robotron

Programmierhandbuch

Teil 1

VEB Robotron-Meßelektronik
»Otto Schön« Dresden

Inhaltsverzeichnis

1. Einleitung

Das Programmierhandbuch des Heimcomputers robotron Z 9001 enthält eine ausführliche Beschreibung der Programmiersprache BASIC, sowie Hinweise und Informationen zum Betriebsystem und weiteren Programmiermöglichkeiten.
Wie Sie sicher bereits wissen, ist Ihr Computer sehr schnell und zuverlässig bei der Ausführung Ihrer Instruktionen, kann aber ohne diese gar nichts für Sie tun. Ihr Wunsch besteht also darin, sich auf möglichst einfache und effektive Weise mit Ihrem Heimcomputer über dessen Aufgaben zu verständigen. Das betrifft sowohl sofort ausführbare Anweisungen und Kommandos zur Beeinflussung der Programmabarbeitung als auch die Formulierung einer wiederholt nutzbaren Problemlösung, also eines Programms.
Beides gestattet die dialogorientierte Programmiersprache BASIC auf komfortable Weise, deshalb ist sie im internationalen Maßstab die Programmiersprache für Heimcomputer.
Mit Hilfe der Programmiersprache BASIC des Heimcomputers robotron Z 900 können sowohl einfache mathematische Operationen sofort ausgeführt, als auch Programme eingegeben, abgearbeitet und auf Magnetbandkassette gespeichert werden. Auch die Ausgabe von Programmen, Rechenergebnissen und anderen Informationen auf einen Drucker ist möglich. Ebenso können die heimcomputer-spezifischen Ergänzungsmodule (Farbmodul, Spielhebel u. a.) sowie der für die grafische Gestaltung bedeutsam Grafik-Zeichensatz (128 spezielle Grafikzeichen) vom BASIC des „robotron Z 9001“ einfach angesprochen werden. Außerdem ist es möglich, Informationen von angeschlossenen Meßgeräten abzurufen und weiter zu verarbeiten und auch Steuersignale an entsprechende Geräte zu übertragen.
Das BASIC des „robotron Z 9001“ ist in seinem Aufbau (Anweisungs- und Kommandovorrat, syntaktische Struktur) dem international weitverbreiteten Standard angeglichen, so daß eine Übertragung von BASIC-Programmen vieler anderer Rechner auf den „robotron Z 9001“ in der Regel leichtfallen wird.
Für Ihren Heimcomputer wird der BASIC-Interpreter in zwei Varianten angeboten. In der Grundvariante ist er als **RAM-BASIC** von einer Magnetbandkassette, der „Grundkassette“, in den RAM-Speicher (Arbeitsspeicher) des Rechners zu laden. Außerdem wird er als **ROM-BASIC** im sogenannten BASIC-Modul angeboten. Die Nutzung dieser Version bringt den Vorteil, daß das BASIC nach dem Einschalten des Rechners sofort

verfügbar ist und dann außerdem wesentlich mehr Speicherplatz (insgesamt etwa 15 kbytes) für Anwenderprogramme zur Verfügung steht. Sie können mit dem „robotron Z 9001“ zahlreiche käuflich erwerbbare BASIC-Anwenderprogramme unterschiedlichsten Inhalts abarbeiten. Diese Programme sind auf Magnetbandkassetten gespeichert und müssen von Ihnen mit Hilfe eines Kassettenrecorders in den Heimcomputer geladen (eingelesen) werden. Wollen Sie ein **Programm** selbst schreiben (programmieren) und dann abarbeiten, so gehört dazu einerseits, daß Sie die für die Realisierung des Programms erforderlichen **Anweisungen** in der richtigen Reihenfolge in den Rechner eingeben (laden). Andererseits sind zum Test, zur Abarbeitung und auch zur Modifizierung eines Programms eine ganze Anzahl von **Kommandos** notwendig.

In der Dialogsprache BASIC sind, im Gegensatz zu anderen Programmiersprachen, wie FORTRAN, PL/1 oder PASCAL, neben den notwendigen Anweisungen auch solche (Steuer-) Kommandos enthalten.

Dementsprechend kennt der BASIC-Interpreter auch zwei typische Arbeitszustände:

- den **Kommando-Modus**, in dem die Tastatureingabe von Kommandos durch den Nutzer erfolgt. Die Ausführung dieser Kommandos ist im wesentlichen Voraussetzung für die Programmabarbeitung bzw. beeinflußt diese (Programm starten, unterbrechen, fortsetzen, speichern, laden, anzeigen, ändern, ...).
- den **Programm-Modus**, innerhalb dessen die Abarbeitung von Anweisungen des gestarteten BASIC-Programms stattfindet. Diese Anweisungen dienen der eigentlichen Problemlösung und werden interpretierend abgearbeitet, d. h. in vorgegebener Reihenfolge einzeln analysiert und unmittelbar ausgeführt.

Wie Sie bereits wissen, ist BASIC in seinen Grundelementen sehr einfach gestaltet, so daß auch diejenigen, die mit der Rechentechnik noch nicht vertraut sind, schon nach kurzer Zeit eigene Programme schreiben können. Trotzdem werden Sie besonders in der ersten Zeit noch viele Fehler machen. Das sollte Sie jedoch nicht entmutigen. Außerdem kann der Rechner durch Fehlbedienung nicht beschädigt werden. Er weist falsche Eingaben in der Regel durch Fehlermeldungen selbständig zurück.

Der in das BASIC des „robotron Z 9001“ einführende Abschnitt 3 vermittelt Grundkenntnisse, die insbesondere für die Nutzung käuflich erworbener Anwenderprogramme erforderlich sind. im Abschnitt 4 werden Sie dann systematisch durch das BASIC Ihres Heimcomputers geführt, so daß Sie nach und nach in der Lage sein werden, auch kompliziertere BASIC-Programme selbst zu erarbeiten.

Sollten Sie bereits über weitergehende Kenntnisse in der Mikrorechentechnik und in der Programmierung verfügen, so können Sie Ihren Heimcomputer auch in **Assembler** oder in der **K-1520-Maschinensprache** programmieren. Dazu müssen Sie jedoch die entsprechenden Assembler-Kassetten bzw. -Module erwerben.

Reicht Ihnen der freie Speicherplatz Ihres Heimcomputers für größere Programme nicht mehr aus, so können Sie durch einen oder zwei **RAM-Erweiterungsmodule** jeweils 16 kbyte Speicherplatz zusätzlich stecken (siehe auch Abschnitt 5 der Bedienungsanleitung).

Falls Sie ihren Heimcomputer mit Hilfe der Bedienungsanleitung bereits ordnungsgemäß angeschlossen und eingeschaltet haben, so können Sie nun beginnen, ihre ersten Handlungen am Computer auszuführen.

2. Tastatur

Nachdem Sie Ihren Heimcomputer gemäß Abschnitt 3.6 der Bedienungsanleitung in Betrieb genommen haben, befindet sich sein Betriebssystem zunächst im Grundzustand, und auf dem Bildschirm erscheint:

In diesem Zustand des „robotron Z 9001“ sind nur wenige Bedienhandlungen möglich, die komplett im Abschnitt 7 beschrieben werden. insbesondere können an dieser Stelle noch keine Rechenoperationen ausgeführt werden. Dazu müssen Sie erst den BASIC-Interpreter starten. Das dafür erforderliche Kommando übermitteln Sie dem Rechner über die Tastatur, die zunächst gesondert betrachtet werden soll.
Die prinzipielle Funktion der Tasten, insbesondere der Umschalttasten [SHIFT], [SHIFT LOCK] sowie der unbeschrifteten Leertasten, wurde bereits in der Bedienungsanleitung (Abschnitt 4.2 und Bild 5) erläutert.
Zur besseren Übersicht ist nachstehend die Tastatur nochmals abgebildet:

Am besten, Sie probieren die Grundfunktion wichtiger Tasten gleich aus. Das ist auch im Grundzustand des Rechners möglich. Drücken Sie also, nacheinander einige Buchstabentasten, und lassen Sie dann den Finger so lange auf einer Taste, z. B. dem Buchstaben "Z", bis etwa eineinhalb Zeilen am Bildschirm beschrieben sind. Danach brechen Sie die Eingabe mit der [STOP]-Taste ab. Nun könnte etwa folgendes Bild sichtbar sein:

```
robotron Z 9001
OS
>ABCD  robotron Z 9001  ABCD  1234567890
ZZZZZZZZZ
OS
>
```

Der Rechner geht im Betriebssystem (OS) wieder in die Eingabebereitschaft, die durch das Aufforderungszeichen ">" angezeigt wird. Zur Gewöhnung sollten Sie diese Eingaben mit verschiedenen Tasten, auch unter Nutzung der Tasten [SHIFT] und [SHIFTLOCK] wiederholen.

Weitere Zeichen, die Grafikzeichen, können Sie mit Hilfe der Taste [GRAPHIC] darstellen.

[GRAPHIC]
schaltet den Rechner in den Grafikmodus (GRAPHIC-Anzeige leuchtet). jetzt können Sie mit Hilfe der Tastatur die im Anhang B dargestellten Grafikzeichen eingeben. Durch nochmalige Betätigung von GRAPHIC wird auf Normaleingabe zurückgeschaltet (GRAPHIC-Anzeige verlischt).

Es ist zweckmäßig, wenn Sie sich vor der Arbeit mit dem BASIC-Interpreter noch über die prinzipielle Wirkung einiger wichtiger Funktionstasten informieren.

[RESET]
Durch diese Taste wird der Computer wieder in den Grundzustand des Betriebssystems gebracht. Dabei werden einige interne Informationen gelöscht, und auf dem Bildschirm erscheint wieder

Der Computer ist erneut eingabebereit. Probieren Sie es aus!

[ENTER]
Der Computer wird angewiesen, die vorher eingetippte Information zu übernehmen und zu verarbeiten. Mit dieser Taste werden im Normalfall alle Eingaben abgeschlossen.

[STOP]
Die eingegebene Information wird nicht übernommen und abgearbeitet. Ein laufendes Programm wird unterbrochen bzw. abgebrochen.
[←] [→]
Diese Kursortasten dienen der Bewegung des Kursors auf einer Zeile. Im Betriebssystemmodus wird durch ← gleichzeitig das Zeichen vor dem Kursor gelöscht, im BASIC ist das nicht der Fall.

[CONTR]
Diese Taste bewirkt in Verbindung mit anderen Tasten die Ausführung bestimmter Sonderfunktionen, von denen viele nur im BASIC bzw. bei vorhandenem Farbmodul sinnvoll sind. Die vollständige Liste dieser Sonderfunktionen (Steuerzeichen) finden Sie im Anhang A.

Wenn Sie Interesse haben, können Sie z. B. die Funktionen

[CONTRL] [L] - Löschen des Bildschirmes (auch "■" verschwindet!)

[CONTRL] [Q] - Tonausgabe bei Tastenbetätigung bzw. Bildschirmanzeige (wird durch nochmaliges Drücken von [CONTRL] [Q] wieder abgestellt) sofort, d. h. auch im Betriebssystemmodus, ausprobieren. Beachten Sie, daß [CONTRL] **gleichzeitig** mit der entsprechenden Taste gedrückt werden muß.

[COLOR]
Diese Taste wirkt nur nach Erweiterung des Grundgerätes "robotron Z 9001.10" auf Farbwiedergabe bzw. bei der Variante "robotron Z 9001.11". Nach COLOR ist stets eine der Tasten 1 bis 8 zu drücken (Näheres siehe Abschnitt 4.16).

[ESC]
Diese Taste kann in Programmen zur Ausführung besonderer Funktionen genutzt werden. Sie muß dabei über ihre Codierung (vgl. Abschnitt 4.14 und Anhang A) ausgewertet werden.

[↑] [↓]
Diese Tasten können in Programmen über ihre Codierungen genutzt werden, um z. B. Positionierungen in vertikaler Richtung vorzunehmen. Im Normalfall haben die Tasten, außer ↓ im EDIT-Modus des BASIC, keine Funktion.

[LIST] [RUN] [PAUSE] [CONT] [INS] [DEL] [CL LN] [I←] [→I]
Diese Tasten sind nur wirksam, wenn Sie im BASIC arbeiten. Eine genaue Funktionsbeschreibung, finden Sie in den Abschnitten 3 und 4.

3. Was Sie über BASIC wissen müssen

Der BASIC-Interpreter ist für den "robotron Z 9001' in zwei Varianten verfügbar, als RAM-BASIC und als ROM-BASIC. Die Nutzung beider Varianten unterscheidet sich nur beim Starten des BASIC-Interpreters nach dem Einschalten des Gerätes sowie im Speicherplatz, der anschließend für den Anwender verfügbar ist.

3.1. Starten des BASIC-Interpreters

RAM-BASIC
Wollen Sie den BASIC-Interpreter von der Grundkassette laden, so bringen Sie den Computer zunächst in den Grundzustand des Betriebssystems. Dieser wird unmittelbar nach dem Einschalten des Gerätes oder nach [RESET] erreicht. Nun kann das RAM-BASIC durch folgende Handlungen von der Grundkassette in den Heimcomputer geladen werden:

1. Legen Sie die Grundkassette in das Kassettenmagnetbandgerät ein, positionieren Sie das Band vor den Anfang des BASIC-Interpreters (5-Sekunden-Vorton beachten!) und halten Sie es dann an.
   Empfehlung:
   Vor dem ersten Ladevorgang sollten Sie sich ruhig den Anfang des BASIC-Interpreters über den Lautsprecher des Kassettengerätes "anhören", damit Sie den Vorton und die kurzen Zwischentöne deutlich unterscheiden lernen.
2. Geben Sie über Tastatur "BASIC" ein, und drücken Sie anschließend die [ENTER]-Taste:

   ```
   BASIC [ENTER]
   ```

   Der Bildschirm hat dann folgendes Aussehen:

   ```
   robotron Z 9001

   OS
   >BASIC

   start tape
   ■
   ```

3. Starten Sie jetzt die Magnetbandkassette durch Drücken der **Wiedergabetaste**.

4. Betätigen Sie die [ENTER]-Taste am Heimcomputer, spätestens beim Ertönen des Vortones.
   Der Heimcomputer beginnt nun, den BASIC-Interpreter einzulesen (zu „laden“). Diesen Vorgang können Sie akustisch und optisch verfolgen. Der Kursor am Bildschirm muß nach jedem kurzen Zwischenton um eine Position nach rechts springen. In dieser Zeit werden 128 Zeichen (1 Record) in den Rechner eingelesen und geprüft. Auf dem Bildschirm haben Sie etwa folgendes Bild:

   ```
   robotron Z 9001

   OS
   >BASIC

   start tape
                  ■
   ```
   ↑ Cursor bewegt sich nach rechts

Das Einlesen des BASIC-Interpreters dauert nicht ganz 2 Minuten. Nach erfolgreichem Einiesen meldet sich der BASIC-Interpreter mit:

```
HC-BASIC
MEMORY SIZE? :
```

Sie können nun das Kassettengerät ausschalten.

Achtung!
Sollte der Einlesevorgang durch Fehlermeldungen der Art
BOS-error: ...
unterbrochen werden, so drücken Sie zunächst die [STOP]-Taste und informieren sich dann im Anhang H des Programmierhandbuches bzw. im Anhang 3 der Bedienungsanleitung über die möglichen Ursachen bzw. Korrekturhandlungen.

Auf die Frage nach der Größe des Speicherplatzes

```
MEMORY SIZE? :■
```

reagieren Sie im Normalfall nur mit [ENTER]. (Falls Sie auch mit Maschinenprogrammen arbeiten wollen, sind hier andere Eingaben erforderlich. Näheres dazu finden Sie im Abschnitt 5.3)
Nach [ENTER] bekommen Sie den aktuellen, für ihre Programme und Daten verfügbaren, Speicherplatz in Bytes (in einem Byte kann jeweils ein Zeichen gespeichert werden) angezeigt.

Der BASIC-Interpreter geht danach in den Kommandomodus über und wartet auf Ihre Eingaben.

```
HC-BASIC

MEMORY SIZE? :
4846 BYTES FREE

OK
>■
```

Achtung!
Die vollständige Abarbeitung von Kommandos und Programmen wird im Kommandomodus des BASIC-Interpreters stets durch "OK" angezeigt. Nach dem Aufforderungszeichen ">" ist dann die Eingabe neuer Kommandos oder Anweisungen möglich.

ROM-BASIC
Wenn Sie den BASIC-Modul gesteckt haben, so müssen Sie nach dem Einschalten des Heimcomputers nur

```
BASIC [ENTER]
```

eingeben. Der Interpreter meldet sich dann sofort mit

```
HC-BASIC

MEMORY SIZE? :■
```

Drücken Sie nun wieder [ENTER], so ist der BASIC-Interpreter arbeitsbereit.

Verlassen, Abbruch und Neustart des BASIC
Die Arbeit im BASIC kann auf zweierlei Arten abgebrochen werden. Durch Eingabe des Kommandos

```
BYE [ENTER]
```

gelangen Sie wieder in den Betriebssystemmodus (OS-Modus).

```
BYE

OS
>■
```


Andererseits können Sie durch [RESET] den Grundzustand des Betriebssystems wiederherstellen. Diese Möglichkeit sollte aber nur genutzt werden, wenn sich der Rechner durch andere Tasten (auch [STOP] bzw. [ENTER]) nicht mehr bedienen läßt bzw. auf keinerlei Eingaben reagiert. Wollen Sie nach dem Abbruch der Arbeit im BASIC wieder neu beginnen, so können Sie das für RAM- und ROM-BASIC in gleicher Weise durch

```
WBASIC [ENTER]
```

In diesem Fall bleiben vorher vorhandene BASIC-Programme erhalten. Der BASIC-Interpreter meldet sich nur kurz durch

```
OK
>■
```

Natürlich können Sie auch wieder durch

```
BASIC [ENTER]
```

starten, wobei dann die vorher vorhandenen Programme gelöscht werden und das BASIC-Anfangsbild wieder erscheint. Probieren Sie diese Möglichkeiten ruhig aus, damit Sie mit dem Rechner und dessen Reaktionen vertraut werden.

3.2. Sofort ausführbare Anweisungen

Nach dem Starten des BASIC-Interpreters können Sie bereits viele einfache Anweisungen (Rechenoperationen) ausführen, unter anderem auch die mathematischen Grundoperationen, etwa im Bereich eines wissenschaftlichen Taschenrechners. Man spricht deshalb gelegentlich auch vom „Taschenrechnermodus“ des Heimcomputers.

