ENTERPRISE®
COMPUTERS

BASIC-Kurs
für
Einsteiger

(c) 1986 by ENTERPRISE Computers GmbH

Inhaltsverzeichnis

Einleitung 5

1. Grundlagen 6

1.1 Computer im Haushalt 6

1.2 Begriffe 8

1.3 Zahlensysteme 13

1.3.1 Dualzahlen 13

1.3.2 Hexadezimalzahlen 21

2. Grundlagen 25

2.1 Direkter Modus 25

2.2 Programmiermodus 29

2.3 Befehle 31

2.4 Anweisungen 42

3. Programmentwicklung 65

3.1 Der Erste Schritt 67

3.2 Der Zweite Schritt 68

3.3 Der Dritte Schritt 70

3.4 Vierter und Fünfter Schritt 78

4. Externe Datenträger 86

4.1 Der Cassettenrecorder 87

4.2 Das Diskettenlaufwerk 100

5. Der Drucker 107

Anhang

Schlußwort.....113

Tabelle.....114

Lösungen.....115

Vorbereitung einer Diskette.....120

Aufbau Mikrocomputer.....121

Beispielprogramm.....122

Formular.....123

Einleitung

In vielen Gesprächen mit Enterprise - Besitzern wurde ich immer wieder auf einen BASIC - Kurs angesprochen. Ich studierte also das Handbuch, und stellte fest, daß es zwar ungewöhnlich gut geschrieben ist, aber dennoch ist es für den Anfänger nicht ausreichend

Bei den Möglichkeiten, die der `Enterprise` bietet, ist das jedoch normal. Bei meinen Gesprächen habe ich mir alle Probleme notiert. Die häufigsten habe ich dann herausgesucht und in einen BASIC - Kurs umgesetzt. Dabei stellte icht fest, daß der Stoff ausreicht, um mehrere Bücher zu füllen.

Was erwartet Sie nun im ersten Teil?

Zunächst einmal müßen wir uns mit allgemeinen Grundlagen beschäftigen. Diese sind wichtig, damit Sie die Arbeitsweise Ihres Computers kennenlernen. Danach geht es dann richtig los. Sie lernen, Programme zu entwickeln, diese richtig zu erstellen und zu dokumentieren. Sie lernen den Umgang mit Cassettenrecorder und Diskettenlaufwerk. Da der Enterprise Drucker nur mit englischer Beschreibung ausgeliefert wird, und das Computerhandbuch den Einsatz des Druckers nicht behandelt, werden wir uns mit diesem Gerät besonders beschäftigen.

Zu diesem Kurs können Sie auch alle Programme auf einem Datenträger erhalten. Die Kassette kostet 10,- DM, die Diskette 15,- DM. Sie erhalten den Datenträger bei mir gegen Einsendung des Betrages oder eines Verrechnungschecks. Bei einer Nachnahmelieferung kommen noch die Versandkosten hinzu.

Doch nun genug der Vorrede. Wir wollen nun endlich mit unserer Arbeit beginnen. Ich wünsche Ihnen viel Spaß und noch mehr Erfolg beim Lernen.

Jürgen Frölje
Schneckenburger Str. 11
8000 München 80

München 1986

1. Grundlagen

1.1 Der Computer im Haushalt

Seit 1979 befinden sich die Home- und Personalcomputer auf dem Vormarsch. Am Anfang waren es die `Computerfreaks`, die Tausende von Mark in ihren ersten Computer steckten. Es wurden nur wenige Computer verkauft, da ein System mit 48 KB RAM und einem Diskettenlaufwerk (ca. 100 KB) über 10.000,-- DM kostete. Sound und Grafik waren für diese ersten Computer noch Fremdworte. Außer einer primitiven Blockgrafik und eventuell einen einheitlichen Signalton war nichts vorhanden.

Als die Firmen Apple, Tandy und Commodore mit ihren ersten Geräten auf den Markt kamen, wurden die Computer langsam erschwinglich. Für ca. 2.000,-- DM gab es schon einen Rechner mit Bildschirm, 4 KB RAM und einen integrierten Cassettenrecorder. Mit diesen `niedrigen` Preisen wurden die Computer nun auch für breite Schichten interessant. Immer mehr Geräte fanden nun ihren Weg in die deutschen Haushalte. Nachdem nun die Preise durch größere Produktionen immer weiter fielen, wurden nun auch Grafik und Sound weiterentwickelt.

Mit dem VC 20 gelang Commodore dann der große Durchbruch. Dieser legendäre Computer bot zum ersten Mal Sound und Farbgrafik für weniger als 900,-- DM. Außerdem benötigte man keinen Monitor mehr, da der VC 20 an jeden normalen Fernseher anschliessbar war. Nun begann auch die Zeit der vielen kleinen Anbieter. In aller Welt entstanden Computer Firmen. Jeder wollte am beginnenden Computerboom teilhaben. Viele dieser Firmen erlebten nicht einmal die Markteinführung ihres Computers. Andere konnten den Preisverfall nicht ausgleichen.

Mit dem C64 gelang der Firma Commodore (mittlerweile zum größten Hersteller aufgestiegen) ein neuerlicher Paukenschlag. Dieser Computer setzte einen Standard. Er verfügt in der Grundausstattung über 64 KB RAM. Grafik wird mit 16 Farben und 320 x 200 Bildpunkten zum Vergnügen. Der Sound kommt sogar mit 3 Tongeneratoren und einem Rauschgenerator. Der C64 kostete am Anfang ca. 1.800,-- DM. Ein vergleichbares System war nicht auf dem Markt. Es dauerte aber nicht lange, und die Konkurrenz brachte ähnlich leistungsstarke Geräte auf den Markt. In der Folgezeit entstand ein Kampf um Marktpositionen, wie er bis dahin nur selten zu sehen war. Mit immer leistungsfähigeren Computern und fallenden Preisen wird um die Gunst der Käufer gerungen.

Was aber macht ein Computer im Haushalt?

Da ein Computer normalerweise für umfangreiche und durchaus ernste Aufgaben benötigt wird, mußten die Hersteller Aufgaben und Bereiche suchen, die in jedem Haushalt zu finden sind.

Ein guter Grund ist der immer stärkere Einsatz von Computern in der Wirtschaft, Forschung, Industrie und Verwaltung. Viele Berufe kommen heute ohne Computer nicht mehr aus. Es ist also von Vorteil, einen Computer bedienen zu können. Was liegt also näher, als sich in seiner Freizeit mit dieser Technik zu beschäftigen. Allein dieser Bereich rechtfertigt aber nicht derart hohe Umsatzzahlen. Früh wurde erkannt, daß ein Computer ein sehr vielseitiges Spielzeug sein kann. Die ersten Computerspiele waren noch recht einfach aufgebaut. Durch die schnelle Entwicklung von Sound und Grafik aber wurden die Spiele immer aufwendiger und komplizierter. Heute gibt es zum Beispiel für einige Computer mehrere tausend verschiedene Spiele. Kein Wunder also, daß der Computer zuhause in über 90% aller Anwendungsfälle als besseres Telespiel benutzt wird.

Man mag nun über diese Seite des Computers denken, was man will (auch alte Programmierhasen spielen gelegentlich mit ihrem Handwerkszeug). Tatsache ist, daß der Computer zu wesentlich mehr eingesetzt werden kann, als nur für Telespiele.

Durch die Programmiersprache `BASIC` wird der Einstieg in die Programmierung sehr leicht gemacht. Und es macht auch viel mehr Spaß, mit selbstprogrammierten Spielen.

Sollten Sie schon Kenntnisse über die < E D V > haben, können Sie die nächsten Seiten überschlagen

1.2 Begriffe

Bevor wir nun in unsere eigene Programmentwicklung einsteigen, müßen wir erst einmal einige Begriffe klären, die immer wieder auftauchen. Wenn Sie einmal ein Prospekt aufschlagen, tauchen da eine Reihe von Bezeichnungen auf, die dem Laien oftmals unverständlich erscheinen. Oft entsteht dann der Eindruck, es mit einer Fremdsprache zu tun zu haben, die nur stellenweise mit unserer deutschen Sprache übereinstimmt. Zum besseren Verständnis werden wir an dieser Stelle nun die wichtigsten Begriffe klären.

Das gesamte Umfeld wird allgemein als EDV bezeichnet. Die Abkürzung E D V steht für " Elektronische Daten - Verarbeitung". Der Begriff wird erst richtig deutlich, wenn Sie sich einmal vor Augen halten, daß das Arbeiten mit Papier, Kugelschreiber und Karteikarten auch eine Datenverarbeitung ist. Hierbei handelt es sich um eine manuelle Datenverarbeitung. Als Daten werden alle Informationen bezeichnet, die sich in irgendeiner Form verwerten, auswerten, manipulieren oder sammeln lassen. Als Daten bezeichnet man zum Beispiel Namen, Adressen, Geburtsdaten, Einkommen, Telefonnummern, Maße, Gewichte usw.. Diese Liste ließe sich nun beliebig fortführen.

Bei der Bezeichnung der Speicher werden zwei Begriffe verwendet. Um einen Computer richtig einschätzen zu können (hierbei geht es nicht um technische Details, sondern um eine Einschätzung, ob das Gerät für die eigenen Aufgaben geeignet ist), ist es wichtig, diese beiden Begriffe in ihrer Bedeutung zu kennen und auch verstanden zu haben.

Der eine Speicher wird als ROM - Speicher bezeichnet.

R = Read

O = Only

M = Memory

Übersetzt würde es etwa lauten:"Nur Lese Speicher". Dieser Speicher ist also nur zum Lesen geeignet. Normalerweise ist eine Datenverarbeitung mit diesem Speicher nicht gut möglich. Jeder Computer benötigt aber Basis-Informationen. Beim Einschalten des Computers muß der Speicher organisiert werden. Weiterhin muß der Computer in die Lage versetzt werden, Arbeiten nach einem bestimmten Muster ausführen zu können. Wenn diese Daten nicht zu Verfügung stehen, ist das Gerät nicht einmal in der Lage, Eins und Eins zu addieren. Alle diese Basis-Informationen bezeichnet man als Betriebssystem. Da diese Informationen immer wieder benötigt werden, und normalerweise nicht verändert werden, werden sie schon im Werk in einen Speicherbaustein eingebrannt. Diese Speicher behalten ihre Information einige Jahre, ohne dabei eine Energie zu benötigen.

Der andere Speichertyp wird als RAM bezeichnet.

R = Random

A = Access

M = Memory

Der RAM-Speicher ist der sogenannte Arbeitsspeicher. Diesen Speicher verwenden Sie zum Beispiel, wenn Sie spielen oder programmieren. Auch andere Programme, die von einem Datenträger in den Computer eingelesen werden, verwenden den RAM-Speicher. Dieser Speicher kann sowohl gelesen, als auch beschrieben werden.

A C H T U N G : RAM-Speicher verlieren ihre Informationen, wenn Sie Ihren Computer ausschalten! ! !

Den RAM-Speicher können Sie jedoch nicht vollständig für Ihre eigenen Programme verwenden. Ein Teil dieses Speichers wird als sogenannter Bildschirmspeicher benötigt. Alle Informationen, die auf dem Bildschirm dargestellt werden, müssen in einen dafür vorgesehenen Teil des Speichers abgelegt werden. Normalerweise wird ein Bild auf dem Monitor 50 mal pro Sekunde neu aufgebaut, d. h. die Informationen aus dem Bildschirmspeicher werden 50 mal pro Sekunde aus diesem gelesen und auf dem Bildschirm dargestellt. Je größer nun die Information auf dem Bildschirm, je größer muß auch der Bildschirmspeicher sein. Damit aber

noch nicht genug. Für bestimmte Aufgaben (Sie werden diese Aufgaben an anderer Stelle noch genau kennenlernen), benötigt der Computer reservierten Speicherplatz. Es wird also zusätzlich weiterer RAM-Speicher benötigt. Bei Ihrem Computer haben Sie je nach Modell 49 KB RAM oder 114 KB RAM zur Verfügung.

Die Angaben der Speichergrößen werden in KByte (KB) ausgedrückt. Beginnen müßen wir aber mit dem Bit. Ein Bit ist die kleinste darstellbare Informationseinheit in einem Computer. Dieses Bit kann zwei Zustände anzeigen oder beinhalten:

1 oder 0

ja oder nein

Strom oder kein Strom

Sie sehen, daß der Computer kann nur zwei Zustände darstellen. Auf diese Basis werden alle (auch noch so komplexen Programme) Vorgänge im Computer durchgeführt.

