENIE'-System außer den arithmetischen Funktionen, viele leistungsfähige Zeichenkettenmanipulations-Funktionen zur Verfügung.

In diesem Kapitel werden wir diese Zeichenkettenmanipulations-Statements, die in unserer erweiteten 'BASIC'-Programiersprache benutzbar sind erläutert. Wir werden von nun an eine Zeichenkette mit ihrem englischen Namen STRING bezeichnen.

Vergleichen von Strings

Mit den relationalen Operatoren (=,<,#,usw.) können Strings auf ihre Gleichheit überprüft und nach ihrer alphabetischen Größe verglichen werden. Wird auf Gleichheit geprüft, so müssen alle Zeichen, führende und abschliessende Leerzeichen (Blanks) eingeschlossen, übereinstimmen, sonst wird auf ungleich entschieden.

Beispiel:

```
>10 IF A$ = "JA" THEN 250
```

Strings werden Zeichen für Zeichen, von links nach rechts, verglichen.
Um genau zu sein: Die ASCII-Darstellung der Zeichen wird verglichen. Ein Zeichen mit einer höheren Codenummer ist größer als eins mit einer niedrigeren. Mit anderen Worten "AC" größer "BC".
Für Buchstaben gilt die alphabetische Reihenfolge (A = Min., Z = Max.).
Werden Strings verschiedener Länge verglichen, so wird der kürzere String als kleiner angenommen, auch wenn seine Zeichen identisch mit denen des längeren sind. Daher ist "B" kleiner als "B ".
Die folgenden relationalen Operatoren können zum Vergleich von Strings eingesetzt werden:

```
> , < , <= , => , = , <>
```

Es gibt nur eine String Operation. Das ist das Aneinanderhängen von Strings. Der Operator ist das '+'-Zeichen.

Beispiel:

```
>10 S1$ = "DIE SONNE"
>20 S2$ = "SCHEINT"
>30 S3$ = ", "
>40 C$=S1$+S2$+S3$+S2$+S3$+S2$+"."
>50 PRINT C$
>60 END
```

Der Programmstart zeigt die Aneinander-
reihung der Strings:

```
>RUN
DIE SONNE SCHEINT, SCHEINT, SCHEINT.
READY
>
```

ASC (String)

Diese Statement berechnet den ASCII-Code des ersten Zeichens des angegebenes Strings. Der String muß in Anführungszeichen eingeschlossen sein. Ein leerer String als Argument erzeugt eine Fehlermeldung.

Beispiel:

```
>100 PRINT"ASCII-CODE VON 'H':";,ASC ("H")
>110 S$="HEIM"
>120 PRINT "DER STRING HEISST:"; S$
>130 PRINT"ASCII-CODE DES 1. BUCHSTABEN:";
>140 PRINT ASC (S$)
>150 END
>RUN
ASCII-CODE VON 'H': 72
DER STRING HEISST: HEIM
ASCII-CODE DES 1.BUCHSTABEN: 72
READY
>
```

Beide Zeilen drucken die gleiche Zahl.
Eine ASCII-Code-Liste finden Sie im Anhang.

CHR$ (Ausdruck)

Dieses Statement macht das genaue Gegenteil der 'ASC'-Funktion. Es erzeugt das zum Wert von Ausdruck gehörige ASCII-Zeichen. Als Argument darf eine Zahl von 0 bis 255, oder ein Ausdruck mit diesem Wert, auftreten. Das Argument muß in Klammern gesetzt werden.

Beispiel:

```
>10 PRINT CHR$ (72)
>RUN
H
READY
>
```

LEFT$ (String,n)

Das Statement erzeugt die ersten n Zeichen des angegebenen Strings. String und n müssen in Klammern stehen. String kann eine Zeichenkettenvariable oder eine Konstante sein, n kann ein numerischer Ausdruck oder eine Konstante sein.

Beispiel:

```
>10 A$ = "ABCDEFG"
>20 B$ = LEFT$ (A$,4)
>30 PRINT B$
>50 END
>RUN
ABCD
READY
>
```

RIGHT$ (String,n)

Erzeugt die n letzten Zeichen von String. Die Argumente müssen in Klammern stehen. String kann eine Zeichenkettenvariable oder eine Konstante sein, n kann eine numerische Variable oder eine Konstante sein. Ist die Länge von String kleiner oder gleich n, wird der gesamte String erzeugt.

Beispiel:

```
>10 A$ = "ABCDEFG" : B$ = RIGHT (A$,3)
>20 PRINT B$ : END
>RUN
EFG
READY
>
```

LEN (String)

Dieses Statement dient zur Abfrage der Länge von Strings, wobei es sich bei dem String um eine Variable, eine Konstante oder einen Zeichenkettenausdruck handeln kann.

Beispiel:

```
>10 A$ = "ABCDEFG"
>20 PRINT"LÄNGE DER ZEICHENKETTE :";LEN (A$)
>30 END
>RUN
LÄNGE DER ZEICHENKETTE IST : 7
READY
>
```

MID$ (String, p, n)

Dieses Statement wird dazu benutzt um einen Teilstring von einem String zu erzeugen. Der Teilstring beginnt an der Position p und ist n Zeichen lang. Die Argumente müssen in Klammern stehen. Der angegebene String kann eine Konstante, eine Variable oder ein Ausdruck sein.

Beispiel:

```
>10 A$ = "ABCDEFG"
>20 B$ = MID$ (A$, 3, 4)
>30 PRINT "DER NEUE STRING HEISST :"; B$
>40 END
>RUN
DER NEUE STRING HEISST : CDEF
READY
>
```

STR$ (Ausdruck)

Verwandelt eine numerische Konstante oder einen Ausdruck in einen String. Das Angebot muß in Klammern stehen.

Beispiel:

```
>10 A = 34.56
>20 B$ = STR$ (A)
>30 PRINT "DAS ERGEBNIS IST :"; B$
>40 END
>RUN
DAS ERGEBNIS IST : 34.56
```

STRING$ (n, Zeichen oder Zahl)

Erzeugt einen String, der aus n mal dem Zeichen besteht.

Beispiel:

```
>10 PRINT STRING$ (10, "* ")
>20 END
>RUN
* * * * * * * * * *
READY
>
```

An Stelle von Zeichen kann man auch eine Zahl von 0 bis 255 angeben. Sie wird als ASCII-Code behandelt und das entsprechende Zeichen oder der graphische Code werden erzeugt.

Beispiel:

```
>10 PRINT STRING$ (10, 72)
>20 END
>RUN
HHHHHHHHHH
READY
>
```

VAL (String)

Macht das Gegenteil der 'STR$'-Funktion.
Erzeugt numerischen Wert aus einem String.

Beispiel:

```
>10 A$ = "56"
>20 B$ = "23"
>30 C = VAL (A$ + "." + B$)
>40 PRINT"DAS ERGEBNIS IST :";C;",";C+100
>50 END
>RUN
```

3.24 Arithmetische Funktionen

In diesem Kapitel werden wir die fest programmierten, arithmetischen Funktionen des GENIE-Systems besprechen. In den meisten Fällen muß ein Argument an die Funktion weitergegeben werden, bevor der Funktionswert berechnet werden kann. Dieses Argument kann eine numerische Konstante, eine numerische Variable oder ein numerischer Ausdruck sein.

Das allgemeine Format ist:

Ergebnis = Funktion (Argument)

Beispiel:

```
>10 A = RND (3)
>20 B = INT (C)
>30 C = SQR (F * G - 4)
```

Wir behandeln folgende Funktionen:

1. ABS (X)
2. ATN (X)
3. CDBL (X)
4. CINT (X)
5. COS (X)
6. CSNG (X)
7. EXP (X)
8. FIX (X)
9. INT (X)
10. LOG (X)
11. RANDOM
12. RND (X)
13. SNG (X)
14. SIN (X)
15. SQR (X)
16. TAN (X)

ABS (X)

Berechnet den Absolutwert von X.

ATN (X)

Berechnet den Arctustangens von X (im Bogenmaß). Um das Ergebnis im Grad zu erhalten, multiplizieren Sie das Ergebnis von ATN (X) mit 57.29578.

CDBL (X)

Erzeugt die doppelt genaue Darstellung von X. Das Ergebnis enthält 17 Stellen, wovon die Stellen, die das Argument enthalten signifikant sind.

CINT (X)

Berechnet die nächste ganze Zahl, die kleiner als das Argument ist. Das Argument muß zwischen -32768 und +32767 liegen.

Beispiel:

CINT (2.6) = 2 CINT (-2.6) = -3

COS (X)

Berechnet den Cosinus von X (im Bogenmaß).
Soll er in Grad berechnet werden :
COS (X * .0174533)

CSNG (X)

Erzeugt die einfach genaue Darstellung von X. Berechnet eine 6stellige Zahl mit 4/5 Rundung bei doppelt genauem X.

EXP (X)

Berechnet die Exponential-Funktion von X, d.h. e hoch X. Das ist die Umkehrfunktion zu LOG (X).

FIX (X)

Trennt die Nachkommastellen von X ab.

Beispiel:

FIX (1.5) = 1 FIX (-1.5) = -1

INT (X)

Erzeugt die ganzzahlige Darstellung von X. Berechnet die größte, ganze Zahl, die nicht größer als X ist.

Beispiel:

INT (3.5) = 3 INT (-3.5) = -4

LOG (X)

Berechnet den natürlichen Logarithmus von X, d.h. LOGe(X). Um einen Logarithmus zu einer anderen Basis zu berechnen, benutzen Sie die folgende Formel:

LOGb(X) = LOGe(X) / LOGe(b)

(b ist die Basis)

RANDOM

Führt ein Computer diese Funktion aus, so erzeugt er einen neuen Vorrat an Zufallszahlen. Diese Funktion benötigt keine Argumente.

RND (X)

Erzeugt eine Pseudozufallszahl aus der aktuellen Menge der Zufallszahlen. (Werden intern generiert, daher ist kein Zugriff vom Benutzer möglich).

RND (0)
erzeugt eine einfach genaue Zahl zwischen 0 und 1.

RND (X)
erzeugt eine eine ganze Zahl zwischen 1 und X. X muß positiv und kleiner als 32768 sein.

SGN (X)

Die Vorzeichenfunktion. Sie erzeugt eine -1, wenn X negativ ist, 0 wenn wenn X Null ist und +1 wenn X positiv ist.

SIN (X)

Berechnet den Sinus des Arguments (im Bogenmaß). Soll er in Grad berechnet werden: SIN (X * .0174533)

SQR (X)

Berechnet die Quadratwurzel von X.

TAN (X)

Berechnet die Tangensfunktion von X (im Bogenmaß). Soll er in Grad berechnet werden:
TAN (X * .174533)

3.25 Graphik Befehle

Zunächst eine Bemerkung für GENIE III/IIIs:

Mit diesen Computern können Sie nicht nur eine Bildschirmeinteilung von 16 Zeilen mit je 64 Zeichen, sondern auch eine Bildschirmeinteilung von 24 Zeilen mit je 80 Zeichen vornehmen. Da die Umschaltkommandos nicht im BASIC sondern im DOS vorhanden sind, können Sie die entsprechenden Befehle im G-DOS Handbuch finden.

Eine Bemerkung für GENIE I/II/IIs und Speedmaster:

Eine Umschaltung auf 24 Zeilen mit je 80 Zeichen ist bei diesen Geräten nicht möglich.

GENIE-Graphik:

Die Graphik mit dem 'PRINT@-Befehl' haben Sie bereits kennengelernt. Viel interessanter für die Graphikdarstellung ist jedoch die PUNKT-GRAPHIK, die Ihr Genie Ihnen bietet. Hierbei wird der Bildschirm in 128 Spalten und 48 Zeilen (160 Spalten und 71 Zeilen) eingeteilt. Numeriert werden sie von 0 bis 127 und von 0 bis 47 (von 0 bis 159 und von 0 bis 70). Man kann die Einteilung mit einem Koordinatensystem vergleichen, bei dem der Nullpunkt mit den Koordinaten X=0 und Y=0 in der linken oberen Ecke liegt. Im Anhang finden Sie eine Bildschirmeinteilung (48 Zeilen mit je 128 Spalten pro Zeile).

Nun wollen wir aber die vier Graphik-Befehle besprechen. Alle Befehle können auch in der aktiven Kommandoebene benutzt werden.

CLS

Mit diesem Befehl kann der Bildschirm gelöscht werden.

SET (X,Y)

Dieser Befehl setzt einen Punkt an den Koordinaten X, Y. Die Werte für X und Y müssen innerhalb der Bildschirmeinteilung liegen.

Beispiel:

```
>10 SET (37,20)
```

Setzt einen Punkt mit den Koordinaten 37, 20.

RESET (X,Y)

Dieser Befehl löscht den Punkt mit den Koordinaten X und Y. Wenn an dieser Stelle kein Punkt gesetzt war, so geschieht nichts. Die Werte für X und Y müssen innerhalb der Bildschirmeinteilung liegen.

Beispiel:

```
>10 RESET (37,20)
```

Löscht den Punkt mit den Koordinaten 37, 20.

Wenn Sie mit diesen beiden Befehlen z.B. eine Linie zeichnen wollen, so müssen Sie die Befehle in Schleifen benutzen. Sie müssen dabei immer darauf achten, daß sich eine der beiden Koordinaten ändert. Es darf auch ein Rechenausdruck in der Klammer stehen.

Beispiel:

```
>10 CLS
>20 FOR A = 1 TO 128
>30 SET (A,0) : NEXT A
>40 FOR B = 1 TO 128 STEP 2
>50 RESET (B,0) : NEXT B
```

Dieses Programm zeichnet eine gepunktete waagerechte Linie in der oberen Linie auf den Bildschirm.

POINT (X,Y)

Mit diesem Befehl kann man überprüfen, ob an dem Punkt mit den Koordinaten X, Y ein Punkt gesetzt ist. Wenn ein Punkt gesetzt ist, so wird ein binäres "wahr" (-1) ausgegeben, sonst eine 0.

```
>10 A = POINT (37,20)
```

Wenn Sie ein GENIE IIs oder GENIE IIIs haben, gibt es für Sie noch eine Menge anderer Graphik Befehle. Sie finden diese im Anhang.

3.26 Zusätzliche BASIC-Befehle

INP (Portnummer)

Liest einen 8-Bit Wert aus dem angegebenen Port. Der GENIE kann 256 Ports adressieren. Sie werden von 0 bis 255 numeriert. Einige Ports sind schon intern verwendet.

```
>10 A = INP (124)
```

Liest einen 8-Bit Wert aus Nr. 124 nach A.

OUT Portnummer, Wert

Gibt einen 8-Bit Wert aus dem angegebenen Port aus. Dieses Statement verlangt zwei Argumente: die Portnummer und den Wert, der an diesem Port ausgegeben werden soll.
Der GENIE kann 256 Ports ansprechen, sie sind von 0 bis 255 adressiert.

Beispiel:

```
>30 OUT 14,20
```

Gibt den Wert 20 an Port 14 aus. Beide Argumente müssen zwischen 0 und 255 liegen.