Numerische Operationen
Wollen Sie z. B. die Aufgabe

235 + 117

lösen, so geben Sie nach dem Aufforderungszeichen zeichenweise über die Tastatur ein

```
PRINT 235+117
```

und drücken danach die [ENTER]-Taste. Auf dem Bildschirm sind dann Aufgabe und Ergebnis in folgender Weise dargestellt:

```
>PRINT 235+117
 352
OK
>■
```

Durch „OK“ wird dabei wieder die vollständige Abarbeitung der Rechenoperation quittiert. Das Aufforderungszeichen „>“ signalisiert die erneute Eingabebereitschaft des Computers. Ab der Position des Kursors ■ können Sie eine neue Aufgabe eingeben.
Damit das Ergebnis auch angezeigt wird, muß jeweils das Schlüsselwort **PRINT** („Drucke“) vorangestellt werden. Zur Vereinfachung der Eingabe kann das häufig benötigte **PRINT** auch durch ein Fragezeichen („?“) abgekürzt werden. Geben Sie z. B. ein:

```
?768-374 [ENTER]
```

Zur Ausführung solcher einfachen Rechenoperationen stehen Ihnen die Operationszeichen

+	für Addition
-	für Subtraktion
*	für Multiplikation
/	für Division
^	für Potenzierung

zur Verfügung. Weiterhin können Sie einige mathematische Standardfunktionen (siehe auch Abschnitt 4.2) wie z. B.

SQR(X)	- Quadratwurzel
SIN(X)	- Sinus (X im Bogenmaß)
LN(X)	- Natürlicher Logarithmus (X > 0)
PI	- Konstante π

nutzen. Wenn Sie zur Übung einige Aufgaben lösen wollen, so beachten Sie bitte, daß jede Anweisung durch [ENTER] abgeschlossen werden muß. Auf dem Bildschirm ergibt sich z. B.

```
>?SIN(PI*45/180)
 .707107
OK
>?3^9.5
 34092
OK
>?0.5*LN(10)/(5-3*5)
-.115129
OK
```



```
>?7*7*PI
 153.938
OK
>■
```

Achtung!

1. Bei nichtganzzahligen Werten muß stets ein Dezimalpunkt eingegeben werden, kein Komma. Das entspricht dem internationalen Standard der Rechentechnik.
2. Zur Unterscheidung von dem Buchstaben 0 wird die Ziffer 0 (Null) auf dem Bildschirm und auf der Tastatur durchgestrichen dargestellt.

Zeichenketten

Neben Zahlen kann Ihr Heimcomputer auch Zeichenketten verarbeiten. Zeichenketten sind Folgen von Buchstaben, Ziffern, Sonderzeichen oder Grafikzeichen, die im allgemeinen durch Anführungszeichen ("...") begrenzt werden. Sie werden zur Textspeicherung und -darstellung benötigt. Geben sie z. B. ein:

```
PRINT "robotron" [ENTER]
```

Der Rechner verarbeitet dann die Zeichenfolge **robotron** als Zeichenkette und gibt diese wieder am Bildschirm aus.

Eingabekorrekturen

Wenn Sie sich bei der Eingabe der Anweisungen und Kommandos in den Rechner gelegentlich vertippen, können Sie mit Hilfe der Kursortasten [←] und [→] den Kursor auf der Eingabezeile bewegen und die falschen Eingaben durch richtige überschreiben (ersetzen). Probieren Sie diese Möglichkeiten anhand selbstgewählter Beispiele aus, und vergessen Sie nicht, daß der Rechner Ihre Aufgaben erst dann abarbeitet, wenn Sie [ENTER] gedrückt haben.

Achtung!

1. Wenn Sie eine Anweisung fehlerhaft eingeben und [ENTER] drücken, reagiert der Rechner mit der Meldung „?SN ERROR“ (Syntaxfehler). Sie können dann die Eingabe wiederholen.
2. Nach [STOP] wird die Anweisung nicht ausgeführt, Sie können erneut eingeben.

Noch wesentlich einfacher können Sie korrigieren, wenn Sie die Tasten [INS], [DEL], [CL LN], [I←] und [→I] nutzen. Diese sind besonders dann

nützlich, wenn mehrere Zeichen innerhalb der aktuellen Eingabezeile gestrichen oder hizugefügt werden sollen.

[←] [→]
Diese Tasten bewegen den Kursor auf das erste bzw. das letzte Zeichen der Eingabezeile.

[DEL]
(DELETE-Taste). Mit [DEL] wird das Zeichen, auf dem sich der Kursor befindet, gelöscht.

[INS]
(INSERT-Taste). Mit [INS] werden in die Eingabezeile ab der Kursorposition Leerzeichen eingefügt, die anschließend durch andere Zeichen ersetzt werden können.

[CL LN]
(CLEAR-LINE-Taste). Durch [CL LN] wird die aktuelle Eingabezeile gelöscht. Der Kursor rückt auf den Zeilenanfang. Wenn Sie sich auch mit diesen Korrekturmöglichkeiten vertraut gemacht haben, so können Sie sich nun der Nutzung vorhandener (z. B. käuflich erworbener) BASIC-Programme zuwenden oder auch versuchen, erste kleine Programme selbst zu schreiben.

3.3. Nutzung vorhandener Programme

Für den Heimcomputer robotron Z 9001 werden zahlreiche BASIC-Anwenderprogramme auf Magnetbandkassetten angeboten. Inhaltlich lassen sich diese Programme (Kassetten) in die Rubriken

Lehre und Lernen
Datenverarbeitung
Spiele
Wissenschaft und Technik
Heim und Hobby

einordnen. Alle Programme sind farbig gestaltet, können aber auch in der Schwarzweißausführung des „robotron Z 9001“ gut verwendet werden. Vor der Nutzung solcher robotron-Anwenderprogramme können Sie sich anhand der zur Kassette gelieferten Programmbeschreibung über den wesentlichen Inhalt der Programme und Besonderheiten bei ihrer Abarbeitung (Spielregeln u. ä.) informieren. Weitere Bedienerhinweise erhalten Sie vom Programm auf dem Bildschirm. Ist Ihr Kassettengerät mit einem Bandzählwerk ausgestattet, können Sie sich vor dem ersten Laden der Programme die Zählerstände der Programmanfänge notieren, die jeweils an einem 5-Sekunden-Vorton erkennbar sind.
Wollen Sie ein Programm in den Rechner einlesen, müssen sie darauf achten, daß der im Rechner noch freie Speicherplatz für das Programm ausreicht. Gegebenenfalls können Sie sich durch die Anweisung

```
PRINT FRE(0)
```
[ENTER]

diesen Speicherplatz (in Bytes) anzeigen lassen.

Achtung!
Einige Programme werden vom Hersteller geschützt. Bei diesen Programmen können die Kommandos CSAVE, EDIT und LIST (vgl. Abschnitt 4) nicht abgearbeitet werden.

Laden von BASIC-Programmen
Die auf Magnetbandkassetten gespeicherten BASIC-Anwenderprogramme können durch das Kommando

```
CLOAD "progname"
```

in den Rechner geladen werden. Dabei muß für *progname* der konkrete Name des gewünschten Programmes eingegeben werden, z. B. also

```
CLOAD "R+HANOI"
```

für das Programm R+HANOI auf der Grundkassette R 0111 des "robotron Z 9001". Die für das Laden der BASIC-Programme erforderlichen Handlungen sind im folgenden aufgeführt:
1. Löschen Sie die noch im Rechner gespeicherten Programme durch

```
NEU
```
[ENTER]

2. Legen Sie die Kassette in das Magnetbandkassettengerät ein, und Positionieren Sie das Magnetband vor das zu ladende Programm. (Der Programmanfang ist an einem etwa 5 Sekunden andauernden Vorton zu erkennen.)

3. Geben Sie am Heimcomputer das Kommando

```
CLOAD "progname"
```

z.B. also

```
CLOAD "R+HANOI"
```

ein. Drücken Sie aber noch nicht [ENTER]

4. Starten Sie nun bitte das Magnetband (Wiedergabetaste!).

5. Beim Ertönen des Vortones drücken Sie [ENTER].
Nach dem Einiesen des ersten Datensatzes vom Magnetband zeigt der Computer den Namen des gefundenen Programms am Bildschirm an:

```
>CLOAD "R+HANOI"
  ***R+HANOI      ■
```

Danach rückt der Kursor bei jedem richtig gelesenen Datensatz (Record, je 128 Bytes) um eine Position nach rechts (etwa nach jeweils einer Sekunde). Nach erfolgreich abgeschlossenem Einlesen des gesamten Programms erscheint die Meldung

```
FILE FOUND
```

(d. h. „Programm gefunden“) auf dem Bildschirm, und der BASIC-Interpreter befindet sich wieder im Kommandomodus. Sie können Ihr Kassettengerät abschalten.

Der Bildschirm hat nun etwa folgendes Aussehen:

```
>CLOAD "R+HANOI"
  ***R+HANOI                          FILE FO
UND
OK
>
>■
```


Achtung!
Sollte das Einlesen eines Programms anders als beschrieben verlaufen, z. B. durch eine der Fehlermeldungen

```
BOS-error: . . .
```

unterbrochen werden, so drücken Sie bitte [STOP] und informieren sich dann im Anhang H des Programmierhandbuches bzw. im Anhang 3 der Bedienungsanleitung über die möglichen Fehler bzw. Korrektur-handlungen.

Programmstart
Nach dem Laden des Programms möchten Sie es sicher starten und abarbeiten. Der Start erfolgt am einfachsten durch Drücken der Taste [RUN]. Nach kurzer Zeit sehen Sie auf dem Bildschirm das erste Bild des Programms, welches in der Regel über Namen und Inhalt dieses Programms informiert.

Programmabarbeitung
Bei der Abarbeitung eines Programms sollten Sie die am Bildschirm angezeigten Bedienerhinweise beachten. Die Programme befinden sich dann in einem Wartezustand und fordern zu Eingaben bzw. Bedienhandlungen auf. Im einfachsten Fall muß dabei nur [ENTER] betätigt werden. Alle Eingaben von Zahlen, Buchstaben und Texten sind durch [ENTER] abzuschließen, sofern in der Programmbeschreibung keine anderen Hinweise gegeben werden. Bei der Eingabe von Zahlen ist entsprechend der bei Rechnern üblichen Schreibweise ein Dezimalpunkt statt eines Kommas zu schreiben, z. B.

```
123.45 [ENTER]
```

Sind mehrere Zahlen einzugeben, so sind diese jeweils durch ein Komma zu trennen. Wird eine Zahl fehlerhaft eingegeben bzw. statt einer Zahl eine Zeichenkette (Buchstabenfolge), so kommt es zur Fehlermeldung

```
?REDO FROM START
```

Die Eingabe kann dann wiederholt werden. Normale Texteingaben werden ebenfalls durch [ENTER] abgeschlossen. Sollte man sich bei den geforderten Eingaben vertippen, so ist eine Korrektur möglich, falls noch nicht [ENTER] gedrückt wurde. Im Abschnitt 3.2 wurde bereits beschrieben, wie Sie solche Eingabekorrekturen ausführen können.

Beachten Sie bitte:

1. Bei einigen Programmen werden bestimmte Tasten für spezielle Bedienhandlungen genutzt. Diese müssen dann nicht durch [ENTER] abgeschlossen werden. Genaue Angaben dazu erhalten Sie in der Programmbeschreibung bzw. über den Bildschirm.
2. Zur Vermeidung von Irrtümern und Fehlbedienungen sollten Sie sich alle Informationen und Fragestellungen am Bildschirm genau durchlesen.
3. Bei Alternativfragen wird im allgemeinen eine der möglichen Antworten wesentlich häufiger zu erwarten sein. Diese Entscheidungsmöglichkeit ist eingeklammert, da sie nur die Betätigung von [ENTER] erfordert, z.B.

   ```
   NEUES SPIEL: (J)/N ■
   ```

   In diesem Fall ist
   [J] [ENTER]
   oder nur [ENTER] für JA bzw.
   [N] [ENTER] für NEIN einzugeben.

4. In einigen Spiel-Programmen ist der Einsatz von Spielhebeln zur Steuerung der Bewegungsabläufe zweckmäßig. Die Spielhebel sind an der entsprechenden Buchse (vgl. Bedienungsanleitung, Abschnitt 3.1) anzuschließen und mit der Aktionstaste in Richtung Bildschirm zu halten.

Programmende bzw. Programmabbruch

Wird ein Programm normal beendet, so wird nach einem kurzen Abschlußbild der Kommandomodus wieder erreicht. Sie können das an der Ausgabe

```
OK
>■
```

erkennen und dann im BASIC weiterarbeiten. Wird ein Programm jedoch vorzeitig mittels der STOP-Taste abgebrochen, so erscheint die Ausschrift

```
BREAK IN nnnn
OK
>■
```

wobei für nnnn die Zeilennummer steht, bei deren Abarbeitung das Programm abgebrochen wurde. Es ist dabei möglich, daß vom Programm

nicht der gesamte Bildschirm als Ausgabebereich freigegeben wurde oder die angegebene Ausschrift sogar nicht vollständig sichtbar wird.

Die Grundeinstellung des Bildschirms wird dann durch Eingab von

```
WINDOW [ENTER]
CLS [ENTER]
```

erreicht (vgl. auch Abschnitte 4.4 und 4.15). Ein erneuter Start des Programms durch [RUN] ist natürlich möglich.
Vor dem Einlesen eines neuen Programms muß durch

```
NEU [ENTER]
```

das alte Programm gelöscht werden. Im Ausnahmefall ist auch der Zeichenkettenspeicherbereich durch

```
CLEAR 256 [ENTER]
```

zurückzusetzen (vgl. Abschnitt 4.17).

3.4. Erste Schritte zur BASIC-Programmierung

Sicher wissen Sie bereits, daß ein Rechnerprogramm aus ein Folge von Anweisungen besteht, die dem Computer mitteilen, welche Operationen mit Zahlen oder Zeichen er in welcher Reihenfolge ausführen soll.
Im BASIC des „robotron Z 9001“ beginnt jede Programmzeile mit einer Zeilennummer und enthält eine oder mehrere Anweisungen, die durch einen Doppelpunkt getrennt werden. Die Zeilennummern legen gleichzeitig die Abarbeitungsreihenfolge der Anweisungen fest, wenn sie nicht durch Sprunganweisung geändert wird.
Bei der Erarbeitung eines Programms wird jede Programmzeile über die Tastatur eingegeben. Zur Erleichterung späterer Programmergänzungen sollte dabei von Beginn an im Zehnerabstand numeriert werden.

Programmeingabe

Als erstes Übungsbeispiel soll ein Programm zur Berechnung des Mittelwertes von drei Zahlen aufgestellt werden. Im einfachsten Fall müssen Sie dazu folgendes eingeben:

```
10  A=5 [ENTER]
```

```
20  B=7  [ENTER]
30  C=12 [ENTER]
40  M=(A+B+C)/3 [ENTER]
50  PRINT"MITTELWERT   :";M [ENTER]
```

Beachten Sie bitte:

1. Die Eingabe jeder Programmzeile muß mit [ENTER] abgeschlossen werden. Erst dann wird die Zeile in das Programm. übernommen.
2. Fehler bei der Eingabe einer Programmzeile (z. B. Tippfehler) können, wie im Abschnitt 3.2 beschrieben, korrigiert werden, falls noch nicht [ENTER] betätigt wurde.
3. Bereits mit [ENTER] übernommene Zeilen können nach Neueingabe mit der gleichen Zeilennummer geändert bzw. überschrieben werden. Eine Programmzeile kann gelöscht werden, indem nur die Zeilennummer, und [ENTER] eingegeben werden. Ausführlicher wird auf Korrekturprobleme im Abschnitt 4.7 (EDIT-Kommando) eingegangen.

Programmabarbeitung

Nach beendeter Eingabe können Sie Ihr Programm starten, indem Sie die Taste [RUN] („Start“) betätigen. Danach arbeitet der Computer die Programmzeilen der Reihe nach ab. Beim angegebenen Mittelwertprogramm passiert dabei folgendes:

Durch die Anweisungen 10 bis 30 werden die Variablen A, B und C mit den Werten 5, 7 und 12 belegt. In Zeile 40 wird der Mittelwert M berechnet und in Zeile 50 auf dem Bildschirm ausgegeben.

```
>RUN
Mittelwert   : 8
OK
>■
```

Durch „OK“ wird das Ende der Programmabarbeitung angezeigt. Der Rechner wartet nun wieder auf eine Anweisung bzw. ein Kommando.

Programmänderung

Will man die Mittelwerte verschiedener Zahlengruppen berechnen, so ist es unzweckmäßig, wenn man die Zeilen 10 bis 30 immer wieder neu schreiben muß. Deshalb gibt es eine Eingabeanweisung, die durch das Schlüsselwort INPUT charakterisiert wird. Mit Hilfe der INPUT-Anweisung können Sie während des Programmlaufs Zahlen oder Zeichen eingeben, d. h. Variable mit Eingabedaten belegen. In unserem Beispiel kann die Zeile 10 einfach überschrieben (ersetzt) werden, indem Sie eingeben:

```
10 INPUT "3 Zahlen eingeben :";A,B,C [ENTER]
```

Das Löschen der jetzt überflüssigen Zeilen 20 und 30 erfolgt einfach durch

```
20 [ENTER]
30 [ENTER]
```

Wenn Sie nun wieder [RUN] betätigen, erscheint durch die INPUT-Anweisung in Zeile 10 die Aufforderung

```
3 Zahlen eingeben : ■
```

auf dem Bildschirm. Der Rechner wartet jetzt auf die Eingabe der 3 Zahlen A, B, C, die durch Kommas getrennt werden müssen, z. B.:

```
5,7,12 [ENTER]
```

Danach wird gemäß Zeile 40 der Mittelwert M berechnet und mit Hilfe der Anweisung 50 auf dem Bildschirm ausgegeben. Insgesamt entsteht folgendes Bild:

```
3 Zahlen eingeben: 5,7,12
Mittelwert   : 8
OK
>■
```

Diese kleine Rechnung läßt sich sicher auch auf einem Taschenrechner sehr schnell ausführen. Wollen wir aber zum arithmetischen Mittelwert gleichzeitig noch das geometrische und quadratische Mittel bestimmen, so ist das im BASIC wesentlich einfacher möglich.
Zur Realisierung dieser Programmerweiterung wird zum bereits gespeicherten Programm folgendes eingegeben:

a) Zur Berechnung der gesuchten Werte wird folgendes eingefügt:

```
41 MG=(A*B*C)^(1/3) [ENTER]
42 MQ=SQR((A*A+B*B+C*C)/3) [ENTER]
```

b) Zur Ausgabe der Werte wird ergänzt:

```
60 PRINT"Geometr. Mittel:";MG [ENTER]
70 PRINT"Quadrat. Mittel:";MQ [ENTER]
```

Die Ausführung des jetzt im Rechner gespeicherten Programms nach [RUN] liefert auf dem Bildschirm

```
3 Zahlen eingeben: 5,7,12
Mittelwert         : 8
Geometr. Mittel: 7.4887
Quadrat. Mittel: 8.52448
OK
>■
```

Selbstverständlich kann dieses Programm beliebig oft abgearbeitet werden, wobei nach [RUN] stets nur die 3 Zahlenwerte für A, B und C einzugeben sind.