Die nächst größere Einheit ist das Byte. Ein Byte besteht aus 8 Bit. Wenn wir nun nach unseren Kenntnissen der Mathematik weiterrechnen, müßte ein KiloByte aus Eintausend Byte bestehen. Doch weit gefehlt. Wir müssen uns daran gewöhnen, daß in der EDV vieles anders ist. Nachdem wir dies nun zur Kenntnis genommen haben, verwundert es uns nicht mehr, daß ein KiloByte nicht aus Eintausend Byte besteht, sondern aus 1024 Byte. Demnach muß ein MegaByte aus 1024 KiloByte bestehen.

Ziel dieses Buches ist es, Ihnen die Grundlagen der Programmier sprache BASIC näher zu bringen. Haben Sie sich schon einmal gefragt, was die Bezeichnung "BASIC" bedeutet? Das Wort "BASIC" ist kein Eigenname sondern wie wie in der EDV üblich, eine Abkürzung.

B = Beginners

A = All Purpose

S = Symbolic

I = Instruction

C = Code

Die Übersetzung dafür lautet: "Programmiersprache für Anfänger und universellen Einsatzbereich". Diese Sprache wurde an dem Dartmouth College in den USA für all diejenigen entwickelt, die bislang nichts mit Computern zu tun hatten, diese aber programmieren wollten oder mußten.

Am Anfang war es eine eher bescheidene Sprache, die dem Programmierer kaum Entfaltungsmöglichkeiten erlaubte. Im Laufe der Jahre wurde diese Sprache zu einer sehr leistungsfähigen Programmiersprache ausgebaut.

Zwei Begriffe, die immer wieder auftauchen sind, die Bezeichnungen:

Hardware

Software

Hardware (übersetzt: Harte Ware) ist alles, was Sie anfassen können. Ihr Computer, der Bildschirm, Drucker, Diskettenlaufwerk, Tastatur usw.. Diese Liste ließe sich noch beliebig fortführen. Wichtig ist nur, daß Hardware alles ist, was sie sehen und anfassen können.

Software (übersetzt: Weiche Ware) ist alles, was der Computer zum Arbeiten benötigt. Genau genommen sind es, das Betriebssystem, alle Programme und Daten. Die Datenträger gehören aber zur Hardware.

Wir haben jetzt zweimal den Begriff Datenträger erwähnt. Es ist nun an der Zeit, auch diesen Begriff zu klären. Als Datenträger bezeichnet man alle Hilfsmittel, auf denen Daten und Programme gespeichert werden können. Dazu gehören Cassetten, Disketten und Festplatten. Es gibt aber auch Datenträgen, die zwar Daten speichern können, sie aber nicht dem Computer zur Verfügung stellen können. Wir haben am Anfang den Begriff manuelle Datenverarbeitung kennengelernt. In dieser Art der Datenverarbeitung gibt es natürlich auch Datenträger. Zum Beispiel Karteikarten, Notizbücher und Papier. Genaugenommen ist auch das Papier, auf dem dieses Buch geschrieben wurde, ein Datenträger. Dadurch, daß Sie dieses Buch lesen, entnehmen Sie ihm ja gewissermaßen Daten.

Wenn Sie nun an Ihren Computer ein Diskettenlaufwerk anschließen wollen, haben Sie zwei Möglichkeiten. Erstens Sie kaufen sich ein komplettes Laufwerk, und stecken dieses dann an den Computer an. Die zweite Möglichkeit ist, Sie kaufen sich nur einen Controller und verwenden dann ein Laufwerk Ihrer Wahl. In beiden Fällen muß aber die Verbindung zwischen Computer und Diskettenlaufwerk vorhanden sein. Als Verbindung meine ich hier nicht das Kabel, sondern eine Einrichtung, die eine Kommunikation zwischen diesen Geräten ermöglicht. Diese Einheit bezeichnet man als Interface. Wenn wir nun ein Diskettenlaufwerk anschließen, schaffen wir durch das Interface die Voraussetzung, diese beiden Geräte miteinander zu verbinden. Das gleiche gilt auch für den Drucker oder einen Cassettenrecorder.

Nachdem wir nun durch ein Interface die Verbindung zwischen Computer und Diskettenlaufwerk geschaffen haben, benötigen wir noch eine Software. Diese Software ist nötig, da normalerweise der Betrieb vom Grundgerät nicht auf die Verwendung eines Diskettenlaufwerkes ausgelegt ist. Diese Software wird als DOS bezeichnet.

D = Disk

O = Operating

S = System

In der DOS-Software wird festgelegt, welche Adresse ein Diskettenlaufwerk belegen kann. Es wird bestimmt, auf welche Art die Disketten beschrieben werden. Weiterhin wird die Datenorganisation und Struktur festgelegt. Beim Enterprise haben Sie gleich Zwei Versionen:

1. EXDOS

2. IS-DOS

An dieser Stelle verlassen wir nun die Begriffe. Für den Anfang soll es auch erst einmal genug sein. Wenn Sie alle diese Begriffe verstanden haben, können Sie wenigstens mitreden, wenn sich Ihre Bekannten im "Computer-Deutsch" unterhalten.

1.3 Zahlensysteme

Wenn Sie sich einmal umschauen, werden Sie feststellen, daß in unserer Umwelt ein Zahlensystem überall Gültigkeit hat: das Dezimalsystem.

Im Dezimalsystem gibt es zehn Zahlen: 0 bis 9. Eine einstellige Zahl kann einen Wert zwischen 0 und 9 darstellen. Wollen wir einen größeren Wert darstellen, benötigen wir dazu eine zweite Stelle. Nun sind wir in der Lage 100 Werte darzustellen. Diese bewegen sich zwischen 0 und 99. Der nächste größere Wert ist 100. Hier benötigen wir eine dritte Stelle. Diese Kette läßt sich, wie wir alle aus der Praxis wissen, beliebig fortführen. Auf das Dezimalsystem wollen wir hier auch nicht näher eingehen, da Sie dieses System ja zur Genüge kennen. Unser Computer allerdings, kann mit diesem Zahlensystem nichts anfangen. Es paßt einfach nicht in seinen Arbeitsbereich. Ich habe dieses Zahlensystem aber trotzdem erwähnt, da wir es in den folgenden Vergleichen immer wieder als bekanntes System heranziehen werden. Auf diese Weise ist es für Sie auch einfacher, die anderen Zahlensysteme zu verstehen.

Mit welchem Zahlensystem arbeitet nun unser Computer?

1.3.1 Dual - System (Binär)

Wir haben im vorigen Kapitel erfahren, daß der Computer nur zwei Zustände verarbeiten kann. Diese beiden Funktionen werden im Computer folgendermaßen dargestellt:

Strom oder kein Strom

Wir bezeichnen diese beiden Zustände mit 1 oder 0. Da es sich bei dieser Darstellung um zwei Zahlen handelt, gibt es dafür auch sicherlich ein Zahlensystem mit den dazu gehörigen Rechenregeln. Dieses Zahlensystem nennt man Dual - System oder auch Binär - System. Zur Darstellung von verschiedenen Werten hier erst-einmal ein Beispiel:

Dezimal	Dual
0	0
1	1
2	10

Dezimal	Dual
3	11
4	100
5	101
6	110
7	111
8	1000

Sie sehen, daß auch in diesem System alle Werte aus dem uns bekannten Dezimal-System darstellbar sind. Die Darstellung ist nur etwas umständlicher, da dieses System ja nur über zwei Werte verfügt. Auch ist es für uns, die wir an das Dezimalsystem gewöhnt sind, sehr schwer, die Wertigkeit einer Zahl sofort zu verstehen. Natürlich können Sie mit dem Dual-System auch rechnen. Uns interessiert hier erst einmal nur die Addition. Weitere Rechenarten werden Sie dann im Teil 2 kennenlernen. Das folgende Beispiel enthält einige Additionen (Rechnungen im Dezimalsystem kennzeichnen wir hier mit D).

```
D   0        00      D   0       00      D  1     01
  + 0      + 00        + 1     + 01       + 1   + 01
  ---      ----        ---     ----       ---   ----
    0        00          1       01         2     10
  ===      ====        ===     ====       ===   ====
```

Bis hier scheint noch alles in Ordnung zu sein, wenn wir einmal von der Tatsache absehen, daß 1 + 1 = 10 ergibt. Aber was geschieht, wenn wir 2 und 2 addieren:

```
D   2        10
  + 2      + 10
  ---      ----
    4       100
  ===      ====
```

Das Ergebnis kann nicht mehr in zwei Stellen erfasst werden. Wir müßen eine dritte Stelle schaffen. Versuchen Sie nun einmal 4 und 4 zu addieren:

```
D   4       100
  + 4       100
  ---       ---
    8       ???
  ===       ===
```

Jede Stelle ist in der Darstellung im Computer ein Bit. Wie Sie in den Grundlagen bisher erfahren haben, sind in einem Byte, acht Bit enthalten. Uns interessiert nun, welchen dezimalen Wert wir in einem Byte darstellen können. Dazu betrachten wir uns einmal den folgenden Auszug aus einer Tabelle:

dezimal	binär (dual)
0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0011
4	0000 0100
5	0000 0101
.....	....
20	0001 0100
21	0001 0101
22	0001 0110
23	0001 0111
.....	....
253	1111 1101
254	1111 1110
255	1111 1111

Wenn wir nun, wie in der EDV üblich, die Null mitzählen, haben wir also die Möglichkeit, in einem Byte 256 verschiedene Werte darzustellen. Betrachten Sie sich noch einmal die Tabelle, und versuchen Sie, hinter das Geheimnis der Darstellung zu kommen. Versuchen Sie weiterhin, die Wertigkeiten der einzelnen Stellen zu ermitteln, bevor wir auf der nächsten Seite darauf eingehen. Das Grundprinzip der Addition haben wir ja schon angesprochen. Führen Sie bitte folgende Addition durch:

```
   0010 1101
+  1011 0100
------------
   ???? ????
============
```

Nehmen Sie sich dazu bitte ein Blatt Papier und führen Sie die Addition durch. Das Ergebnis muß dann wie folgt aussehen:

```
   0010 1101
+  1011 0100
------------
=  1110 0001
============
```

Nachdem Sie nun das Ergebnis haben, wollen wir es auch mit dem uns bekannten Dezimal-System überprüfen. Dafür legen wir uns nun drei Tabellen an.

```
Einer  I  Nullen
-------I--------
       I
       I
       I
       I
       I
```

So sollten Ihre drei Tabellen aussehen.

Bevor wir nun unsere Einträge in die Tabellen eintragen, müssen wir noch die Wertigkeiten der einzelnen Stellen kennen.

255 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 128

Stellen = 1 2 3 4 5 6 7 8

Diese Wertigkeit ist bei der Darstellung eines Bytes von rechts nach links zu lesen. Wenden wir uns nun unserer ersten Tabelle zu:

Schreiben Sie vor die erste Tabelle auch die erste Dualzahl.

0010 1101

```
Einer  I  Nullen
-------I--------
       I
   1   I
       I    2
   4   I
   8   I
       I   16
  32   I
       I   64
       I   128
-----------
  45   +  210
```

Wir lesen nun die Dualzahl von rechts nach links. Tragen Sie dabei immer gleich die Wertigkeit der entsprechenden Stelle ein

Am Ende addieren wir beide Seiten. Das Ergebnis wird nun wieder addiert.

Das Ergebnis beider Zahlen muß 255 lauten. Unsere gesuchte Zahl beträgt 45, da wir ja nur die Einer benötigen, um den Wert einer Dual - Zahl zu bestimmen. Die Nullen werden nur zur Gegenprobe addiert. Wenn Einser und Nullen dann 255 ergeben, können wir davon ausgehen, daß wir richtig gerechnet haben.