PEEK (Adresse)

Diese Funktion holt einen 8-Bit Wert aus der Speicherzelle, die mit Adresse dezimal angegeben wurde. Der Wert von 'PEEK' ist ebenso dezimal und liegt zwischen 0 und 255.

Beispiel:

```
>20 B = PEEK (3000)
```

Schreibt den Inhalt von Speicherzelle 3000 nach B.

POKE Adresse, Wert

Dieses Statement schreibt einen 8-Bit Wert in die durch Adresse dezimal angegebene Speicherzelle. Es braucht zwei Argumente: Adresse und Wert. Der Wert muß zwischen 0 und 255 liegen.

Beispiel:

```
>10 A = 250
>20 REM SCHREIBE A IN ZELLE 19000
>30 POKE 19000, A
>40 REM HOLE DEN WERT VON ZELLE 19000
>50 B = PEEK (19000)
>40 DAS ERGEBNIS IST :"; B
>50 END
```

MEM

Gibt die Anzahl der nicht benutzten Bytes im Speicher an.
Hier sollte man noch darauf hinweisen, daß, auch wenn der Computer 64, 128 oder 256 KByte Speicherplatz besitzt nur ungefähr 30 KByte zur freien Verfügung im BASIC vorhanden sind. Dies hängt mit dem Mikroprozessor des Computers zusammen.
Wenn Sie die genaue Anzahl der Bytes im Speicher erfahren möchten, dann können Sie in der aktiven Kommandoebene 'PRINT MEM' eingeben. Sie können dieses Befehl aber auch in Programmen verwenden.

Beispiel:

```
>10 IF MEM < 200 THEN GOTO 400
```

INKEY$

Liest ein Zeichen von der Tastatur. Wurde zum Zeitpunkt der Ausführung der 'INKEY'-Funktion keine Taste betätigt, so wird ein Leerstring (" ") erzeugt.
Die von der 'INKEY'-Funktion eingelesenen Daten erscheinen nicht auf dem Bildschirm.

Beispiel:

```
>10 CLS
>20 PRINT "GIB PASSWORT EIN"
>30 A$ = INKEY$:IF A$ = "O" THEN 40 ELSE 30
>40 B$ = INKEY$:IF B$ = "K" THEN 50 ELSE 40
>50 PRINT "WILLKOMMEN"
...
```

Dies ist ein möglicher Programmanfang, durch den nur Personen, die das Passwort kennen, in das Hauptprogramm gelangen können.

POS (Dummyargument)

Es wird eine Zahl von 0 bis 63 (79), die die momentane Position des Cursors angibt, erzeugt. 'Dummyargument' kann irgendeine Zahl sein, meist wird die 0 benutzt.

Beispiel:

```
>100 A = POS (0)
```

USR (Argument)

Ruft ein Unterprogramm in Maschinensprache auf. Das Maschinenprogramm kann mit 'POKE' erzeugt oder von Band eingelesen werden. Wenn der Benutzer in der Programmierung von Maschinensprache noch nicht sicher ist, ist vom Einsatz dieser Funktion abzuraten.

Die Startadresse des Unterprogramms wird mit 'POKE' in die Adressen 16526 und 16527 mit dem am wenigsten signifikanten Byte in 16526 geschrieben.

Um ein Argument in das Unterprogramm zu übergeben, sollte das Unterprogramm an seinem Anfang ein 'CALL 0A7FH (2687 dezimal), ausführen. Das Argument wird dann in ein HL-Register geschrieben.

Um in das BASIC zurück zu kommen, ohne einen Wert zurückzugeben, wird das Unterprogramm mit 'RET' abgeschlossen. Soll ein Wert zurückgegeben werden, so wird er in das HL-Registerpaar geladen und am Ende des Unterprogramms ein JP 0A9AH (2714 in dezimal) ausgeführt. Die Werte werden als 2 Byte, ganze Varzeichenzahl behandelt. Die 'USR'-Funktion reserviert 8 Stack Ebenen für das Unterprogramm. Das Unterprogramm sollte in obersten Speicherbereich geschrieben werden.

Um es davor zu schützen, das es vom BASIC als Programmspeicher betrachtet und damit sein Inhalt zerstört wird, sollte der Benutzer den Speicherplatz schützen.

VARPTR (Variablenname)

Die Adresse, an der der Wert der Variablen im Speicher steht, wird berechnet.

```
>10 K = VARPTR (A)
```

In den folgenden Erklärungen heißt:

LSB das am wenigsten signifikante Byte.
MSB das am höchsten signifikante Byte.

Für die verschiedenen Variablentypen bedeutet K dann folgendes:

a) 2-Byte, ganzzahlige Variablen (A%)

```
K     = LSB
K + 1 = MSB
```

b) Variablen einfacher Genauigkeit (A)

```
K     = LSB
K + 1 = Nächstes MSB
K + 2 = MSB
K + 3 = Exponent
```

c) Variablen doppelter Genauigkeit (A#)

K = LSB
K + 1 = nächstes MSB
.
.
.
K + 6 = MSB
K + 7 = Exponent

d) Zeichenkettenvariablen (STRINGS) (A$)

K = Länge des Strings
K + 1 = LSB der Anfangsadresse
K + 2 = MSB der Anfangsadresse

3.27 Bit, Byte, Hexadezimal

Diesen 3 Begriffen werden Sie im folgenden noch öfter begegnen. Daher will ich sie Ihnen kurz erklären.
Das "Herz" Ihres Computers ist eine Z80-CPU.
Ein Mikroprozessor selbst versteht kein BASIC, sondern die sogenannte Maschinensprache. Der eingebaute BASIC-Interpreter übersetzt ein BASIC-Programm in die für den Mikroprozessor verständliche Maschinensprache.

Die Maschinensprache funktioniert auf binärer Ebene. Die grundlegende Informationseinheit ist dabei ein "Bit". Ein Bit kann nur 2 Zustände annehmen: 0 und 1, an und aus, Transistor leitet oder schließt ...
Nun faßt man 8 dieser Bits zu einem "Byte" zusammen. Der Z80 ist ein 8-Bit Mikroprozessor, d.h. er kann jeweils 8 Bits = 1 Byte auf einmal verarbeiten. Da ein Byte 8 Bit enthält, sind 256 Zustände möglich (von 0...255 einschließlich).
Da die Darstellung eines Bytes in Bit-Schreibweise (z.B. 01101111 binär = 111 dezimal) recht umständlich ist, nimmt man je 4 Bit aus einem Byte und gibt diese dann durch ein Hexadezimaläquivalent aus:

Binär:	Hex.:	Dezimal:
0000	0	1
0001	1	2
0010	2	3
0011	3	4
0100	4	5
0101	5	6
0110	6	7
0111	7	8
1000	8	9
1001	9	10
1010	A	11
1011	B	12
1100	C	13
1101	D	14
1110	E	15
1111	F	16

Ein Byte, das, wie schon gesagt, aus 8 Bit besteht, wird dann durch 2 Hexadezimalziffern dargestellt.

Z.B.: 0111 1110 = 7E Hex. = 126 dez.

Der Z80-Mikroprozessor kann maximal 65536 Speicherzellen adressieren. Jede Speicherzelle entspricht einem Byte, das 256 verschiedene Zustände annehmen kann.

Die Adresse einer Speicherzelle wird durch 16 Bits angegeben. Hier werden dann 4 mal 4 Bits zu je einer Hexadezimalziffer zusammengefasst.

Ein Beispiel:

0110 1111 1100 1110 = 6FCE Hex. = 28622 Dez.

Die Adressen reichen also von 0000H bis FFFFH.

Als nächstes finden Sie ein kleines BASIC-Programm, welches es Ihnen ermöglicht dezimale Zahlenwerte in hexadezimaler Schreibweise ausgeben zu lassen.

```
>10 CLS
>20 DIM A$ (15)
>30 FOR A = 0 TO 15
>40 READ A$ (A)
>50 NEXT A
>60 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
>70 INPUT "Dezimalzahl";Z
>80 A$ =""
>90 A=Z
>100 A1=FIX(A/16)
>110 A2=A-A1*16
>120 A$ =A$ (A2)+A$
>130 A=A1
>140 IF A>0 THEN 100
>150 IF LEN(A$) < 4 THEN A$="0"+A$ : GOTO150
>160 PRINT Z;" dez. ----> ";A$ ;" hex."
>170 GOTO 70
```

ANHANG A

Reservierte Worte:

ABS	DEFINT	FRE	LINE	POS	SQR
AND	DEFSNG	GET	LIST	PRINT	STEP
ASC	DEFUSR	GOSUB	LOAD	PUT	STOP
ATN	DEFSTR	GOTO	MEM	RANDOM	STRING$
CDBL	DELETE	IF	MID$	READ	STR$
CHR$	DIM	INKEY$	NAME	REM	TAB
CINT	EDIT	INP	NEW	RESET	TAN
CLEAR	ELSE	INPUT	NEXT	RESTORE	THEN
CLOSE	END	INSTR	NOT	RESUME	TROFF
CLS	ERL	INT	ON	RETURN	TRON
CONT	ERR	KILL	OR	RIGHT$	USING
COS	ERROR	LEFT$	OUT	RND	USR
DATA	EXP	LET	PEEK	SET	VAL
DEFDBL	FIX	LSET	POINT	SGN	VARPTR
DEFFN	FOR	LEN	POKE	SIN	

Control Codes:

Mit "PRINT CHR$ (X)" und Werten von X<32 können diverse Steuerfunktionen ausgeführt werden, wobei nicht alle Werte von 0...31 belegt sind, sondern nur die hier aufgelisteten:

8	Löscht letztes Zeichen	26	Cursor ↓
10	Neue Zeile	27	Cursor ↑
14	Cursor "Ein"	28	Cursor auf 0,0
15	Cursor "Aus"	29	Cursor zum Zeilenanfang
23	32 Zeichen / Zeile	30	bis Zeilenende löschen
24	Cursor ←	31	Bildschirmspeicher ab Cursor löschen
25	Cursor →	128	Leerzeichen
129 ... 191	Diverse Grafikzeichen		
192 ... 255	0 bis 63 Leerzeichen		

Mit folgendem kleinen Programm können Sie die Steuercodes ausprobieren:

```
>10 CLS
>20 INPUT "Steuercodeeingabe (8...31)";X
>30 PRINT"ABC";CHR$ (X);"DEF"
>40 GOTO 20
```

ASCII-Code-Tabelle:

hex	dez.	ASCII	hex	dez.	ASCII	hex	dez.	ASCII	hex	dez.	ASCII
00	0	NUL	20	32	Space	40	64	@	60	96	\`
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	—	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6C	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	'	7C	124	'
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	↑	7E	126	~
1F	31	US	3F	63	? ▶	5F	95	—	7F	127	DEL

BASIC-Befehlstabelle:

v = Variable	x = Rechenausdruck	z = Zeilennummer	k = Konstante

Kommandos	RUN z	Startet ein Programm
	LIST z - z	Gibt Programmliste aus
	NEW	Löscht Programmspeicher
	CONT	Fortsetzung Programmlauf
Anweisungen	PRINT x, x; x oder ?	Ausgabebefehl
	LET v = x oder v = x	Wertzuweisung an Variable
	INPUT V, V V	Aufforderung zur Dateneingabe
	READ v, v, v	Einlesen von Daten
	DATA k, k, k	Daten im Programm
	GOTO z	Unbedingter Sprung
	FOR v = x TO x STEP x	Schleifendefinition
	NEXT v	Schleifenende
	IF v x THEN ELSE	Bedingte Befehle
	IF v x GOTO z	Bedingter Sprung
	GOSUB z	Anruf Unterprogramm
	ON V GOTO z	Errechneter Einsprung
	RETURN	Rücksprung zum Hauptprogramm
	DIM v (k,k)	Speicherplatzreservierung
	STOP	Anhalten des Programmlaufs
	END	Ende des Programms
	REM oder ▼	Für Kommentarzeilen

Operatoren	+	Addition	=	Gleichheit
	-	Substraktion	<>	Ungleich
	*	Multiplikation	>	Größer als
	/	Division	>=	Größer oder gleich
	^	Potenzieren	<	Kleiner als
	(x)	Vorrang	<=	Kleiner oder gleich

Numerische Funktionen	ABS (X)	Absolutwert	INT (X)	Ganze Zahl
	SGN (X)	Vorzeichen	SQR (X)	Quadratwurzel
	SIN (X)	Sinus	LOG (X)	Logarithmus (Basis e)
	COS (X)	Cosinus	RND (X)	Zufallszahl
	TAN (X)	Tangens	EXP (X)	e^x
	ATN (X)	Arcustangens	TAB (X)	Tabulatorstellen

Zeichenketten Funktionen	LEN (X$)	Länge der Zeichenkette
	ASC (X$)	ASCII Code des 1. Zeichens, dezimal
	CHR$ (X)	Zeichen entsprechend ASCII dezimal
	VAL (X$)	Wandelt Ziffernkette in Zahl
	STR$ (X)	Wandelt Zahl in Ziffernkette
	LEFT$ (X$,n)	Trennt die n vorderen Zeichen ab
	RIGHT$ (X$,n)	Trennt die n hinteren Zeichen ab
	MID$ (X$,m,n)	Trennt vom mten Zeichen n Zeichen ab

. Dezimal Trennung	: Neuer Befehl	, Daten Trennung	; Dichtdruck	, Kolonnen
X$ Zeichenkette	X% Ganze Zahl	X! 6 Stellen	X# 16 Stellen	

Cassette	CLOAD# – 1, "A"	Lädt Programm "A" von Kassette 1
	CSAVE# – 1, "A"	Speichert Programm "A" auf Kassette 1
	CLOAD ?	Prüft nächstes Programm
	PRINT# – 1,d,d,d	Schreibt Daten auf Kassette 1
	INPUT# 1,v,v,v	Holt Daten von Kassette 1 in Variable
Programm	AUTO z[n]	Gibt Zeilennr. von z mit Abstand n
	TRON	Schaltet Trace-Betrieb ein
	TROFF	Schaltet Trace-Betrieb aus
	CLEAR	Setzt alle Variablen auf 00
	CLEAR n	Reserviert n Speicherplätze
	DELETE z - z	Löscht Befehle von Zeile bis Zeile
Variable definieren	DEFINT v - v	Bestimmt Variable von v - v als Integer
	DEFSNG v - v	Bestimmt Variable von v - v als einf. genau
	DEFDBL v - v	Bestimmt Variable von v - v als doppelt genau
	DEFSTR v - v	Bestimmt Variable von v - v als Zeichenketten
Bild-schirm	PRINT @ s,x,x	Zeigt Ausdrücke ab Schreibstelle S
	PRINT USING x$;x	Zeigt x entsprech. x$ formatiert
	PRINT TAB (x) x;	Zeigt X an Tabulatorstelle (x)
	PRINT MEM	Zeigt freien Speicherraum
Ein-Ausgabe	INKEY$	Übernimmt Zeichen von Tastatur
	INP (port-nr.)	Übernimmt ein Byte von Eingangs-Port
	OUT prt-nr., wert	Gibt Wert an Ausgangs-Port
Grafik	CLS	Löscht Bildschirm
	SET (x,y)	Setzt Punkt mit Koordinaten x und y
	RESET (x,y)	Löscht Punkt mit Koordinaten x und y
	POINT (x,y)	Prüft Punkt mit Koordinaten x und y
Verschie-denes	RANDOM	Erneuert Zufallsgenerator
	VARPTR (v)	Gibt Adresse wo v gespeichert ist
	RESTORE	Bei nächstem READ von Anfang DATA
	RESUME	Abschluß Fehlerbehandlung
	ON ERROR GOTO z	Bei Fehler Sprung nach z
	ERROR code	Zur Simulierung von Fehlern
	EDIT z	Anruf Editor
Maschinen-sprache	PEEK (adr)	Dezimalwert gespeichert in Adresse
	POKE adr,wert	Speichert Wert in Adresse
	USR (argument)	Anruf Unterprogramm in Maschinensprache
Druck	LLIST [z - z]	Druckt Programm aus dem Speicher
	LPRINT x,x;x;x	Allgemeiner Befehl für Drucken
Verknüp-fung	AND	UND-Verknüpfung
	OR	ODER-Verknüpfung