Darstellung von Programmen auf dem Bildschirm

Nach einigen Modifikationen des Programms muß man sich in der Regel wieder einen genauen Überblick über das aktuell gespeicherte Programm verschaffen. Dazu wird der Programmtext durch Drücken der [LIST]-Taste am Bildschirm ausgegeben („gelistet“). Wurde das Programm zur Mittelwertberechnung wie oben beschrieben eingegeben und ergänzt, so wird es nach [LIST] Wie folgt angezeigt:

```
>LIST

10 INPUT"3 Zahlen eingeben:";A,B,C
20 M=(A+B+C)/3
30 MG=(A*B*C)^(1/3)
40 MQ=SQR((A*A+B*B+C*C)/3)
50 PRINT "MITTELWERT     : ";M
60 PRINT "Geometr. Mittel: ";MG
70 PRINT "Quadrat. Mittel: ";MQ
OK
>■
```


Zeichenketten

Neben den im ersten Beispiel verwendeten numerischen Variablen zur Speicherung von Zahlen kann sich der BASIC-Interpreter auch Zeichenketten und Zeichenkettenvariable merken. An die Bezeichnung dieser Variablen wird jeweils das Zusatzzeichen $ (Dollarzeichen) angefügt, z. B.

B$, A$, Z$, T$.

Eine Wertzuweisung für solche Variablen erfolgt, wie auch für numerische Variable, durch das Gleichheitszeichen oder auch eine INPUT-Anweisung.

Beispiele:

```
20 A$="ZEICHENFOLGE"
21 T$="robotron"
22 Z$="beliebiger Text :"
40 INPUT"Texteingabe :";W$
```

Bevor Sie ein kleines Programm mit einer Zeichenkettenvariablen ausprobieren, müssen Sie das "alte" Programm durch

```
NEW [ENTER]
```

löschen. Kontrollieren Sie die Wirkung von NEW, indem Sie anschließend [LIST] drücken!
Geben Sie nun ein:

```
10 INPUT"IHR GEBURTSTAG:";T$ [ENTER]
20 PRINT"Sie wurden am ";T$;" geboren." [ENTER]
```

und starten Sie durch [RUN]. Nach der entsprechenden Aufforderung geben Sie Ihren Geburtstag ein, z. B.

```
28. Februar 1950 [ENTER]
```

Nachdem Sie sich die Antwort des Rechners angesehen haben, können Sie das Programm erneut starten. Geben Sie Ihren Geburtstag nun anders ein, z. B. also

```
28.2.1950 [ENTER]
```

Was macht Ihr Computer? Er belegt die „Zeichenkettenvariable“ T$ in Zeile 10 mit der von Ihnen eingegebenen Zeichenfolge und gibt Sie in Zeile 20 zwischen zwei festen Texten („Zeichenkettenkonstanten“) wieder aus.
Im Abschnitt 4.13 wird die Arbeit mit solchen Zeichenketten eingehend erläutert.

Löschen des Bildschirmes
Sie werden unterdessen festgestellt haben, daß die Inhalte der Bildschirmzeilen nach oben rücken, sobald der untere Bildrand erreicht ist. Gelegentlich möchten Sie jedoch einen „sauberen“ (leeren) Bildschirm zu Beginn eines Programms haben. Dazu dient die CLS-Anweisung, die sowohl sofort, d. h. im Kommandomodus, als auch im Programm ausgeführt werden kann.
Probieren Sie Z. B.

```
CLS [ENTER]
```

aus, und ergänzen Sie anschließend das vorhandene Programm durch die Zeile 5:

```
5 CLS [ENTER]
```

Die Wirkung dieser Anweisung erkennen Sie durch mehrmaligen Programmstart. Versuchen Sie nun, Ihr erworbenes Wissen zu festigen, indem Sie eigene kleine Programme in den Rechner eingeben und abarbeiten. Dabei können Sie analog zu den oben angeführten Beispielen vorgehen. Wenn Sie jedoch mehr über BASIC erfahren wollen, so wenden Sie sich den systematischen und ausführlichen Darstellungen im Abschnitt 4 zu.

4. Programmierung in BASIC

Aus den Abschnitten 1 und 3 wissen Sie jetzt schon eine ganze Menge über BASIC. Bevor Sie sich nun intensiver mit dieser leicht erlernbaren Programmiersprache befassen, hier noch einige Ratschläge:
Die folgende BASIC-Beschreibung ist so angelegt, daß Sie sowohl in der Reihenfolge der Abschnitte die Sprache und ihre Anwendung auf einfache

Weise erlernen können als auch, was hin und wieder nötig sein wird, durch Nachschlagen schnell die wesentlichen Informationen finden können.
Die bei der Darstellung der einzelnen Anweisungen und Kommandos gegebenen, teilweise sehr detaillierten Hinweise können Sie bei einem ersten Durcharbeiten ohne Nachteil übergehen. Wichtig ist, daß Sie praktische Erfahrungen sammeln, indem Sie möglichst viele Beispiele nachvollziehen. Denn Sie wissen ja - „Probieren geht über Studieren“!
Trotzdem: Der Heimcomputer robotron Z 9001 will es, wie jeder andere Computer, von Ihnen stets genau wissen. Damit Sie die „Grammatik“ der Sprache, in der Sie sich mit ihm verständigen, möglichst schnell und sicher überblicken können, werden im nächsten Abschnitt einige wenige Darstellungsregeln eingeführt, die später durchgängig benutzt werden.

4.1. Schreibweise und Darstellungsregeln

Was ist ein BASIC-Programm, und wie schreibt man es? Das BASIC-Programm ist eine Folge von numerierten BASIC-Programmzeilen (kurz: BASIC-Zeilen), die Anweisungen enthalten und die in aufsteigender Folge ihrer Zeilennummern nacheinander ausgeführt werden. Die BASIC-Programmzeile enthält eine oder mehrere BASIC-Anweisungen, die durch Doppelpunkt : voneinander getrennt werden. Sie kann bis zu 72 Zeichen (also fast zwei Bildschirmzeilen) lang sein. Leerzeichen zählen dabei mit, die vorangestellte Zeilennummer nicht. Beispiel:

```
10 CLS:PRINT "BASIC IST EINFACH!"
```

Die BASIC-Anweisung
dient der Formulierung einer bestimmten Aufgabe, die der Heimcomputer zum Zeitpunkt ihrer Ausführung erfüllen soll. Sie kann natürlich nicht länger als eine BASIC-Programmzeile sein.

Die BASIC-Zeilennummer
muß im Bereich von 1 bis 65527 liegen, ganzzahlig sein und darf nur einmal auftreten.

Die BASIC-Programmeingabe
kann in Groß- oder Kleinbuchstaben erfolgen, wobei letztere automatisch in Großbuchstaben gewandelt werden, wenn sie nicht in Anführungszeichen eingeschlossen (als Zeichenkette), als Eingabedaten für Zeichenkettenvariable oder in Kommentaranweisungen auftreten. Leerzeichen

werden in der BASIC-Programmzeile mitgespeichert. Sie sind ohne Bedeutung, falls nicht, wenn man sie wegläßt, ein BASIC-Schlüsselwort (siehe Anhang G) an falscher Stelle entsteht.

Das BASIC-Kommando
und die sofort ausführbare BASIC-Anweisung unterscheiden sich von der BASIC-Programmanweisung äußerlich durch das Fehlen einer BASIC-Zeilennummer. Zur Darstellung der Anweisungen und Kommandos werden in diesem Handbuch folgende Vereinfachungen benutzt:

Darstellung	Bedeutung
Worte in Großbuchstaben	Schlüsselworte von BASIC, die exakt so geschrieben werden müssen (z. B. **AUTO**).
Worte in Kleinbuchstaben	Parameter (Ersatzworte), die vom Programmierer durch aktuelle Zahlen, Zeichen, Variablenbezeichnungen, Ausdrücke u. ä. ersetzt werden müssen (z. B. *zeilennummer*).
Sonderzeichen	Mit Ausnahme von [ ] Sonderzeichen von BASIC, die exakt so geschrieben werden müssen.
[text]	Der *text* (ohne []) kann wahlweise auftreten oder entfallen.
[text] ...	Der *text* (ohne []) kann wahlweise mehrfach hintereinander auftreten (bis zur Maximallänge von 72 Zeichen/BASIC-Zeile).
0	Null (zur Unterscheidung vom Buchstaben O); wird nur im BASIC-Text verwendet.
[TASTE]	Taste der Tastatur Ihres Heimcomputers

In den angeführten Beispielen wird in der Regel auf die Darstellung des Eingabeabschlusses mit [ENTER] und auch der Vollzugsmeldung OK sowie des Aufforderungszeichens > verzichtet.

4.2. Elemente von BASIC

A, 8, d, *, I, ...	Zeichensatz
123, 4.75, "HAUS", ...	Konstanten
X, AB, Z$, TX$, ...	Variable
ABS, ATN, COS, EXP. INT, LN	Standard
SGN, SIN, SQR, TAN, PI	-funktionen
A-EXP(X-3), "NAME: "+NA$(I)	Ausdrücke

Alle Anweisungen und Kommandos in BASIC bestehen aus Schlüsselwörtern, die der englischen Sprache entstammen (z. B. READ für „Lies“ oder GOTO für „Setze fort bei“), und Elementen, die in ähnlicher Weise im Mathematikunterricht jeder Schule Verwendung finden. Außer mit Zahlen kann jedoch Ihr Heimcomputer auch noch mit Text, sogenannten Zeichenketten, operieren. Er kann diese ähnlich wie Zahlen miteinander verknüpfen, vergleichen oder ein- und ausgeben. Allerdings muß der Programmierer beachten, ob er in einer Anweisung mit Zahlen oder Zeichenketten arbeitet, denn nur in wenigen Anweisungen dürfen diese Datentypen auch gemischt auftreten.
Bevor Sie mit der Formulierung einzelner Anweisungen oder Kommandos im BASIC beginnen, informieren Sie sich zunächst über die „Bausteine“, die Sie dabei immer wieder verwenden werden.
Für den „Einsteiger“ in die BASIC-Programmierung genügt zunächst ein Überblick. Einzelheiten sollten bei Bedarf später nachgeschlagen werden.

Zeichensatz
Der Zeichensatz besteht aus alphanumerischen und Sonderzeichen sowie 128 speziellen Grafikzeichen. Außerdem enthält er eine Reihe von Steuerzeichen, die nicht auf dem Bildschirm dargestellt werden können. Jedes Zeichen wird intern in einem Byte (8 Bits) verschlüsselt. Mit Ausnahme der Grafik- und einiger Steuerzeichen geschieht das nach dem international gebräuchlichen ASCII-Code. Anhang A zeigt den vollständigen Zeichensatz des „robotron Z 9001“.

Konstanten
Konstanten sind feste, d. h. unveränderliche, Werte, die im BASIC-Programm benutzt werden. Nach dem Datentyp ist zwischen numerischen Konstanten (Zahlen) und Zeichenkettenkonstanten (Text) zu unterscheiden:

Numerische Konstanten

a) Ganze Zahlen und Festkommazahlen
Sie werden wie üblich benutzt. Zu beachten ist lediglich, daß
- statt des Dezimalkommas unbedingt ein Dezimalpunkt gesetzt werden muß (1/2 darf also nicht 0,5, sondern muß 0.5 geschrieben werden),
- bei Eingabe von mehr als 6 gültigen Ziffernstellen auf 6 Stellen gerundet wird (1.23456789 wird intern zu 1.23457)

- positives Vorzeichen und auch die 0 vor dem Dezimalpunkt entfallen können (statt +5.5 genügt 5.5, statt 0.33 genügt .33)

Beispiele: 100, -1.345, -3.001, -3, 0

b) Gleitkommazahlen
Zur Unterstützung der Lesbarkeit wandelt Ihr Heimcomputer alle Zahlen außerhalb der Bereiche von ± 0.01 bis ± 999999 bei der Bildschirmausgabe in Gleitkommazahlen (wissenschaftliche Darstellung) um. Sie können jedoch auch jede Zahl des Zahlenbereiches in dieser Form eingeben. Die Darstellung hat die Form

mantisse* E *exponent

Z. B. -1.03E+7
Der Wert der Zahl berechnet sich zu

zahlenwert = *mantisse* * $10^{exponent}$

d. h., für das obige Beispiel ergibt sich der Wert -10300000. Zu beachten ist, daß
- die Mantisse eine ganze oder Festkommazahl sein muß,
- der Exponent ganzzahlig sein muß, wobei ein positives Vorzeichen auch entfallen kann.

Beispiele:

ganze Zahl, Festkommazahl	Gleitkommazahl
30000	3E4
-20500	-20.5E+3
12300000	1.23E+7
0.000000123	1.23E-7

c) Zahlenbereich
Der Bereich zulässiger Zahlenwerte wird durch die interne 4-Byte-Gleitkommadarstellung sämtlicher Zahlen bestimmt. Der zulässige Zahlenbereich wird gebildet durch ± 9.40396E-39 bis ± 1.70141E+38 und die 0.

Zahlen, deren Absolutwert größer ist, führen zu einem Überlauffehler. Bei Unterschreitung des zulässigen Zahlenbereiches wird automatisch der Wert 0 angenommen.

Zeichenkettenkonstanten
Zeichenkettenkonstanten können aus alphanumerischen, grafischen oder Sonderzeichen des Zeichensatzes gebildet werden. Sie werden mit Anführungszeichen vom übrigen BASIC-Text abgegrenzt und dürfen 0 bis maximal 255 Zeichen lang sein. Im Sonderfall einer Zeichenkette der Länge 0 Zeichen spricht man von einer „leeren“ Zeichenkette. Diese darf nicht verwechselt werden mit einer Zeichenkette, die ein oder mehrere Leerzeichen enthält! Leerzeichen in Zeichenketten werden wie jedes andere Zeichen behandelt.

Beispiele:
"HEIMCOMPUTER robotron Z 9001"
"" (leere Zeichenkette!)
" " (Zeichenkette, die 2 Leerzeichen enthält)
"1234" (numerische Zeichenkette)
"/*" (Zeichenkette mit Sonderzeichen)

Variable
Unter einer Variablen versteht man eine Größe, deren Wert (im Gegensatz zur Konstanten) verändert werden kann. Im BASIC- Programm erhält jede Variable, wie in der Mathematik auch, einen Namen. Der Computer reserviert außerdem für jede Variable Speicherplatz, auf dem er sich den aktuellen Wert dieser Variablen merkt. Somit kann er mit einer Variablen rechnen. Er kann ihr Werte zuweisen oder auch Operationen mit ihr ausführen.

Beispiel:
Statt der Konstanten 3 und 4 in der Anweisung

```
PRINT 3+4
```

könnten Sie ebensogut mit 2 Variablen arbeiten, denen Sie vor der Addition Ihre gewünschten Werte zuweisen, also

```
A=3
B=4
PRINT A+B
```

Die Verwendung dieser Variablen A und B hat den Vorteil, daß unsere Zahlenwerte unter ihrem Namen erhalten bleiben, solange keine neue

Wertzuweisung für sie vorgenommen wird. Wir könnten im obigen Beispiel weitere Anweisungen ergänzen, z. B.

```
PRINT B-A
```

oder auch (mit Änderung des Wertes der Variablen A auf 21)

```
A=14+A+B
```

Erst die Verwendung von Variablen ermöglicht in den meisten Fällen die Lösung von praktischen Aufgaben. Sie gestatten es, Eingabedaten, Zwischenergebnisse usw. zu speichern und wieder aufzufinden. Außerdem gelingt mit ihrer Hilfe die von den Daten des speziellen Anwendungsfalles unabhängige und deshalb wiederverwendungsfähige Lösung.

Ebenso wie bei den Konstanten sind zwei Arten (Typen) von Variablen zu unterscheiden:

Numerische Variable und
Zeichenkettenvariable

Der Typ wird jeweils bei der Namensvergabe festgelegt: Eine Zeichenkettenvariable wird durch das Zeichen $ am Ende ihres Namens gekennzeichnet. Sonst können die Variablennamen bei Beachtung folgender Einschränkungen frei gewählt werden:

- Sie müssen mit einem Buchstaben beginnen und dürfen beliebig lang sein. Für den Heimcomputer sind jedoch nur die ersten beiden Zeichen bedeutsam, die übrigen werden zwar mitgespeichert, dienen aber nicht zur internen Unterscheidung verschiedener Variablen.

- Im Namen dürfen an keiner Stelle BASIC-Schlüsselworte enthalten sein (siehe Anhang G).

Beispiele:

Variablenname	Bemerkung
NAME	richtig, aber nur die ersten zwei Buchstaben sind von Bedeutung
X	richtig
A1	richtig
1Y	falsch, Variablenname muß mit einem Buchstaben beginnen
B!	falsch, Sonderzeichen außer $ sind unzulässig
C X	richtig, Leerzeichen sind bedeutungslos
TOR	falsch, enthält das reservierte Wort OR (logischer Operator)
NA	richtig, aber identisch mit der Variablen NAME
TEXT$	richtig, Zeichenkettenvariable
na	richtig, der BASIC-Interpreter wandelt die Kleinbuchstaben in Variablenbezeichnungen bei der Eingabe in Großbuchstaben um, identisch mit (NAME und NA)
TE$	richtig, identisch mit TEXT$

Die bislang verwendeten Variablen heißen auch „einfache Variable“, da jede einzelne Variable ihren eigenen Namen besitzt. Es können auch Speicherplätze für mehrere Werte unter einem gemeinsamen Namen zusammengefaßt werden. Man spricht dann von einer Feldvariablen oder kurz, einem Feld (siehe Abschnitt 4.8). Hierbei verkörpern die Feldelemente Einzelwerte und sind durch Angabe von Indizes (in runden Klammern hinter dem Feldnamen) eindeutig bestimmt. Ein solches Feldelement heißt deshalb auch „indizierte Variable“ und ist inhaltlich einer einfachen Variablen gleichzusetzen. Die Regeln zur Namensvergabe gelten gleichermaßen für Feldvariable wie für einfache Variable.

Beispiele:

```
A (5)=13
```

Dem Feldelement des Feldes A mit dem Index 5 wird der Wert 13 zugewiesen. (Die Zählung der Feldelemente beginnt stets mit Null!)

```
PRINT T$(3,5)
```

Eine Zeichenkette des Zeichenkettenfeldes T$ wird ausgegeben.

Standardfunktionen
Anstelle von Konstanten oder Variablen können Sie in Anweisungen ebensogut eine der im BASIC enthaltenen mathematischen Standard-funktionen einsetzen, z. B.

```
A=128/2
PRINT SQR(A)
```

Dann wird die Quadratwurzel des vorher bestimmten Wertes der Variablen A berechnet und ausgegeben. (Beachten Sie auch, daß für die Division / statt : geschrieben werden muß!)

Folgende Standardfunktionen stehen Ihnen zur Verfügung:

ABS(X) - Absoluter Betrag
ATN(X) - Arcustangens
COS(X) - Cosinus (X im Bogenmaß)
EXP(X) - Exponentialfunktion ex (x < 87. 3366)
INT(X) - Nächstkleinere ganze Zahl zu X
LN(X) - Natürlicher Logarithmus (X > 0)
SGN(X) - Vorzeichenfunktion : SGN(X)
1 fürX>0
0 fürX=0
1 fürX<0
SIN(X) - Sinus (X im Bogenmaß)
SQR(X) - Quadratwurzei (X - 0)
TAN(X) - Tangens (X im Bogenmaß)
PI - Konstante π = 3.14159

Ausdrücke
Ein Ausdruck besteht im einfachsten Fall aus einer einzelnen Konstanten, im allgemeinen aus einer Verknüpfung von Operanden (Konstanten, Variablen, Funktionsaufrufen, Ausdrücken) mittels Operationszeichen (kurz: Operatoren).