Rechnen Sie nun die zweite und dritte Zahl anhand Ihrer noch leeren Tabellen aus. Die zweite gesuchte Zahl muß dezimal 180 ergeben. Die dritte Zahl dient uns nun zur Probe. Ihr Dezimalwert beträgt 225. Wir können nun die Probe machen.

```
D    45          0010 1101   =  45
   + 180         1011 0100   = 180
   -----         ---------     ---
     225         1110 0001   = 225
```

Wenn Sie das gleiche Ergebnis haben, ist Ihre Rechnung korrekt. Hier nun noch ein Tip, wie Sie am besten vorgehen. Bei der Addition von Binärzahlen gehen Sie genauso vor, wie bei der Dezimal - Addition, mit der Ausnahme, daß 1 und 1 nicht 2 sondern 0 mit Übertrag 1 ergibt. Auf diese Art werden wir nun die Dezimalzahlen 11 und 7 addieren.

```
D    11          0000 1011
   +  7        + 0000 0111
   ----        -----------
   = 18        = 0001 0010
   ====        ===========
```

Rechnung: wir beginnen rechts: 1 + 1 = 0 mit Übertrag 1
2. Stelle: 1 + 1 + Übertrag 1 = 1 mit Übertrag 1
3. Stelle: 0 + 1 + Ü 1 = 0 mit Übertrag 1
4. Stelle: 1 + 0 + Ü 1 = 0 mit Übertrag 1
5. Stelle: 0 + 0 + Ü 1 = 1 ohne Übertrag

Die Stellen Sechs bis Acht ergeben durchweg 0.

Wir rechnen von nun an immer mit acht Stellen. Da wir mit diesen, genau ein Byte darstellen können.

Nun gibt es wieder Arbeit für Sie. Wandeln Sie anhand von Tabellen alle drei Binärzahlen in Dezimalzahlen um. Vergessen Sie nicht die Nullen zwecks Gegenprobe auch zu addieren. Wenn das Ergebnis, mit dem oben aufgeführten Ergebnis identisch ist, haben Sie Ihre Aufgabe gut gelöst.

Bevor wir nun die Addition verlassen, wollen wir das eben Gelernte durch einige Übungen noch ein wenig vertiefen.

Ü 1.3.1

```
  1011 0000          0001 1011
+ 0001 1011        + 0111 1111
-----------        -----------
= ???? ????        = ???? ????
===========        ===========

  1100 0111          0010 1010
+ 0010 0111        + 1010 0111
-----------        -----------
= ???? ????        = ???? ????
===========        ===========

  0110 0000          0000 1111
+ 1001 1111        + 1111 0000
-----------        -----------
= ???? ????        = ???? ????
===========        ===========
```

Rechnen Sie die Aufgaben aus und überprüfen Sie die Ergebnisse mit Hilfe von Tabellen. Die richtigen Lösungen finden Sie im Anhang. Rechnen Sie aber bitte erst die Übungen durch, bevor Sie in den Anhang schauen.

Oftmals müssen Sie eine Dezimalzahl in eine Dualzahl umwandeln. Hierzu gibt es eine einfache Methode. Wir benutzen dazu wieder einmal eine Tabelle:

```
1 x 2^0  =   1
1 x 2^1  =   2
1 x 2^2  =   4
1 x 2^3  =   8
1 x 2^4  =  16
1 x 2^5  =  32
1 x 2^6  =  64
1 x 2^7  = 128
```

Wenn wir nun die Dezimalzahl 177 in eine Dualzahl umwandeln wollen, beginnen wir von links nach rechts und in

der Tabelle von unten nach oben.

Ist 128 in 177 enthalten? Wenn wir, wie in diesem Fall, die Frage mit ja beantworten können, ist die achte Stelle unserer Dualzahl eine 1. Nun ziehen wir von 177 den Wert der achten Stelle ab. Es bleiben 49. Ist 64 in 49 enthalten? Nein, also ist die 7. Stelle eine Null. Ist 32 in 49 enthalten? Ja, die 6. Stelle ist 1 und wir ziehen den Wert der 6. Stelle von 49 ab. Es bleiben 17. Ist 16 in 17 enthalten? Ja, die 5. Stelle ist wieder 1. Auch jetzt ziehen wir den Wert der 5. Stelle ab. Es bleiben 1. Ist 8 in 1 enthalten? Nein, also 4. Stelle eine Null. Ist 4 in 1 enthalten? Nein, 3. Stelle ist Null. Ist 2 in 1 enthalten? Nein, 2. Stelle ist Null. Ist 1 in 1 enthalten? Ja, die erste Stelle wird 1 und wir ziehen den Wert von 1 ab. Es muß Null herauskommen.

Haben Sie das alles verstanden? Ich auch nicht. Da das ganze ein wenig unverständlich war, schauen wir uns die Umrechnung noch einmal an.

```
1... ....     128 in 177 enthalten

                177
              - 128
              -----
                 49  Rest

10.. ....     64 nicht in 49 enthalten

101. ....     32 in 49 enthalten

                49
              - 32
              ----
                17  Rest

1011 ....     16 in 17 enthalten

                17
              - 16
              ----
                 1  Rest

1011 0...     8 in 1 nicht enthalten

1011 00..     4 in 1 nicht enthalten

1011 000.     2 in 1 nicht enthalten

1011 0001     1 in 1 enthalten
```

```
  1
- 1
---
  0  Rest
=======
```

Auf diese Art können wir nun jede Dezimalzahl in eine Dualzahl umwandeln. Um ein wenig in Übung zu kommen, haben wir wieder ein paar Aufgaben für Sie. Auch hier sollten Sie die Regel beachten, erst rechnen und dann nachschlagen.

Ü. 1.3.2

Wandeln Sie folgende Dezimalzahlen in Dualzahlen um:

25 - 127 - 65 - 139 - 228 - 254 - 89 - 111

Ü. 1.3.3 Wandeln Sie die folgenden Dezimalzahlen in Dualzahlen um. Berechnen Sie das Ergebnis an hand der Dualzahlen, und überprüfen Sie die Ergebnisse mit Hilfe von Tabellen.

```
  23        44       128       71       99
+ 46      +  6     +  25     + 99     + 13
----      ----     -----     ----     ----
= ??      = ??     =  ??     = ??     = ??
====      ====     =====     ====     ====


  144       158       36       111      199
+  23     +  77     +201     + 132    +  22
-----     -----     ----     -----    -----
= ???     = ???     =???     = ???    = ???
=====     =====     ====     =====    =====
```

Soweit das Dual-System. Sie werden sich jetzt sicherlich fragen: "Was soll ich damit? Ich will doch nur in BASIC programmieren". Sie haben bestimmt recht. Zur normalen Programmierung benötigen Sie dieses Wissen nicht. Aber schon wenn Sie tiefer in die Programmierung einsteigen, ist dieses Wissen Grundvoraussetzung.

Es ist sehr umständlich, ein Byte in der uns bis jetzt bekannten Art darzustellen. Wenn Sie mehrere Bytes in dieser Form darstellen, verlieren Sie schnell die Übersicht. Aber wir haben ja noch ein weiteres Zahlensystem.

1.3.3 Hexa-Dezimal-System

Bevor wir uns diesem neuen Zahlensystem zuwenden, muß ich noch einmal darauf hinweisen, daß unser Computer nur im Dual-System arbeiten kann. Dieses neue Zahlensystem dient nur zu unserer Erleichterung.

Betrachten wir uns noch einmal die ersten vier Stellen eines Bytes.

.... 1111 = Dezimal 15

Wenn wir die Null mit einrechnen, haben wir also 16 Werte, die wir mit 4 Bit darstellen können. Was liegt also näher, als ein Zahlensystem mit der Basis 16 zu verwenden. Dazu betrachten wir uns einmal die nächste Tabelle:

Dezimal	Hex	Binär
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	1001
...	...	....
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Sie sehen, die Stellen von 0 bis 9 sind im Hexadezimalsystem und im Dezimalsystem gleich. Danach wird es im Dezimalsystem 2-stellig. Da sich unser Dezimalsystem auf die Basis 10 stellt, haben wir nur die Zahlen 0 bis 9 zur Verfügung. Im Hexadezimalsystem benötigen wir aber noch Fünf weitere Zahlen, bevor es 2-stellig wird. Wir nehmen also die Buchstaben A bis F noch dazu. Damit haben wir aber erst die Hälfte eines Bytes dargestellt. Schauen wir uns dazu einige Beispiele an.

Dezimal	Binär	Hex
10	0000 1010	0A
16	0001 0000	10
20	0001 0100	14
32	0010 0000	20
64	0100 0000	40
128	1000 0000	80
255	1111 1111	FF

Sie sehen in der letzten Zeile, daß der Wert 255 Dezimal in der Hexadezimalen Schreibweise mit FF dargestellt wird. Es ist also der letzte "zweistellige" Wert. Zum besseren Verständnis bauen wir uns wieder eine Tabelle auf, die die Schlüsselwerte enthält. Eine komplette Tabelle finden Sie im Anhang. Wir werden in dieser kurzen Tabelle ausschließlich Werte finden, die einem vielfachen der Basis entsprechen. Im Dezimalsystem würden wir eventuell folgende Werte darstellen 10, 20, 30, 40, 50 usw. Alle diese Zahlen entsprechen immer einem Vielfachen der Basis. Nun aber zu unserer Tabelle.

16	Dezimal	=	10	Hex
32	"	=	20	"
48	"	=	30	"
64	"	=	40	"
80	"	=	50	"
96	"	=	60	"

112	Dezimal	=	70	Hex
128	"	=	80	"
144	"	=	90	"
160	"	=	A0	"
176	"	=	B0	"
192	"	=	C0	"
208	"	=	D0	"
224	"	=	E0	"
240	"	=	F0	"

Fehlen uns jetzt nur noch 15 für unser Byte. Wir erinnern uns, daß Dezimal 15 in Hex mit F dargestellt wird. Damit ergibt Dezimal 255 = FF Hex. Damit Sie von vorn herein richtig arbeiten, werden wir in Zukunft Hex-Zahlen immer mit einem vorangestellten $ schreiben. Diese Schreibweise wurde in der EDV eingeführt, um eine Verwechslung mit Dezimalzahlen auszuschließen. Wenn Sie also folgende Zahl sehen > $10 < , wissen Sie, daß es sich nur um eine Hex-Zahl handeln kann. In diesem Fall mit dem Dezimalwert 16. Um die ganze Sache nun noch ein wenig zu erleichtern, teilen wir das Byte einfach in zwei Hälften. Diese tragen nun die Bezeichnungen Erstes Halbbyte und Zweites Halbbyte. Jetzt wissen Sie auch, warum ich bei der Darstellung eines Bytes nach 4 Stellen einen Zwischenraum gelassen habe.

```
Beispiel:    1011 0101     Dezimal = 181
           $   B    5      B in der zweiten Stelle ist 176
                           dazu addieren wir 5 und erhalten
                           181.
```

Betrachten wir uns noch einmal die Wertigkeiten. Im Ersten Halbbyte können wir die Werte von 0 bis 15 darstellen. Im zweiten Halbbyte die Werte 16 bis 240. Wenn wir beide maximalen Werte addieren erhalten wir 255. Damit Ihnen nun das Hex-System geläufig wird, wollen wir wieder einige Aufgaben rechnen. Zunächst jedoch erst einmal wieder ein paar Beispiele.

```
  $ 5          5      Probe  $10 = 16
+ $ 11       + 17             $16 = 22 (16 + 6)
------       ----
= $ 16       = 22
======       ====
```

```
   $ A7          167          Probe:  $ F0  = 240
 + $ 52        +  82                  $ F9  = 249
 ------        -----
 = $ F9        = 249
 ======        =====
```

Zur Vertiefung des Ganzen jetzt noch einige Übungen. Bitte denken Sie daran: Erst rechnen und dann im Anhang blättern.

Ü. 1.3.4 Wandeln Sie folgende Dezimalzahlen in Hexadezimalzahlen um.

8 - 23 - 75 - 112 - 180 - 224 - 254

Sollten Sie Probleme bei der Umwandlung haben? Denken Sie an die Umwandlung im Dualsystem. Sollten Sie aber dennoch Schwierigkeiten haben, wird Ihnen im Anhang geholfen.

Ü. 1.3.5 Wandeln Sie folgende Dualzahlen in Hexadezimale Zahlen um.

1010 0111 - 1110 1011 - 0101 1010

1101 1111 - 1100 0011 - 1011 0111

Ü. 1.3.6 Rechnen Sie folgende Aufgaben durch

```
  $ 23     $ 0F     $ A0     $ B7     $ AA
+ $ 17   + $ E9   + $ 11   + $ 05   + $ 11
------   ------   ------   ------   ------
```

Ü. 1.3.7 Wandeln Sie folgende Dezimalaufgaben in Hex um. Rechnen Sie dann die Aufgaben durch. Danach wandeln Sie alle Aufgaben in Binär um, und überprüfen Sie die Ergebnisse mit Hilfe von Tabellen.

```
  65     79     83     159    128    16
+ 93   + 120  + 22   +  83  + 127  + 15
----   -----  ----   -----  -----  ----
```

Nachdem Sie nun diese Aufgaben erfolgreich gelöst haben, werden wir die allgemeinen Grundlagen verlassen. Bevor Sie aber weiterarbeiten, sollten Sie 30 Minuten spaziern gehen. Es wird Ihnen gut tun.