Fehlercode-Tabelle:

1	NF	NEXT ohne FOR
2	SN	Syntaxfehler
3	RG	RETURN ohne GOSUB
4	OD	Nicht genügend Daten in DATA
5	FC	Unzulässige Funktion aufgerufen
6	OV	Überfließen
7	OM	Speicherbereich zu klein
8	OL	Sprung in nicht definierte Zeile
9	BS	Index außerhalb des zulässigen Bereichs
10	DD	Feldvariable neu dimensionieren
11	/0	Division durch Null
12	ID	Unzulässiges Kommando
13	TM	Durcheinander bei Variablen-Typen
14	OS	Nicht ausreichend Speicher für Zeichenketten
15	LS	Zeichenkette zu lang
16	ST	Zeichenkettenformel zu komplex
17	CN	Kein Weiterlauf des Programms möglich
18	NR	RESUME fehlt
19	RW	RESUME ohne Error
20	UE	Fehler kann nicht angezeigt werden
21	MO	Operand fehlt
22	FD	Daten auf Datei nicht lesbar

Editor-Befehlstabelle:

EDIT Zeile	NewLine	Ruft Editor auf	Space und ←
X Space	und X ←	Bewegt Cursor	
Löschen	D	Löscht ein Zeichen	
	X D	Löscht x Zeichen	
Ändern	C	1 Zeichen zur Änderung frei	Alle Zeichen gültig
	X C	x Zeichen zur Änderung frei	
Ein-fügen	I	Einfügen am Cursor	
	X	Anhängen am Ende	
	H	Einfügen und Rest löschen	
Ende Einfüg.	SH ↑	Beendet Einfüge-Modus	← SP
Zeile	L	Bleibt im Editor	
Anzeigen	NewLine	Ende Editor, Änderung übernehmen	Alle

ANHANG B

Speicherbereiche und Grenzen der Programmierung:

Bereiche:

ganze Zahlen	-32768 bis +32767
einfache Genauigkeit	+/- 1.701411E+38
doppelte Genauigkeit	-1.70141183424455 6D+38 bis +1.70141183424455 6D+38
String	bis 255 Zeichen
Zeilennummern	0 bis 65529
Programmzeilenlänge	bis zu 240 Zeichen

Speicherplatzbedarf:

Eine Programmzeile braucht minimal	5 Bytes
Zeilennummer	2 Bytes
Zeilenpointer	2 Bytes
Zeilenvorschub (Carriage Return)	1 Byte

Dazu braucht jeder Befehl, Operator, Variablenname, Spezialzeichen und Konstantenziffer ein Byte

Dynamische Speicherplatzzuteilung:
(bei Programmausführung)

ganzzahlige Variablen	5 Bytes
Variablen einfacher Genauigkeit	7 Bytes
Variablen doppelter Genauigkeit	11 Bytes

String Variablen	6 Bytes minimal (3 Variablenname, 3 für Länge u. Pointer + 1 pro Zeichen)
Feldvariablen	12 Bytes minimal (3 Variablennamen, 2 für Größe, 1 pro Dimension + 1 pro Zeichen)

Jede aktive 'FOR-NEXT'-Schleife	16 Bytes
Jedes aktive 'GOSUB' (ohne 'RETURN')	6 Bytes
Jede Klammerebene	4 Bytes
(für Zwischenergebnisse	12 Bytes)

ANHANG C

Graphik Befehle (nur für GENIE IIIs):

Anleitung zum RDLBASIC
(Graphik-Betriebssystem)

Um die graphischen Fähigkeiten Ihres Computers auch nutzen zu können, wird zum GDOS-Betriebssystem des Rechners eine Graphikunterstützung mitgeliefert. Diese besteht aus drei Bestandteilen, nämlich:

- Dem eigentlichen Betriebssystem (RDLBASIC/CMD)

- Der Druckerunterstüzung (alle /DRV-Files und INST/BAS)

- Verschiedene Schriftarten (alle /RZS-Files)

RDLBASIC/CMD ersetzt das Disk-BASIC völlig, wenn Sie mit Graphikunterstützung arbeiten wollen. Statt BASIC <RETURN> geben Sie einfach RDLBASIC <RETURN> ein, um das System zu starten.

RDLBASIC bietet eine Reihe neuer BASIC-Befehle, die sowohl im aktiven Eingabemodus als auch von Programmen aus ausgeführt werden können.

Da der Computer seine gesamte Bildschirmausgabe über ein IC steuert, welches verschiedene Bildschirmformate ermöglicht (z.B. 80x24, 64x16, ...), ist ist auch die Graphikauflösung frei wählbar (bei 80x24 Zeichen z.B. 640x264 Punkte). RDLBASIC paßt sich an diese Auflösung selbstständig an, sodaß Sie immer vernünftige Graphikansteuerung haben.

Übersicht über die neuen BASIC-Befehle:

HON 0-2, 0-1	Bildschirmseiten einblenden
HOFF 0-1	Bildschirmseiten ausblenden
HCLS	Momentan aktive Bildschirmseite löschen
HNEG	Momentan aktive Bildschimseite invertieren
HFLEX 0-1	Momentan aktive Bildschirmseite horizotal und vertikal spiegeln
HCOLOR 0-2	Zeichenart für die nächsten Punkte setzen (setzen, löschen oder invertieren)
HPLOT x,y, 0-2	Punkt oder Linie zeichnen

HMOVE x,y, 0-2	Punkt setzen
HDRAW x,y, 0-2	Linie ziehen
HCARO xl,yl,x2,y2, 0-2	Rechteck ausfüllen
HPEN	Mit den Pfeiltasten zeichnen
HTEXT x,y, 0-2, text	Texte zeichnen
HBIT x,y, 0-2, b, h, text	Texte vergrößert zeichnen
HCODE filename	Schriftart von Diskette laden
HSAVE filename	Graphikbildschirm auf Diskette speichern
HLOAD	Graphikbildschirm von Diskette laden
HMERGE filename	Graphikbildschirm mit Diskettenfile mischen
HPRINT br, lr	Graphikbildschirm ausdrucken

Beschreibung der einzelnen Befehle:

(Anmerkung: Werden Parameter zu einem Befehl nicht angegeben, so werden sie als Null angenommen. Beispiel: HON entspricht HON 0,0.)

HON

Format: HON s,v

Mit diesem Befehl bestimmen Sie, welche der drei Bildschirme Ihres Computers auf dem Monitor sichtbar sind. Hierbei gibt s die Nummer des sichtbaren Schirms an:

s = 0 Graphikseite 0 ist sichtbar

s = 1 Graphikseite 1 ist sichtbar

s = 2 Textbildschirm ist sichtbar

und v die Nummer des Graphikbildschirms, der verändert werden kann (entfällt, bei s = 2):

v = 0 Graphikseite 0 kann verändert werden

v = 1 Graphikseite 1 kann verändert werden

HOFF

Format: HOFF s

Dieser Befehl blendet Bildschirmseiten aus. Der Parameter s gibt an, ob die Graphikseite oder die Textseite ausgeblendet werden soll:

s = 0 Graphikbildschirm ausblenden

s = 1 Textbildschirm ausblenden

HCLS

Format: HCLS

Dieser Befehl löscht den Graphikbildschirm, der momentan verändert werden kann (entspricht CLS auf dem Textbildschirm).
Wurde vorher ein HNEG ausgeführt, wird der Bildschirm vollständig weiß geschrieben.

HNEG

Format: HNEG

Der momente veränderbare Graphikbildschirm wird invertiert, d.h. ein nicht gesetzter Punkt wird gesetzt, ein schon gesetzter Punkt wird gelöscht.

Dieser Befehl ändert auch die logische Funktion der HCOLOR-Parameter 0 und 1, sodaß 'Setzen' nach einem HNEG einen Punkt löscht.

HFLEX

Format: HFLEX a

Der momentan veränderbare Graphikbildschirm wird um eine Mittelachse gespiegelt. Hierbei bewirkt der Parameter a:

a = 0 Spiegelung um vertikale Mittelachse

a = 1 Spiegelung um horizontale Mittelachse

HCOLOR

Format: HCOLOR f

Mit diesem Befehl wird der Zeichenmodus für Punkte und Linien, die mit dem HPLOT-Befehl erzeugt werden, eingestellt.

Hierbei bedeutet der Parameter f:

f = 0 Punkt löschen

f = 1 Punkt setzen

f = 2 Punkt invertieren

HPLOT

Format: HPLOT x,y,n

Die Koordinaten x und y (Bereich abhängig vom Bildschirmformat) geben einen Punkt auf dem momentan veränderbaren Graphikbildschirm an, mit dem je nach Parameter n folgendermaßen verfahren wird:

n = 0 Der Punkt x,y wird als Startpunkt einer Linie gespeichert, aber nicht gesetzt.

n = 1 Der Punkt x,y ist Zielpunkt einer Linie, deren Start punkt vorher definiert wurde. Die Linie wird entsprechend dem mit HCOLOR gewählten Modus gezeichnet.
Der Endpunkt der Linie wird als Startpunkt einer neuen Linie übernommen.

n = 2 Der Punkt x,y wird entsprechend dem gewählten Graphikmodus gesetzt, aber nicht als Startpunkt einer Linie übernommen.

HMOVE

Format: HMOVE x,y,f

Der Punkt x,y wird entsprechend dem Graphikmodus f gesetzt und als Startpunkt einer Linie abgespeichert. Danach wird f als neuer Graphikmodus abgespeichert.

HDRAW

Format: HDRAW x,y,f

Von einem definierten Startpunkt (s. HMOVE) aus wird eine Linie zum Punkt x,y im Graphikmodus f gezogen. Dann wird f als neuer Graphikmodus abgespeichert.

Im Gegensatz zu HPLOT zeichnet HMOVE schon den ersten Punkt der Linie, HDRAW zeichnet diesen dann nicht noch einmal. Diese Befehle sind sinnvoll, wenn der Graphikmodus häufig gewechselt werden soll. Man erspart sich dann den ständigen Aufruf von HCOLOR.

HCARO

Format: HCARO xl,yl,x2,y2,f

Die Punktpaare xl,yl und x2,y2 legen zwei diagonale Ecken eines Rechtecks fest, das entsprechend dem Graphikmodus f ausgefüllt wird.

HPEN

Format: HPEN

Dieser Befehl ermöglicht Ihnen, die manuelle Eingabe von Graphiken auf dem Bildschirm. Die momentane Position wird durch ein kleines Kreuz dargestellt, daß Sie mit Hilfe der 4 Pfeiltasten bewegen können. Drücken Sie mehrere Pfeiltasten zusammen, kombinieren sich deren Effekte.

Halten Sie die Leertaste fest, so bewegt sich der Cursor bei jedem Druck auf eine Pfeiltatse nur einen Punkt weiter, so ist genaues Positionieren möglich.

Zusätzliches Betätigen der Tasten <CLEAR> (f=0), <SHIFT> (f=1) oder <RETURN> (f=2) erzeugt Punkte im entsprechenden Graphikmodus.

Des weiteren existieren folgende Befehle unter HPEN:

<C> Bildschirm löschen (wie HCLS)

<N> Bildschirm invertieren (wie HNEG)

<H> Bildschirm horizontal spiegeln
(wie HFLEX 1)

<V> Bildschirm vertikal spiegeln
(wie HFLEX 0)

<P> (ohne <CLEAR>, <SHIFT> oder <RETURN>)
Aktuelle Cursorposition als Startpunkt einer Linie abspeichern.

<P> (mit <CLEAR>, <SHIFT> oder <RETURN>)
Punkt setzen und als Startpunkt einer Linie speichern

<L> (nur mit <CLEAR>, <SHIFT> oder <RETURN>)
Linie vom Startpunkt zur aktuellen Cursorposition ziehen.

<R> (nur mit <CLEAR>, <SHIFT> oder <RETURN>)
Das Rechteck, das durch den Startpunkt und die aktuelle Cursorposition bestimmt wird, wird ausgefüllt

<E> Rücksprung ins BASIC

HTEXT

Format: HTEXTx,y,f,text

Der angegebene Text wird im Graphikmodus f gezeichnet, wobei der Punkt x,y die linke unter Ecke des Textes darstellt. Wird x,y weggelassen, wird am Ende des letzten Textes weitergemacht, so ist es möglich, Texte nahtlos aneinanderzufügen.

HBIG

Format: HBIG x,y,f,b,h,text

Der angegebene Text wird in x-Richtung um den Faktor b und in y-Richtung um den Faktor h vergrößert. Ist einer der Faktoren Null, so wird die Textgröße in dieser Richtung halbiert. Ansonsten entspricht dieses Kommando dem Befehl HTEXT.

HCODE

Format: HCODE filename

Die angegebene Datei wird von Diskette in den Zeichensatz übernommen. Alle folgenden HTEXT- und HBIG-Befehle benutzen diesen Zeichensatz.

Auf Ihrer GDOS-Diskette finden Sie drei Schriftarten, nähmlich:

SCRIPT/RZS	Schreibschrift,
BLOCK/RZS	Blockschrift und
NORMAL/RZS	die normale Schrift des Computers.

HSAVE

Format: HSAVE filename

Der Inhalt des momentan veränderbaren Graphikbildschirms wird unter dem angegebenen Namen auf Diskette gespeichert.

HLOAD

Format: HLOAD filename

Der angegebene File wird in den momentan veränderbaren Graphikbildschirm gelesen. Hierbei wird, falls nötig, eine Anpassung an das momentane Bildschirmformat vorgenommen.

HMERGE

Format: HMERGE f,filename

Der Inhalt des angegebenen Files wird mit dem Inhalt des momentan veränderbaren Graphikbildschirm unter Beachtung des Parameters f gemischt:

f = 0 Alle in File gesetzten Punkte werden auf dem Bildschirm gelöscht

f = 1 Alle in File gesetzten Punkte werden auf dem Bildschirm gesetzt

f = 2 Alle in File gesetzten Punkte werden auf dem Bildschirm invertiert

HPRINT

Format: HPRINT br,lr

Der Inhalt des momentan veränderbaren Bildschirms wird auf einem angeschlossenen Drukker ausgedruckt. Der Parameter br gibt an, welche der 4 möglichen Breiten, (siehe auch das Kapitel, 'Druckerkonfigurierung') benutzt wird, lr gibt die Breite eines linken Randes an.