Er stellt wiederum genau einen aktuellen Wert dar, der numerisch oder eine Zeichenkette sein kann. Beispielsweise hat der Ausdruck PI/2 den (numerischen) Wert 1.5708. Demzufolge gibt es numerische und Zeichenkettenausdrücke, deren Wert, falls er einer Variablen zugewiesen werden soll, vom selben Typ wie die Variable sein muß. Wichtig ist, daß die Wertbestimmung eines Ausdrucks nach festen Regeln (Reihenfolge!) erfolgt, die mit den in der Elementarmathematik üblichen Rechenregeln übereinstimmen.

Operationen mit Zahlen
Die nachfolgende Tabelle zeigt Rang und Schreibweise der Operatoren:

Rang	Operator	Bedeutung		
1	^	Potenzierung	\	
2	-	negatives Vorzeichen	I	
3	*	Multiplikation	I	arithmetische
	/	Division	I	Operatoren
4	+	Addition	I	
	-	Subtraktion	/	
5	<	kleiner als	\	
	<=	kleiner als oder gleich	I	
	=	gleich	I	
	Vergleichs-			
	<>	ungleich	I	
	Operatoren			
	>	größer als	I	
	>=	größer als oder gleich	/	
6	NOT	Negation (Verneinung)	\	logische
	AND	Konjunktion (UND)	I	
	Operatoren			
	OR	Disjunktion (ODER)	/	.

Die Berechnung eines Ausdrucks erfolgt unter Berücksichtigung des Rangs der Operatoren. Dem Rang 1 kommt die höchste Priorität zu, diese Operation wird zuerst ausgeführt. Besitzen zwei Operatoren gleichen Rang, erfolgt ihre Abarbeitung von links nach rechts. Klammerungen und Funktionsaufrufe werden noch vor den in der Tabelle mit Rang angegebenen Operationen ausgewertet.
Trifft der BASIC-Interpreter während der Auswertung eines Ausdrucks auf eine Division durch Null, so wird die Fehlermeldung /0 angezeigt.
Wird der Betrag des Ergebnisses einer Berechnung größer als der maximal erlaubte Wert (1.70141E+38), reagiert der BASIC- lnterpreter mit der Fehlermeldung OV.
Der logische Operator NOT ist, ebenso wie das negative Vorzeichen, ein einseitiger Operator, d. h., er bezieht sich nur auf einen Operanden, vor dem er steht. Die übrigen Operatoren sind zweiseitig und verknüpfen zwei logische bzw. arithmetische Operanden.

Ausdrücke, die logische Operatoren und Vergleichsoperatoren enthalten, bezeichnet man als logische Ausdrücke. Sie liefern einen logischen Wert, der „wahr“ oder „falsch“ sein kann. Das Ergebnis kann in Verbindung mit der Anweisung IF ... THEN ... ELSE zur Steuerung des Programmablaufes genutzt werden.
Wenn in einem Ausdruck auch arithmetische Operatoren auftreten, werden nach der oben angegebenen Reihenfolge diese zuerst abgearbeitet.

Beispiele:

```
A+B<(X-1)/Y
```

Dieser Ausdruck ist „wahr“, wenn der Wert von A+B kleiner ist als der Wert von X-1 dividiert durch Y.

```
IF  SIN(X)<0 THEN 1000
IF I>10 OR K<0 THEN 40
```

Das sind zwei Beispiele, wie das Ergebnis von logischen Ausdrücken zur Steuerung von Programmabläufen genutzt werden kann.
Beispiele zur Bildung von numerischen Ausdrücken:

Nr.	BASIC-Darstellung	mathematische Schreibweise	Bemerkungen
1	A/B/C	$\frac{A}{B - C}$	richtig
2	A/(B*C)	$\frac{A}{B * C}$	richtig, Beispiele 1 und 2 liefern unter Umständen verschiedene Werte und sind nicht identisch (wegen unterschiedlicher Abarbeitungsreihenfolge, Rundung!)
3	K*-52	-52 * K	richtig
4	K*(-52)	-52 * K	richtig
5	-52*K	-52 * K	richtig
6	4*PI*R^3/3	$4/3\pi * R^3$	richtig
7	SQR(A^2+B^2)	$\sqrt{A^2 + B^2}$	richtig, Verwendg. einer Funktion
8	A+FELD(-3)	-	falsch, -3 ist ein unzulässig. Index

Beispiele für syntaktisch richtige und falsche Vergleichs- und logische Ausdrücke:

Nr.	BASIC-Darstellung	mathematische Schreibweise	Bemerkungen zur BASIC-Schreibweise
1	NOT(A>B)	$\neg(A > B)$	richtig
2	A <= B	$A \leq B$	richtig, logisch gleichwertig zu Beispiel 1
3	A<X AND X<=B	$A < X < B$	richtig
4	A OR B*(I$J)	-	falsch
5	X< =Y>Z	-	falsch

Operationen mit Zeichenketten
Zeichenketten können durch das Operationszeichen + miteinander verkettet werden.

Beispiel:

```
10 X$="BAD" : Y$="WETTER"
20 PRINT X$+"E"+Y$
>RUN
BADEWETTER
```

Außerdem können Zeichenketten mit Hilfe der Vergleichsoperatoren verglichen werden. Der Vergleich erfolgt zeichenweise von links nach rechts. Dabei wird die interne Darstellung der jeweiligen Zeichen miteinander verglichen. Sind alle Zeichencodes der beiden Zeichenketten gleich, so sind auch die Zeichenketten identisch. Andernfalls gilt die Zeichenkette als die kleinere, in welcher zuerst ein Zeichen mit niedrigerem ASCII-Code auftritt.
Wird während eines Vergleichs das Ende einer Zeichenkette erreicht, so gilt die kürzere als die kleinere Zeichenkette.
Die folgenden logischen Ausdrücke liefern den Wert „wahr“:

```
"AA"<"AB"
"X%">"X$" AND "a"<"b"
"SPACE ">"SPACE"
D$<"84."+M$++"11"
```

(mit D$="84.11.08" und M$="11. ")

Den Wert „falsch“ liefern dagegen die logischen Ausdrücke

```
"B"<"A" AND "C"<"D"
A$>B$
```

mit A$="TEXT1" und B$="TEXT2".
Auch bei der Bestimmung des Wertes von Zeichenkettenausdrücken gilt, daß Klammerungen und Funktionsaufrufe zuerst ausgewertet werden. Ebenso hat der Operator + Vorrang vor den Vergleichs- bzw. logischen Operatoren.

4.3. Programmeingabe, -anzeige und -start

NEW		Löschen von Programmen
programmzeile		direkte Programmeingabe
AUTO		automatische Zeilennumerierung
LIST,	[LIST]	Programmanzeige
LINES		Zeilenanzahl je Anzeigeschritt
RUN,	[RUN]	Programmstart

Bevor Sie mit der praktischen Programmierung beginnen können, benötigen Sie noch einige „Werkzeuge“. Der BASIC-Interpreter des „robotron Z 9001“ stellt sie Ihnen mit den oben angegebenen Kommandofunktionen zur Verfügung. Mit ihrer Hilfe können Sie BASIC-Programmzeilen über die Tastatur eingeben und in den Arbeitsspeicher übernehmen, die gespeicherten BASIC-Zeilen auf dem Bildschirm anzeigen und schließlich auch die Ausführung des Programms starten.

Beachten Sie bitte.
- Voraussetzung für die Eingabe eines Kommandos bzw. für die direkte Programmeingabe ist, daß sich der BASIC-Interpreter im Kommandomodus befindet.
- Ihre Eingaben sind mit der [ENTER]-Taste abzuschließen. Dies entfällt, wenn Sie von einer BASIC-Funktionstaste Gebrauch machen ([LIST], [RUN]).
- Benutzen Sie zur Eingabe- bzw. Korrekturerleichterung die Tasten [←], [→], [I←], [→I], [CL LN], [DEL], [INS], die im Abschnitt 3.2 beschrieben wurden.

- Nach Ausführung des jeweiligen Kommandos kehrt der BASIC-Interpreter in den Kommandomodus zurück.
- Erkennt der BASIC-Interpreter Fehler bei der Kommandoausführung, erscheint eine Fehlermitteilung (siehe Anhang H), und es wird ebenfalls der Kommandomodus erreicht.

Löschen von Programmen

Format[1]): **NEW**
Funktion:
Das im Arbeitsspeicher befindliche BASIC-Programm wird gelöscht.

Hinweise:
1. Dieses Kommando sollte vor der Neueingabe eines BASIC-Programmes gegeben werden, um eine ungewollte Überlagerung mit einem vorhandenen Programm zu vermeiden.
2. Durch NEW wird eine abweichend vom Standard festgelegte Größe des Speicherbereiches für Zeichenkettenvariable (siehe Abschnitt 4.17) nicht verändert.
3. Nach NEW haben numerische Variable den Wert Null, und Zeichenkettenvariable bekommen die leere Zeichenkette zugewiesen.

Direkte Programmeingabe
Format[2]): *zeilennummer anweisung[:anweisung]...*
Funktion:
Die BASIC-Programmzeile wird nach Betätigung der [ENTER]-Taste entsprechend ihrer Zeilennummer in das im Arbeitsspeicher befindliche BASIC-Programm eingeordnet.

Hinweise:
1. Eine unter der angegebenen Zeilennummer bereits vorhandene BASIC-Zeile wird durch die zuletzt eingegebene ersetzt.

_____
1) Als Format wird im folgenden eine formale Beschreibung der Anweisungen, Kommandos und BASIC-Funktionen bezeichnet.
2) Eckige Klammern kennzeichnen wahlfreie Angaben und dienen zur Unterscheidung von Pflichtangaben. Beachten Sie, daß die eckigen Klammern selbst nicht geschrieben werden.

2. Die Eingabe einer Zeilennummer ohne nachfolgende BASIC-Anweisung (*zeilennummer* [ENTER]) bewirkt das Streichen einer vorhandenen BASIC-Zeile mit dieser Nummer.

Beispiel:
Geben Sie folgende BASIC-Zeilen ein:

```
10 PRINT "1.ZEILE" [ENTER]
20 PRINT "2.ZEILE" [ENTER]
```

Betätigen Sie nun die [LIST]-Taste und überzeugen Sie sich, daß der BASIC-Interpreter Ihre Programmzeilen gespeichert hat.

Geben Sie jetzt ein:

```
15 PRINT "2.ZEILE" [ENTER]
20 PRINT "3.ZEILE" [ENTER]
5 PRINT "UEBERSCHRIFT" [ENTER]
```

Wenn Sie anschließend erneut die [LIST]-Taste betätigen, finden Sie die gewünschte BASIC-Zeilenfolge in Ihrem aktuellen „Programm vor:

```
5 PRINT "UEBERSCHRIFT" [ENTER]
10 PRINT "1.ZEILE" [ENTER]
15 PRINT "2.ZEILE" [ENTER]
20 PRINT "3.ZEILE" [ENTER]
```

Besonders bei der Eingabe von BASIC-Zeilen in geordneter Reihenfolge ersparen Sie sich Eingabeaufwand, wenn Sie die Zeilennummernvergabe dem Heimcomputer übertragen.

Programmeingabe mit automatischer Zellennumerierung

Format:
AUTO [*zeilennummer* [,*schrittvveite*]]
zeilennummer - erste vom Heimcomputer zu vergebende Zeilennummer (Standardwert: 10)
schrittweite - Differenz zweier aufeinanderfolgender Zeilennummern (Standardwert: 10)

Funktion:
Erzeugung aufsteigender BASIC-Zeilennummern, die vom Nutzer zu vollständigen BASIC-Programmzeilen ergänzt werden können.
Hinweise:
1. Erzeugt der Heimcomputer eine Zeilennummer, unter der im Arbeitsspeicher bereits eine BASIC-Zeile existiert, erscheint ein Stern * hinter der Zeilennummer auf dem Bildschirm. Sie können dann
   - den * ignorieren, die BASIC-Zeile vervollständigen und dadurch die bereits vorhandene ersetzen oder
   - durch eine Leereingabe (*zeilennummer* [ENTER]) die bereits vorhandene BASIC-Zeile streichen oder
   - mittels der [STOP]-Taste die Programmeingabe mit automatischer Zeilennumerierung ohne Veränderung der vorhandenen BASIC-Zelle beenden.
2. Ebenso wie bei der direkten Programmeingabe führt der Versuch, eine **nicht** vorhandene BASIC-Zeile mittels Leereingabe zu streichen, zu einer Fehlermitteilung (UL ERROR).
3. Sie beenden den AUTO-Modus durch [STOP].
4. Soll eine bereits eingegebene BASIC-Zeile geändert werden, so ist das durch Überschreiben möglich. Wesentlich bessere Korrekturmöglichkeiten für umfangreiche Änderungen bestehen jedoch bei Verwendung des EDIT-Kommandos (siehe Abschnitt 4.7).

Beispiel:
Die erneute Eingabe der vier BASIC-Zeilen des vorangegangenen Beispiels bereiten wir mit den Kommandos

```
NEW [ENTER]
AUTO [ENTER]
```

vor.
Sie erhalten nun die erste BASIC-Zeilennummer auf dem Bildschirm und vervollständigen die BASIC-Zeile:

```
>AUTO
10 PRINT "UEBERSCHRIFT" [ENTER]
```

Mit den Folgezeilen verfahren Sie ebenso und beenden die Programmeingabe nach der Zeile 40 mit

```
50 [STOP]
```

Danach können Sie sich mittels der [LIST]-Taste das aktuelle BASIC-Programm aus dem Arbeitsspeicher auf dem Bildschirm anzeigen lassen. Um bei der Programmeingabe mit automatischer Zeilennumerierung dieselbe Zeilennummernfolge wie im vorangegangenen Beispiel zu erzielen, hätten Sie das Kommando

```
AUTO 5,5 [ENTER]
```

verwenden können.

Programmanzeige

Format:
LIST [*zeilennummer*] oder [LIST]
zeilennummer - bestimmt die BASIC-Zelle, ab der das Programm angezeigt werden soll
(Standard: niedrigste vorhandene Zeilennummer)

Funktion:
Das im Arbeitsspeicher befindliche BASIC-Programm wird abschnittsweise auf dem Bildschirm angezeigt („gelistet").

Hinweise:
1. Die Anzeige erfolgt in aufeinanderfolgenden Abschnitten mit einer festen Anzahl von BASIC-Zeilen, die durch LINES (siehe unten) festgelegt werden kann. Standardmäßig werden jeweils 10 Zeilen angezeigt. Reicht die freie Zeilenzahl auf dem Bildschirm für die Anzeige der BASIC-Zeilen nicht aus, „rollt" das Bild um die entsprechende Zeilenzahl nach oben.
2. Nach der Anzeige jedes Abschnittes wartet der Heimcomputer darauf, daß Sie die Anzeige des nächsten Abschnittes verlangen. Sie tun dies durch [ENTER]. Der Vorgang wiederholt sich, bis entweder die höchste Zeilennummer Ihres BASIC-Programms erreicht wurde oder Sie die Programmanzeige mit [STOP] beenden.
3. Die Verwendung der [LIST]-Taste hat dieselbe Wirkung wie die Eingabe des LIST-Kommandos ohne Zeilennummer, d. h. das BASIC-Programm wird ab der niedrigsten Zeilennummer angezeigt.
4. Die [LIST]-Tastenbenutzung erfordert die Gültigkeit der unteren Tastenbelegung, d. h. [SHIFT] bzw. [SHIFT LOCK] dürfen nicht wirken.

Beispiel:
Probieren Sie die gezielte Anzeige ab einer vorgegebenen Zeilennummer
aus! Bezüglich des letzten Beispieles erhalten Sie nach

```
LIST 30 [ENTER]
```

die Bildschirmanzeige

```
30 PRINT "2.ZEILE"
40 PRINT "3.ZEILE"
```

Anschließend befinden Sie sich wieder im Kommandomodus.

Auf einfache Weise können Sie jedoch auch die Anzahl der in einem Abschnitt anzuzeigenden BASIC-Zeilen selbst festlegen:

Format:
LINES *zeilenanzahl*
zeilenanzahl - ganzzahliger Wert im Bereich von 1 bis 65535
(Standard: 10)

Funktion:
Festlegung der maximalen Anzahl von BASIC-Zeilen, die bei Ausführung des LIST-Kommandos ohne Unterbrechung nacheinander auf dem Bildschirm angezeigt werden.

Hinweis:
Die mit dem LINES-Kommando getroffene Festlegung gilt bis zur erneuten Eingabe eines LINES-Kommandos.

Beispiel:
Lassen Sie sich Ihr aktuelles BASIC-Programm noch einmal Zeilenweise anzeigen:

```
LINES 1 [ENTER]
[LIST]
[ENTER]
[ENTER]
   .
   .
   .
[STOP]
```

Mit dem Kommando

```
LINES 10 [ENTER]
```

können Sie anschließend die beim Einschalten des Heimcomputers automatisch wirksame Einstellung wieder herstellen.

Programmstart

Format:
RUN [*zeilennummer*] oder [RUN]
zeilennummer - BASIC-Zeilennummer, ab der die Programmabarbeitung beginnt (Standard: niedrigste vorhandene BASIC-Zeilennummer)

Funktion: Start der Programmabarbeitung

Hinweise:
Vor der Abarbeitung der ersten BASIC-Anweisung werden sämtliche Variablen gelöscht. Numerische Variable erhalten den Wert Null, Zeichenkettenvariable die leere Zeichenkette (" ") zugewiesen. Außerdem erfolgt die Organisation der Arbeit mit internen Daten (DATA, siehe 4.9) noch vor Ausführung der ersten Anweisung.

Beispiel:
Starten Sie das zuletzt eingegebene Beispielprogramm zum Vergleich mit der Taste [RUN] und mit dem Kommando

```
RUN20 [ENTER]
```

4.4. Einfache Anweisungen

[**LET**]	VVertzuweisung
PRINT	Ausgabeanweisung
CLS	Bildschirm löschen
INPUT	Eingabeanweisung
REM oder **!**	Kommentaranweisung
BEEP	akustisches Signal

In diesem Abschnitt werden Sie die einfachsten BASIC-Anweisungen kennenlernen, so daß Sie dann in der Lage sind, eigene kleine Programme zu formulieren und auszuprobieren. Zu den ständig benötigten Anweisungen eines dialogorientierten BASIC-Programms gehören Wertzuweisungen für Variable ebenso wie Anweisungen zur Ein- und Ausgabe von Daten über Tastatur bzw. Bildschirm. Demgegenüber sind erläuternde Kommentare im Programm sowie die Möglichkeit der Ausgabe eines Summertones („Pieps“) als hilfreiche Ergänzungen anzusehen.

Noch ein wichtiger Hinweis:
In diesem und den folgenden Abschnitten wird auf den Eingabeabschluß mittels [ENTER] in der Regel nicht mehr hingewiesen!