2. B A S I C - G r u n d l a g e n

In diesem Teil werden wir uns mit Grundlagen beschäftigen, die einen direkten Bezug zur BASIC - Programmierung haben. Wir werden auch schon einige Funktionen der Sprache BASIC kennenlernen, ohne direkt auf das Warum einzugehen. Im Teil Befehle und Anweisungen, werden wir dann auch auf deren Bedeutung und Anwendung eingeben. Wir werden uns jedoch nur mit einem kleinen Teil der zur Verfügung stehenden Befehle und Anweisungen beschäftigen. Eine vollständige Liste können Sie Ihrem Handbuch entnehmen. Am Ende des Buches finden Sie dann auch noch eine Liste von Befehlen und Anweisungen, mit denen wir uns dann in den Bänden 2 und 3 dieses Kurses beschäftigen werden.

Sie haben die Möglichkeit, Ihren Computer in zwei Betriebsarten zu betreiben. Dazu brauchen Sie den Computer nicht umschalten, sondern nur einige Regeln zu beachten.

2.1 Direkter Modus

Anweisungen, die im Direktmodus eingegeben werden, werden vom Computer sofort nach dem Betätigen der `ENTER` - Taste ausgeführt.

Wie aber merkt der Computer, daß er sich im Direktmodus befindet?

Der Computer erkennt eine Anweisung im direkten Modus daran, daß keine Zeilennummer vor der Anweisung steht. Wenn also eine Zeile ohne vorstehende Zeilennummer eingegeben wird, versucht der Computer, diese sofort auszuführen.

W I C H T I G ! ! !

Anweisungen im Direktmodus werden nicht

in den Programmspeicher übernommen.

Nach Ausführung werden sie vom Computer

wieder vergessen.

Bevor wir uns die ganze Geschichte in Form von Beispielen etwas näher betrachten, wollen wir uns erst einmal die Anweisungen, die dort verwendet werden, etwas näher ansehen. Wir werden hier nur zwei Anweisungen verwenden.

1. PRINT

2. LET

Mit der PRINT - Anweisung sagen wir dem Computer, daß er etwas auf dem Bildschirm darstellen soll. Wir können nun direkt in einer PRINT - Anweisung den Text, den wir auf dem Bildschirm haben wollen, unterbringen. Wir können aber auch einen Text oder eine Zahl in Form einer Variablen in einer PRINT - Anweisung unterbringen. Um diese Variable mit einem Text oder einer Zahl zu belegen, benötigen wir die LET - Anweisung. Genug der Vorrede: Wir fangen am besten gleich mit unseren Beispielen für den Direktmodus an. Schauen Sie sich dabei die verwendeten Anweisungen bitte genau an. Da die Anweisungen direkt ausgeführt werden, haben Sie die Möglichkeit, die Bedeutung direkt zu erfahren und zu verstehen. Die Eingabe der Beispiele geht nun folgendermaßen vor sich. Sie tippen alle Teile, die groß geschrieben sind, und alle Teile, die in Klammern stehen. Worte, die in diesen Klammern < > stehen, stellen Tasten auf Ihrem Computer dar, die Sie betätigen sollen.

Beispiele

2.1.1 PRINT "HALLO" <ENTER>

Nach dieser Eingabe erscheint auf dem Bildschirm

```
HALLO
OK
```

Der Computer schreibt den Inhalt der Anführungszeichen auf den Bildschirm.
Damit ist seine Arbeit beendet. Die Anweisung

PRINT "HALLO"

befindet sich nicht mehr im Speicher.

OK sagt dem Anwender, daß der Computer auf weitere Anweisungen wartet.

2.1.2 PRINT 12+6 <ENTER>

Nach dieser Eingabe erscheint auf dem Bildschirm

```
18
OK
```

PRINT 12+6*(6-3) <ENTER>

Es erscheint auf dem Bildschirm

```
30
OK
```

Rechnen Sie einmal das Ergebnis auf dem Papier oder mit einem Taschenrechner nach. Sie werden feststellen, daß der Computer sich strikt an die mathematischen Regeln hält. Hier in unserem Fall: Punktrechnung geht vor Strichrechnung (ausgenommen sie steht in Klammern). Das werden wir gleich einmal probieren.

PRINT (12+6) * (6-3) <ENTER>

Auf dem Bildschirm erscheint

```
54
OK
```

Soweit die PRINT - Anweisung. Wie bereits erwähnt, die Anweisungen sind nicht mehr im Speicher. Sie stehen aber noch auf dem Bildschirm. Der Enterprise bietet nun eine Funktion, die nicht bei allen Computern zu finden ist. Betätigen Sie einmal den Cursor mit Hilfe des Steuerhebels auf Ihre letzte Anweisung. Nun betätigen Sie die <ENTER> - Taste. Aha, die Anweisung wird nochmal ausgeführt. Es besteht also eine Verbindung zwischen Bildschirmspeicher und Computer, die in zwei Richtungen wirkt. Aber das ist noch nicht alles. Wenn Sie nun den Steuerhebel weiter nach oben bewegen, werden Eingaben sichtbar, die nicht mehr auf dem Bildschirm zu sehen waren. Eine sehr nützliche Einrichtung die Sie bei anderen Computern dieser Preisklasse vergeblich suchen werden. Wie nützlich diese Einrichtung ist, werden Sie an anderer Stelle noch erfahren.

Kommen wir nun zur LET - Anweisung. Auf die Bedeutung von Variablen werden wir noch später eingehen. Sie sollen hier nur einen Vorgeschmack von dem erhalten, was Ihr Computer alles leistet.

Beispiele:

2.1.3 LET A=10 <ENTER>

LET B=20 <ENTER>

PRINT A*B <ENTER>

Es erscheint auf dem Bildschirm

```
200
OK
```

Gehen wir noch einen Schritt weiter:

LET A=15 <ENTER>

LET B=20 <ENTER>

LET C=A*B <ENTER>

PRINT C <ENTER>

Es erscheint auf dem Bildschirm

```
300
OK
```

Nun geben Sie bitte noch folgende Anweisungen ein:

PRINT A <ENTER>

Bildschirmausgabe

```
15
OK
```

PRINT B <ENTER>

Bildschirmausgabe

```
20
OK
```

Wie Sie sehen, bleiben Ihre Belegungen erhalten. Bevor wir nun in den zweiten Modus einsteigen, sollten Sie noch ein wenig üben. Den direkten Modus werden Sie aber in den seltensten Fällen benötigen. Er ist nur dann von Interesse, wenn Sie schnell etwas rechnen wollen.

2.2 Programmiermodus (Indirekter Modus)

Nun kommen wir in den Modus, den Sie am häufigsten gebrauchen werden. Immer wenn Sie ein Programm erstellen wollen, werden Sie in diesem Modus arbeiten. Damit nun aber Ihr Computer "merkt", daß Sie ein Programm schreiben wollen, müssen Sie nun Zeile für Zeile eine sogenannte Zeilennummer verwenden. Beginnen wir doch gleich einmal mit einem Beispiel:

```
1 PRINT "HALLO"     <ENTER>
```

Es passiert nichts ! ! !

Der Computer hat erkannt, daß Sie ein Programm schreiben wollen. Er verhält sich also ruhig und wartet auf Ihre nächste Eingabe.

Gehen wir nun einmal davon aus, daß unser Programm nur aus dieser einen Zeile besteht. Um dieses Programm nun zur Ausführung zu bringen, muß es erst gestartet werden. In der Programmiersprache BASIC gibt es dafür einen Befehl. Wir starten also unser Programm mit

```
RUN       <ENTER>
```

Bildschirmausgabe

```
HALLO
OK
```

Das Ergebnis ist das gleiche, wie im direkten Modus. Der Unterschied liegt aber darin, daß dieses Programm im Speicher erhalten bleibt. Sie können es also jeder Zeit durch die Eingabe von

```
RUN     <ENTER>
```

erneut starten. Dieses Programm bleibt nun so lange im Speicher erhalten, bis Sie es entweder überschreiben, oder bis Sie den Computer ausschalten.

Ein Programm wird normalerweise nicht nur aus einer Zeile bestehen. Im direkten Modus haben wir mit zwei verschiedenen Anweisungen gearbeitet. Diese wollen wir nun in einigen Beispielen verwenden.

Beispiele:

2.2.1 Programm 1

```
1 PRINT 12+6              <ENTER>

2 PRINT 12+6*(6-3)        <ENTER>

3 PRINT (12+6)*(6-3)      <ENTER>
```

Nachdem Sie nun das Programm eingetippt haben (hoffentlich haben Sie auch nach jeder Anweisung die `ENTER - Taste betätigt), können wir es mit dem Befehl

```
RUN       <ENTER>
```

starten. Das Programm läuft nun im Computer ab, und wird es beginnend mit Zeile 1, zeilenweise abarbeiten. Die Bildschirmausgabe erfolgt dabei dann auch in der Reihenfolge des Programmes.

Bildschirmausgabe

```
18
30
54
OK
```

Damit haben wir unsere Rechenbeispiele aus dem Direktmodus in einem Programm zusammengefasst. Sie können dieses Programm nun beliebig oft durch die Eingabe

```
RUN    <ENTER>
```

wiederholen.

2.2.2 Programm 2

Im nächsten Programm werden wir alle Beispiele aus dem direkten Modus in einem Programm zusammenfassen und dann als ein Programm ablaufen lassen. Wir werden dabei mit der Zeilennummer 4 beginnen. Dadurch bleiben die ersten 4 Zeilen, also das komplette erste Programm, erhalten.

```
 4 LET A=10              <ENTER>
 5 LET B=20              <ENTER>
 6 PRINT A*B             <ENTER>
 7 LET A=15              <ENTER>
 8 LET B=20              <ENTER>
 9 LET C=A+B             <ENTER>
10 PRINT C               <ENTER>
11 PRINT "ENDE"          <ENTER>
```

Damit ist unser Programm 2 erstellt. Es besteht aus elf Zeilen. Wir starten das Programm mit

```
RUN    <ENTER>
```

Auf dem Bildschirm erscheint

```
18
30
54
200
300
ENDE
OK
```

Auch dieses Programm können Sie jederzeit wiederholen. Damit haben Sie die beiden Betriebsarten Ihres Computers kennengelernt. Wir wenden uns nun einem anderen Thema zu.

2.3 Befehle

Im vorigen Abschnitt haben wir zwei Anweisungen und einen Befehl kennengelernt. PRINT und LET sind Anweisungen. Um das Programm zu starten, haben wir aber einen Befehl benötigt. Es ist der Befehl "RUN". Wir werden uns hier nicht mit allen Befehlen beschäftigen. Für den ersten Teil sind folgende Befehle ausgewählt worden:

RUN
LIST
NEW
AUTO
LOAD
SAVE
RENUMBER
DELETE

Befehle werden normalerweise zur Programmsteuerung und zur Unterstützung der Programmierung eingesetzt.

RUN

Wie wir gesehen haben, benötigen wir den Befehl `RUN` um ein Programm zu starten. Wir haben aber auch die Möglichkeit, das Programm an beliebiger Stelle zu starten. Nehmen wir z. B. unser elfzeiliges Programm. Mit der Eingabe

RUN 5 <ENTER>

starten wir das Programm in Zeile 5. Das bedeutet, daß die Zeilen 1 bis 4 ignoriert werden. Mit der Angabe einer Zeilennummer können Sie Programme ab einer bestimmten Stelle starten. Oftmals werden Sie diese Möglichkeit beim Testen von Teil-Programmen sinnvoll einsetzen können.

LIST

Stellen Sie sich einmal vor, Sie haben ein Programm geschrieben, und dieses dann auf Cassette abgespeichert. Nach dem Laden des Programmes wollen Sie etwas verändern. Sie wissen aber nicht mehr welche Zeilennummer Sie verändern müssen. Nachdem Sie das Programm geladen haben, befindet es sich zwar im Speicher. Es ist aber nicht auf dem Bildschirm zu sehen. Sie benötigen also einen Befehl, mit dem Sie das Programm sichtbar machen können. Dafür gibt es den Befehl

LIST

Wenn Sie Ihr Programm noch im Speicher haben, geben Sie nun folgenden Befehl ein:

LIST <ENTER>

Auf dem Bildschirm erscheint nun Ihr ganzes Programm. Leider sind Ihre Programme aber in den seltensten Fällen so kurz. Was tun wir denn, wenn wir aus einem sehr langen Programm nur einen Teil sehen wollen. Unser Programm hat elf Zeilen und wir wollen uns die Zeilen 4 bis 7 ansehen.