Beschreibung der einzelnen Parameter:

Für alle Parameter, die Zahlenangaben erfordern, gilt:

- Ein weggelassener Parameter wird als 0 angenommen

- Es können Zahlen, Variablen oder Funktionen angegeben werden

Für Parameter, die Texte angeben, gilt:

- Ein weggelassener Parameter wird als Nullstring angenommen

- Texte bestehen entweder aus Zeichenketten in Anführungszeichen (wie "TEST-TEXT"), Textvariablen (wie A$), Textfunktionen (wie "Hallo "+ MID $ (A$, x ,5)). Außerdem können Texte als Zahlen eingegeben werden. Sie werden dann als ASCII-Werte interpretiert. Eine zulässige Eingabe für den Text 'Peter sagte: "Hallo Klaus" ' ist also:

  "Peter sagte: ",34, "Hallo Klaus",34

  Die filespecs-Parameter müssen als normale 'GDOS'-Filespecs, bestehend aus Name/Erweiterung: Laufwerk, angegeben werden.

Druckerkonfigurierung:

Auf ihrer 'GDOS'-Diskette finden sie das File PRINTER/DRV, sowie die Files ITOH/DRV und STAR/DRV. Letztere sind vordefinierte Files für den Drucker ITOH 8510 bzw. die STAR-Drucker 'GEMINI 10X', 'DELTA 10 'und 'RADIX 10'.
Um RDLBASIC auf ihren Drucker zu konfigurieren, genügt es, den entsprechenden File auf den File PRINTER/DRV zu kopieren

Beispiel: COPY STAR/DRV PRINTER/DRV

Sollten sie jedoch einen Drucker besitzen, der mit keinem dieser Files läuft, können sie sich mit Hilfe des Instalationsprogramms INST/BAS eigene Drucker-Files erstellen. Das Programm fragt sie nach

dem Namen des Drucker
(wird als Filename benutzt)

Den Einleitungsequenzen für PICA, ELITE, COMPRESSED und PROPORTIONAL-Schrift
(Sollte ihr Drucker eine dieser Schriftbreiten nicht unterstützen, geben sie einfach nichts ein!).
Geben sie diese Sequenzen als durch Kommata getrennte Dezimalzahlen ein

Beispiel: ESC S entspricht 27,83

- Der Einleitungsequenz für die hochauflösende Graphik

- Die Sequenz zur Einstellung eines Linefeeds von etwa 1/8 Inch.

- Wie die Längenangabe bei der Einleitung zur hochauflösenden Graphik erfolgt.

  Hierbei bedeutet eine Eingabe von:

  0 1 Byte binär
  1 2 Byte binär (LSB/MSB-Format)
  2 2 Zeichen ASCII (also 00 bis 99)
  3 3 Zeichen ASCII (also 000 bis 999)
  4 4 Zeichen ASCII (also 0000 bis 9999)
  5 5 Zeichen ASCII (also 00000 bis 99999)

- Wie die einzelnen Datenbits organisiert sind.

  Hierbei bedeutet:

  0 Bit 0 entspricht dem obersten,
    Bit 7 dem untersten Punkt

  1 Bit 0 entspricht dem untersten,
    Bit 7 dem obersten Punkt

  Die entsprechenden Informationen entnehmen Sie bitte dem Handbuch ihres Druckers.

INST/BAS generiert aus diesen Eingaben ein File, das sie dann (wie oben beschrieben) auf PRINT/DRV kopieren können.

ANGANG D

Graphik Befehle:

(nur für Speedmaster und GENIE IIs mit Graphikzusatzkarte)

Anleitung zum HMHPLOT
(Graphik-Betriebssystem)

Um die graphischen Fähigkeiten Ihres Computers nutzen zu können, wird das Disk-BASIC des Rechners mit Hilfe des Programms HMHPLOT um einige Befehle erweitert.

Das Graphik-Modul wird als DOS-Call geladen und definiert das HIMEM. Es muß daher vor dem BASIC-Aufruf geladen werden.

Der Befehl lautet:

HMHPLOT,n,B

Wird der Parameter n weggelassen, so bittet das System um die Eingabe einer "0" oder "1", wodurch die Höhe des zu setzenden HIMEM bestimmt wird. Reicht der stets vorhandene Standard-Schriftcode aus, so ist eine "0" einzugeben. Andernfalls werden für ggf. hinzuzuladende Schönschrift-Codes zusätzlich 7000 Bytes des Speichers reserviert (HIMEM wird um 7000 erniedrigt).
Bei unmittelbarer Eingabe von n=0, 1 oder 2 entfällt die Frage, wobei n=2 die Erhaltung des momentan im Graphik-Speicher abgelegten Bildes bewirkt.

Anschließend wird automatisch initialisiert, und das System meldet sich mit READY. Falls der Parameter "B" nicht gesetzt wird, meldet sich das System direkt mit READY, und BASIC kann getrennt initialisiert werden. Von nun an werden die nachfolgend beschriebenen Graphik-Befehle als vermeintliche BASIC-Befehlsworte erkannt und ausgeführt.

ACHTUNG:

Folgt einer der nachfolgend aufgelisteten Befehle einem "THEN" oder "ELSE", so muß dazwischen ein ":" eingefügt werden.

Befehlsübersicht:

HON, HOFF, HCLS, HNEG, HFLEX

Graphik-Bildschirm ein-, ausblenden, löschen, negieren und spiegeln.

HCOLOR, HPLOT, HMOVE, HDRAW, HCARO, HPEN

Graphik-Modus setzen, Zeichen-Befehle, Cursor-Funktion.

HTEXT, HBIG, HCODE

Schreib- und Lade-Befehle für verschiedene Schriftarten.

HSAVE, HLOAD, HMERGE, HPSAVE, HPLOAD

Lade-Befehle zwischen Graphik-Bildschirm und Diskette

HPDEF, HPRINT, HPDISK

Befehle für Drucker-Auswahl und Daten-Übertragung auf Drucker.

Einzel-Beschreibungen:

HON

Format: HON

Der Graphik-Bildschirm wird in den Monitor zum System-Bildschirm eingeblendet. Beide Bildschirme bleiben jedoch voneinander unabhängig. D.h. Graphik-Befehle haben keinen Einfluß auf den System-Bildschirm, und sämtliche anderen BASIC-Befehle wirken nicht auf den Graphik-Bildschirm.

HOFF

Format: HOFF

Der Graphik-Bildschirm wird aus dem Monitor ausgeblendet, bleibt aber intern erhalten.

HCLS

Format: HCLS

Der Graphik-Bildschirm wird gelöscht, bleibt aber im Monitor eingeblendet (gleiche Funktion wie CLS auf dem System-Bildschirm). Im Falle der Negation wird der Bildschirm komplett aufgehellt.

HNEG

Format: HNEG

Der Graphik-Bildschirm wird negiert. D.h. statt hellen Linien auf dunklem Untergrund erhält man dunkle Linien auf hellem Untergrund und umgekehrt.

HFLEX

Format: HFLEX

Der Graphik-Bildschirm wird horizontal gespiegelt. D.h. zur Bildmitte symetrisch gelegene Graphik-Spalten werden paarweise vertauscht.

HCOLOR

Format: HCOLOR m

Mit HCOLOR wird, insbesondere für den Zeichen-Befehl HPLOT, der Graphik-Modus m voreingestellt.

Es bedeuten, auf einen Graphik-Punkt angewandt:

m=0 Punkt löschen

m=1 Punkt setzen

m=2 Punkt alternieren. D.h. der Punkt wird gelöscht, wenn er vorher gesetzt war, bzw. umgekehrt.

Die Bedeutungen von m=0 und m=1 werden "physikalisch" vertauscht, wenn HNEG ausgeführt wird, sodaß "logisch" ihre ursprüngliche Funktion erhalten bleibt.

HPLOT

Format: HPLOT x,y,n

Durch die Koordinaten x (horizontal zwischen 0 und 479) und y (vertikal zwischen 0 und 191) wird die Lage des anzufahrenden Punktes auf dem Bildschirm definiert. Dabei ist der Punkt (0,0) in der linken unteren Ecke positioniert. Der Parameter n hat folgende Bedeutung:

n=0 Der Punkt (x,y) wird als Startpunkt für eine nachfolgend zu zeichnende Linie abgespeichert, jedoch nicht gesetzt.

n=1 Der Punkt (x,y) ist Zielpunkt einer Linie, ausgehend von einem zuvor definierten Startpunkt. Die Linie wird entsprechend dem voreingestellten Graphik-Modus m (HCOLOR) gezeichnet (Start- und Endpunkt werden mit gezeichnet).

Anschließend wird (x,y) für einen nachfolgenden Plot-Befehl als neuer Startpunkt abgespeichert.

n=2 Der Punkt (x,y) wird entsprechend dem Graphik-Modus m (HCOLOR) gesetzt, jedoch nicht als Startpunkt für eine später zu zeichnende Linie übernommen.

Dieser Befehl eignet sich insbesondere für Graphiken, welche in einem festen Graphik-Modus erstellt werden, und bei welchen die "Move"- und "Draw"-Befehle abhängig von variablen Parametern gesteuert werden. Diese können dann unmittelbar nach n übergeben werden.

HMOVE

Format: HMOVE x,y,m

Der Punkt (x,y) (vgl. HPLOT) wird entsprechend dem Graphik-Modus (vgl. HCOLOR) gesetzt und als Startpunkt für eine anschliessend zu zeichnende Linie abgespeichert. Gleichzeitig wird HCOLOR durch Eingabe von m neu definiert.

HDRAW

Format: HDRAW x,y,m

Von einem zuvor definierten Startpunkt aus wird eine Linie entsprechend dem Graphik-Modus m nach dem Zielpunkt (x,y) gezogen (Parameter wie in HMOVE).
Dieser wird als neuer Startpunkt abgespeichert. Im Gegensatz zu HPLOT ist hier der Startpunkt nicht Teil der zu zeichnenden Linie.
HMOVE und HDRAW sind insbesondere dann zu empfehlen, wenn die Linien festgelegt sind, und der Graphik-Modus öfter gewechselt werden soll.
Man erspart sich hierdurch jeweils den Aufruf HCOLOR.

HCARO

Format: HCARO u,v,x,y,m

Durch die Koordinaten-Paare (u,v) und (x,y) werden zwei diagonal gegenüberliegende Punkte eines achsenparallelen Rechtecks dargestellt, dessen Inhalt entsprechend dem Graphik-Modus m ausgefüllt wird. Hierbei entsprechen (u,v) dem Startpunkt und (x,y) dem Zielpunkt in HPLOT und HDRAW. Nach Ausführung des Befehls besitzt das Paar (u,v) dieselben Werte wie (x,y).

HPEN

Format: HPEN

Diese Routine erwartet manuelle Eingriffe des Benutzers und erlaubt in ihrer Grundfunktion die Bewegung eines Cursors auf dem Bildschirm mit Hilfe der Pfeil-Tasten. Gleichzeitiges Drücken zweier verschiedener Pfeil-Tasten mischt die entsprechenden Richtungen. Ein Verlassen dieser Routine ist nur durch Drücken der Taste <E> möglich.

Eigentliches Ziel der Cursor-Funktion ist das manuelle Zeichnen von Linien, wobei der Graphik-Modus m (HCOLOR) durch gleichzeitiges Drücken einer der Funktionstasten <CLEAR> (m=0), <SHIFT> (m=1) und <ENTER> (m=2) festgelegt wird. Ist keine dieser Tasten gedrückt, so bewegt sich der Cursor ohne weitere Funktion. Ansonsten wird die jeweils aktuelle Cursor-Position als Startpunkt (z.B. für HPLOT, HDRAW, HCARO) abgespeichert.

Um exakte Positionen anfahren zu können, wird die Bewegung des Cursors in Einzelschritten durch Drücken der <Space>-Taste und gleichzeitig fortgesetztes Antippen der entsprechenden Pfeil-Taste (mit oder ohne Funktionstaste <CLEAR>, <SHIFT>, <ENTER>) ermöglicht.

Ferner existieren innerhalb von HPEN folgende Sonderfunktionen:

<C>	Bildschirm löschen (wie HCLS)
<N>	Bildschirm negieren (wie HNEG)
<S>	Bildschirm spiegeln (wie HFLEX)
<P>	(ohne Funktionstasten <CLEAR>, <SHIFT>, <ENTER>) Aktuelle Cursor-Position als Startpunkt abspeichern (wie HPLOT x,y,0)
<P>	(mit Funktionstasten) Punkt in der aktuellen Cursor-Position setzen, und diese als Startpunkt abspeichern (wie HMOVE)
<L>	(nur in Verbindung mit einer Funktionstaste) Linie von vor4efiniertem Startpunkt zur Cursor-Position zeichnen (wie HDRAW)
<R>	(nur in Verbindung mit einer Funktionstaste) Rechteck, bestimmt durch die Diagonale zwischen vordefiniertem Startpunkt und aktueller Cursor-Position, entsprechend dem Graphik-Modus m ausfüllen (wie HCARO)
<E>	Rücksprung in den Eingabe-Modus oder das BASIC-Programm

HTEXT

Format: HTEXT x,y,m,t

Der Text-String t wird entsprechend dem Graphik-Modus m auf den Graphik-Bildschirm geschrieben. Hierbei bezeichnen (x,y) die Koordinaten der linken unteren Ecke des ersten im Text-String enthaltenen Zeichens. Nach Ausführung des Befehls ist der Cursor (intern) am Ende des geschrienbenen Textes positioniert. Durch Weglassen von x und y in der Parameter-Liste lassen sich Texte nahtlos aneinanderfügen (vgl. Abschnitt über Eingabe-Parameter).

HBIG

Format: HBIG x,y,m,fx,fy,t

Gleiche Funktion wie HTEXT mit dem Unterschied, daß Breite und Höhe der Schrift variiert werden können.

Es bedeuten:

fx=0	Halbierung der Schriftbreite
fx=k	Vervielfachung der Schriftbreite um den Faktor k>0
fy=0	Halbierung der Schrifthöhe
fy=k	Vervielfachung der Schrifthöhe um den Faktor k>0.

Bei der Halbierung der Schriftbreite bzw. Schrifthöhe ist ggf. mit einer Entstellung der Schrift zu rechnen, da deren Auflösung halbiert wird.

HCODE

Format: HCODE flspec
HCODE 0
HCODE 1

Mit Hilfe dieser Befehle kann zwischen verschiedenen Schriftarten gewählt werden, falls nach dem LÖaden des Moduls HMHPLOT/CMD eine "1" eingegeben wurde. Dabei lädt der erste Befehl die unter 'flspec' angegebene Datei von der Diskette in den hierfür mittels HIMEM reservierten Speicher. Die geladene Datei enthält die Zeichen-Codes der gewünschten Schönschrift, welche fortan von HTEXT und HBIG benutzt wird. Möchte man auf die erste gespeicherte Standard-Schrift zurückgreifen, so gibt man einfach den zweiten Befehl ein. Mit dem dritten Befehl kann man ausschließlich wieder auf die bereits geladene Schönschrift umschalten.