Wertzuweisung

Format:
[**LET**] *variable* = *ausdruck*
variable - Bezeichnung einer einfachen oder indizierten Variablen (numerische oder Zeichenkettenvariable)
ausdruck - Ausdruck, **vom selben Typ** wie die Variable

Funktion:
Der aktuelle Wert des rechts stehenden Ausdrucks wird bestimmt und der Variablen zugewiesen.

Hinweise:
1. Es ist darauf zu achten, daß sämtliche im rechtsstehenden Ausdruck benutzten Variablen zum Zeitpunkt der Anweisungsausführung mit den beabsichtigten Werten belegt sind. (Zum Zeitpunkt des Programmstarts (RUN) werden alle Variablen gelöscht.)
2. Das Zeichen = fungiert in der Anweisung nicht als Gleichheitszeichen sondern als Zuweisungsoperator. Ein Vertauschen von Variabler und Ausdruck ist nicht zulässig.
3. Das BASIC-Schlüsselwort LET wird in der Praxis meist weggelassen.
4. Die Anweisung kann (ohne BASIC-Zeilennummer) ebenso wie die meisten anderen BASIC-Anweisungen im Kommandomodus sofort ausgeführt werden.

Beispiele:
Berechnen Sie die Länge der Hypotenuse c = $\sqrt{a^2+b^2}$ eines rechtwinkligen Dreiecks mit bekannten Seitenlängen a = 10 und b = 20 !

Hier das BASIC-Programm:

```
10 LET A=10
20 LET B=10
30 LET C=SQR(A*A+B*B)
40 PRINT C
```

Starten Sie es mit [RUN], ergibt sich

```
22.3607
```

für den gewünschten Wert C.

Ändern Sie jetzt eine Seitenlänge durch Ersetzen der entsprechenden BASIC-Zeile, z. 8. (unter Verzicht auf LET)

```
20 B=15
```

und starten Sie erneut, so erhalten Sie nun

```
18.0278
```

als Hypotenusenlänge.

Und nun noch ein Beispiel mit Zeichenkettenvariablen:

```
100 X$=" Z9001"
110 Y$="Heimcomputer"
120 X$=Y$+"? Glueckwunsch zum "+X$+"!"
130 PRINT X$
```

Nach

```
RUN 100
```

erscheint dann auf dem Bildschirm

```
Heimcomputer? Glueckwunsch zum Z9001!
```

Achten Sie bitte auf die doppelte Verwendung von X$ in Zeile 120! Wenn die BASIC-Zeilen 10 bis 40 noch nicht gelöscht wurden und Sie die Programmabarbeitung mittels [RUN] wiederholen, werden beide Beispiele nacheinander abgearbeitet. Überlegen Sie: warum?

Ausgabeanweisung[1])

Format:
PRINT [*ausdruck* [*trennzeichen ausdruck*] ...]
ausdruck - numerischer oder Zeichenkettenausdruck
trennzeichen - kann sein
1. Komma (,) für (Standard-) Tabellenausgabe
2. Semikolon (;) für fortlaufende Ausgabe
Funktion:
Die aktuellen Werte der in der PRINT-Anweisung angegebenen Ausdrücke werden bestimmt und entsprechend den Darstellungsregeln sowie den verwendeten Trennzeichen auf dem Bildschirm angezeigt.
Werden nach PRINT keine Ausdrücke angegeben, wird eine Leerzeile ausgegeben.

Hinweise:
1. Darstellungsregeln
   a) Darstellung von Zahlen
      Vor und nach jeder Zahl erscheint stets ein Zeichen: vor der Zahl ihr Vorzeichen (Minuszeichen oder Leerzeichen, falls positiv) und nach der Zahl ein Leerzeichen zur Trennung von eventuell anschließenden Ausgaben. Je Zahl werden bis zu 6 aufeinanderfolgende Dezimal-ziffern (außer Exponent) entsprechend der Rechengenauigkeit ausgegeben. Nullen am Ende einer Zahl (nach dem Dezimalpunkt) werden unterdrückt.
      Zahlen die betragsmäßig kleiner als 0.01 oder größer als 999999 sind, werden in Gleitkommadarstellung ausgegeben, alle übrigen Zahlen in ihrer natürlichen Darstellung, d. h. als ganze Zahl oder Festkommazahl ohne Exponent.

[1]) Erweiterte Ausgabemöglichkeiten sind in den Abschnitten 4.15 und 4.16 dargestellt

Beispiele:

PRINT-Anweisung	Bildschirmdarstellung
? 0.00123456789	1.23457E-03
? 0.0123456789	0.0123457
? 0.123456789	.123457
? 1.23456789	1.23457
? 1234.56789	1234.57
? 123456.789	1.23457
? 1234567.89	1.234567E+06
? 10000000	1E+07

b) Darstellung von Zeichenketten
Zeichenketten werden unverändert und ohne zusätzliche Zeichen ausgegeben.

2. Wirkung der Trennzeichen
   a) Trennzeichen Komma (,)
   Die 40 Zeichen fassende Bildschirmzeile wird gedanklich in 3 Zonen zu je 13 Zeichen unterteilt (am rechten Rand bleibt eine Zeichenposition frei). Die Werte der durch Komma getrennten Ausdrücke werden dann jeweils linksbündig in aufeinanderfolgenden 13-Zeichen-Zonen ausgegeben. Dabei werden die Darstellungsregeln selbstverständlich eingehalten. Sind mehr als 3 Werte auszugeben, wird die Ausgabe automatisch in der ersten Ausgabezone der nächsten Zeile fortgesetzt usw.
   b) Trennzeichen Semikolon (;)
   Die Werte werden fortlaufend unmittelbar nacheinander ausgegeben. Bei vollständig gefüllter Zeile erfolgt ebenfalls automatischer Zeilenwechsel, bis die Ausgabe aller Ausdrücke abgeschlossen ist.

3. Jede PRINT-Anweisung beginnt mit ihren Ausgaben auf einer neuen Zeile (Ausnahmen sind im Abschnitt 4.15 beschrieben). Welche Zeile des Bildschirms die nächste Zeile ist, wird durch vorangegangene PRINT-, INPUT-, CLS- und WINDOW-Anweisungen sowie eventuelle Fehlermeldungen des Betriebssystems bestimmt. Ist der Bildschirm vollständig gefüllt, und es erfolgt eine weitere Ausgabe, so „rollt“ das Bild um jeweils eine Zeile nach oben.

4. Zur Vereinfachung der Eingabe können Sie statt PRINT ohne weiteres auch einfach nur ? eingeben. Bei einer Programmanzeige mittels LIST wird das Zeichen ? wieder durch PRINT ersetzt.

Beispiele:
Die nachfolgende Tabelle zeigt ihnen einige Beispiele, die Sie durch eigene schnell ergänzen können. Überzeugen Sie sich von der Wirkung der Trennzeichen!

BASIC-Anweisung	Bildschirmausgabe
? X	123
? X,Y,Z,0.123456789	123 1E-03 -2.22222E-04 .123457
? A$	ABC
? " X"," X^2"," X^3"	X X^2 X^3
? X,X^2,X^3	123 15129 1.86087E+06
?	
? "A";"B";"C"	ABC
? "PRINT-"+A$,X*Y	PRINT-ABC .123
? "SIN(PI/4)= ";SIN(PI/4)	SIN(PI/4)= .707107
? 2^10;B$;	1024 Byte
? C$;1;B$;" PRO ZEICHEN":?	1 Byte PRO ZEICHEN
? "ZEICHENKETTE SEI","9999999", "ZAHL WIRD",9999999; "OK?"	ZEICHENKETTE SEI 9999999 ZAHL WIRD 1E+07 OK?

```
|           |           |           |
0         12 13        25 26...39
```

```
|           |           |           |
0         12 13        25 26        39
```

^ Spaltennummer der Bildschirmanzeige

***) Anmerkungen:** **1. ? steht abkürzend für PRINT**
2. Vorausgesetzte Variablenwerte:

```
X=123              A$="ABC"
Y=0.001            B$="Bytes"
Z=-2.22222E-04     C$=""
```

Bildschirm löschen

Format **CLS**

Funktion: Die Bildschirmanzeige wird gelöscht.

Hinweise:
1. Der Löschvorgang wirkt auf den aktuellen Ausgabebereich auf dem Bildschirm, der (mittels WINDOW, siehe Abschnitt 4.15) auch abweichend vom gesamten Bildschirm festgelegt werden kann.
2. Nach Ausführung von CLS wird die Ausgabe am linken oberen Rand des aktuellen Ausgabebereiches fortgesetzt.
3. CLS kann auch im Kommandomodus vorteilhaft eingesetzt werden.

Eingabeanweisung

Format:
INPUT ["*hinweis*"] *variable* [,*variable*] ...
variable - numerische oder Zeichenkettenvariable, für die ein Wert eingegeben werden soll
hinweis - Zeichenkette, die Informationen für den Programmanwender enthält

Funktion:
Nach Ausgabe des *hinweis*-Textes wird die laufende Programmabarbeitung unterbrochen, und über Tastatur einzugebende Werte werden den in der INPUT-Anweisung aufgeführten Variablen zugewiesen. Ist dies vollständig geschehen, wird der Programmlauf fortgesetzt.

Hinweise:
1. Die INPUT-Anweisung ist nur im BASIC-Programm zulässig, nicht im Kommandomodus!
2. Auf dem Bildschirm wird der *hinweis*-Text an der Stelle im Programm ausgegeben, an der die nächste PRINT-Ausgabe erfolgen würde. Ist auf den *hinweis* verzichtet worden, erscheint statt dessen ein Fragezeichen (?). Anschließend geben Sie Daten ein und beenden die Eingabe mit [ENTER].
3. Reihenfolge, Typ und Anzahl der einzugebenden Werte müssen mit der Liste der Variablen in der INPUT-Anweisung übereinstimmen. Zeichenkettenkonstanten können, falls sie keine Leerzeichen oder Kommas enthalten, ohne die begrenzenden Anführungszeichen ("…") eingegeben werden. Das Komma dient als Trennzeichen zwischen den einzugebenden Werten und darf nicht mit dem Dezimalpunkt verwechselt werden.
4. Sind zu wenige Werte eingegeben worden, erfolgt mittels ?? die Aufforderung zur Eingabe der restlichen Werte. Sind zu viele Werte eingegeben worden, werden die überflüssigen ignoriert. Die Mitteilung EXTRA IGNORED erscheint, und die Programmabarbeitung wird fortgesetzt.
5. Bei Eingabe einer Zeichenkette statt einer numerischen Variablen erhalten Sie die Ausschrift ? REDO FROM START und müssen dann die Dateneingabe für diese INPUT-Anweisung von Anfang an wiederholen.
6. Bei einer Leereingabe für eine Variable wird deren aktueller Wert nicht verändert. Die Leereingabe erzeugen Sie durch sofortige [ENTER]-Betätigung.

Beispiel:
Berechnen Sie noch einmal die längste Seite eines rechtwinkligen Dreiecks. Im Gegensatz zum Beispiel bei der Erläuterung der LET-Anweisung sollen jetzt die Seitenlängen a und b erst während der Programmabarbeitung eingegeben werden.
Nach dem Kommando NEW geben Sie ein:

```
10 INPUT "Werte fuer a und b? ";A,B
20 C=SQR(A*A+B*B)
30 PRINT "c=";C
```

Starten Sie nun mit [RUN], erscheint zunächst

```
Werte fuer a und b?
```

Sie geben nun Ihre Zahlenwerte ein, z. B.

```
10,20 [ENTER]
```

und erhalten anschließend die Lösung:

```
Werte fuer a und b? 10,20
c= 22.3607
```

Natürlich hätten Sie die Werte für a und b auch einzeln abfragen können, z. B. durch

```
10 INPUT "Wert fuer a? ";A
20 INPUT "Wert fuer b? ";B
```

Probieren Sie auch die Fehlermöglichkeiten einmal aus!

Kommentaranweisung

Format:
REM [*kommentar*]
kommentar - beliebiger Text

Funktion:
Kommentare dienen der besseren Lesbarkeit eines BASIC-Programms. Auf die Programmabarbeitung haben sie keinen Einfluß.

Hinweise:
1. Statt des Schlüsselwortes REM kann abkürzend ein Ausrufezeichen (!) geschrieben werden.
2. Tritt in einer BASIC-Zeile eine Kommentaranweisung auf, kann dahinter keine weitere BASIC-Anweisung in derselben Zeile stehen, da der gesamte Text bis zum Zeilenende als Kommentar betrachtet wird.

Beispiel:
Zur Illustration nachfolgend Ausschnitte aus einem BASIC-Programm.

```
10 REM  PROGRAMM XYZ
20 REM STAND: 11.11.1984
30 A1=999  :  A2=1E+06  :! Anfangswerte
```

Akustisches Signal

Format: BEEP

Funktion:
Der Summer ihres Heimcomputers wird veranlaßt, einen kurzzeitigen Ton zu erzeugen.

Hinweis:
Die Anweisung ist nützlich, um die Aufmerksamkeit des Programmnutzers zu lenken, z. B. auf das Auftreten von Fehlern oder den Abschluß länger andauernder Rechnungen.

Beispiel:
Eine Zahleneingabe wird geprüft. Ist die Zahl kleiner als Null, erfolgt eine Fehlermeldung und erneute Eingabe.

```
200 INPUT"ZAHL?";Z
210 IF Z>=0 THEN 250 :! Fallunterscheidung
220 BEEP
230 PRINT "NEGATIVE ZAHL UNZULÄSSIG! "
240 GOTO 200
250 PRINT "Eingegeben wurde: ";Z
```

4.5. Anweisungen zur Steuerung des Programmablaufes

GOTO	unbedingte Verzweigung
ON ... GOTO	berechnete Verzweigung
IF ... THEN ... :ELSE...	bedingte Verzweigung
FOR ... NEXT	Wiederholungsanweisung
PAUSE	Programmunterbrechung
STOP	Programmabbruch
END	Programmende

Mit der Kenntnis der im vorangegangenen Abschnitt behandelten Anweisungen können Sie ohne weiteres bereits eigene kleine BASIC-Programme formulieren. Sie werden dabei jedoch schnell bemerken, daß die starre Abarbeitungsreihenfolge der BASIC-Anweisungen gemäß ihrer Zeilennumerierung oft stark einschränkend ist. Die Lösung praktischer Aufgaben erfordert häufig z. 8. die Unterscheidung verschiedener Fälle, verbunden mit der Abarbeitung unterschiedlicher Programmteile, die mehrfache Ausführung von Anweisungen oder auch die Unterbrechung des Programmlaufs an einer vorgegebenen Stelle.
Sie finden anschließend BASIC-Anweisungen erläutert, mit deren Hilfe Sie solche Probleme auf einfache Weise lösen können.

Unbedingte Verzweigung (Sprung)

Format:
GOTO *zeilennummer*
zeilennummer - BASIC-Zeilennummer, ab der die Ausführung des Programms fortgesetzt werden soll

Funktion:
Die Programmabarbeitung wird in der BASIC-Programmzeile fortgesetzt, deren Zeilennummer angegeben wurde.

Beispiel:
Fortlaufende Berechnung der Quadratwurzeln einzugebender Zahlen.

```
10 INPUT"Eingabezahl ? ";N
20 PRINT „SQR(X)= ";SQR(X)
30 GOTO 10:! Sprung nach Zeile 10
```

Dieses Programm muß mit [STOP] beendet werden.

Berechnete Verzweigung

Format:
ON *ausdruck* **GOTO** *zeilennummer* [,*zeilennummer*) ...
ausdruck - numerischer Ausdruck mit einem Wert größer als oder gleich Null, dessen ganzer Anteil benutzt wird
zeilennummer - BASIC-Zeilennummer, ab der die Ausführung des Programms fortgesetzt werden soll

Funktion:
Der ganzzahlige VVert des Ausdrucks wird bestimmt und sei gleich I. Das Programm wird dann mit der Zeilennummer fortgesetzt, die an I-ter Stelle (von links) in der angegebenen Liste von Zeilennummern steht.

Hinweise:
1. ist der ganzzahlige Wert des Ausdrucks gleich Null oder größer als die Anzahl der angegebenen Zeilennummern, wird das Programm mit der auf die ON ... GOTO-Anweisung folgenden BASIC-Anweisung fortgesetzt.
2. Ein negativer Wert des Ausdrucks führt zu einer Fehlermeldung.

Beispiel:
Im folgenden Demonstrationsbeispiel soll ein "Menü" auf dem Bildschirm erscheinen und der Nutzer zur Wahl einer Programmfunktion aufgefordert werden. Der angegebene Programmausschnitt zeigt eine Möglichkeit zur Organisation der Programmfortsetzung nach der Funktionsauswahl.

```
100 CLS   :! LOESCHEN DES BILDSCHIRMES
110 PRINT "AUSWAHL PROGRAMMFUNKTION"
120 PRINT : PRINT : PRINT
130 PRINT "1 – EINGABE" : PRINT
140 PRINT "2 – BERECHNUNG" : PRINT
150 PRINT "3 – AUSGABE" : PRINT
160 PRINT "4 – PROGRAMMENDE" : PRINT
170 PRINT : PRINT : PRINT
180 !EINGABE KENNZAHL
190 KZ=0 : INPUT "KENNZAHL FUNKTION?";KZ
200 ON KZ GOTO 500,800,1000,1500
210 BEEP : GOTO 190
500 CLS   :! Programmteil EINGABE
510 PRINT "Aufruf EINGABE"
520 PAUSE 30 :! 3 Sekunden Pause
  .
  .
730 GOTO 100
800 CLS   :! Programmteil BERECHNUNG
810 PRINT "Aufruf BERECHNUNG"
820 PAUSE 30 :! 3 Sekunden Pause
  .
  .
980 GOTO 100

1000 CLS   :! Programmteil AUSGABE
1010 PRINT "Aufruf AUSGABE"
1020 PAUSE 30 :! 3 Sekunden Pause
  .
  .
1410 GOTO 100

1500 CLS   :! PROGRAMMENDE
1510 PRINT "PROGRAMMENDE"
  .
  .
1600 END
```

Bedingte Verzweigung

Format 1:
IF *bedingung* THEN *zeilennummer* [:ELSE *zeilennummer*]

Format 2:
IF *bedingung* THEN *anweisung* [:*anweisung*] ...
[:ELSE *anweisung* [:*anweisung*] ...]

bedingung - Vergleichs- oder logischer Ausdruck, Wert gleich Null entspricht "falsch" bzw. "Bedingung nicht erfüllt", Wert ungleich Null entspricht "wahr" bzw. "Bedingung erfüllt"

zeilennummer - BASIC-Zeilennummer, ab der das Programm fortgesetzt werden soll

anweisung - beliebige BASIC-Anweisung mit Ausnahme erneuter IF-Anweisung.

Funktion:
Ist die angegebene Bedingung erfüllt, werden die Anweisungen nach THEN ausgeführt, andernfalls die Anweisungen nach ELSE. Wird nicht aus dem betreffenden Anweisungszweig zu einer anderen BASIC-Zeile verzweigt, erfolgt die Programmfortsetzung mit der auf die IF-Anweisung folgenden BASIC-Zeile. Das trifft auch zu, wenn kein ELSE-Zweig existiert und die angegebene Bedingung nicht erfüllt ist.