LIST 4-7 <ENTER>

Nun erscheinen nur die Zeilen 4 bis 7. Aber der LIST - Befehl bietet noch mehr. Auf die gleiche Art können wir uns zum Beispiel nur eine Zeile ansehen:

LIST 4 <ENTER>

Jetzt haben wir nur Zeile 4 gelistet. Sie sehen: Der LIST - Befehl ist recht umfangreich einsetzbar.

NEW

Ich hoffe, Sie haben Ihr Programm noch immer im Speicher. Wir wollen nun ein neues Programm schreiben, und wissen nicht, ob wir alle Zeilennummern wieder belegen. Es gibt nun eine einfache Möglichkeit, den Speicher zu löschen. Wir schalten ganz einfach unseren Computer ab. Wenn wir ihn nun wieder einschalten, ist der Speicher leer und neu formatiert. Doch halt!!! Es gibt doch auch für diese Arbeit einen Befehl. Geben Sie nun

NEW <ENTER>

ein. Nachdem Sie die ENTER - Taste betätigt haben, meldet sich Ihr Computer nur mit einem lakonischen OK. Nun wollen wir die Angelegenheit einmal überprüfen. Wozu haben wir den LIST - Befehl.

LIST <ENTER>

Außer dem schon bekannten OK ist nichts auf dem Bildschirm zu sehen. Nun wissen wir, daß unser Speicher ist leer.

A c h t u n g
==========

Verwenden Sie den NEW - Befehl nur dann, wenn

Sie sicher sind, daß Sie das im Speicher be-
---

findliche Programm auch wirklich löschen
---

wollen. Nach dem NEW ist das Programm un-
---

wiederbringlich gelöscht ! ! !

AUTO

Wenden wir uns nun einem Befehl zu, der die angeborene Faulheit des Programmierers unterstützt. Sollten Sie nun der Ansicht sein, daß Sie nicht zu den von Natur aus faulen Menschen gehören, können Sie diesen Abschnitt ruhig überspringen. Sie werden diesen Befehl dann sowieso nicht verwenden.

Stellen wir uns einmal vor, wir müssen ein längeres BASIC Programm schreiben. Da wir nun jeder Programmzeile eine Zeilennummer vorstellen müssen, und diese in aufsteigender Reihenfolge sind, liegt es nahe, diese Arbeit durch den Computer ausführen zu lassen. Geben Sie nun bitte folgende Schritte ein:

AUTO <ENTER>

Auf dem Bildschirm erscheint eine 10. Probieren wir den Befehl doch gleich einmal aus.

```
<100>  PRINT "HALLO"   <ENTER>
<110>  PRINT "TEST"    <ENTER>
<120>  PRINT 10*23     <ENTER>
<130>  LET A=234       <ENTER>
<140>  PRINT A*10      <ENTER>
```

<150>

Hier wollen wir nun unsere Arbeit beenden. Der Computer bringt aber stur die nächste Zeilennummer. Ganz oben rechts am Computer befindet sich eine rote Taste. Es handelt sich um die STOP - Taste. Ein leichter Druck genügt, und die AUTO - Funktion wird abgebrochen. Auch der Befehl AUTO bietet noch einige Variationen. Wir haben jetzt ein Programm mit 6 Zeilen, beginnend bei 100. Unsere letzte Zeilennummer war 150. Wenn wir nun unser Programm weiter schreiben wollen, können wir nicht einfach AUTO eingeben, da der Computer wieder bei Zeile 100 beginnen würde. Dadurch würden wir aber die bereits bestehenden Zeilen ja wieder löschen. Also müssen wir bei 160 fortfahren.

```
AUTO AT 160 STEP 10      <ENTER>

<160> PRINT "AUTO TEST"  <ENTER>

<170>                    <STOP>
```

Mit AUTO AT geben wir die Zeilennummer an, an der wir beginnen wollen. Mit STEP 10 sagen wir dem Computer, daß wir mit einer Schrittweite von 10 arbeiten wollen.

Als wir die AUTO - Funktion zum ersten mal aufgerufen haben, war die erste Zeilennummer 100 und die Schrittweite 10. Wir aber haben unser erstes Programm mit den Zeilennummern 1, 2, 3 usw geschrieben. Warum also schreibt der Computer in Zehner - Schritten. Stellen Sie sich einmal vor, Sie haben ein sehr langes Programm geschrieben. Nach einer Prüfung des Programmes stellen Sie fest, Sie haben in der Mitte des Programmes einige Zeilen vergessen. Wenn Sie nun mit einer Schrittweite von 1 gearbeitet haben, können Sie das Programm ab den fehlenden Anweisungen neu schreiben. Wenn Sie aber mit einer Schrittweite von 10 oder mehr gearbeitet haben, können Sie an den entsprechenden Stellen neue Programmzeilen einfügen. Zum Schluß sehen wir uns jetzt noch eine andere Art von AUTO an.

Dazu löschen wir aber erst einmal den Speicher. Ich gebe Ihnen an dieser Stelle keine Hilfe. Wenn Sie bis hierher sorgfältig gearbeitet haben, wissen Sie auch ohne Hilfe, wie Sie den Speicher zu löschen haben. Nachdem nun der Speicher gelöscht ist, tippen wir

```
AUTO STEP 15

<100> PRINT "HALLO"
```

```
<115>  PRINT "ENDE"          <ENTER>

<130>                        <STOP>
```

Die Funktion AUTO beginnt bei 100. Diesmal haben wir auch nur die Schrittweite bestimmt. Soweit der Befehl AUTO. Ich glaube, Sie werden ihn in Zukunft noch oftmals nutzen.

LOAD

Die bisherigen Befehle waren überwiegend für die Bearbeitung von Programmen bestimmt. Sie können für das Buch auch einen Datenträger erhalten. Wie Sie sich denken können, sind auf dem Datenträger Programme. Wenn Sie nun eines dieser Programme in den Speicher laden wollen, benötigen Sie dazu einen neuen Befehl.

Zuerst werden wir den Befehl in einer Form ansprechen, wie ihn nur Anwender mit einem Cassettenrecorder verwenden können. Wenn Sie Ihren Cassettenrecorder noch nicht benutzt haben, lesen Sie zunächst einmal den Anhang über den Cassettenrecorder Betrieb.

Nachdem Sie nun das Band eingelegt haben, überzeugen Sie sich davon, daß es auch zurück gespult ist. Nun geben Sie folgenden Befehl ein:

```
LOAD   <ENTER>
```

Nachdem Sie die <PLAY> - Taste am Recorder gedrückt haben, beginnt der Recorder zu laufen. Der Computer sucht sich nun das erste auf dem Band befindliche Programm heraus, und beginnt, es in den Speicher zu laden. Nachdem das Programm komplett im Speicher ist, meldet sich der Computer mit OK.

Sie können das Programm nun starten oder mit LIST auf den Bildschirm bringen. Soweit so gut. Der Anwender mit einem Diskettenlaufwerk kann den LOAD-Befehl in dieser Form nicht anwenden, da die Programme auf der Diskette nicht sequentiell gespeichert sind. Auf die Art der Speicherung wollen wir im Moment nicht eingehen. Wichtig für uns ist im Moment nur, daß wir bei der Diskette stets einen Namen angeben müssen. Wir werden nun den LOAD-Befehl in einer anderen Form kennenlernen.

LOAD "PROG1" <ENTER>

Von der Diskette wird nun das Programm mit dem Namen "PROG1" geladen. Aber auch Anwender mit dem Cassettenrecorder können diesen Befehl verwenden. Der Computer sucht dann so lange nach dem Programm "PROG1", bis dieses gefunden ist, egal an welcher Stelle es auch stehen mag.

An dieser Stelle verlassen wir den LOAD - Befehl. Wir werden uns an anderer Stelle noch einmal mit diesem Befehl beschäftigen müssen. Für den Moment reicht es aber, wenn Sie Ihre Programme laden können.

SAVE

Wie wir ein Programm laden können, haben Sie ja nun gelernt. Aber was nützt es schon, wenn wir unsere eigenen Programme nicht speichern können. Für diese Funktion haben wir den SAVE - Befehl. Die Besitzer von Cassettenrecordern können diesen Befehl in seiner einfachen Form nutzen:

SAVE <ENTER>

Diese Form ist aber nicht zu empfehlen. Wenn Sie ein Programm auf diese Art speichern, erhält es keinen Namen. Der Nachteil ist, daß Sie dieses Programm nicht gezielt wieder laden können. Die Folge ist, Sie müssen alle Programme auf der Cassette durchlaufen lassen. Die Anwender von Diskettenlaufwerken können diesen Befehl in dieser einfachen Form ohnehin nicht anwenden. Nachdem Sie nun das Programm "PROG1" geladen haben, wollen wir es wieder abspeichern. Die Anwender von Cassettenrecordern verwenden einfach eine neue Cassette. Bei der Diskette wird es nun etwas schwieriger. Wenn Sie bisher noch nicht mit Ihrem Diskettenlaufwerk gearbeitet haben, lesen Sie erst im Anhang den Diskettenteil.

Nachdem Sie nun eine formatierte Diskette haben, können wir unser Programm speichern. Verwenden Sie bitte die neue formatierte Diskette.

SAVE "TEST" <ENTER>

Nachdem sich nun der Computer mit OK gemeldet hat, löschen wir unseren Speicher. Hierzu verwenden wir den

Befehl NEW.

Zur Kontrolle werden wir unser Programm "TEST" wieder laden. Wie das funktioniert wissen Sie ja. Sollten Sie es nicht mehr wissen, empfehle ich, den vorigen Abschnitt noch einmal zu lesen.

RENUMBER

Wir haben uns ja schon über die Zeilenabstände unterhalten. Es kann aber mal vorkommen, das ein Zeilenabstand von 10 nicht ausreichend ist. Bei vielen Computern müßten wir nun das Handtuch werfen, und das Programm neu schreiben. Nicht so beim Enterprise.

Wenn Sie "PROG1" nicht mehr im Speicher haben, laden Sie es jetzt bitte ein. Nachdem das Programm nun geladen ist, listen wir es auf dem Bildschirm. Die Zeilennummern entsprechen nun absolut nicht unseren Anforderungen. Wir haben keine Möglichkeit, auch nur eine Zeile einzufügen. Wir wollen aber zwischen die Zeilen 6 und 7 drei neue Zeilen einfügen. Da wir einen Enterprise Computer haben, ist das für uns kein Problem.

```
1  PRINT "PROG1"
2  PRINT "Dieses Programm ist ein Beispiel"
3  PRINT
4  PRINT "Rechenbeispiel"
5  LET A=10
6  LET B=20
7  PRINT A*B
8  PRINT "Ende"
```

Wir geben nun folgenden Befehl ein:

```
RENUMBER     <ENTER>
```

Nachdem wir diesen Befehl eingegeben haben, listen wir das Programm noch einmal. Nun entspricht das Programm unseren Anforderungen. Der Programmanfang liegt bei 100, und der Zeilenabstand beträgt 10. Wir können nun unsere neuen Anweisungen fogendermaßen eintippen.

```
153 PRINT                      <ENTER>

155 PRINT "10 * 20 = ";        <ENTER>
```

Nachdem Sie die beiden neuen Zeilen geschrieben haben, listen wir das Programm noch einmal. Die neuen Zeilen werden automatisch an die richtigen Stellen gesetzt.

Auch der RENUMBER - Befehl verfügt über verschiedene Variationsmöglickeiten. Geben Sie nun folgenden Befehl ein:

```
RENUMBER 100 TO 500 STEP 20
```

Nachdem Sie das Programm gelistet haben, stellen Sie fest, daß die erste Zeile bei 500 beginnt. Alle weiteren Programmzeilen folgen in einem Abstand von je 20. Sie können diesen Befehl auch noch anders eingeben:

```
RENUMBER FIRST TO 500 STEP 30
```

Diese Eingabe ist sinnvoll, wenn Sie ein langes Programm bearbeiten. Wenn Sie am Ende des Programmes stehen, müßten Sie erst das Programm listen, um zu erfahren, wo die erste Zeile steht. Mit der Angabe FIRST, wählt sich der Computer automatisch die erste Zeile aus.