Es stehen die Schriftarten 'Script' und 'Block' zur Verfügung, welche wahlweise, jedoch nicht gleichzeitig geladen werden können. Das Wechseln auf einen anderen Schönschrift-Code erfolgt über den Befehl 1.

HSAVE

Format: HSAVE flspec

Der Inhalt des Graphik-Bildschirms wird ohne Umwandlung der einzelnen Bytes auf die Diskette geschrieben. Es werden 13 Einheiten Disketten-Platz benötigt.

HLOAD

Format: HLOAD flspec

Die durch HSAVE auf der Diskette abgespeicherte Datei 'flspec' wird in den Graphik-Bildschirm kopiert. Dessen bisheriger Inhalt wird hierbei gelöscht. Da keine Byte-Umwandlung vorgenommen wird, benötigt der Ladevorgang nur etwa 5 Sekunden.

HMERGE

Format: HMERGE m,flspec

Eine mittels HSAVE abgespeicherte Datei wird unter Berücksichtigung des Graphik-Modus m dem Inhalt des Graphik-Bildschirms hinzugefügt.

HPSAVE

Format: HPSAVE flspec

Die Bit-Konfiguration des Graphik-Bildschirms erlaubt keine unmittelbare Übertragung von durch HSAVE auf Diskette abgespeicherten Dateien auf den Drucker. Deshalb wird der Inhalt des Graphik-Bildschirms vor dem Abspeichern konvertiert. Hierdurch reduziert sich der belegte Speicherplatz auf der Diskette auf 9 Einheiten.

HPLOAD

Format: HPLOAD flspec

Beim Laden von durch HPSAVE abgespeicherten Dateien muß deren Inhalt zurückkonvertiert werden. Daher benötigt dieser Ladevorgang etwa 14 Sekunden.

HPDEF

Format: HPDEF k

Da Drucker verschiedener Fabrikationen mit unterschiedlichen Steuercodes arbeiten, ist es notwendig, zuerst den anzusprechenden Druckertyp einzustellen.

Es sind 3 Druckertypen berücksichtigt,
welche wie folgt anzusprechen sind:

k=0	EPSON FX-80
k=1	EPSON MX-80/82
k=3	ITOH 8510A

HPRINT

Format: HPRINT a,b,c,d,e

Es bedeuten:

a	Schreibdichte
b	Tabulator
c	Zeilenvorchub
d	Bildgröße
e	Anzahl der Druckzeilen, welche maximal ausgegeben werden sollen

Die Schreibdiche (a) und deren Auswahl ist abhängig vom Druckertyp. Es werden die maximalen Anschläge pro Zoll/Zeile angegeben:

FX-80:

0 : 60/480 (entspr. ESC"*",CHR$(0))
1 : 120/960
2 : 120/960 (dopp. Geschwindigkeit)
3 : 240/1920
4 : 80/640
5 : 72/576 (1:1 Graphik)
6 : 90/720 (entspr. ESC"*",CHR$(6))

MX-80/82:

0 : 60/480 bzw. 72/576   (entspr. ESC"K")
1 : 120/960 bzw. 144/1152 (entspr. ESC"L")

ITOH 8510A:

0 : 80/640   (entspr. ESC"N")
1 : 96/768   (entspr. ESC"E")
2 : 160/1280 (entspr. ESC"P")
3 : 136/1088 (entspr. ESC"Q")

Der Tabulator (b) kann beliebig gewählt werden und wird in Anschlägen (Dots) der angewählten Schreibdichte gezählt.

Der Zeilenvorschub (c) ist je nach Drucker-typ wie folgt einstellbar:

FX-80:

n/216 Zoll (0 < n < 256)   (entspr. ESC"3")

MX-80/82:

n/72 Zoll (0 < n < 86)   (entspr. ESC"A")

ITOH 8510A:

n/144 Zoll (0 < n < 100)   (entspr. ESC"T")

Um Graphik ohne Lücken und ohne Überlappung zu erhalten, wird n=23 für den FX-80, n=8 für den MX-80/82 und n=17 für den ITOH 8510A empfohlen.

Es können drei verschiedene Bildgrößen (d) unabhängig von der Wahl in (a) und (c) ausgewählt werden:

d=0	Standard-Größe
d=1	Bild wird horizontal und vertikal verdoppelt und um 90 Grad verdreht ausgedruckt
d=2	Bild wird nur vertikal verdoppelt und aufrecht gezeichnet

Die Beschränkung (e) gibt an, wieviele Druckzeilen höchstens ausgegeben werden sollen. Sie entsprechen nur im Fall d=0 den Zeilen des Graphik-Bildschirms (24 Zeilen zu jeweils 8 Dots).

HPDISK

Format: HPDISK a,b,c,d,e,flspec

Mittels HPDISK können durch HPSAVE auf Diskette abgespeicherte Inhalte des Graphik-Bildschirms direkt auf den Drucker übertragen werden. Dies ist insbesondere bei Verwendung solcher "Daten"-Disketten auf Rechnern ohne Graphik-Karte von Bedeutung, da sie auch dort ausgedruckt werden können. Die Parameter a,b,c und e sind identisch mit jenen aus HPRINT.

Für d gilt:

d=0 Übertragung des Original-Bildes
d=1 Übertragung des negierten Bildes (vgl. HNEG)

Der Text-String 'flspec' schließlich gibt wieder den Namen des auszudruckenden Disketten-Files an.

Eingabe einer Parameter-Liste:

Falls der abzuarbeitende Graphik-Befehl die Eingabe einer Paraameter-Liste verlangt (z.B. HPRINT), gibt es grundsätzlich 3 Möglichkeiten, diese einzugeben:

HPRINT a,b,c,d,e <ENTER>
HPRINT,a,b,c,d,e <ENTER>
HPRINT (a,b,c,d,e) <ENTER>

Die Unterschiede sind rein optischer Natur und haben keinen Einfluß auf den Funktionsablauf.

Für die Parameter selbst werden mit Ausnahme von 't' und 'flspec' grundsätzlich nicht-negative ganze Zahlen erwartet, welche entweder direkt oder als Variablen-Werte (nicht indiziert!) übergeben werden können.

Für 't" und 'flspec' werden Text-Strings erwartet. Auch hier kann man zwischen der direkten Eingabe und der Übergabe einer String-Variablen wählen. Da i.a. nicht alle im jeweiligen Code-Satz vorhandenen Zeichen direkt über die Tastatur angesprochen werden können, ist zusätzlich die Eingabe eines Zeichen-Codes als Integer-Wert zwischen 0 und 127 bzw. einer entsprechenden Variable innerhalb des Text-Strings möglich:

```
t = "Text",32,0,1," String"
```

Da unter den Codes 0 und 1 griechische Buchstaben abgespeichert sind und 32 standardmäßig mit " " belegt ist, werden hinter dem String "TEXT", ein Leerzeichen, zwei griechische Buchstaben, sowie die Zeichenkette " String" ausgegeben.
Es ist also die Aneinanderreihung einer beliebigen Zahl von Text-Segmenten möglich. Denn der Text-String ist stets am Ende der Parameter-Liste angeordnet und wird nur durch die Ende-Kennung des BASIC-Befehls abgeschlossen.

'flspec' steht grundsätzlich für einen Dateinamen für die Diskette, bestehend aus Name, Extension und Laufwerks-Nummer. Hierbei ist nur der Name ohne Extension zwingend.

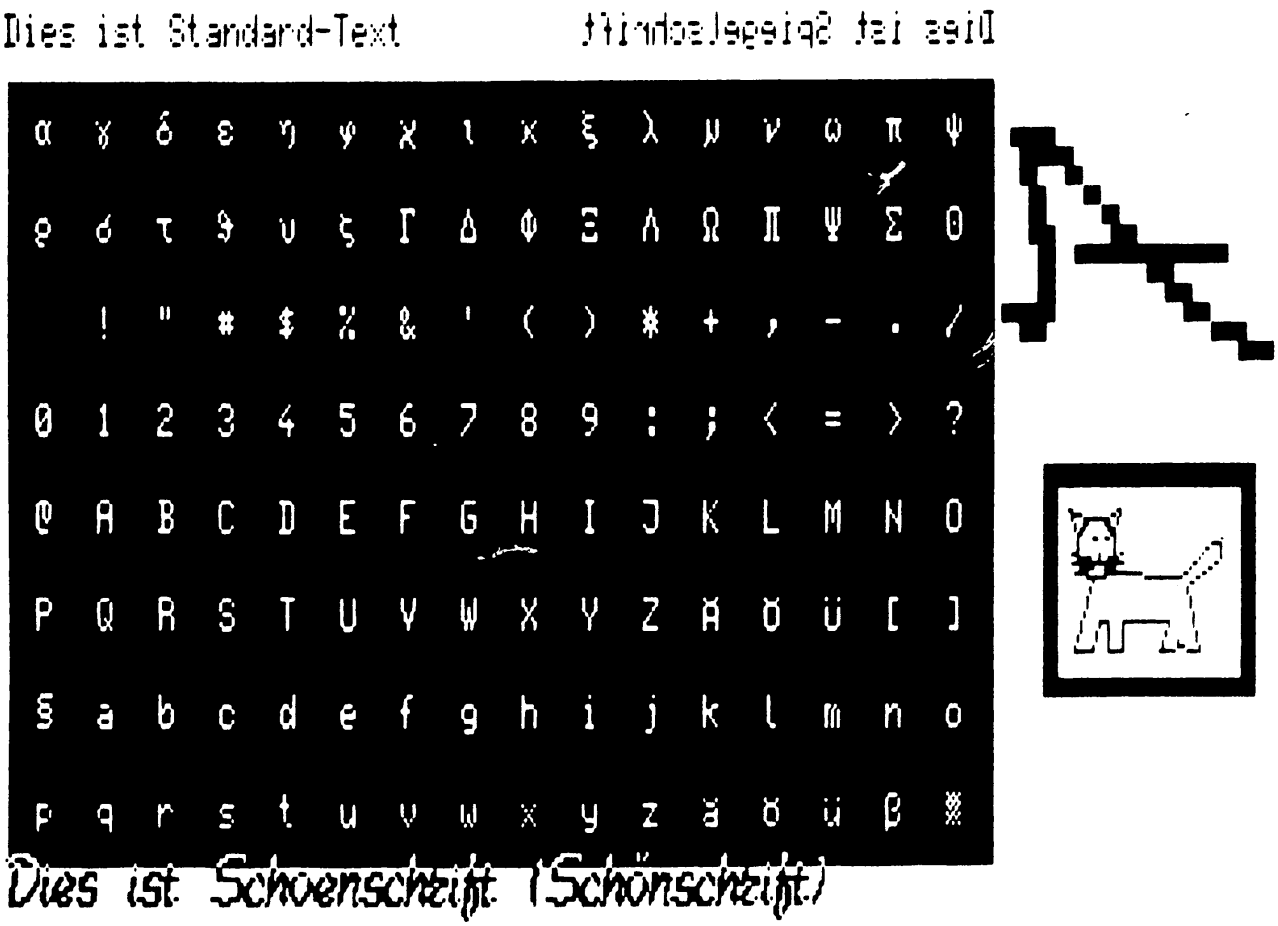
Es werden folgende Standard-Extensionen angenommen:

HCODE	Ext = /PCD
HSAVE, HLOAD, HMERGE	Ext = /BLD
HPSAVE, HPLOAD, HPDISK	Ext = /PRT

Man beachte, daß PCD-, BLD- und PRT-Files aufgrund verschiedener Formatierungen nicht austauschbar sind!

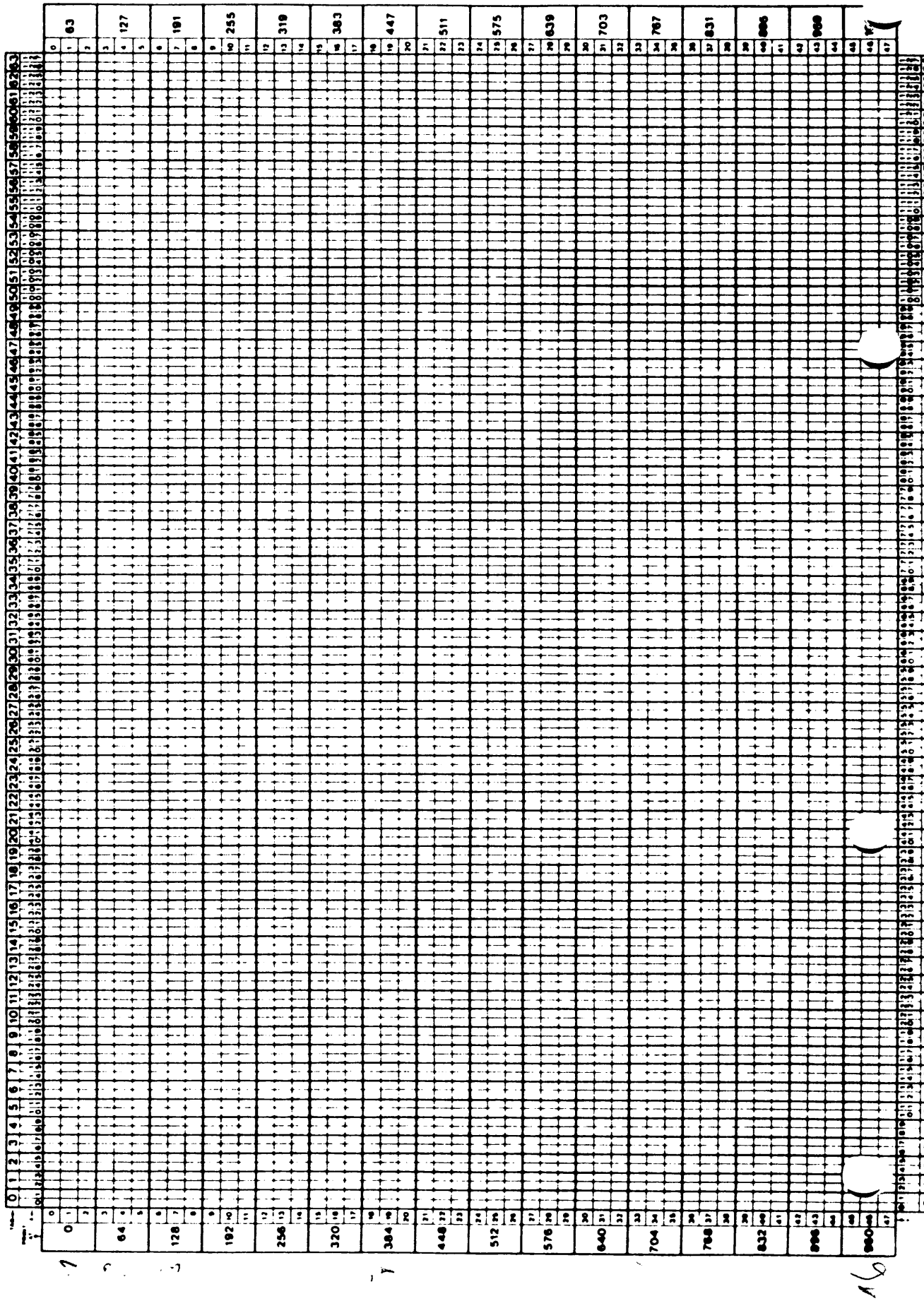
Bei den ganzzahligen Parametern gibt es noch die Möglichkeit, sie einfach wegzulassen, wenn alte Werte angenommen werden sollen. (Ausnahme sind HTEXT und HBIG, wo die (x,y)-Koordinaten nicht den alten, sondern die am Text-Ende gelegenen Werte darstellen). Soll auf die Eingabe von "sämtlichen" oder "restlichen" Integer-Werten verzichtet werden, so gibt man hinter dem Befehlswort lediglich den verbleibenden Text-String (welcher niemals weggelassen werden darf) oder überhaupt nichts ein. Werden jedoch hinter wegzulassenden Werten andere Werte angegeben, so müssen die "Lücken" durch Kommata eingeschlossen sein. Insbesondere muß dann zwischen der ersten "Lücke" und dem Befehlswort ein Komma stehen! In bestimmten Fällen (z.B. HCOLOR m, HCODE 0, HCODE 1) ist das Weglassen der Eingabe-Parameter nicht zulässig und wird ggf. mit einer Fehlermeldung beantwortet.