Hinweise:
1. Fehlt der ELSE-Zweig, gehören sämtliche hinter THEN stehenden Anweisungen dieser BASIC-Zeile zum THEN-Zweig.
2. Beachten Sie bei der Formulierung von Bedingungen (Vergleichsausdrücken), daß die interne Zahlendarstellung Ihres Heimcomputers für „praktisch gleich große“ Zahlen, die auf verschiedene Weise gewonnen wurden, infolge Rundungsunterschieden geringfügig unterschiedlich ausfallen kann. Ein Test auf Gleichheit würde in solchen Fällen mit dem Ergebnis „falsch“ enden und Ihr Programm möglicherweise an ungewollter Stelle fortgesetzt werden.
3. Erinnern Sie sich, daß Schlüsselwörter des BASIC (siehe Anhang G) nicht in gewählten Bezeichnungen oder Zusammenziehungen solcher mit Schlüsselwörtern (ohne trennendes Leerzeichen) auftreten dürfen! Beispielsweise würde

```
999 IFB<ATHEN200
```

wegen des darin enthaltenen Schlüsselwortes AT zum Syntaxfehler [1]) führen, nicht aber

```
999 IF B<A THEN 200
```

Beispiel:
Probieren Sie das folgende Zahlenratespiel aus!

```
10 ! Spieler A gibt eine ganze Zahl vor
20 INPUT "Zu erratende Zahl? ";A : CLS
30 ! Spieler B darf nun raten
40 INPUT "Gesuchte Zahl ist? ";B
50 IF B=A THEN 80
60 IF B<A THEN PRINT B; "IST ZU KLEIN"
     :ELSE PRINT B; "IST ZU GROSS"

70 GOTO 40
80 PRINT "RICHTIG! ";B; "IST DIE GESUCHTE ZAHL"
90 A$="J":INPUT"NOCH EINMAL: (J)/N";A$
100 IF A$="J" THEN 20
110 IF A$<>N THEN BEEP:GOTO90
120 PRINT"ENDE"
```

Wiederholungsanweisung (Schleife)

Format:
FOR *laufvariable=anfwert* **TO** *endwert* **STEP** *schrittweite*
.
.
.
NEXT [*laufvariable*]

laufvariable - numerische Variable, deren Wert nach jedem Schleifendurchlauf um die Schrittweite verändert wird.

[1]) Ein Fehler, der sich durch einen Verstoß gegen die in den Formaten beschriebenen Regeln ergibt, wird als Syntaxfehler bezeichnet.

anfwert - numerischer Ausdruck, der den Wert der Laufvariablen beim ersten Durchlauf bestimmt.

endwert - numerischer Ausdruck, dessen Wert nach jedem Schleifendurchlauf mit dem veränderten Wert der Laufvariablen verglichen wird.

schrittweite - numerischer Ausdruck, dessen Wert nach jedem Durchlauf zum Wert der Laufvariablen addiert wird (Standardwert: 1)

Funktion:

Die zwischen FOR... und NEXT befindlichen BASIC-Anweisungen werden wiederholt ausgeführt. Die Zahl der Wiederholungen wird dadurch bestimmt, daß der Wert der Laufvariablen beim nächsten Schleifendurchlauf den vorgegebenen Endwert nicht überschreiten (bei positiver Schrittweite) bzw. nicht unterschreiten (bei negativer Schrittweite) darf. Jede Schleife wird mindestens einmal durchlaufen. Nach dem letzten Schleifendurchlauf wird das Programm mit der auf NEXT folgenden BASIC-Anweisung fortgesetzt (siehe Beispiel).

Hinweise:

1. Innerhalb der Anweisungsfolge einer Wiederholungsanweisung ist das Auftreten weiterer FOR ... NEXT-Anweisungen zulässig. Man spricht dann von „geschachtelten Schleifen“. Dabei ist zu beachten, daß jede durch FOR und NEXT begrenzte „innere“ Schleife vollständig innerhalb der „äußeren“, d. h. zeitlich zuvor begonnenen, Schleife liegt und daß in jedem Fall unterschiedliche Laufvariable verwendet werden.
2. Enden mehrere geschachtelte Schleifen an derselben Programmstelle, kann ein gemeinsamer Schleifenabschluß mit NEXT, gefolgt von den Laufvariablen in der richtigen Reihenfolge, geschrieben werden.
3. Wird auf die Angabe einer Laufvariablen hinter NEXT verzichtet, bezieht sich NEXT stets auf die zuletzt eröffnete Schleife. Wiederholungsanweisung (Schleife)
4. Der Eintritt in eine Wiederholungsanweisung darf nur über FOR... erfolgen. Ein Hineinspringen (z. B. mittels GOTO) in die Anweisungsfolge innerhalb einer Schleife führt bei Erreichen von NEXT zum Fehler.
5. Eine Wiederholungsanweisung kann vorzeitig verlassen werden (z. B. mittels einer IF- oder GOTO-Anweisung). Der Heimcomputer gibt aber dann den für die Schleifenorganisation benötigten Speicherplatz (16 Bytes) nicht wieder frei, so daß es bei häufigem vorzeitigen Verlassen von Schleifen zu Speicherüberlauf kommen kann. Besser ist es in solchen Fällen, innerhalb der Schleife den Wert der Laufvariablen auf den Endwert zu setzen und die Schleife über NEXT zu verlassen.

Beispiele:

Berechnen Sie doch zunächst einmal, wie Ihr Sparguthaben wächst, wenn Sie es eine bestimmte (ganze) Zahl von Jahren bei einem Zinssatz von 3,25 % unangetastet lassen:

```
10 INPUT "GUTHABEN (MARK)? ";G
20 INPUT "WIEVIELE JAHRE KEINE ABHEBUNG?";AJ
30 IF AJ<1 THEN PRINT "BEDAURE!":GOTO 90
40 FOR J=1 TO AJ
50 G=G+G*0,0325
60 NEXT J
70 G=INT(G*100+0.5)/100: Rundung auf Pfennige
80 PRINT:PRINT"DANACH WAEREN ES";GMARK! ":PRINT
90 PRINT "NOCH EINMAL? RUN! "
```

Wenn Sie sich z. B. für den Guthabenzuwachs nach Monaten oder sogar Tagen (oder auch nur für BASIC) interessieren, ändern Sie das Programm nach Ihren Vorstellungen ab!

Hier noch ein Beispiel für „geschachtelte Schleifen“:

```
10 FOR I=3 TO 1 STEP -1
20 FOR J=1 TO 4 STEP2
30 PRINT I,J
40 NEXT J
50 NEXT I
```

Starten Sie dieses Programm mit RUN, erhalten Sie

```
3             1
3             3
2             1
2             3
1             1
1             3
```

Für die Zeilen 40 und 50 hätten Sie auch schreiben können:

```
40 NEXT J,I
```

Programmunterbrechung

Format:
PAUSE [*dauer*]

dauer - numerischer Ausdruck, dessen Wert (positiv und ganzzahlig) die Dauer der Programmunterbrechung in Zehntelsekunden bestimmt.

Funktion:
Das Programm wird, ohne den Programmodus zu verlassen, in seiner Ausführung unterbrochen. Nach Betätigung der Taste [CONT] oder nach Ablauf einer angegebenen Unterbrechungsdauer wird es mit der nachfolgenden BASIC-Anweisung fortgesetzt.

Hinweis:
Während der Programmunterbrechung können Sie, da Sie sich nicht im Kommandomodus befinden, keine Anweisungen oder Kommandos abarbeiten lassen.

Beispiel:
Ein Text soll wiederholt (zeilenweise) ausgegeben werden. Die Zeit zwischen zwei Ausgaben wird mit einer PAUSE-Anweisung festgelegt.

```
10 FOR I=1 TO 10
20 PRINT "*** robotron Z 9001 ***"
30 PAUSE 30
40 NEXT I
```

Bitte variieren Sie durch Neueingabe der Zeile 30 die Ausgabegeschwindigkeit. Probieren Sie auch PAUSE ohne Angabe der Unterbrechungsdauer und PAUSE I*2 aus.

Programmabbruch

Format: **STOP**

Funktion:
Das laufende BASIC-Programm wird bei Erreichen der STOP-Anweisung abgebrochen, die aktuelle BASIC-Zeilennummer angezeigt, und der BASIC-Interpreter kehrt in den Kommandomodus zurück.

Hinweise:
1. Die Abbruchmitteilung des BASIC-Interpreters lautet

   ```
   BREAK IN zeilennummer
   OK
   >■
   ```

   wobei *zeilennummer* die BASIC-Zeile der den Abbruch aulösenden STOP-Anweisung kennzeichnet.
2. Nach STOP bleiben alle aktuellen Variablenwerte und Systemzustände erhalten
3. Im Gegensatz zur PAUSE-Anweisung können Sie nach STOP-Anweisungen Kommandos ausführen lassen. lnsbesondere können Sie das Programm mittels RUN neu starten, wobei aktuellen Variablenwerte wieder zurückgesetzt werden. Wollen Sie das Programm mit den aktuellen Variablenwerten fortsetzen, so können Sie das mit der Anweisung

   GOTO *zeilennummer*.

   Dabei ist die zur Programmfortsetzung vorgesehene BASIC-Zeilennummer anzugeben. Gegebenenfalls können Sie Variablen vor Ausführung der GOTO-Anweisung durch sofort ausführbare Anweisungen modifizieren.
4. Mit dem Kommando CLEAR (siehe Abschnitt 4.17) können Sie alle Variablen vor der Programmfortsetzung löschen. Bei Benutzung der Kommandos AUTO, EDIT, RENUMBER und DELETE gehen die aktuellen Variablenwerte ebenfalls verloren.
5. Nach STOP ist auch eine Programmfortsetzung mit der Taste [CONT] möglich, falls Sie zwischenzeitlich keine anderen Kommandofunktionen (AUTO, EDIT, LIST, ...) benutzt haben.
6. STOP-Anweisungen können an beliebigen Stellen im Programm stehen. Sie lassen sich u. a. zum Programmtest vorteilhaft einsetzen.

Beispiel:
Fügen Sie in das vorhergehende Beispiel zur PAUSE-Anweisung ein:

```
15 IF I>5 THEN STOP
```

und überzeugen Sie sich Mittels [RUN] von der Wirkung Kontrollieren Sie nach Abbruch des Programms den aktuellen Wert der Laufvariablen mittels

```
? I
```

und modifizieren Sie z. B.

```
I=9
```

Danach setzen Sie die Ausgabe durch [CONT] oder

```
GOTO 20
```

fort.

Programmende

Format **END**

Funktion:
Das laufende BASIC-Programm wird beendet. Der BASIC-Interpreter kehrt ohne eine Mitteilung in den Kommandomodus zurück.

Hinweis:
Die END-Anweisung ist nicht zwingend erforderlich. Ein Programm wird im Normalfall nach der letzten Anweisung beendet. Trotzdem ist diese Anweisung empfehlenswert, da nach dem logischen Programmende weitere BASIC-Anweisungen im Arbeitsspeicher stehen können (z. B. Unterprogramme, andere Programme). Fehlt die END-Anweisung, würden diese anschließend ausgeführt.

Beispiel:
Sie sollten künftig Programme stets mit END abschließen, Fügen Sie an das letzte Beispiel an

```
50 END
60 PRINT "NAECHSTER PROGRAMMBEGINN"
```


und starten Sie erneut. Wollen Sie die Anweisung in Zelle 60 erreichen, erinnern Sie sich an das Kommando RUN 60.

4.6. Steuerung des Programmablaufes durch den Nutzer

[RUN]	Programmstart
[STOP]	Programmabbruch
[PAUSE]	Programmunterbrechung
[CONT]	Programmfortsetzung

Haben Sie schon diese sehr wichtige Eigenschaft Ihres Heimcomputers bemerkt? Er läßt sich von Ihnen selbst dann noch steuern, wenn er gerade Ihr BASIC-Programm abarbeitet!
Während die im vorherigen Abschnitt angegebenen Anweisungen zur Steuerung des Programmablaufes unter denselben Bedingungen natürlich stets an denselben Programmstellen wirken, haben Sie es mit den oben angegebenen BASIC-Funktionstasten selbst in der Hand, den Zeitpunkt Ihres Eingreifens zu bestimmen.
Es gibt vielfältige Gründe, die Sie zu einem solchen Eingriff veranlassen können, z. B. wenn Sie

- ein nicht mehr benötigtes oder fehlerhaft ablaufendes BASIC-Programm vorzeitig abbrechen wollen,
- Zwischenergebnisse einer Berechnung ansehen möchten, für die programmseitig keine Ausgabe vorgesehen war,
- Variablenwerte verändern möchten,
- ein bestimmtes Bildschirmbild länger als im Programm vorgesehen betrachten wollen.

Im zuletzt genannten Fall empfiehlt sich die Anwendung von [PAUSE]/[CONT], in den anderen Fällen [STOP] mit entsprechender Fortsetzung. Die Wirkung der Tasten [PAUSE] und [STOP] entspricht der gleichnamigen Anweisungen (siehe Abschnitt 4.5). Bezüglich der Fortsetzungsmöglichkeiten sind Sie an dieser Stelle ebenfalls informiert worden.

Beachten Sie bitte noch, daß

- die Taste [STOP] auch zur Beendigung von Kommandofunktionen des BASIC-Interpreters benutzt wird (AUTO, LIST, EDIT), die [PAUSE]-Taste in diesen Fällen jedoch unwirksam ist,

- nach [STOP] (oder der STOP-Anweisung) die zuletzt, z. B. im abgebrochenen Programm, erzeugten Systemzustände (einschl. der durch WINDOW, CLEAR, LINES und NULL festgelegten) erhalten bleiben.

Über Abbruch und Unterbrechung hinaus gibt es weitere Möglichkeiten, auf den Ablauf Ihres Programms Einfluß zu nehmen. Diese müssen Sie dann allerdings programmseitig vorbereiten. Naheliegend ist die mit INPUT zu realisierende Aufforderung zur Eingabe von Steuerdaten, die im Programm ausgewertet werden und seinen weiteren Ablauf bestimmen. Dieser Möglichkeit bedient man sich z. B. bei der sogenannten „Menü-Technik“, die Sie schon kennengelernt haben (Beispiel zu ON ... GOTO im Abschnitt 4.5).
Steuernden Einfluß auf Ihr Programm können Sie auch mit Hilfe der im Abschnitt 4.14 beschriebenen speziellen Eingabefunktionen ausüben. Beispielsweise gestattet Ihnen die Funktion INKEY$, Funktionstasten zu vereinbaren, deren Betätigung den weiteren Programmablauf bestimmt. Der wesentliche Unterschied zur INPUT-Anweisung besteht darin, daß das Programm auch ohne ihr Eingreifen „weiterläuft“. Der Zeitpunkt, zu dem steuernd eingegriffen wird, kann somit vom Nutzer bestimmt werden.

4.7. Programmänderung

EDIT Änderung von Programmzeilen
DELETE Streichen von Programmzeilen
RENUMBER Neunumerieren von Programmzeilen

Jetzt sind Sie sicher bereits in der Lage, eine Reihe von Aufgaben mit einem selbst geschriebenen kleinen BASIC-Programm zu lösen. Sie werden dabei festgestellt haben, daß die Programmänderung durch Neueingabe kompletter BASIC-Zellen manchmal recht umständlich ist. Mit den Kommandos EDIT und DELETE stehen Ihnen komfortablere Korrekturmöglichkeiten zur Verfügung.
Nach umfangreichen Programmänderungen ist oftmals eine neue, durchgängige Zeilennumerierung nützlich. Durch das Kommando RENUMBER können Sie auch das erreichen.

Ändern von Programmzeilen

Format:
EDIT *zeilennummer*
zeilennummer - Nummer der BASIC-Zeile, die als erste zur Änderung bereitgestellt werden soll.

Funktion:
Beginnend mit der angegebenen Zeilennummer, werden in aufsteigender Numerierungsfolge BASIC-Zeilen einzeln angezeigt und zur Änderung bereitgestellt. Sie können die betreffende Zeile direkt ändern und sich dazu auch der Tasten [←[, [→], [I←], [→I], [CL LN], [DEL], [INS] (siehe Abschnitt 3.2) bedienen. Die BASIC-Zeilennummer läßt sich nicht ändern.
Jede einzelne BASIC-Zeile wird mit der Taste [ENTER] oder der [↓]-Taste beendet, anschließend wird, falls vorhanden, diejenige mit der nächsthöheren Zeilennummer bereitgestellt. Wurde die Zeile Mit [ENTER] abgeschlossen, wird die entsprechende Zeile im Arbeitsspeicher durch die geänderte ersetzt. Im Sonderfall der Leereingabe (*zeilennummer* [ENTER]) erfolgt die vollständige Streichung der betreffenden Zelle. Wird die Zeile mit der [↓]-Taste abgeschlossen, bleibt die ursprüngliche Zeile im Arbeitsspeicher unverändert erhalten, unabhängig davon, ob ihr angezeigtes Abbild geändert wurde oder nicht.

Hinweise:
1. Nutzen Sie beim Ausführen von Änderungen in einer Zeile die im Abschnitt 3.2. beschriebenen Tasten zur Kursorbewegung sowie zum Streichen bzw. Einfügen von Zeichen.
2. Der EDIT-Modus kann vor Erreichen der letzten Programmzeile mit [STOP] beendet werden. Dabei bleibt die zuletzt angezeigte BASIC-Zeile unverändert.
3. Tritt ein Stern (*) hinter der Zeilennummer der bereitgestellten Zeile auf, bedeutet das, daß die Zeile die Länge von 72 Zeichen überschreitet. Die Ursache liegt dann darin, daß Sie bei der Eingabe dieser Zeile abkürzend ? statt PRINT geschrieben haben. Der BASIC-Interpreter bietet Ihnen im EDIT-Modus (ebenso wie bei LIST) jedoch die „rückübersetzte“ Form der Zeile an, d. h. PRINT (5 Zeichen) statt ? (1 Zeichen). Soll die Zeile unverändert erhalten bleiben, dürfen Sie sie

jetzt nicht mit [ENTER] abschließen (da Sie in diesem Fall durch die angezeigten 72 Zeichen ersetzt wird) sondern mit [↓] bzw. [STOP].

4. Falls Sie eine nicht existierende Zeilennummer im EDIT-Kommando angeben, erhalten Sie eine Fehlermitteilung.

Beispiel:
Wiederholen Sie die eine oder andere Programmänderung in Verbindung mit den zuvor gezeigten Beispielen. Nutzen Sie dazu jetzt das EDIT-Kommando! Wollen Sie zum Beispiel mit der Zeile 40 beginnend ändern, geben Sie ein:

```
EDIT 40
```

Streichen von Programmzeilen

Format:
DELETE *zeilennummer1* [,*zeilennummer2*]
zeilennummer1,2 - kennzeichnet niedrigste bzw. höchste zu streichende BASIC-Zeile

Funktion:
Der Programmabschnitt, der durch die angegebenen BASIC-Zeilennummern begrenzt ist, wird gestrichen.