Für den Anfang soll es damit genug sein. Der RENUMBER - Befehl verfügt über weitere Möglichkeiten. Diese werden Sie aber an anderer Stelle kennenlernen, da Sie mit den bisherigen informationen im allgemeinen auskommen.

DELETE

Mit dem RENUMBER - Befehl haben wir eine Möglichkeit kennen gelernt, Platz für neue Anweisungen zu schaffen. Es kann aber auch vorkommen, daß Sie Programmzeilen löschen müssen. Eine einfache Lösung bietet sich auch ohne Befehl. Wenn Sie nur eine Zeile löschen wollen, geben Sie einfach die zu löschende Zeilennummer ein.

Beispiel: 500 <ENTER>

Damit ist die Zeile 500 gelöscht. Wenn Sie aber eine ganze Reihe von Zeilen löschen wollen, ist dieses Verfahren einfach zu umständlich. Hier haben wir dann den Befehl DELETE. Laden Sie nun das Programm "PROG2" in Ihren Computer. Dieses Programm hat keine weitere Bedeutung. Wir verwenden es nur, um den DELETE - Befehl kennenzulernen. Es sollen nun die Zeilen 40 bis 60 gelöscht werden.

DELETE 40 TO 60 <ENTER>

Durch LIST können Sie sich nun davon überzeugen, daß die Zeilen 40, 50 und 60 gelöscht wurden. Es ist aber auch folgende Eingabe möglich:

DELETE 40 - 60 <ENTER>

Diese Eingabe hat die gleiche Wirkung. Wir können aber auch verschiedene Blöcke und Zeilen löschen. Geben Sie folgenden Befehl ein:

DELETE 10 - 30,70,90,120 - 150 <ENTER>

Mit diesem Befehl haben Sie nun die Zeilen 10 bis 30, 70, 90 und die Zeilen 120 bis 150 gelöscht. Soweit der Befehl DELETE.

Soweit unsere kleine Reise durch die Welt der BASIC - Befehle. Abschließend finden Sie noch einmal alle Befehle mit ihrer Bedeutung aufgelistet.

RUN	= startet ein Programm
LIST	= der LIST - Befehl dient zum Listen eines Programmes auf den Bildschirm
NEW	= löscht alles, was sich im Speicher befindet
AUTO	= erlaubt eine automatische Zeilennummerierung
LOAD	= dient zum Laden von Programmen von Cassette oder Diskette
SAVE	= dient zum Speichern von Programmen auf Cassette oder Diskette
RENUMBER	= wird benötigt, um eine neue Zeilenfolge zu erstellen
DELETE	= dient zum Löschen von Programmzeilen

Sie können die Befehle natürlich auch klein schreiben. Der Enterprise akzeptiert seine Befehle in jeder schreibweise.

2.4 Anweisungen

Bisher haben wir nur Befehle kennen gelernt, mit denen wir BASIC - Programme bearbeiten können. Nun wollen wir aber endlich anfangen, selber Programme zu schreiben. Um aber ein Programm zu schreiben, benötigen wir BASIC - Anweisungen. In diesem ersten Teil werden wir folgende Anweisungen besprechen:

LET
PRINT
INPUT
IF
FOR NEXT
GOTO
GOSUB RETURN
ON
GET
DATA
READ
RESTORE
CLEAR

Das soll uns für den Anfang genügen. Anweisungen, die sich mit der Cassette, dem Drucker und dem Diskettenlaufwerk befassen, werden in gesonderten Kapiteln behandelt. Doch nun steigen wir endlich in die Programmierung ein.

LET

Die LET - Anweisung haben wir ja bereits kennengelernt. Bevor wir jedoch weiter arbeiten, müssen wir noch einmal einen kurzen Abstecher in die Theorie' wagen.

Wenn in einem Programm mit Hilfe einer LET - Anweisung eine Zuordnung getroffen wird, handelt es sich um eine Konstante. Dieser Name besagt lediglich, daß es sich hier um einen nicht veränderbaren Wert handelt. Das Gegenstück zur Konstanten ist die Variable. Eine Variable beinhaltet einen Wert, der sich je nach Bedarf ändern kann. Wenn wir nun in einem Programm mit einer LET - Anweisung arbeiten, können wir den Wert nur durch eine weitere Anweisung ver-

ändern. Für das Programm bedeutet das, daß der Wert dieser Konstanten immer wieder in der gleichen Form verwendet wird.

Sinnvoll ist der Einsatz von Konstanten dann, wenn es sich um unveränderbare Werte handelt. Z.B. der Mehrwertsteuersatz in einem Rechnungsprogramm verändert sich normalerweise nicht. Multiplikatoren, die immer wieder benötigt werden, sind sinnvoller Weise auch als Konstante definiert.

Abschließend sei noch erwähnt, daß Sie sich die Schreibweise der LET - Anweisung vereinfachen können.

```
A=10
```

ist von der Ausführung her identisch mit

```
LET A=10
```

Sie können aber auch Texte als Konstante definieren. In diesem Fall müssen Sie dem A ein Dollarzeichen vorstellen. Der Text muß in Anführungszeichen geschrieben werden.

```
A$="DM"

LET A$="DM"
```

Der Enterprise bietet die Möglichkeit, Konstante oder Variable nicht nur mit Buchstaben zu definieren. Sie können ganze Worte verwenden.

```
LET MEHRWERTSTEUER=14

    MEHRWERTSTEUER=14

LET WÄHRUNG$="DM"

    WÄHRUNG$="DM"
```

Es ist sehr sinnvoll, ausgeschriebene Namen als Bezeichnungen zu wählen. Wenn Sie erst einmal ein größeres Programm geschrieben haben, werden Sie es schätzen lernen, anstatt A, AB, C$ richtige Namen wie MEHRWERTSTEUER, PREIS, BEZEICHNUNG in den Programmlisten zu finden. Es ist sofort ein Bezug zur Sache vorhanden.

PRINT

Die PRINT - Anweisung haben wir ja auch schon behandelt. Die PRINT - Anweisung in dieser Form dient ausschließlich zur Darstellung auf dem Bildschirm. Wir können auch Zahlen und Texte in einer PRINT - Anweisung mischen.

```
PRINT "Hausnummer ";16      <ENTER>

PRINT "Summe ";230;" DM"    <ENTER>
```

Beide Anweisungen haben wir im Direktmodus ausgeführt. Durch das Semikolon haben wir den Computer veranlasst, keinen Zwischenraum zu lassen. Probieren Sie nun folgende Anweisungen aus.

```
PRINT "Hausnummer",16       <ENTER>

PRINT "Summe ",230," DM"    <ENTER>
```

Wenn wir anstatt einem Semikolon, ein Komma verwenden, läßt der Computer einen Zwischenraum. Geben Sie nun folgendes Programm ein:

```
10 PRINT "T E S T"          <ENTER>

20 PRINT "-------"          <ENTER>

30 PRINT                    <ENTER>

40 PRINT "Summe  ";         <ENTER>

50 PRINT 230;               <ENTER>

60 PRINT "  DM"             <ENTER>
```

Bevor wir nun das Programm analysieren, starten Sie es erst einmal.

Die Zeilen 10 und 20 sind wohl klar. Zeile 30 verwenden wir um eine Leerzeile auf dem Bildschirm zu bekommen. Zeile 40 bis 60 bringen etwas neues. Wenn Sie eine PRINT - Zeile mit einem Semikolon beenden, wird keine neue Zeile begonnen. Der Computer wird veranlaßt, in der gleichen Zeile weiter zu schreiben.

Als nächstes wollen wir noch eine besondere Form der PRINT - Anweisung ansprechen. Wenn Sie einen Bestimmten Bildschirmaufbau erstellen wollen, müssen Sie nicht mit vielen PRINT - Anweisungen arbeiten um entsprechende Leerzeilen zu schreiben. Probieren Sie einmal folgende Anweisung aus:

```
PRINT AT 2,15 "Hallo ich bin hier"    <ENTER>
```

Sie werden Ihren Text nun in der zweiten Zeile, beginnend mit der 15. Position wiederfinden. Auf diese Art können Sie Ihre Bildschirmausgaben an jede beliebige Stelle bringen.

INPUT

Mit der INPUT - Anweisung kommen wir nun zur ersten neuen Anweisung. Wenn wir mit der LET - Anweisung eine Konstante definiert haben, so haben wir mit der INPUT - Anweisung die Möglichkeit, eine Variable zu definieren und ständig mit neuen Werten zu belegen. Schauen wir uns dazu ein Beispiel an:

```
NEW                 <ENTER>

10 INPUT A          <ENTER>

20 INPUT B          <ENTER>

30 C=A*B            <ENTER>

40 PRINT A;" * ";B;" = ";C
```

Bevor Sie das Programm starten, wollen wir es uns etwas näher betrachten.

In Zeile 10 und 20 werden die Variablen A und B definiert und der Anwender aufgefordert, diese mit Werten zu belegen. Wenn das Programm abläuft erscheint als Aufforderung, einen Wert einzugeben, ein Fragezeichen. In der Zeile 30 wird der Variablen C der Wert zugeordnet, der aus der Multiplikation von A und B entsteht. Zeile 40 bringt uns dann das Ergebnis auf den Bildschirm.

Nun starten Sie bitte das Programm. Denken Sie daran, Sie haben mit A und B ausschließlich numerische Variable de-

finiert. Sie dürfen hier nur Zahlen eingeben. Wenn nun das Fragezeichen erscheint, geben Sie irgendeine Zahl ein und bestätigen Sie Ihre Eingabe mit der <ENTER> - Taste.

Dieses Programm können Sie nun beliebig oft wiederholen und immer mit neuen Zahlen arbeiten. Wenn Sie aber nun das Programm starten, und eine andere Person soll die Werte eingeben, müssen Sie dieser Person erst einmal erklären, was sie zu tun hat. Ein gutes Programm kommt aber ohne lange Erklärungen aus. Es erklärt sich selbst. Unser Programm verfügt leider nicht über diese Eigenschaft.

Das ist aber kein Problem. Wenn wir die uns bisher bekannten Befehle sinnvoll einsetzen, soll das schon funktionieren. Löschen Sie bitte den Speicher, wir schreiben unser nächstes Programm.

```
10 PRINT "Geben Sie bitte eine Zahl ein ";     <ENTER>
20 INPUT A                                     <ENTER>
30 PRINT "Geben Sie bitte eine Zahl ein ";     <ENTER>
40 INPUT B                                     <ENTER>
50 C=A*B                                       <ENTER>
60 PRINT A;" * ";B;" = ";C                     <ENTER>
```

Wenn Sie dieses Programm ablaufen lassen, brauchen Sie keine Erklärungen mehr abzugeben. Wir haben allerdings zwei zusätzliche PRINT - Anweisungen benötigt.

Erinnern Sie sich noch an den DELETE - Befehl? Ja? Sehr gut, dann löschen Sie bitte die Zeilen 10 bis 40. So, nun werden wir die vier Anweisungen um zwei kürzen. Tippen Sie folgende beiden Zeilen neu ein:

```
10 INPUT PROMPT "Geben Sie bitte eine Zahl ein":A  <E>
20 INPUT PROMPT "Geben Sie bitte eine Zahl ein":B  <E>
```

Lassen Sie sich bitte nicht durch das <E> täuschen, es ist auch hier die ENTER - Taste gemeint. Doch nun zum Thema:

Da wir nun die INPUT - Anweisung durch den Zusatz PROMPT erweitert haben, sparen wir uns eine PRINT - Anweisung. Der Ausdruck PROMPT teilt dem Computer mit, daß vor der Variableneingabe noch ein Text auszudrucken ist.

Bei der PRINT - Anweisung haben wir gelernt, daß wir den Ausdruck an jede beliebige Stelle des Bildschirmes bringen können. Diese Möglichkeit haben wir auch bei der INPUT - Anweisung. Hier ein Beispiel:

```
10 INPUT AT 2,15,PROMPT "Eingabe Zahl":A  <E>
```

Mit dieser Anweisung wird die "Eingabe ..." in der zweiten Zeile ab der 15. Stelle gezeigt. Soweit die INPUT - Anweisung.

IF THEN

Wie oft kommt es vor, daß Sie einen Wert mit einem anderen vergleichen müssen. Diese Funktion übernimmt die IF - Anweisung. Bevor wir jedoch näher auf diese Anweisung eingehen, müssen wir uns noch mit einer Besonderheit der Programmiersprache BASIC beschäftigen.