Graphik-Beispiel mit Code-Tabelle:



ANHANG E

Bildschirmtabelle:



ANHANG F

Programme:

Auf den folgenden Seiten finden Sie die Listings folgender vier Programme:

Panzerschlacht

Regierungsspiel

Gleichung

Tilgung

Nun noch einige allgemeine Worte zur Handhabung dieser Listings:

Alle vier Programme sind von routinierten Programmierern geschrieben. Daher enthalten sie eine Vielzahl von Programmiertricks. Seien Sie also nicht verärgert, wenn Sie nicht alle Programmschritte verstehen. Vorsichtig sollten Sie beim Abtippen der Programme sein. Eine falsche Zahl o.ä. kann Ihnen z.B. eine Menge Schwierigkeiten bereiten. Die beste Lösung ist, wenn Ihnen jemand die Programmzeilen diktiert, der möglichst gleichzeitig Ihre Eingaben kontrolliert. Sollten Sie aber die Programme allein eingeben, kontrollieren Sie am besten jede Zeile, nachdem Sie sie eingegeben haben. Dies kostet zwar etwas Zeit, kann Ihnen aber eine schwierige Fehlersuche ersparen.

Panzerschlacht:

Panzerschlacht ist ein Spiel, das man zu zweit gegeneinander spielen kann. Sinn ist es natürlich, den Gegenspieler abzuschießen, bevor dieser es tut.
Die Steuerung wird durch das Programm selbst erklärt.

```
10 CLEAR150
20 DEFINTA-Z
30 DIMA$(8),A(8),B(100),B$(3)
40 B$(1)="*"
50 B$(2)="O"
60 B$(3)="+"
70 GOTO260
80 A=PEEK(14368)
90 IFA>127THEND=D+1:IFD=9THEND=1
100 IF(AAND64)THEND=D-1:IFD=0THEND=8
110 IF(AAND16)THENIFPEEK(P+15361+A(D))=32THE
NPRINT@P,"   ";:P=P+A(D)ELSEIFPEEK(P+15361+A
(D))<128THENPRINT@P,"   ";:P=P+A(D):GOTO1930
120 IF(PEEK(14338)AND32)THENIFS=0THENS=P:R=D
130 IFS<>0THENPRINT@S,"   ";:IFPEEK(15361+S+
A(R))=32THENS=S+A(R):PRINT@S+1,CHR$(140);ELS
EIFS=P1-A(D)THEN1880ELSES=0
140 PRINT@P,A$(D);
150 IF(PEEK(14337)AND8)THEND1=D1-1:IFD1=0THE
ND1=8
160 IF(PEEK(14340)AND64)THEND1=D1+1:IFD1=9TH
END1=1
170 A=PEEK(14344)
180 IF(AAND1)THENIFPEEK(P1+15361+A(D1))=32TH
ENPRINT@P1,"   ";:P1=P1+A(D1)ELSEIFPEEK(P1+1
5361+A(D1))<128THENPRINT@P1,"   ";:P1=P1+A(D
1):GOTO1880
190 IF(AAND4)THENIFS1=0THENS1=P1:R1=D1
200 IFS1<>0THENPRINT@S1,"   ";:IFPEEK(15361+
S1+A(R1))=32THENS1=S1+A(R1):PRINT@S1+1,CHR$(
140);ELSEIFS1=P-A(D1)THEN1930ELSES1=0
210 PRINT@P1,A$(D1);
220 G=G+1
230 IFG>BOTHENG=1
```

```
240 PRINT@B(G),B$(RND(3));
250 GOTO80
260 CLS
270 PRINT@22,"P A N Z E R S C H L A C H T"
280 FORA=1TO33
290 FORA=10TO127
300 SET(A,4)
310 NEXT
320 FORA=4TO46
330 SET(127,A)
340 SET(126,A)
350 NEXT
360 FORA=127TO10STEP-1
370 SET(A,46)
380 NEXT
390 FORA=46TO4STEP-1
400 SET(10,A)
410 SET(11,A)
420 NEXT
430 PRINT@140,"BEI DIESEM SPIEL KANN MAN ZWE
I PANZER MIT ";
440 PRINT@204,"FOLGENDEN TASTEN STEUERN:";
450 PRINT@268,"SPIELER NR. 1:";
460 PRINT@349,"<V> - RECHTSDREHEN";
470 PRINT@413,"<C> - LINKSDREHEN";
480 PRINT@477,"<X> - FAHREN";
490 PRINT@541,"<Z> - FEUER";
500 PRINT@588,"SPIELER NR. 2:";
510 PRINT@669,"</> - RECHTSDREHEN";
520 PRINT@733,"<.> - LINKSDREHEN";
530 PRINT@797,"<,> - FAHREN";
540 PRINT@861,"<M> - FEUER";
550 PRINT@908,"ZUM START <S> DRUECKEN ";
560 A$(1)=CHR$(188)+CHR$(143)+CHR$(188)
```

```
570 A$(2)=CHR$(188)+CHR$(188)+CHR$(131)
580 A$(3)=CHR$(179)+CHR$(191)+CHR$(140)
590 A$(4)=CHR$(143)+CHR$(143)+CHR$(176)
600 A$(5)=CHR$(143)+CHR$(188)+CHR$(143)
610 A$(6)=CHR$(176)+CHR$(143)+CHR$(143)
620 A$(7)=CHR$(140)+CHR$(191)+CHR$(179)
630 A$(8)=CHR$(131)+CHR$(188)+CHR$(188)
640 FORA=1TO50
650 GOSUB2190
660 NEXT
670 P=960
680 FORA=1TO16
690 PRINT@P,A$(1);
700 GOSUB2190
710 PRINT@P,"   ";
720 P=P-64
730 NEXT
740 GOSUB2190
750 PRINT@0,A$(2);
760 GOSUB2190
770 PRINT@0,A$(3);
780 GOSUB2190
790 FORA=3TO21
800 PRINT@A," "CHR$(140);
810 GOSUB2190
820 NEXT
830 PRINT@A," ";
840 PRINT@0,A$(4);
850 GOSUB2190
860 FORA=1TO16
870 P=P+64
880 PRINT@P,A$(5);
890 GOSUB2190
900 PRINT@P,"   ";
```

```
910 NEXT
920 FORA=1TO20
930 GOSUB2190
940 NEXT
950 GOTO260
960 CLS
970 PRINT@320,CHR$(23);
980 INPUT"NAME DES 1. SPIELERS";B$
990 INPUT"NAME DES 2. SPIELERS";C$
1000 PRINT
1010 INPUT"BEI WIEVIELEN TREFFERN SOLL DAS S
PIEL ZU ENDE SEIN";T
1020 IFLEN(B$)>11THENB$=LEFT$(B$,12)
1030 IFLEN(C$)>11THENC$=LEFT$(C$,12)
1040 INPUT"MIT WIEVIELEN BOMBEN WOLLEN SIE S
PIELEN";BO
1050 IFBO<2ORBO>100THEN1040
1060 CLS
1070 FORA=4TO125
1080 SET(A,4)
1090 SET(A,46)
1100 NEXT
1110 FORA=4TO46
1120 SET(4,A)
1130 SET(5,A)
1140 SET(124,A)
1150 SET(125,A)
1160 NEXT
1170 FORA=4TO10
1180 SET(64,A)
1190 SET(65,A)
1200 SET(64,50-A)
1210 SET(65,50-A)
1220 NEXT
```

```
1230 FORA=16TO29
1240 SET(A,10)
1250 SET(A,40)
1260 SET(129-A,10)
1270 SET(129-A,40)
1280 NEXT
1290 FORA=16TO34
1300 SET(16,A)
1310 SET(17,A)
1320 SET(112,A)
1330 SET(113,A)
1340 NEXT
1350 FORA=22TO28
1360 SET(28,A)
1370 SET(29,A)
1380 SET(100,A)
1390 SET(101,A)
1400 NEXT
1410 FORA=34TO40
1420 SET(40,A)
1430 SET(41,A)
1440 SET(88,A)
1450 SET(89,A)
1460 SET(40,50-A)
1470 SET(41,50-A)
1480 SET(88,50-A)
1490 SET(89,50-A)
1500 NEXT
1510 FORA=41TO53
1520 SET(A,40)
1530 SET(A,10)
1540 SET(129-A,10)
1550 SET(129-A,40)
1560 NEXT
```

```
1570 PRINT@133,"*";
1580 PRINT@955,"*";
1590 FORA=1TOBO
1600 B=RND(8)*128+RND(19)*3+2
1610 IFPEEK(15360+B)=32THENPRINT@B,"*";:B(A)
=BELSE1600
1620 NEXT
1630 H=2
1640 O1=0
1650 O=0
1660 PRINT@40,"NEUES SPIELFELD ?";
1670 A$=INKEY$
1680 A$=INKEY$
1690 IFA$="N"THEN1710
1700 IFA$="J"THENCLS:PRINT@640,;:GOTO1040ELS
E1680
1710 PRINT@40,CHR$(30);
1720 D=1
1730 P1=132
1740 P=954
1750 D1=5
1760 S=0
1770 S1=0
1780 PRINT@0,"  "B$,O1,"  "C$,O;
1790 A(1)=-64
1800 A(2)=-61
1810 A(3)=3
1820 A(4)=67
1830 A(5)=64
1840 A(6)=61
1850 A(7)=-3
1860 A(8)=-67
1870 GOTO80
1880 FORA=1TO100
```

```
1890 PRINT@P1+RND(3)-1,CHR$(RND(160)+32);
1900 NEXT
1910 O=O+1
1920 GOTO1970
1930 FORA=1TO100
1940 PRINT@P+RND(3)-1,CHR$(RND(160)+32);
1950 NEXT
1960 O1=O1+1
1970 IFT<>OANDT<>O1THENPRINT@P,"    ";:PRINT@
P1,"    ";:PRINT@S,"  ";:PRINT@S1,"  ";:GOTO1
710
1980 CLS
1990 A$="* ERGEBNIS *"
2000 FORA=4TO-4STEP-1
2010 FORB=1TO6
2020 PRINT@28+B*(64+A),A$" ";
2030 PRINT@859-B*(64+A)," "A$;
2040 NEXT
2050 NEXT
2060 FORA=1TO12
2070 PRINT@40+A*60,CHR$(232);
2080 NEXT
2090 FORA=1TO200
2100 NEXT
2110 CLS
2120 PRINT@22,"*** ERGEBNIS ***"
2130 PRINTSTRING$(64,61)B$" HAT"O1"PUNKTE ER
REICH."
2140 PRINT
2150 PRINTC$" HAT"O"PUNKTE ERREICHT."
2160 FORA=1TO5000
2170 NEXT
2180 RUN
2190 FORO=1TO20
```

```
2200 IFINKEY$="S"THEN960
2210 NEXT
2220 RETURN
```

Regierungsspiel:

Dieses Programm macht sehr viel Spaß, vor allem wenn man es mit Leuten spielt, die etwas vom Regieren verstehen.
Jeder Mitspieler regiert einen kleinen Stadtstaat, der groß und mächtig werden soll.
Ihr Erfolg wird durch den Titel deutlich, den Ihnen das Programm verleiht. Wer als erster "HRH König" ist, hat gewonnen. Das Programm gibt auf Wunsch noch eine kleine Anleitung, die einige Tips enthält.
Ansonsten kann einem die Erfahrung helfen, die vielen Entscheidungen, die jedes Regierungsjahr anfallen, weise zu treffen.

```
10 RANDOM
20 CLEAR700
30 DEFINTA-J,M-Q,T,V,W
40 Y(0)=1400
50 DATA"Sir ","Baron ","Graf ","Marquis ","H
erzog ","Grossherzog ","Prinz ","* HRH Koeni
g "
60 DATA"Lady ","Baroness ","Graefin ","Marqu
ise ","Herzogin ","Grossherzogin ","Prinzess
in ","* HRH Koenigin "
70 DATA"Santa Paravia","Fiumaccio","Torricel
la","Molinetto","Fontanile","Romagna"
80 CLS
90 PRINT
100 PRINT@316,CHR$(23);"    Santa Paravia un
d Fiumaccio"
110 FORA=0TO555
120 NEXT
130 CLS
140 PRINT
150 FORA=1TO16
160 READA$
170 NEXT
180 PRINT" Wieviele Mitspieler ? (Eingabe zw
ischen 1 und 6)";
190 GOSUB6170
200 F=VAL(A$)
210 IFF<1ORF>6THENCLS:GOTO180
220 FORA=1TOF
230 READT$(A)
240 CLS
250 PRINT
260 PRINT"Wer ist Regent von ";T$(A);
270 INPUTN$(A)
```

```
280 N$(A)=N$(A)+" von "+T$(A)
290 CLS
300 PRINT"Ist ";N$(A);" ein Mann oder eine F
rau";
310 V(A)=0
320 GOSUB6170
330 IFA$<>"M"ANDA$<>"F"THEN310
340 IFLEFT$(A$,1)="F"THENV(A)=8
350 G(A)=25
360 H(A)=10
370 I(A)=5
380 J(A)=2
390 O(A)=1420+RND(35)
400 K(A)=1000
410 L(A)=10000
420 R(A)=5000
430 T(A)=1
440 U(A)=1
450 N(A)=4
460 P(A)=25
470 Q(A)=5
480 M(A)=25
490 S(A)=2000
500 NEXT
510 FORA=1TOF
520 RESTORE
530 B=V(A)+T(A)
540 FORC=1TOB
550 READT$(A)
560 NEXTC
570 NEXTA
580 PRINT
590 PRINT"Wuenschen Sie eine Einweisung ? (J
/N)";
```

```
600 GOSUB6170
610 IFA$="N"THEN640
620 IFA$<>"J"THEN580:GOSUB6200
630 GOSUB6200
640 CLS
650 PRINT"1. Anfaenger  2. Fortgeschrittener
  3. Profi  4. Grossmeister"
660 PRINT"Eingabe der Spielstaerke";
670 GOSUB6170
680 U(0)=VAL(A$)
690 IF U(0)<1U(0)=1
700 IF U(0)>4U(0)=4
710 U(0)=U(0)+5
720 E=E+1
730 IF T(E)=-1E=E+1
740 IF(T(1)<1)AND(T(2)<1)AND(T(3)<1)AND(T(4)
<1)AND(T(5)<1)AND(T(6)<1)THEN6400
750 IFE>FTHENE=0:Y(0)=Y(0)+1:GOTO720
760 IFY(0)>O(E)THEN720
770 IFY(0)=O(E)THEN960
780 GOSUB1330
790 GOSUB2050
800 GOSUB2780
810 GOSUB2410
820 GOSUB3520
830 GOSUB4870
840 GOSUB5320
850 GOTO720
860 CLS
870 PRINT"Adelige Soldaten Klerus Haendler
Sklaven  Land   Staatskasse"
880 PRINT
890 FORA=1TOF
900 PRINTT$(A);N$(A)
```

```
910 PRINTN(A);TAB(7)P(A);TAB(16)Q(A);TAB(23)
M(A);TAB(32)S(A);TAB(40)L(A);TAB(50)K(A)
920 NEXT
930 PRINT
940 INPUT"  (Druecke Enter)";A$
950 RETURN
960 CLS
970 PRINT
980 PRINT"Sehr schlechte Nachrichten"
990 PRINT
1000 PRINTT$(E);N$(E);" ist gerade gestorben
,"
1010 T(E)=-1
1020 Y=RND(8)
1030 IFY(0)>1450PRINT"nach langer Regentscha
ft und in hohem Alter":GOTO1090
1040 IFY<4PRINT"an Lungenentzuendung nach ei
nem kalten Winter in seinem Schloss"
1050 IFY=5PRINT"waehrend einer Pockenepidemi
e"
1060 IFY=4PRINT"an Thyphus nach Genuss vom s
chmutzigem Wasser"
1070 IFY=6PRINT"nach einem Raubueberfall wae
hrend einer Reise"
1080 IFY>6PRINT"an Fleischvergiftung"
1090 PRINT
1100 INPUT"(Druecke New Line)";A$
1110 IFF=1THEN6400
1120 GOSUB3520
1130 GOSUB860
1140 GOTO720
1150 I!=INT(I!)
1160 RETURN
1170 C!=INT(C!)
```

```
1180 RETURN
1190 S!=INT(S!)
1200 RETURN
1210 K(E)=INT(K(E))
1220 RETURN
1230 Z=RND(A)*S(E)/100
1240 Z%=Z
1250 PRINTZ%;"Sklaven in diesem Jahr geboren
"
1260 S(E)=S(E)+Z%
1270 RETURN
1280 Z=RND(A)*S(E)/100
1290 Z%=Z
1300 PRINTZ%;"Sklaven in diesem Jahr gestorb
en"
1310 S(E)=S(E)-Z%
1320 RETURN
1330 W=(RND(5)+RND(6))/2
1340 ONWGOTO1350,1370,1390,1410,1430
1350 W$="Trocken   eine Hungersnot droht"
1360 GOTO1450
1370 W$="Schlechtes Wetter   duerftige Ernte
"
1380 GOTO1450
1390 W$="Normales Wetter   durchschnittliche
 Ernte"
1400 GOTO1450
1410 W$="Schoenes Wetter   gute Ernte"
1420 GOTO1450
1430 W$="Exzellentes Wetter   hervorragende
Ernte"
1440 GOTO1450
1450 R=RND(50)
1460 R(E)=(R(E)*100-R(E)*R)/100
```

```
1470 X=L(E)
1480 Y=(S(E)-D(E)*100)*5
1490 IF Y<0Y=0
1500 IFY<XTHENX=Y
1510 Y=R(E)*2
1520 IFY<XTHENX=Y
1530 R(E)=R(E)-X/2
1540 Y=W-.5
1550 H!=X*Y
1560 R(E)=R(E)+H!
1570 D!=N(E)*100+C(E)*40+M(E)*30+P(E)*10+S(E
)*5
1580 L=(3*W+RND(6)+RND(6)+10)/10
1590 IFW=1THENL=L-1
1600 IFH!<1Y=2:GOTO1630
1610 Y=D!/H!
1620 IF Y>2Y=2
1630 IF Y<.8Y=.8
1640 L=L*Y
1650 L=INT(L*10)
1660 L=L/10
1670 Z=6-W
1680 G=(Z*3+RND(5)+RND(5))/5*Y*20
1690 RETURN
1700 PRINT
1710 PRINT"Ratten vernichteten ";R;" % Deine
r Getreidereserven"
1720 PRINTW$;" (";H!;" Zentner)"
1730 PRINT
1740 IFK(E)<32766GOSUB1210
1750 PRINT"Getreide     Getreide     Preis vo
n  Preis von  Staatskasse"
1760 PRINT"Reserve      Bedarf       Getreide
   Land"
```

```
1770 PRINTR(E);TAB(13)D!;TAB(24)G;TAB(36)L;T
AB(48)K(E)
1780 PRINT"Zentner      Zentner     1000 Z.
   Hektar      Gulden "
1790 RETURN
1800 J=(J(E)*300-500)*T(E)
1810 ONJ(E)GOTO1820,1840,1860,1880
1820 J$="sehr gerecht"
1830 GOTO1890
1840 J$="gemaessigt"
1850 GOTO1890
1860 J$="streng"
1870 GOTO1890
1880 J$="zuegellos"
1890 Y=150-G(E)-H(E)-I(E)
1900 IF Y<1Y=1
1910 C!=(N(E)*180+Q(E)*75+M(E)*20)*(Y/100)+U
(E)*100
1920 S!=(N(E)*50+M(E)*25+U(E)*10)*(Y/100)*(5
-J(E))/2
1930 I!=N(E)*250+U(E)*20+(10*J(E)*N(E))*(Y/1
00)
1940 C!=C!*G(E)/100
1950 IFC!<32760GOSUB1170
1960 S!=S!*H(E)/100
1970 IFS!<32760GOSUB1190
1980 I!=I!*I(E)/100
1990 IFI!<32760GOSUB1150
2000 PRINT"Staatseinnahmen    ";J+C!+S!+I!;"
Gulden"
2010 PRINT"Zollgebuehren","Umsatzsteuer","Ei
nk.steuer","Gericht"
2020 PRINTG(E);"%",H(E);"%",I(E);"%",J$
2030 PRINTC!,S!,I!,J;"Gl."
```

```
2040 RETURN
2050 CLS
2060 PRINT
2070 PRINTT$(E);N$(E)
2080 GOSUB1700
2090 PRINT
2100 PRINT"1.Getr.kauf   2.Getr.verkauf    3
.Landkauf   4.Landverkauf"
2110 PRINT"(Tippe 0 zur Fortsetzung)";
2120 GOSUB6170
2130 I!=VAL(A$)
2140 PRINT
2150 IFI!>4THENCLS:GOTO2100
2160 IFI!<1THENRETURN
2170 ONI!GOTO2180,2270,2320,2360
2180 PRINT
2190 INPUT"Wieviel Getreide moechtest Du kau
fen";I!
2200 K(E)=K(E)-(I!*G/1000)
2210 R(E)=R(E)+I!
2220 CLS
2230 PRINT
2240 PRINTT$(E);N$(E)
2250 GOSUB1730
2260 GOTO2090
2270 INPUT"Wieviel Getreide moechtest Du ver
kaufen";I!
2280 IFI!>R(E)PRINT"... so viel hast Du nich
t":PRINT:GOTO2270
2290 K(E)=K(E)+(I!*G/1000)
2300 R(E)=R(E)-I!
2310 GOTO2220
2320 INPUT"Wieviel Hektar moechtest Du kaufe
n";I!
```

```
2330 L(E)=L(E)+I!
2340 K(E)=K(E)-(I!*L)
2350 GOTO2220
2360 INPUT"Wieviel Hektar moechtest Du verka
ufen";I!
2370 IFI!>(L(E)-5000)PRINT"... so viel kanns
t Du nicht verkaufen":GOTO2360
2380 L(E)=L(E)-I!
2390 K(E)=K(E)+(I!*L)
2400 GOTO2220
2410 CLS
2420 PRINT
2430 PRINTT$(E);N$(E)
2440 PRINT
2450 GOSUB1800
2460 PRINT
2470 PRINT"1. Zollgebuehren  2. Umsatzsteuer
  3. Eink.steuer  4. Gericht"
2480 PRINT"(Tippe Indexnr. zum [ndern, 0 zur
 Fortsetzung)";
2490 GOSUB6170
2500 I=VAL(A$)
2510 PRINT
2520 IFI>4CLS:GOTO2460
2530 IFI<1THEN2740
2540 ONIGOTO2550,2600,2640,2680
2550 INPUT"Neue Zollgebuehren (0 bis 100 %)"
;I
2560 IF I>100I=100
2570 IF I<0I=0
2580 G(E)=I
2590 GOTO2410
2600 INPUT"Neue Umsatzsteuer (0 bis 50 %)";I
2610 IF(I>50)OR(I<0)I=5
```

```
2620 H(E)=I
2630 GOTO2410
2640 INPUT"Neue Einkommensteuer (0 bis 25 %)
";I
2650 IF(I<0)OR(I>25)THENI=0
2660 I(E)=I
2670 GOTO2410
2680 PRINT"Gericht: 1.sehr gerecht  2.gemaes
sigt  3.streng  4.zuegellos";
2690 GOSUB6170
2700 I=VAL(A$)
2710 IF(I>4)OR(I<1)THENI=1
2720 J(E)=I
2730 GOTO2410
2740 K(E)=K(E)+C!+S!+I!+J
2750 IFK(E)<0THENK(E)=K(E)*1.5
2760 IFK(E)<(-10000*T(E))THEN5800
2770 RETURN
2780 PRINT
2790 INPUT"Wieviel Getreide gibst Du fuer de
n Verbrauch frei";G!
2800 IFG!<(R(E)/5)PRINT"Du musst mindestens
20% Deiner Getreidereserven freigeben":GOTO2
790
2810 IFG!>(R(E)-(R(E)/5))PRINT"Du musst mind
. 20% zurueckbehalten":GOTO2790
2820 R(E)=R(E)-G!
2830 CLS
2840 PRINT
2850 PRINTT$(E);N$(E)"!"
2860 PRINT
2870 Z=G!/D!-1
2880 IF Z>0Z=Z/2
2890 IF Z>.25Z=Z/10+.25
```

```
2900 Z%=50-G(E)-H(E)-I(E)
2910 IF Z%<0Z%=Z%*J(E)
2920 Z%=Z%/10
2930 IF Z%>0Z%=Z%+3-J(E)
2940 Z=Z+(Z%/10)
2950 IF Z>.5Z=.5
2960 IFG!<(D!-1)THEN3250
2970 A=7
2980 GOSUB1230
2990 A=3
3000 GOSUB1280
3010 IF(G(E)+H(E))<35M(E)=M(E)+RND(4)
3020 IFI(E)<RND(20)N(E)=N(E)+RND(2)-1:Q(E)=Q
(E)+RND(3)-1
3030 IFG!<(D!+D!*.3)THEN3190
3040 Z%=S(E)/1000
3050 Z=(G!-D!)/D!*10
3060 Z=Z*Z%*RND(25)+RND(40)
3070 IF Z>32000Z=32000
3080 Z%=Z
3090 Z=RND(Z%)
3100 PRINTZ;"Sklaven kamen in die Stadt"
3110 S(E)=S(E)+Z
3120 U(E)=U(E)+.5
3130 Z%=Z/5
3140 Z=RND(Z%)
3150 IF Z>50Z=50
3160 M(E)=M(E)+Z
3170 N(E)=N(E)+1
3180 Q(E)=Q(E)+2
3190 IFJ(E)<3THEN3240
3200 J!=S(E)/100*(J(E)-2)*(J(E)-2)
3210 J!=RND(J!)
3220 S(E)=S(E)-J!
```

```
3230 PRINTJ!;"Sklaven fliehen vor hartem Ger
icht"
3240 GOTO3350
3250 X=(D!-G!)/D!*100-9
3260 X%=X
3270 IFX>65THENX=65:M(E)=M(E)/2
3280 IFX<0X%=0:X=0
3290 A=3
3300 GOSUB1230
3310 A=X%+8
3320 GOSUB1280
3330 IFZ%>1000THENU(E)=U(E)/2
3340 GOTO3190
3350 Z=A(E)*75
3360 K(E)=K(E)+Z
3370 IFZ>0PRINT"Dein Markt brachte";Z;"Gulde
n Pacht ein"
3380 IFS(E)<32766S!=S(E):GOSUB1190:S(E)=S!
3390 Z=D(E)*(55+RND(250))
3400 IFZ>0K(E)=K(E)+Z:PRINT"Deine Wollfabrik
 hatte";Z;"Gulden Gewinn zu verzeichnen"
3410 Z=P(E)*3
3420 PRINT"Du hast Deinen Soldaten";Z;"Gulde
n Sold ausbezahlt"
3430 K(E)=K(E)-Z
3440 IF(L(E)/1000)>P(E)THEN5970
3450 IF(L(E)/500)<P(E)THEN3500
3460 FORA=1TOF
3470 IFA=ETHEN3490
3480 IFP(A)>(P(E)*2.4)THEN5970
3490 NEXT
3500 INPUT"(Druecke New Line)";A$
3510 RETURN
3520 CLS
```

```
3530 L%=(L(E)/1000)
3540 IFL%<10X=80:Y=27:GOTO3620
3550 IFL%<30X=80:Y=27-(L%-10):GOTO3620
3560 IFL%<50X=60:Y=27-(L%-30):GOTO3620
3570 IFL%<70X=40:Y=27-(L%-50):GOTO3620
3580 IFL%<90X=20:Y=27-(L%-70):GOTO3620
3590 IFL%<110X=1:Y=27-(L%-90):GOTO3620
3600 X=1
3610 Y=7
3620 FORZ=XTO127
3630 SET(Z,Y)
3640 NEXTZ
3650 FORZ=YTO47
3660 SET(X,Z)
3670 NEXTZ
3680 IF(P(E)-5)<(L(E)/1000)THEN3940
3690 FORA=X+1TOX+6
3700 FORB=Y+1TOY+5
3710 SET(A,B)
3720 NEXTB
3730 NEXTA
3740 SET(X,Y-1)
3750 SET(X+2,Y-1)
3760 SET(X+4,Y-1)
3770 SET(X+6,Y-1)
3780 IF(P(E)/2)<(L(E)/1000)THEN3940
3790 FORA=X+7TOX+10
3800 FORB=Y+1TOY+5
3810 SET(A,B)
3820 NEXTB
3830 NEXTA
3840 SET(X+8,Y-1)
3850 SET(X+10,Y-1)
3860 RESET(X+3,Y+2)
```

```
3870 RESET(X+7,Y+4)
3880 SET(X+1,Y-1)
3890 SET(X+9,Y-1)
3900 SET(X,Y-2)
3910 SET(X+2,Y-2)
3920 SET(X+8,Y-2)
3930 SET(X+10,Y-2)
3940 Z=C(E)+1
3950 IFZ>7THENZ=7
3960 ONZGOTO4310,4240,4170,4120,4070,4020,39
70
3970 FORA=96TO110
3980 SET(A,30)
3990 NEXTA
4000 RESET(102,30)
4010 RESET(104,30)
4020 FORA=96TO99
4030 FORB=24TO29
4040 SET(A,B)
4050 NEXTB
4060 NEXTA
4070 FORA=107TO110
4080 FORB=24TO29
4090 SET(A,B)
4100 NEXTB
4110 NEXTA
4120 FORB=22TO24
4130 SET(103,B)
4140 NEXTB
4150 SET(102,23)
4160 SET(104,23)
4170 FORA=101TO105
4180 FORB=25TO26
4190 SET(A,B)
```

```
4200 NEXTB
4210 NEXTA
4220 RESET(101,25)
4230 RESET(105,25)
4240 FORA=100TO106
4250 FORB=27TO29
4260 SET(A,B)
4270 NEXTB
4280 NEXTA
4290 RESET(102,29)
4300 RESET(104,29)
4310 Z=B(E)*2
4320 IFZ=0THEN4440
4330 IFZ>10SET(80,33):SET(82,33):SET(96,33):
SET(98,33):SET(81,32):SET(97,32)
4340 IFZ>8Z=9:FORA=87TO91:FORB=31TO33:SET(A,
B):NEXTB:NEXTA:RESET(88,33):RESET(90,32):SET
(88,30):SET(90,30):SET(89,29)
4350 FORA=(89-Z)TO(89+Z)
4360 FORB=34TO36
4370 SET(A,B)
4380 NEXTB
4390 NEXTA
4400 FORA=(90-Z)TO(90+Z)STEP2
4410 RESET(A,35)
4420 NEXTA
4430 RESET(89,36)
4440 Z=S(E)-D(E)*100
4450 IFZ<1THENZ=1
4460 Z=Z*5/L(E)*10+1
4470 IFZ>10THENZ=10
4480 Z=(Z/10)*(45-Y)
4490 Z=INT(47-Z)
4500 FORA=119TO127
```

```
4510 SET(A,Z)
4520 NEXTA
4530 RESET(122,Z)
4540 RESET(123,Z)
4550 RESET(125,Z)
4560 FORA=119TO127STEP2
4570 SET(A,Z+1)
4580 NEXTA
4590 SET(118,Z-1)
4600 SET(127,Z-1)
4610 Z=A(E)*2
4620 IFZ=0THEN4700
4630 IF Z>((126-X)-2)Z=((126-X)-2)
4640 FORA=XTOX+ZSTEP2
4650 SET(A,39)
4660 SET(A+1,39)
4670 SET(A+1,40)
4680 SET(A+1,41)
4690 NEXTA
4700 Z=D(E)
4710 IFZ=0THEN4810
4720 IF Z>(126-X)Z=126-X
4730 FORA=126-ZTO127
4740 FORB=45TO47
4750 SET(A,B)
4760 NEXTB
4770 NEXTA
4780 FORA=127-ZTO126STEP2
4790 RESET(A,46)
4800 NEXTA
4810 PRINT@644,"Jahr";
4820 PRINT@707,Y(0);
4830 PRINT@0,T$(E);N$(E);"    ";
4840 PRINT"(Druecke eine Taste)"
```

```
4850 GOSUB6170
4860 RETURN
4870 CLS
4880 PRINT
4890 PRINTT$(E);N$(E)
4900 PRINT"Staatsanschaffungen"
4910 PRINT
4920 PRINT"1. Marktplatz                    1000
Gulden"
4930 PRINT"2. Wollfabrik                    2000
Gulden"
4940 PRINT"3. Palast (Teil)                 3000
Gulden"
4950 PRINT"4. Kathedrale  (Teil)            5000
Gulden"
4960 PRINT"5. Tausch, 20 Soldaten fuer 20 Sk
laven   500 Gulden"
4970 PRINT
4980 PRINT"Du hast ";K(E);"Gulden"
4990 PRINT
5000 PRINT"Tippe: 0 zur Fortsetzung, 6 zum V
ergleich der Staende"
5010 PRINT
5020 PRINT"Deine Wahl ?";
5030 GOSUB6170
5040 I=VAL(A$)
5050 CLS
5060 IFI<1RETURN
5070 IFI>5GOSUB860:GOTO4870
5080 ONIGOTO5130,5090,5180,5230,5280
5090 D(E)=D(E)+1
5100 K(E)=K(E)-2000
5110 U(E)=U(E)+.25
5120 GOTO4870
```

```
5130 A(E)=A(E)+1
5140 M(E)=M(E)+5
5150 K(E)=K(E)-1000
5160 U(E)=U(E)+.1
5170 GOTO4870
5180 B(E)=B(E)+1
5190 N(E)=N(E)+RND(2)
5200 K(E)=K(E)-3000
5210 U(E)=U(E)+.5
5220 GOTO4870
5230 C(E)=C(E)+1
5240 Q(E)=Q(E)+RND(6)
5250 K(E)=K(E)-5000
5260 U(E)=U(E)+1
5270 GOTO4870
5280 P(E)=P(E)+20
5290 S(E)=S(E)-20
5300 K(E)=K(E)-500
5310 GOTO4870
5320 Z=0
5330 A=A(E)
5340 GOSUB5690
5350 A=B(E)
5360 GOSUB5690
5370 A=C(E)
5380 GOSUB5690
5390 A=D(E)
5400 GOSUB5690
5410 A=K(E)/5000
5420 GOSUB5690
5430 A=L(E)/6000
5440 GOSUB5690
5450 A=M(E)/50
5460 GOSUB5690
```

```
5470 A=N(E)/5
5480 GOSUB5690
5490 A=P(E)/50
5500 GOSUB5690
5510 A=Q(E)/10
5520 GOSUB5690
5530 A=S(E)/2000
5540 GOSUB5690
5550 A=U(E)/5
5560 GOSUB5690
5570 A=Z/U(0)-J(E)+1
5580 A=INT(A)
5590 IF A>8A=8
5600 IF(Y(0)+2)=O(E)T(E)=T(E)+1
5610 IFT(E)>=ATHEN5680
5620 T(E)=A
5630 RESTORE
5640 FORB=1TO(T(E)+V(E))
5650 READT$(E)
5660 NEXT
5670 IFT(E)=8THEN5730
5680 RETURN
5690 IFA>10:A=10
5700 A=INT(A)
5710 Z=Z+A
5720 RETURN
5730 CLS
5740 PRINT
5750 PRINT"Das Spiel ist vorbei ";T$(E);N$(E
)
5760 PRINT"hat gewonnen"
5770 GOSUB3530
5780 GOSUB860
5790 GOTO6420
```

```
5800 CLS
5810 PRINT
5820 PRINTT$(E);N$(E);" ist bankrott"
5830 PRINT
5840 PRINT"Glaeubiger haben das meiste Deine
s Eigentums beschlagnahmt"
5850 PRINT
5860 INPUT"(Druecke New Line)";A$
5870 A(E)=0
5880 B(E)=0
5890 C(E)=0
5900 D(E)=0
5910 L(E)=6000
5920 U(E)=1
5930 K(E)=100
5940 M(E)=M(E)/2
5950 R(E)=4000
5960 RETURN
5970 Z=0
5980 FORA=1TOF
5990 IFA=ETHEN6030
6000 IFP(A)<P(E)THEN6030
6010 IFP(A)<(1.2*(L(A)/1000))THEN6030
6020 IF P(A)>P(Z)Z=A
6030 NEXT
6040 IFZ=0T$(0)=" Baron ":N$(0)="Peppone von
 Monterana hat":A!=RND(9000)+1000:GOTO6060
6050 A!=P(Z)*1000-L(Z)/3
6060 IF A!>(L(E)-5000)A!=(L(E)-5000)/2
6070 PRINTT$(Z);N$(Z);" angegriffen und besc
hlagnahmt";A!
6080 PRINT"Hektar Land!"
6090 L(Z)=L(Z)+A!
6100 L(E)=L(E)-A!
```

```
6110 Z=RND(40)
6120 IF Z>(P(E)-15)Z=P(E)-15
6130 PRINTT$(E);N$(E);Z;" Soldaten in der Sc
hlacht"CHR$(10)"gefallen"
6140 P(E)=P(E)-Z
6150 INPUT"(Druecke New Line)";A$
6160 RETURN
6170 A$=INKEY$
6180 IFA$=""THEN6170
6190 RETURN
6200 CLS
6210 PRINT"Santa Paravia und Fuimaccio"
6220 PRINT" Du regierst einen italienischen
Stadtstaat aus dem 15.Jht."
6230 PRINT"Regierst Du gut, so bekommst Du h
oehere Titel. Der 1. Spieler"
6240 PRINT"der Koenig/in wird hat gewonnen.
Regierst Du schlecht so wirst"
6250 PRINT"Du nicht lange genug leben um zu
gewinnen."
6260 PRINT"Der Computer zeichnet Dir eine Ka
rte des Reiches. Die Groesse"
6270 PRINT"des Gebietes innerhalb der Mauer
waechst , wenn Du mehr Land"
6280 PRINT"kaufst. Die Groesse des Turms lin
ks oben zeigt Dir die Ver-"
6290 PRINT"teidigungsfaehigkeit an. Wenn er
schrumpft brauchst Du mehr "
6300 PRINT"Soldaten. Wenn das Pferd und der
Landarbeiter die obere Mauer"
6310 PRINT"erreichen, wird Dein ganzes Land
genutzt. Anderenfalls brauchst"
6320 PRINT"Du mehr Sklaven, die in Dein Land
 kommen wenn Du mehr Getreide"
```

```
6330 PRINT"als notwendig anbaust. Baust Du z
u wenig Getreide an, sterben"
6340 PRINT"die Einwohner. Hohe Steuern bring
en Geld ein, ver-"
6350 PRINT"schlechtern aber das wirtschaftli
che Wachstum."
6360 PRINT"(Druecke <New Line> um zu beginne
n)";
6370 INPUTA$
6380 CLS
6390 RETURN
6400 GOSUB3520
6410 GOSUB860
6420 PRINT
6430 PRINT"Das Spiel ist vorbei, <New Line>
fuer neues Spiel ";
6440 INPUTA$
6450 GOTO10
```