Hinweise:
1. Soll nur eine Zeile mit DELETE gestrichen werden, darf nur *zeilennummer1* angegeben werden.
2. Die mit *zeilennummerl,2* spezifizierten BASIC-Zeilen müssen im Arbeitsspeicher vorhanden sein, andernfalls führt die Ausführung des Kommandos zu einer Fehlermitteilung.

Beispiel:
In einem BASIC-Programm, das sich im Arbeitsspeicher befindet und in Zehnerschritten von 10 bis 5660 numeriert ist, sollen die Zeilen 2840 bis 3620 (einschließlich!) gestrichen werden. Das Kommando lautet

```
DELETE 2840,3620
```


Neunumerieren von Programmzeilen

Format:
RENUMBER [*zlnralt1* [,*zlnralt2* [,*zlnrneu1* [,*schrittweite*]]]]
zlnralt1,2 -kennzeichnet niedrigste bzw. höchste alte Zeilennummer des neu zu numerierenden Programmabschnittes (Standardwerte: *zlnralt1*: niedrigste vorhandene Zeilennummer *zlnralt2*: höchste vorhandene Zeilennummer)
zlnrneu1 - kennzeichnet niedrigste Zeilennummer des neu numerierten Programmabschnittes (Standardwert: *zlnralt1*)
schrittweite - Differenz zweier aufeinanderfolgender Zeilennummern (Standardwert: 10)

Funktion:
Das im Arbeitsspeicher befindliche BASIC-Programm, bzw. ein angegebener Abschnitt daraus, wird gemäß Vorgabe oder standardmäßig neu numeriert. Dabei werden auch alle Bezugnahmen auf Zeilennummern (GOTO .... IF ... THEN ... usw.) entsprechend verändert.

Hinweise:
1. Mit dem RENUMBER-Kommando ist es nicht möglich, die Reihenfolge der Programmzeilen zu verändern! Deshalb müssen Sie streng darauf achten, daß durch abschnittsweises Neunumerieren die aufsteigende Zeilennumerierung nicht verletzt wird und keine doppelten Zeilennummern entstehen. Andernfalls kommt der BASIC-Interpreter in Schwierigkeiten, die möglicherweise nur durch [RESET] behoben werden können.
2. Lassen Sie im RENUMBER-Kommando Parameter weg, weil Sie die Standardwerte nutzen wollen, so ist dies nur von rechts nach links möglich.

Beispiel:
Ein von 5 bis 250 in 5er-Schritten numeriertes Programm soll in 10er-Schritten numeriert werden. Es soll außerdem mit der Zeilennummer 10 beginnen. Das RENUMBER-Kommando lautet:

```
RENUMBER 5,250,10,10
```

Der letzte Parameter (10) hätte auch entfallen können (Standardwert). Nach Ausführung des Kommandos hat das Programm die Numerierung von 10 bis 500 in 10er-Schritten. Überzeugen Sie sich an einem

selbstgewählten Beispiel, indem Sie das neunumerierte Programm anschließend mit [LIST] anzeigen!

4.8. Felder

DIM Feldvereinbarung

Unter einem Feld versteht man die Zusammenfassung mehrerer Variablen desselben Typs (numerisch oder Zeichenkette) unter einer gemeinsamen Bezeichnung, dem Feldnamen.
Der Zugriff auf die einzelnen Variablen, die Feldelemente, erfolgt durch Angabe eines Index (oder mehrerer Indizes) nach dem Feldnamen. Wird nur ein Index verwendet, spricht man von einem eindimensionalen Feld, d. h., die Feldelemente oder indizierten Variablen sind in einer Reihe angeordnet. Bei zwei Indizes ist es üblich, von Zeilen und Spalten zu sprechen, in denen die Feldelemente stehen.
Ihr BASIC-Interpreter kann Felder verarbeiten, die theoretisch bis zu 255 Dimensionen haben könnten. Für die meisten praktischen Probleme sind jedoch Felder mit bis zu 3 Dimensionen ausreichend. Außerdem begrenzen die maximale BASIC-Zeilenlänge und der verfügbare Arbeitsspeicher die Zahl der verwendbaren Dimensionen auf einen wesentlich, niedrigeren Wert. Damit Sie Ihre Felder „nach Maß“ vereinbaren können, steht Ihnen die folgende Anweisung zur Verfügung:

Format:
DIM *feldname* (*index* [,*index*] ...) [,*feldname*(*index*[,*index*] ...)] ...

feldname - nach den für Variable gültigen Regeln gewählte Bezeichnung (siehe Abschnitt 4.2)
index - numerischer Ausdruck, der die höchste Ordnungsnummer der Feldelemente einer Dimension festlegt und dessen ganzzahliger Wert zwischen 0 und 32766 liegen muß

Funktion:
Es wird Speicherplatz für die in der DIM-Anweisung festgelegten Feldelemente reserviert. Gleichzeitig bekommen alle Elemente eines numerischen Feldes den Wert 0 und die eines Zeichenkettenfeldes die leere Zeichenkette ("") zugewiesen.

Hinweise:
1. Die Zählung der Feldelemente einer Dimension beginnt stets mit 0. Es werden also je Dimension *index*+1 Feldelemente vereinbart.
2. Standardmäßig gelten 11 Feldelemente (eindimensional) als vereinbart, wenn ein Feldname im BASIC-Programm benutzt wird, ohne daß dieses Feld mit einer DIM-Anweisung vereinbart wurde.
3. Eine DIM-Anweisung muß in der Reihenfolge der Programmabarbeitung vor der ersten Benutzung des Feldnamens in einer BASIC-Anweisung stehen. Andernfalls erfolgt die Dimensionierung gemäß 2, und bei Erreichen der DIM-Anweisung für das Feld wird es neu vereinbart. Dabei werden alle Feldelemente gelöscht.
4. Bei Zeichenkettenfeldern ist darauf zu achten, daß der Speicherbereich für Zeichenketten zur Zeit des Programmlaufs ausreichend groß ist (siehe Abschnitt 4.17/CLEAR).

Beispiel:
Zunächst vereinbaren Sie ein eindimensionales Feld, weisen einigen Feldelementen Werte zu und geben aus:

```
10 DIM W(5)
20 FOR I=! TO 3 : W(I)=10+I : NEXT I
30 FOR I=0 TO 5 : PRINT "W(";I; ")=";W(I)
40 NEXT I
```

Nach dem Programmstart erhalten Sie auf dem Bildschirm:

```
W( 0 )= 10
W( 1 )= 11
W( 2 )= 12
W( 3 )= 13
W( 4 )= 0
W( 5 )= 0
```

Hätten Sie die Zeile 10 weggelassen (oder vergessen), hätte alles genauso funktioniert, da das Feld weniger als 11 Elemente besitzt. Überzeugen Sie sich, indem Sie Zeile 10 streichen und erneut starten! Nachteilig ist in diesem Fall, daß Sie zusätzlichen Speicherplatz für die nichtbenötigten Feldelemente W(6) bis W(l0) reservieren.

Benutzen Sie jetzt Felder, deren Größe Sie erst zum Zeitpunkt der Programmabarbeitung festlegen:

```
10 INPUT "WIEVIELE PERSONEN?"; N
20 IF N<1 THEN BEEP: GOTO 10
30 DIM NAME$(N-1,1), ALTER(N-1)
40 FOR I=0 TO N-1
50 INPUT "NAME, VORNAME, ALTER?";
   NAME$(I,0),NAME$(I,1),ALTER(I)
60 NEXT I
70 PRINT "AELTER ALS 17 JAHRE SIND:":
   PRINT: KP=0
80 FOR I=0 TO N-1
90 IF ALTER(I)>17 ZHEN PRINT NAME$(I,1);
   " ";NAME$(I,0): KP=1
100 NEXT I
110 IF KP=0 THEN PRINT "KEINE PERSONEN"
120 PRINT: END
```

Bevor Sie [RUN] drücken, überlegen Sie, ob die Gesamtzahl der einzugebenden Zeichen für das Feld NAME$ größer als 256 Zeichen werden wird. Wenn ja, vergrößern Sie den Speicherbereich für Zeichenketten, mit CLEAR 1000 z. B. auf maximal 1000 Zeichen (siehe Abschnitt 4.17). Haben Sie nun das Programm gestartet und die Daten der von Ihnen festgelegten Anzahl von Personen eingegeben, erhalten Sie anschließend die Liste der Personen über 17 Jahre. Wie wäre es, wenn Sie das Mini-„Recherche-Programm“ noch etwas ausbauen? Zum Beispiel könnten sie weitere Daten der Personen erfassen und auch das „Recherchekriterium“ wählen lassen.

4.9. Interne Daten

DATA	Vereinbarung von Daten
READ	Lesen von Daten
RESTORE	Setzen des Datenzeigers

Wie Sie wissen, kann ihr „robotron Z 9001“ beachtliche Datenmengen speichern. Ebenso wie Ihr BASIC-Programm wollen Sie die „Problemdaten“ jedoch oft nicht bei jeder Benutzung erneut eingeben. Mit den im folgenden erläuterten Anweisungen gelingt es ihnen ohne weiteres, auch größere Datenmengen direkt in das Programm aufzunehmen und diese gemeinsam mit dem Programm auf Magnetbandkassette zu speichern.

Format:
DATA *konstante* [,*konstante*] ...
konstante - numerische oder Zeichenkettenkonstante

Funktion:
Die angegebenen Daten werden gespeichert. Der Zugriff auf diese Daten erfolgt ausschließlich mittels der READ-Anweisung.

Hinweise:
1. Die DATA-Anweisung ist nur im Programmodus erlaubt, d. h. nicht als sofort ausführbare Anweisung im Kommandomodus.
2. Da alle DATA-Anweisungen bei Programmstart (RUN) ausgewertet werden, ist ihre Stellung (Anordnung) im Programm beliebig. Sie können also auch nach den entsprechenden READ-Anweisungen auftreten. Während des Programmlaufs werden DATA-Anweisungen übergangen.
3. Anführungszeichen ("...") zur Begrenzung von Zeichenketten-konstanten können entfallen, wenn keine Schlüsselwörter, Kommas bzw. führende oder nachfolgende Leerzeichen in ihnen enthalten sind.

Format:
READ *variable* [, *variable*] ...
variable - Name einer numerischen oder Zeichenkettenvariablen, die den Wert einer Konstanten aus einer DATA-Anweisung erhält

Funktion:
Den Variablen der READ-Anweisung(en) werden Konstanten aus DATA-Anweisungen als aktueller Wert zugewiesen. Die Zuweisung erfolgt fortlaufend in der Reihenfolge des Auftretens der READ-Anweisungen und der Variablennamen in ihnen. Dabei werden die Konstanten der DATA-Anweisungen in ihrer Folge gemäß aufsteigender Zeilennumerierung zugewiesen.

Hinweis:
Der Typ der Variablen (numerisch oder Zeichenkette) muß mit dem Typ der ihr zugewiesenen Konstanten aus der DATA-Anweisung übereinstimmen.

Beispiel:

```
10 DATA 1065,75, "MEYER, FRITZ"
20 DATA 960,35, "ENDER, SYLVIA"
30 READ M,N$
40 PRINT: PRINT N$;" verdient";M; "MARK": PRINT
```

Bei Abarbeitung dieses Programms erhalten Sie zunächst auf dem Bildschirm

```
MEYER, FRITZ verdient 1065,75 MARK
```

Geben Sie nun GOTO 30 ein und überprüfen Sie das Ergebnis!

Format:
RESTORE [*zeilennummer*]
zeilennummer - BASIC-Zeilennummer einer DATA-Anweisung (Standard: niedrigste Zeilennummer einer DATA-Anweisung)

Funktion:
Die Wertzuweisung für die nächsten READ-Anweisungen erfolgt, beginnend mit dem ersten Datenelement der DATA-Anweisung, in der angegebenen BASIC-Zeile bzw. der ersten DATA-Anweisung im Programm.

Beispiel:
Vielleicht haben Sie beim vorigen Beispiel ein zweites Mal versucht, das Programm mit GOTO 30 fortzusetzen und einen OD-Fehler erhalten, da keine weiteren Daten verfügbar waren.

In diesem Falle hätte

```
RESTORE 10
```

oder einfach

```
RESTORE
```

genügt, um mit

```
GOTO 20
```

erneut beginnen zu können.

Was passiert, wenn sie

```
RESTORE 20
```

eingeben, bevor Sie mit

```
GOTO 30
```

oder [RUN] fortsetzen?
Wenden Sie sich nun noch einmal dem Beispiel aus Abschnitt 4.8 zu. Angenommen, Sie wollen die „Personaldaten“ gemeinsam mit Ihrem Programm speichern und nicht vor jeder „Re- cherche“ neu eingeben. Dann ersetzen Sie, wie im folgenden Programm gezeigt, die INPUT-Dateneingabe durch DATA-Anweisungen, deren Daten Sie mittels READ einlesen.

```
10 DATA 8 :!ANZAHL PERSONEN
20 READ N :! EINLESEN PERSONENANZAHL
30 DIM NAME$(N-1,1),ALTER(N-1)
40 ! EINLESEN DATEN
50 FOR I=0 TO N-1
60 READ NAME$(I,0),NAME$(I,1),ALTER(I)
70 NEXT I
80 ! AUSWERTUNG DATEN
90 INPUT "ALTERSGRENZE?";AG
100 IF AG<1 THEN BEEP: GOTO 90
110  PRINT  "AELTER  ALS";AG;  "JAHRE  SIND:
";PRINT:
    KP=0
120 FOR I=0 TO N-1
130 IF ALTER(I)>AG THEN PRINT NAME$(I,1);" ";
    NAME$(I,0):KP=1
140 NEXT I
150 IF KP=0 THEN PRINT "KEINE PERSONEN"
```

```
200 !DATEN
210 DATA HINZ,MARIO,16,KUNZ,INES,13
220 DATA FENDER,TOM,23,COHN,SARAH,19
230 DATA TILLE,HANS-JUERGEN,34,SPETH,ROBERT,20
240 DATA SCHNEIDER,OLGA,67,SACHSE,CAESAR,12
300 END
```

Mit Hilfe des EDIT-Kommandos können Sie Ihre Daten „pflegen“. Der Umfang läßt sich erweitern, indem Sie weitere DATA-Anweisungen hinzufügen.

4.10. Zufallszahlen

RND Erzeugen einer Zufallszahl

Zufällig erzeugte Zahlen, die der Nutzer eines Programms nicht vorhersagen kann oder will, haben große praktische Bedeutung unter anderem für die

- Programmierung von Spielen (z. 8. wenn sie elektronisch „würfeln“ wollen),
- Zufällige Auswahl von Fragen in Lehr- und Lernprogrammen (der Computer zieht eine Frage),
- Erzeugung von Testdaten (oder würden Sie lieber 300 Zahlen von Hand eintippen, um ein Sortierprogramm auf Richtigkeit und Geschwindigkeit zu testen?).

In diesen und vielen anderen Fällen verwenden Sie die BASIC-Funktion RND.

Format:

RND (*ausdruck*)

ausdruck - numerischer Ausdruck, dessen Wert die zu erzeugende Zufallszahl beeinflußt.

Typ: BASIC-Funktion[1])

Funktion:

Es wird eine Zufallszahl zwischen 0 und 1 erzeugt. Abhängig vom Wert des als Argument angegebenen Ausdrucks wird folgendes ausgeführt:

Argument (X = Ausdruck)	Funktionswert: 0<RND(X)
X > 0	nächste Zahl einer Folge von Zufallszahlen, die ihrerseits vom Wert des Arguments abhängt
X > 0	nächste Zahl einer Folge von Zufallszahlen, die ihrerseits vom Wert des Arguments abhängt .
X=0	identisch mit Funktionswert beim vorangegangenen Funktionsaufruf, d. h. Wiederholung der letzten Zufallszahl
X < 0	wie im Fall X > 0, jedoch wird danach eine neue Folge von Zufallszahlen begonnen.

Hinweise:

1. Die praktisch auftretenden Funktionswerte (Zufallszahlen) liegen etwa zwischen 10^{-6} und $1 - 10^{-6}$.
2. Auch für unverändertes Programm und ein- und denselben Vorgang bei Programmstart, z. B. nach Einschalten des Rechners, kann ein jeweils unterschiedlicher Beginn der Zufallszahlenfolge erreicht werden. Der Absolutwert des RND-Funktionsarguments muß dann seinerseits zufällig sein. Das läßt sich z. B. durch Abfrage des aktuellen Sekundenstandes der eingebauten Systemuhr (siehe Abschnitt 5.1) erreichen.

Beispiel:

Programmieren Sie sich einen elektronischen „Würfel“, der jeweils nach [ENTER]-Betätigung eine Zahl zwischen 1 und 6 anzeigt!

```
10 PRINT "Z9001 WUERFELT FUER SIE! ENTER! "
20 PRINT : N=0
30 INPUT"";X$
40 N=N+1 : Z=1+INT(6*RND(1))
50 PRINT,N; "-TER WURF: ";Z
60 GOTO 30
```

[1]) Eine BASIC-Funktion ist keine selbständige Anweisung. Sie wird, wie bereits für die mathematischen Standardfunktionen erläutert, benutzt, d. h. sie besitzt stets einen aktuellen Funktionswert und kann in Ausdrücken bzw. wie ein solcher verwendet werden.

Dieses Programm können Sie nur durch [STOP] beenden. Beachten Sie weiterhin, daß die BASIC-Funktion INT den ganzen Anteil ihres Argumentwertes liefert. Zur Übung ersetzen Sie vielleicht noch die Vorgabe der zu ratenden Zahl im Beispiel des Abschnitts 4.2 (IF-Anweisung) durch eine im Programm erzeugte Zufallszahl!

4.11. Nutzerfunktionen

DEF Fn*name*	Definition einer Nutzerfunktion
Fn*name*	Aufruf einer Nutzerfunktion

Die Benutzung von Funktionsaufrufen in Ausdrücken (Formeln) ist nicht auf die fest im BASIC enthaltenen Funktionen beschränkt. Genauso wie Sie numerische Standardfunktionen einsetzen können (siehe Abschnitt 4.2), können Sie auch Funktionen benutzen, für die Sie zuvor selbst festlegen, wie der Funktionswert bestimmt wird.

Format:
DEF FN*name* [(*parameter*)] = *ausdruck*

name - zweiter Teil des Funktionsnamens, gebildet wie der Name einer numerischen Variablen

parameter - Name einer numerischen Variablen, die im rechtsstehenden Ausdruck verwendet werden kann

ausdruck - numerischer Ausdruck, der die Berechnungsvorschrift für den Funktionswert bildet und unter anderem numerische Variable des Programms sowie weitere Funktionsaufrufe enthalten kann.

Funktion:
Eine vom Nutzer vorzugebende numerische Funktion wird vereinbart. Wahlweise kann ein formaler Parameter zur Bestimmung des Funktionswertes benutzt werden, der zum Zeitpunkt des Funktionsaufrufes durch den aktuellen Argumentwert ersetzt wird.