Bisher haben wir für jede Anweisung eine neue Zeile verwendet. Wir können aber auch mehrere Anweisungen in einer Zeile unterbringen. Dazu müssen wir die einzelnen Anweisungen durch einen Doppelpunkt trennen.

Beispiel:

```
10 PRINT A:PRINT B:PRINT C
```

Hier werden alle PRINT - Anweisungen nacheinander ausgeführt, genauso als hätten wir sie in drei Zeilen untergebracht.

Wir behandeln hier zwar die IF - Anweisung, trotzdem lernen Sie nun eine neue Anweisung kennen, für die sich ein eigener Abschnitt nicht lohnt. Sie können Ihre Programme an jeder beliebigen Stelle beenden, wenn Sie die END - Anweisung verwenden.

Beispiel:

```
...
100 END
```

Das Programm wird nun bis Zeile 99 abgearbeitet und in Zeile 100 beendet, egal was noch folgt. Nun aber zur IF - Anweisung. Beginnen wir gleich mit einem kleinen Programm:

```
10 INPUT PROMPT "Zahl 1 ":A

20 INPUT PROMPT "Zahl 2 ":B

30 IF A>B THEN PRINT "Zahl 1 ist größer als Zahl 2":
   END

40 PRINT "Zahl 1 ist kleiner als Zahl 2"
```

Die Zeilen 10 und 20 sind uns schon geläufig. In Zeile 30 vergleicht der Computer beide Werte miteinander. Ist A kleiner als B wird die PRINT - Anweisung in der gleichen Zeile ausgeführt. Die END - Anweisung brauchen Sie nicht in eine neue Zeile zu schreiben. Ich habe das hier nur aus Platzmangel getan. Wenn nun aber A nicht kleiner ist als B, geht das Programm gleich in die nächste Zeile, da die Bedingung nicht erfüllt ist.

Starten Sie das Programm nun noch einmal, und geben Sie zwei gleiche Werte ein. Auch in diesem Fall wird Ihnen das Programm mitteilen, daß A kleiner ist als B. Diese Aussage aber ist falsch. Nun kann aber der Computer nichts dafür. Wir haben je nur eine Möglichkeit eingebaut. Wir verändern nun unser Programm entsprechend.

```
35 IF A=B THEN PRINT "Beide Zahlen sind gleich groß":
   END
```

So, nun haben wir alles berücksichtigt. Sie können das Programm nun mit verschiedenen Zahlen testen. Hier nun noch die Bedeutungen der Sonderzeichen, die zum Vergleich herangezogen werden:

```
<   bedeutet kleiner als

=   ist gleich

>   bedeutet größer als
```

Sie können bis zu zwei Vergleiche gleichzeitig durchführen.

Beispiel: 100 IF A<>B

In unserem Beispiel wird die Bedingung dann erfüllt, wenn A ungleich B ist.

Beispiel: 100 IF A<=B

Hier ist die Bedingung erfüllt, wenn A kleiner oder gleich B ist.

Beispiel: 100 IF A>=B

Im letzten Beispiel ist die Bedingung erfüllt, wenn wenn A gleich oder größer B ist.

FOR NEXT

Wenden wir uns nun der nächsten Anweisung zu. Oftmals benötigen wir einen bestimmten Bereich den wir in Form einer Schleife mehrfach wiederholen können. Mit den Anweisungen FOR und NEXT können wir Schleifen aufbauen, die beliebig oft wiederholt werden können. Wir betrachten uns das aber am Besten an Hand eines Programmes:

```
10 PRINT "Schleifen"     <ENTER>
20 PRINT     <ENTER>
30 FOR X=1 TO 10      <ENTER>
40 INPUT PROMPT "Zahl 1":A     <ENTER>
50 PRINT A     <ENTER>
60 NEXT X      <ENTER>
70 PRINT "Ende"     <ENTER>
80 END     <ENTER>
```

Die Zeilen 10 und 20 sind uns bekannt. Uns interessieren hier nur die Zeilen 30 und 60. In der Zeile 30 wird eine Schleife definiert, die alle Anweisungen, die zwischen FOR und NEXT stehen, 10 mal wiederholt. Aber lassen Sie das Programm erst einmal laufen.

Ihre Zahlen konnten Sie 10 mal eingeben. Wir gehen jetzt noch einen Schritt weiter. Wir werden nun noch eine zweite

Schleife in unsere erste integrieren. Wir wollen verhindern, daß direkt nach der Eingabe einer Zahl, diese sofort ausgedruckt wir. Hierzu benötigen wir eine Schleife, die nur die Aufgabe hat, die Ausführung der PRINT - Anweisung zu verzögern.

```
45 FOR Y=1 TO 1000          <ENTER>
47 NEXT Y                   <ENTER>
```

Wenn Sie Ihr Programm nun ablaufen lassen, werden Sie die Veränderung sofort spüren.

Eine Besonderheit hat diese Schleife noch:

```
10 FOR X=1 TO 500 STEP 10
...
xx NEXT X
```

Durch den Zusatz STEP 10 veranlassen wir den Computer, die Schleife in 10er Schritten durchzuführen. Diese Schleife wird also nicht 500 mal durchlaufen, sondern nur 50 mal.

GOTO

Als nächste Anweisung behandeln wir eine Sprung-Anweisung. Diese Anweisung zwingt ein Programm, zu einer bestimmten Zeile zu springen, um dort mit der Arbeit fortzufahren. Betrachten wir auch zu dieser Anweisung erst einmal ein kleines Programm. Am besten, Sie tippen es gleich wieder ab.

```
10 PRINT "Sprünge"          <ENTER>
20 PRINT                    <ENTER>
30 INPUT PROMPT "Zahl 1 ":A      <ENTER>
40 INPUT PROMPT "Zahl 2 ":B      <ENTER>
```

```
 50 IF A=B THEN GOTO 150          <ENTER>
 60 IF A<B THEN GOTO 200          <ENTER>
 70 IF A>B THEN GOTO 250          <ENTER>
 80 END
150 PRINT "A ist gleich B ";A;" = ";B     <ENTER>
160 GOTO 10
200 PRINT "A ist kleiner B  ";A;" < ";B   <ENTER>
210 GOTO 10
250 PRINT "A ist größer B  ";A;" > ";B    <ENTER>
260 GOTO 10
270 END
```

Soweit das Programm. Für uns sind zunächst einmal die Zeilen 50, 60, 70 und die Zeilen 160, 210 und 260 von Interesse. Die IF - Anweisung kennen wir ja. Wenn die Bedingung wahr ist, führen wir aber keine PRINT - Anweisung aus. Wir zwingen das Programm, an einer anderen Stelle fortzufahren. Wenn das Programm eine GOTO - Anweisung durchführt, werden alle Anweisungen, die zwischen der GOTO - Anweisung und der darin enthaltenen Zeilennummer, nicht berücksichtigt. Nach der Ausführung einer GOTO - Anweisung in Zeile 50, arbeitet unser Programm also ab Zeile 150 weiter.

Betrachten wir uns nun die Zeile 160. Hier steht eine GOTO - Anweisung allein. Sie zwingt das Programm, bei Zeile 10 fortzufahren. Bei Zeile 10 aber beginnt unser Programm. Wenn Sie das Programm starten, werden Sie feststellen, daß es sich immer wiederholt. Wenn Sie das Programm beenden wollen, drücken Sie die berühmte rote Taste (STOP).

Zwei Anweisungen sind in unserem Programm völlig sinnlos. Spielen Sie das Programm einmal auf dem Papier durch. Es werden zwei Anweisungen übrig bleiben, die Sie nie treffen. Löschen Sie die entsprechenden Zeilen. Wenn Sie danach das Programm laufen lassen, wird es ohne Probleme arbeiten.

Bevor wir uns nun der GOSUB - Anweisung zuwenden, wollen wir uns erst einmal ein wenig entspannen. Dazu werden wir das nachfolgende Programm in unseren Computer eintippen. Es handelt sich dabei um ein einfaches Zahlenspiel.

```
 10 PRINT "Zahlenspiel"
 20 PRINT "-----------"
 30 PRINT
 50 LET ZAHL=RND(10)
 60 PRINT "Sie haben 3 Versuche, eine Zahl zwischen 1
          und 10 zu erraten"
 70 FOR X=1 TO 3
 80 INPUT PROMPT "Ihr Tip ? ": IHREZAHL
 90 IF IHREZAHL=ZAHL THEN GOTO 200
100 LET WERT=3-X
110 PRINT "Leider nicht. Sie haben noch ";WERT;" Ver-
         suche"
120 NEXT X
130 PRINT "Sie haben die Zahl leider nicht erraten"
140 PRINT "Die gesuchte Zahl lautet: ";ZAHL
150 GOTO 300
200 PRINT "Sie haben gewonnen ! ! !"
210 FOR Y=1 TO 500
220 NEXT Y
230 GOTO 300
300 INPUT PROMPT "Wollen Sie weiterspielen ? J ";W$
310 IF W$="J" THEN GOTO 10
320 PRINT "Schade. Auf wiedersehen."
330 END
```

Un nun viel Spaß beim Spielen.

Ich hoffe, Sie haben sich gut entspannt. Bevor wir nun mit der nächsten Anweisung fortfahren, wollen wir uns unser kleines Zahlenspiel etwas näher betrachten.

Zunächst einmal fällt uns auf, daß wir anstatt nichtssagender Buchstaben, ganze Worte für die Variablen verwendet haben. Unbekannt dagegen dürfte die Zeile 50 sein:

```
50 LET ZAHL=RND(10)
```

Mit der Funktion RND(10) veranlassen wir unseren Computer, eine Zufallszahl im Bereich von 0 bis 9 zu generieren. Durch diese Funktion war es uns möglich, eine gewisse Spannung in das Spiel zu bringen, da uns die Zahl ja nicht bekannt war. Außerdem haben wir in unserem Beispiel zum ersten mal eine Varialble mit einem String (Folge von verschiedenen Zeichen, die nicht nur aus Zahlen besteht) belegt. In der folgenden IF - Anweisung haben wir dann diesen String verglichen. Wenn es sich, wie in unserem Fall um eine Konstante handelt, muß der String in Anführungszeichen stehen. Als Eingabe genügt aber ein Buchstabe ohne Anführungszeichen.

GOSUB RETURN

Stellen Sie sich vor, Sie schreiben ein Programm, in dem eine Funktion immer wieder benötigt wird. Diese Funktion besteht aus 3 Programmzeilen. Sie wird aber in drei verschiedenen Programmbereichen benötigt. Sie haben nun die Möglichkeit, diese drei Zeilen in jeweils drei Programmbereiche unterzubringen. Wir wollen uns aber nun mit einer viel eleganteren Form beschäftigen. Zunächst aber ein Beispiel mit den uns bekannten Anweisungen.

```
10 PRINT "Beispiel Routine"
20 INPUT PROMPT "Eingabe Zahl ":Z
30 IF Z=1 THEN PRINT "Diese Zahl ist nicht erlaubt"
40 IF Z=0 THEN PRINT "ENDE":END
50 PRINT "Rechenbeispiel"
.....
```

Die Zeilen 20 bis 40 werden nun drei weitere Male in diesem Programm benötigt. Wenn wir nach unseren bisherigen Methoden weiterarbeiten, tauchen diese Zeilen dann eventuell noch einmal ab Zeile 200 und ab Zeile 400 auf. Mit einer GOTO - Anweisung können wir nicht arbeiten, da nach der Routine an drei verschiedenen Stellen weitergearbeitet wird. In einer GOTO - Anweisung können wir aber nur eine Adresse angeben. Um dieses Problem zu lösen, und um die Wiederholung von immer wiederkehrenden Programmteilen zu vermeiden, haben wir die GOSUB - Anweisung. Wie immer schauen wir uns dazu erst einmal ein Programm an.

```
 10 PRINT "Beispiel Sub Routine"
 20 GOSUB 100
 30 PRINT "Rücksprung 1  Zahl ";A
 40 GOSUB 100
 50 PRINT "Rücksprung 2  Zahl ";A
 60 GOSUB 100
 70 PRINT "Rücksprung 3  Zahl ";A
 80 GOSUB 100
 90 PRINT "Rücksprung 4  Zahl ";A
 95 END
100 PRINT "Routine zur Eingabe einer Zahl"
110 PRINT
120 INPUT PROMPT "Eingabe Zahl ":A
130 RETURN
```

Zunächst einmal analysieren wir unser Programm. In den Zeilen 30, 50, 70 und 90 werden in den PRINT - Anweisungen 4 verschiedene Konstante verwendet. Die GOSUB - Anweisungen in den Zeilen 20, 40, 60 und 80 verweisen alle auf die Zeile 100. Ab der Zeile 100 finden wir dann unsere Routine zur Eingabe einer Zahl. Wichtig ist in diesem Fall, daß das Programm nach Eingabe einer Zahl an die richtige Stelle im Programm zurückkehrt. Auslöser ist die RETURN - Anweisung. Wenn nun unser Programm in Zeile 20 veranlasst wird, in die Zeile 100 zu springen. Kehrt es

nach Ausführung an die gleiche Stelle zurück. Es wird mit Zeile 30 weiter arbeiten. Wenn es dann bei Zeile 40 wieder zur Zeile 100 springt, muß es nach Ausführung bei Zeile 50 weiter arbeiten. Am besten starten Sie das Programm nun, und überzeugen sich selbst von der Richtigkeit.