Gleichungen mit 3 Unbekannten:

Dieses Mathematikprogramm berechnet die 3 Unbekannten von 3 Gleichungen der Form AX+BY+CZ=D.

Sollte keine Lösung des eingegeben Gleichungssystems existieren, führt dies zu einem "Division by zero Error".

```
10 CLS
20 PRINT"DIESES PROGRAMM BERECHNET AUS DREI
GLEICHUNGEN DER FORM"
30 PRINT"AX+BY+CZ=D (A,B,C,D GEGEBEN) DIE UN
BEKANNTEN X,Y UND Z"
40 PRINT
50 INPUT"a1,b1,c1,d1";A1,B1,C1,D1
60 INPUT"a2,b2,c2,d2";A2,B2,C2,D2
70 INPUT"a3,b3,c3,d3";A3,B3,C3,D3
80 D=A1*B2*C3+B1*C2*A3+C1*A2*B3-A3*B2*C1-B3*
C2*A1-C3*A2*B1
90 DX=D1*B2*C3+B1*C2*D3+C1*D2*B3-D3*B2*C1-B3
*C2*D1-C3*D2*B1
100 DY=A1*D2*C3+D1*C2*A3+C1*A2*D3-A3*D2*C1-D
3*C2*A1-C3*A2*D1
110 DZ=A1*B2*D3+B1*D2*A3+D1*A2*B3-A3*B2*D1-B
3*D2*A1-D3*A2*B1
120 PRINT"X="DX/D
130 PRINT"Y="DY/D
140 PRINT"Z="DZ/D
```

Tilgungsplan:

Das folgende Programm hilft Ihnen bei der Berechnung der Abzahlung einer Ratenschuld. Dabei kann man wählen, ob die Schuld mit konstanter Annuität

(Gesamtbelastung=Zinsen+Tilgung)

oder mit konstanter Tilgung erfolgen soll.

```
10 CLS
20 PRINT"T I L G U N G S P L A N   EINER RAT
ENSCHULD"
30 PRINT
40 CLEAR
50 INPUT"KONSTANTE ANNUITAET (1), KONSTANTE
TILGUNG (2)";B
60 ON B GOTO 70, 220
70 INPUT "ANNUITAET, ZINS, KAPITAL EINGEBEN"
;A, P, K#
80 PRINT
90 PRINT"ZEIT";TAB(13)"ZINS";TAB(26)"TILGUNG
";TAB(39)"RESTSCHULD"
100 K1#=K#
110 X=X+1
120 ZI=INT(K#*P+.5)/100
130 T=A-ZI
140 IF T>=K#THEN T=K#
150 R#=K#-T
160 D$="###      #######.##      #######.##
  #######.##"
170 PRINT USING D$; X; ZI; T; R#
180 IF T <=0 THEN PRINT"EINE TILGUNG IST NIC
HT MOEGLICH. DIE ZINSEN BETRAGEN";ZI;"DM UND
 SIND GROESSER ALS DIE TILGUNG VON";T;"DM":
GOTO 490
190 IF R#<=0 THEN PRINT"IN";X;"ZEITPERIODEN
IST DIE SCHULD VON";K1#;"DM GETILGT": GOTO 4
90
200 K#=R#
210 GOTO 110
220 DIM Z(1000)
230 A$="###  #######.##  ######.##  #######.
##  #######.##"
```

```
240 B$="K=#######.##   N=###   P=###.###"
250 PRINT
260 INPUT"BITTE KAPITAL,ZEIT,ZINSSATZ EINGEB
EN";K, N, P
270 K1=K
280 PRINT
290 PRINT
300 PRINT
310 PRINT USING B$; K; N; P
320 PRINT"ZEIT";TAB(7)"ZINSEN";TAB(18)"TILGU
NG";TAB(28)"ANNUITAET" TAB(39)"RESTSCHULD"
330 PRINT
340 T=INT(100*K/N+.5)/100
350 FOR I=1 TO(N-1)
360 IF I=11 OR I=21 OR I=31 OR I=41 OR I=51
OR I=61  THEN INPUT "WEITER";C$
370 Z=INT(K*P+.5)/100
380 PRINT USING A$; I; Z; T; T+Z; K-T
390 K=K-T
400 Z(I)=Z
410 NEXT I
420 U=INT(K*P+.5)/100
430 Z(N)=U
440 PRINT USING A$; N; U; K; U+K; 0
450 FOR I=1 TO N
460 Z1=Z1+Z(I)
470 NEXT I
480 PRINT"GESAMTE ZAHLUNGEN=";Z1+K1
490 INPUT "ANDERE BERECHNUNG (J/N)";C$
500 IF LEFT$(C$, 1)="J" OR LEFT$(C$, 1)="j"
THEN 10
510 END
```