Hinweise:
1. Die Anweisung DEF FN ist nur im Programmodus erlaubt, d. h. nicht als sofort ausführbare Anweisung im Kommandomodus.
2. Die Länge der Anweisung darf insgesamt eine BASIC-Zeile nicht überschreiten.
3. Die Anweisung DEF FN muß vor der ersten Verwendung der Nutzerfunktion durchlaufen werden. Der Aufruf zur Berechnung einer Nutzerfunktion erfolgt durch Angabe ihres Namens einschließlich eines eventuell vereinbarten Arguments (aktueller Parameter).

Beispiel:
Mit Hilfe der BASIC-Standardfunktionen ATN(X) und SQR(X) soll die Berechnung von

$x \arcsin(x) = \arctan(x/\sqrt{1-x^2})$

über eine Nutzerfunktion erfolgen.

```
10 DEF FNAS(X)=ATN(X/SQR(1-X*X))
   :
   :
   :
100 A=.707
110 B=FNAS(A)
120 PRINT "arc sin(";A; ")=";B
```

Nach [RUN] erscheint auf dem Bildschirm

```
arc sin( .707 )= .785247
```

Besonders für häufig im Programm zu berechnende Ausdrücke bringt die Verwendung einer Nutzerfunktion erhebliche Vorteile.

4.12. Unterprogramme

GOSUB	Aufruf eines Unterprogramms
RETURN	Rückkehr aus Unterprogrammen
ON ... GOSUB	Berechneter Aufruf von Unterprogrammen

Der Einsatz von Unterprogrammen (auch „Subroutinen“ genannt) dient vorrangig einer effektiven Programmgestaltung:

- klare, übersichtliche Programmstruktur und damit verbunden gute Testbarkeit und Änderungsfreundlichkeit,
- kürzerer, speicherplatzsparender Programmtext durch mehrfache Verwendung nur einmal formulierter Anweisungsblöcke.

Außerdem gelingt damit auf einfache Weise die Wiederverwendung vorhandener Programmlösungen beim Erarbeiten neuer Programme, einschließlich des Einsatzes von käuflich erhältlichen Unterprogrammen für Standardaufgaben.

Ein Unterprogramm ist für sich allein nicht abarbeitungsfähig. Es bedarf eines übergeordneten Programms, von dem aus es aufgerufen wird. Aufruf eines Unterprogramms

Format:
GOSUB *zeilennummer*
zeilennummer - BASIC-Zeilennummer, ab der die Programmabarbeitung fortgesetzt wird

Funktion:
Die Programmabarbeitung wird mit der angegebenem BASIC-Zelle fortgesetzt. Nach Auftreten einer RETURN-Anweisung (siehe unten) erfolgt die Rückkehr und Programmfortsetzung mit der Anweisung, die auf die GOSUB-Anweisung folgt.

Hinweise:
1. Eine Parameterübermittlung (etwa wie bei Nutzerfunktionen) ist nicht möglich. Alle im Programm verwendeten Variablen sind uneingeschränkt gültig. Es ist darauf zu achten, daß keine Fehler durch ungewollte Mehrfachnutzung von Variablennamen entstehen!
2. Das mit der GOSUB-Anweisung aufgerufene Unterprogramm wird wegen 1. auch nicht gesondert vereinbart. Es empfiehlt sich, eine kennzeichnende Kommentaranweisung voranzustellen.
3. Endet das übergeordnete Programm bezüglich der Zellennumerierung vor einem Unterprogramm, so muß es durch END abgeschlossen werden, da sonst das nachfolgende (Unterprogramm ungewollt erreicht wird.

Rückkehr aus Unterprogrammen

Format: **RETURN**

Funktion:
Das Ende des Unterprogramms ist erreicht. Die Programmabarbeitung wird im übergeordneten Programmteil fortgesetzt. Die Fortsetzung erfolgt unmittelbar nach der GOSUB-Anweisung, die den Eintritt in das Unterprogramm bewirkte.

Hinweis:
Ein Unterprogramm kann an mehreren Stellen (logisch) enden, die jeweils durch RETURN gekennzeichnet werden müssen.

Beispiel:
Angenommen, Sie sollen für eine nach Punkten bewertete schriftliche Leistungskontrolle einer Schulklasse die entsprechenden Noten sowie den Klassendurchschnitt nach Noten und Punkten bestimmen. Sie können dafür z. B. folgendes kleine BASIC-Programm benutzen:

```
10 CLS:PRINT"AUSWERTUNG LEISTUNGSKONTROLLE"
20 PRINT:PRINT
30 INPUT"ANZAHL DER SCHUELER? ";A$
40 INPUT"HOECHSTPUNKTZAHL LEISTUNGSKONTROLLE?"
   ;HP
50 PRINT:PRINT
60 DIM ERG(AS-1,1) :! Feld fuer Punkte und Noten
70 FOR I=1 TO AS-1 :! Einzelergebnisse
80 PRINT"PUNKTE DES";I+1; "-TEN SCHUELERS";
90 INPUT P: ERG(I,0)=P :!Punktzahlspeicherung
100 GOSUB 300
110 ERG(I,1)=N     :! Notenspeicherung
120 PRINT:PRINT"NOTE";N:PRINT
130 NEXT I
140 PRINT:PRINT: "KLASSENDURCHSCHNITT":PRINT
150 KD=0:GOSUB 500:! Punktzahldurchschnitt
160 PRINT"NACH PUNKTEN: ";D; "(VON";HP; ")":
    PRINT
170 KD=1:GOSUB 500:! Notendurchschnitt
180 PRINT"NACH NOTEN: ";D;:PRINT:PRINT
190 END       :! Ende Hauptprogramm
300 ! Unterprogramm Notenermittlung
310 PP=100*P/HP   :! P zu HP in Prozent
```

```
320 PP=INT(PP+.5) :! Rundung zu Prozent
330 IF PP>=96 THEN N=1 : RETURN
340 IF PP>=80 THEN N=2 : RETURN
350 IF PP>=60 THEN N=3 : RETURN
360 IF PP>=37 THEN N=4 : RETURN
370 N=5 : RETURN
500 ! Unterprogramm Durchschnittsberechnung
510 D=0
520 FOR I=0 TO AS-1
530 D=D+ERG(I,KD) :! Punkte
540 NEXT I
550 D=D/AS
560 D=INT(D*100+.5)/100:! Rundung
570 RETURN
```

Sicher hätten Sie bei diesem einfachen Beispiel auch ohne Unterprogramme auskommen können. Berücksichtigen Sie jedoch, daß eine ausgebaute Problemlösung zum Beispiel auch noch Korrekturmöglichkeiten, eine Datenspeicherung auf Magnetbandkassette sowie weitere Auswertemöglichkeiten umfassen sollte! Sie werden dann die Vorteile einer modularen Programmgestaltung, d. h. die Realisierung von fest umrissenen Teilfunktionen in getrennten Unterprogrammen, schnell zu schätzen wissen.

Berechneter Aufruf von Unterprogrammen

Format:
ON *ausdruck* **GOSUB** *zeilennummer* [,*zeilennummer*] ...
ausdruck - numerischer Ausdruck mit einem Wert größer als oder gleich Null, dessen ganzer Anteil benutzt wird
zeilennummer - BASIC-Zeilennummer des aufzurufenden Unterprogrammes

Funktion:
Der ganzzahlige Wert des Ausdrucks wird bestimmt und sei gleich 1. Das Programm wird dann durch Eintritt in ein Unterprogramm bei der Zeilennummer fortgesetzt, die an 1-ter Stelle in der angegebenen Zeilennummernliste steht.

Nach Auftreten einer RETURN-Anweisung erfolgt die Rückkehr und Programmfortsetzung mit der Anweisung, die auf die ON ... GOSUB-Anweisung folgt.

Hinweis:
Ist der ganzzahlige Wert des Ausdrucks gleich Null oder größer als die Anzahl der angegebenen Zeilennummern, wird das Programm mit der auf die ON ... GOSUB-Anweisung folgenden Anweisung fortgesetzt.

Beispiel:
Zur Übung verwenden Sie das Beispiel zu ON ... GOTO aus Abschnitt 4.5. Ersetzen Sie die ON ... GOTO-Anweisung durch die

```
200 ON KZ GOSUB 500,800,1000,1500
205 IF KZ>0 AND KZ<4 THEN 100
```

und fügen Sie an den entsprechenden Stellen RETURN ein.

4.13. Zeichenkettenfunktionen

ASC	ASCII-Code zu gegebenem Zeichen
CHR$	Zeichen zu gegebenem ASCII-Code
VAL	Zahl aus gegebener Zeichenkette
STR$	Zeichenkette aus gegebener Zahl
LEN	Länge einer Zeichenkette
LEFT$	
RIGHT$	Übernahme einer Teilzeichenkette
MID$	
INSTR	Suche einer Zeichenkette in einer anderen Zeichenkette
STRING$	Wiederholung einer Zeichenkette

Diese Zeichenkettenfunktionen sind im BASIC Ihres „robotron Z 9001“ ständig verfügbar. Sie bieten Ihnen wirkungsvolle Unterstützung beim Umgang mit Zeichenketten, d. h. bei der „nichtnumerischen Datenverarbeitung“. Ihre Benutzung erfolgt in der gleichen Weise wie die der numerischen Funktionen.

Bitte beachten Sie:
- Ebenso wie Zeichenkettenvariable enden die Namen derjenigen Zeichenkettenfunktionen, die eine Zeichenkette als Funktionswert liefern, mit dem Sonderzeichen Dollar ($).
- Zeichenketten können mit dem Operationszeichen + verkettet, d. h. aneinandergehängt, werden (z. B. liefert "AB"+"BA" die Zeichenkette "ABBA")
- Die „leere Zeichenkette“ enthält kein Zeichen und hat die Länge Null.

Wandlung Zeichen ⟷ ASCII-Code

Format:
ASC(*zeichenkette*)
zeichenkette - beliebiger Zeichenkettenausdruck, nur erstes Zeichen wird ausgewertet

Typ: BASIC-Funktion

Funktion:
Als Funktionswert wird der (dezimale) Wert des ASCII-Codes des ersten Zeichens der angegebenen Zeichenkette angenommen (zur Codierung vgl. Anhang A).

Hinweis:
Der Aufruf von ASC mit der leeren Zeichenkette als Funktionsargument führt zu einem Programmabbruch (FC ERROR).

Beispiel:

```
>A$="AB" : PRINT ASC(A$)
```

- Diese sofort ausführbare Anweisung liefert den Wert 65, ebenso wie auch

```
>PRINT ASC("AB")
```

oder

```
>PRINT ASC("A")
```

(Das Zeichen A hat die interne Codierung 01000001. Berechnen Sie den Dezimalwert dieser Binärzahl, so erhalten Sie 1*2^0+1*2^6=1+64=65.)

Format:
CHR$(*ausdruck*)
ausdruck - numerischer Ausdruck, dessen ganzzahliger Wert zwischen 0 und 255 liegen muß

Typ: BASIC-Funktion

Funktion:
Als Funktionswert wird das Zeichen geliefert, das den als Funktionsargument angegebenen (dezimalen) Wert des ASCII-Codes hat.

Beispiel:

```
10 I=32
20 A$=CHR$(I+33)
30 PRINT A$
```

Dieses Programmstück bewirkt die Ausgabe des Zeichens A auf dem Bildschirm, da das Zeichen A die (dezimale) ASCII-Codierung 65 hat.
Schauen Sie sich nun einmal die darstellbaren Zeichen des Zeichensatzes auf dem Bildschirm an:

```
10 CLS
20 FOR I=32 TO 127
30 PRINT CHR$(I);
40 NEXT I
50 PRINT:PRINT
60 PAUSE
70 FOR I=128 TO 255
80 PRINT CHR$(I);
90 NEXT I
100 PRINT:PRINT:END
```

Nach der Anzeige der alphanumerischen Zeichen betätigen Sie [CONT] und erhalten die Grafikzeichen.

Wandlung Zeichenkette ←→ Zahl

Format:
VAL(*zeichenkette*)
zeichenkette - Zeichenkettenausdruck, der die Zeichenkettendarstellung einer Zahl enthält

Typ: BASIC-Funktion

Funktion:
Als Funktionswert wird die Zahl geliefert, die der im Argument übergebenen Zeichenkette entspricht. Beginnt diese nicht mit einer Ziffer, einem Dezimalpunkt oder einem Vorzeichen, wird der Funktionswert Null angenommen.

Hinweis:
In der Argumentzeichenkette nach einer Ziffer auftretende Sonderzeichen oder Buchstaben werden, wie alle nachfolgenden Zeichen, nicht weiter ausgewertet. Davon ausgenommen sind Zahlen in Gleitkomma-darstellung.

Beispiele:
Vollziehen Sie die nachfolgenden Anweisungen im Kommandomodus nach!

```
> A$="1234": PRINT VAL(A$)+1000
2234
> PRINT VAL(A$+".5")
1234.5
> PRINT VAL("5152-45-56047")
5152
```


Format:
STR$(ausdruck)
ausdruck - numerischer Ausdruck

Typ: BASIC-Funktion

Funktion:
Als Funktionswert wird die Zeichenkette geliefert, die d des im Argument angegebenen numerischen Ausdruck stellt.

Hinweis:
Wenn der Wert des numerischen Ausdrucks nicht negativ ist, bekommt die Zeichenkette ein Leerzeichen (auf der Position des Vorzeichens) vorangestellt.

Beispiele:

```
>PPINT STR$(1234+1000)
2234
>A=-11.77:PRINT A,STR$(A)
-11.77   -11.77
```

Länge einer Zeichenkette

Format:
LEN(*zeichenkette*)
zeichenkette - zeichenkettenausdruck

Typ: BASIC-Funktion

Funktion:
Die Anzahl der in der Zeichenkette enthaltenen Zeichen (einschließlich nicht darstellbarer und Leerzeichen) wird als Funktionswert geliefert.

Beispiel:

```
>T$="robotron Z9001":PRINT LEN(T$)
  14
>T$="":PRINT LEN(T$)
  0
```

Übernahme einer Teilzeichenkette

Format:
LEFT$(*zeichenkette*, *zeichenanzahl*)
zeichenkette - Zeichenkettenausdruck
zeichenanzahl - Anzahl der aus der Zeichenkette (von links beginnend) zu übernehmenden Zeichen; Wert zwischen 0 und 255

Typ: BASIC-Funktion

Funktion:
Der Funktionswert entspricht der Zeichenkette, die, von links beginnend, aus der angegebenen Anzahl Zeichen der Zeichenkette im Argument gebildet wird.

Hinweis:
Ist die geforderte Zeichenanzahl größer als die vorhandene, wird die gesamte Zeichenkette übernommen. Falls die gewünschte Zeichenanzahl Null ist, wird die leere Zeichenkette zugewiesen.

Beispiel:

```
>A$="ABCDEF":PRINT LEFT$(A$,3)
ABC
```

Format:
RIGHT$(*zeichenkette*, *zeichenanzahl*)
zeichenkette - Zeichenkettenausdruck
zeichenanzahl - Anzahl der aus der Zeichenkette (von rechts beginnend) zu übernehmenden Zeichen; zwischen 0 und 255

Typ: BASIC-Funktion

Funktion:
Der Funktionswert entspricht der Zeichenkette, die, von rechts beginnend, aus der angegebenen Anzahl Zeichen der Zeichenkette im Argument gebildet wird.

Hinweis:
Siehe Hinweis zu LEFT$

Beispiel:

```
>A$="ABCDEF":PRINT RIGHT$(A$,3)

DEF
```

Format:
MID$(*zeichenkette*, *ab_position* [,*zeichenanzah*l])
zeichenkette - Zeichenkettenausdruck
ab_position - Position des ersten zu übernehmenden Zeichens in *zeichenkette*; Wert zwischen 1und 255
zeichenanzahl - Anzahl der aus *zeichenkette* (von *ab_position* nach rechts fortschreitend) zu übernehmenden Zeichen; Wert zwischen 0 und 255

Typ: BASIC-Funktion

Funktion:
Der Funktionswert entspricht der Zeichenkette, die, von der angegebenen Position nach rechts fortschreitend, aus der Zeichenkette im Argument gebildet wird. Ist eine Zeichenanzahl angegeben, werden maximal so viele Zeichen übernommen.

Hinweise:
1. Liegt die *ab_position* hinter dem letzten vorhanden Zeichen, wird die leere Zeichenkette geliefert.
2. Ist die geforderte Zeichenanzahl ab vorgegebener Position größer als die vorhandene, wird die gesamte restliche Zeichenkette zugewiesen.

Beispiel:

```
>A$="ABCDEF":PRINT MID$(A$,3,2)

DEF
```

Suche einer Zeichenkette in einer Zeichenkette

Format:
INSTR(*zeichenkette1*, *zeichenkette2*)
zeichenkette1,2 - Zeichenkettenausdrücke

Typ: BASIC-Funktion

Funktion:
Es wird ermittelt, ob die *zeichenkette1* vollständig in der *zeichenkette2* enthalten ist. Als Funktionswert wird die Position (des ersten Zeichens) der *zeichenkette1* bezüglich ihres ersten Auftretens in der *zeichenkette2* angenommen. Wird die *zeichenkette1* nicht gefunden, ergibt sich der Funktionswert Null.

Beispiel:

```
T1$="01":PRINT INSTR(T1$,"robotron Z9001")

13
```

Wiederholung einer Zeichenkette

Format:
STRING$(*wiederholungen*, *zeichenkette*)
wiederholungen - numerischer Ausdruck, dessen ganzzahliger Wert festlegt, wie oft die Zeichenkette zu wiederholen ist; Wert zwischen 1 und 255
zeichenkette - Zeichenkettenausdruck

Typ: BASIC-Funktion

Funktion:
Der Funktionswert entspricht der Zeichenkette, die durch die angegebene Anzahl von Wiederholungen der Zeichenkette im Argument entsteht.

Hinweis:
Die Länge der erzeugten Zeichenkette darf 255 Zeichen nicht überschreiten.

Beispiele:

```
>PRINT STRING$(5, "Z9001 ")
Z9001 Z9001 Z9001 Z9001 Z9001
```

Unterstreichen Sie nun noch eine Überschrift durch Wiederholung des Zeichens mit dem ASCII-Code 160. Die LEN-Funktion nimmt Ihnen die Mühe ab, die Zahl der erforderlichen Wiederholungen zu ermitteln.

```
10 NE$="*** robotron Z9001 ***"
20 PRINT NE$
30 PRINT STRING$(LEN(NE$),CHR$(160))
```

Informieren Sie sich bitte auch im Abschnitt 6.2. über weitere Anwendungsmöglichkeiten der Zeichenkettenfunktionen.

Dieses Programmierhandbuch wurde verfaßt
von einem Autorenkollektiv
des VEB Robotron-Meßelektronik »Otto Schön« Dresden
Dr.-Ing. Hans-Jürgen Busch
Dr.-Ing. Joachim Haase
Dr. rer. nat. Gert Keller
Dr.-Ing. Hans-Jörg Nowottne

DEWAG Dresden . ATN 33442 031/4 . Regie: Mros; Gestaltung: Bucher
6/85a . Jt 2234/85 und Jt 2235/85 . II-13-1