Wenn ein Programm eine GOSUB - Anweisung ausführt, merkt es sich, an welcher Stelle dieses geschehen ist. Nach Ausführung dieser Routine kehrt es an die gleiche Stelle zurück. Wir werden uns später noch ausführlich mit der GOSUB - Anweisung geschäftigen.

ON

Bisher haben wir uns nur mit einfachen Dingen beschäftigt. Was ist aber, wenn wir von einer Stelle aus an verschiedene Programmstellen springen wollen. Wir wollen zum Beispiel ein Programm schreiben, mit dem wir Rechnungen schreiben können. Am Anfang dieses Programmes wollen wir dem Anwender die Möglichkeit schaffen, drei verschiedene Programmfunktionen aufrufen zu können. Betrachten wir dazu ein Beispiel.

```
 10 PRINT "Auswahl"
 20 PRINT
 30 PRINT "Rechnung drucken   (1) "
 40 PRINT "Rechnung erstellen (2) "
 50 PRINT "Rechnung speichern (3) "
 60 INPUT PROMPT "Wählen Sie 1 - 3 ":A
 70 IF A=1 THEN GOTO 150
 80 IF A=2 THEN GOTO 200
 90 IF A=3 THEN GOTO 250
100 GOTO 60
150 PRINT "Teilprogramm  Rechnung drucken"
```

```
160 END

200 PRINT "Teilprogramm   Rechnung erstellen"

210 END

250 PRINT "Teilprogramm   Rechnung speichern"

260 END
```

Es ist wohl nicht nötig, dieses Programm zu erklären. Wir wollen auch nur auf die Zeilen 70 bis 90 eingehen. Wenn Sie das Programm bereits in Ihren Computer eingegeben haben, müssen Sie nun die Zeilen 70 bis 90 löschen. In diesen drei Zeilen werden die Sprünge zu den entsprechenden Teilprogrammen ausgeführt. Die Sprache BASIC bietet uns nun die Möglichkeit, alles in einer Anweisung zu erledigen.

Nachdem die drei Zeilen gelöscht sind, geben Sie nun folgende Zeile ein:

```
70 ON A GOTO 150,200,250
```

Wenn Sie nun eine Zahl zwischen 1 und 3 eingeben, springt das Programm in die entsprechende Zeilennummer.

GET

Erinnern Sie sich noch an unser kleines Zahlenspiel? Dort haben wir in Zeile 300 gefragt, ob Sie weiter spielen wollen. Egal wie Sie die Frage auch beantwortet haben, Sie mußten Ihre Antwort mit der <ENTER> - Taste bestätigen. Wir lernen nun eine Möglichkeit kennen, die uns die <ENTER> - Taste erspart. Damit Sie das ganze Spiel nicht noch einmal eintippen müssen, sei an dieser Stelle erwähnt, daß sich das Spiel unter dem Namen "SPIEL1" auf dem erhältlichen Datenträger befindet. Laden Sie also zunächst einmal das Spiel.

Nun löschen Sie die Zeilen 230 und 300 und geben dafür folgende Anweisungen ein:

```
230 PRINT "Wollen Sie weiterspielen ?   J ";
240 GET W$
250 IF W$="" THEN GOTO 240
```

Nachdem Sie nun diese kleine Änderung vorgenommen haben, dürfen Sie wieder spielen. Achten Sie dabei auch auf die kleine Veränderung in unserem Programm.

Sie sehen, je tiefer wir in die BASIC - Programmierung einsteigen, desto mehr Möglichkeiten haben Sie, Ihre Programme anwenderfreundlich zu gestalten.

DATA READ RESTORE

Als nächstes müssen wir gleich drei Anweisungen auf einmal behandeln. Ein Computer soll ja mehr leisten, als nur kleine Spiele zum Zeitvertreib. Wir werden uns nun mit der ursprünglichen Aufgabe eines jeden Computers befassen: der Datenverarbeitung. Sie werden zunächst ein Programm kennen lernen, über das wir noch viel zu sagen haben.

```
 10 PRINT "Adressenliste"
 20 PRINT
 30 INPUT PROMPT "Eingabe des zu suchenden Namen ":NAME$
 40 READ NAME1$,NAME2$,PLZ,ORT$,STRASSE$,TEL$
 50 IF NAME$=NAME1$ GOTO 100
 60 PRINT "Name nicht gefunden ! ! !"
 70 GOSUB 200
 80 GOTO 10
100 PRINT "Name 1       :  ";NAME1$
110 PRINT
120 PRINT "Name 2       :  ";NAME2$
```

```
130 PRINT "Plz und Ort   :  ";PLZ;"  ";ORT$
140 PRINT
150 PRINT "Straße        :  ";STRASSE$
160 PRINT
170 PRINT "Telefon       :  ";TEL$
180 GOSUB 200
190 GOTO 10
200 PRINT "Wollen Sie weitere Adressen sehen  J  ";
210 GET W$
220 IF W$="" THEN GOTO 210
230 IF W$<>"J" THEN GOTO 270
240 RESTORE
250 RETURN
270 END
300 DATA Meier,Hubert,8000,München,Gasteig 3,089/1234567
310 DATA Schulz,Kai,2900,Oldenburg,91er Str.2,0441/12345
320 DATA Müller,Eva,2000,Hamburg,Am DOM 7,040/2876549
330 DATA Bit,Klaus,6000,Frankfurt,Byte 3,069/3456789
```

Verwenden Sie anstatt der Beispiele lieber reale Adressen. Sie können die Zeilen mit den Adressen auch beliebig fortsetzen. Bevor wir das Programm analysieren, sollten Sie es erst einmal testen.

Wenn Sie nun das Programm auf einen eigenen Datenträger abspeichern, haben Sie Ihr erstes eigenes Programm. Doch nun wollen wir uns das Programm näher betrachten.

Beginnen wir gleich mit den Zeilen 300 bis ?. Diese Zeilen speichern unsere Daten. Diese Art der Datenspeicherung eignet sich besonders gut, wenn es sich um unveränderbare Daten handelt. Damit wir diese Daten lesen können, benötigen wir die READ - Anweisung. Die Variablen in der READ - Anweisung bestimmen, wieviel Daten in einer Anweisung zu lesen sind. Die Daten in der DATA - Zeile

werden jeweils durch ein Komma getrennt. Wenn wir nun, nachdem wir den ersten Datensatz eingelesen haben, einen weiteren Satz auf den Bildschirm bringen wollen, müssen wir dem Computer mitteilen, daß er die DATA - Zeilen wieder von vorne lesen muß. Dies erreichen wir duch die Anweisung RESTORE. Mit RESTORE wird der interne Zähler wieder zurückgesetzt. Bei der nächsten READ - Anweisung beginnt das Programm wieder mit der ersten DATA - Zeile. Sie sollten sich nun Ihr neues Programm noch einmal gründlich ansehen. Sollten dann noch unverständliche Begriffe auftauchen, lesen Sie die entsprechenden Teile noch einmal.

CLEAR

Wenn Sie sich die beiden bisherigen Programme ansehen, werden Sie feststellen, daß der Beginn des Programmes in der nächsten freien Zeile erfolgt. Wir befassen uns nun mit der Möglichkeit, den Bildschirm in einem Programm zu löschen und damit in der ersten Zeile zu beginnen. Die dazu passende Anweisung ist die CLEAR - Anweisung. Da auch die CLEAR - Anweisung wesentlich mehr bietet als wir es im Moment brauchen, konzentrieren wir uns hier nur auf die eine Funktion, nämlich den Bildschirm zu löschen.

Geben Sie bitte die nachfolgende Anweisung im Direktmodus ein.

```
CLEAR SCREEN          <ENTER>
```

Sie sehen, der Bildschirm wird gelöscht. Wir könnten nun unsere Programme laden und dann verändern. Doch wir wollen ja die Programmierung lernen. Um dies aber wirklich zu tun, werden wir eben ein neues Programm schreiben. Da wir nun eine ganze Reihe von Anweisungen kennen, ist es für uns kein Problem ein umfangreiches Programm zu schreiben. Im nachfolgenden Programm können Sie dann auch gleich Ihren Wissensstand überprüfen. Tippen Sie also nicht einfach alles ab. Versuchen Sie zunächst einmal am geschriebenen Programm dessen Bedeutung und seinen Ablauf zu verstehen. Wenn Sie dann sicher wissen, welche Aufgabe dieses Programm hat, werden Sie auch sicher einen passenden Namen finden. Diesen können Sie dann in Zeile 20 anstatt der Fragezeichen einsetzen.

```
 10 CLEAR SCREEN
 15 RESTORE
 20 PRINT AT 1,5:"??????????????"
 30 PRINT AT 2,5:"---------------"
 40 PRINT AT 5,5:"Adressen listen    (1) "
 50 PRINT AT 7,5:"Rechnung erstellen (2) "
 70 PRINT AT 11,5:"Programmende      (3) "
 80 PRINT AT 15,5:"Wählen Sie 1 - 3      ";
 90 INPUT A
100 ON A GOTO 200,500,1000
110 GOTO 10
200 CLEAR SCREEN
210 PRINT AT 1,10:"Adressenliste"
220 PRINT AT 2,10:"-------------"
230 PRINT AT 5,1:"Firma:"
240 PRINT AT 8,1:"Name:"
250 PRINT AT 11,1:"Straße:"
260 PRINT AT 14,1:"Plz u. Ort:"
270 PRINT AT 17,1:"Telefon:"
280 FOR X=1 TO 10
290 READ FIRMA$,NAME$,STR$,PLZ,ORT$,TEL$
300 PRINT AT 5,20:FIRMA$
310 PRINT AT 8,20:NAME$
320 PRINT AT 11,20:STR$
330 PRINT AT 14,20:PLZ;"  ";ORT$
340 PRINT AT 17,20:TEL$
350 PRINT AT 20,15:"Weiter = irgend eine Taste"
```

```
360 GET W$
370 IF W$="" THEN GOTO 360
380 PRINT AT 5,20:"                    "
390 PRINT AT 8,20:"                    "
400 PRINT AT 11,20:"                    "
410 PRINT AT 14,20:"                    "
420 PRINT AT 17,20:"                    "
430 NEXT X
440 GOTO 10
500 CLEAR SCREEN
505 ZPREIS=0
510 PRINT AT 1,10:"Rechnung erstellen"
520 PRINT AT 2,10:"------------------"
530 PRINT AT 5,1:"Menge       : "
540 PRINT AT 8,1:"Artikel     : "
550 PRINT AT 11,1:"Preis       : "
560 FOR X=1 TO 5
570 INPUT AT 5,15:MENGE(X)
580 INPUT AT 8,15:ART$(X)
590 INPUT AT 11,15:PREIS(X)
600 LET GPREIS(X)=MENGE(X)*PREIS(X)
605 LET ZPREIS=ZPREIS+GPREIS(X)
610 INPUT AT 16,30,PROMPT "Weiter J ":Z$
620 IF Z$<>"J" THEN GOTO 650
630 NEXT X
640 GOSUB 1100
650 CLEAR SCREEN
```

```
660 PRINT AT 1,10:"Rechnungsausgabe"
670 PRINT AT 2,10:"----------------"
690 PRINT AT 10,1:"Menge"
700 PRINT AT 10,8:"Artikel"
710 PRINT AT 10,20:"EP"
720 PRINT AT 10,30:"Gesamt"
730 PRINT AT 11,1:"----------------------------------
    ----"
740 FOR Y=1 TO X
750 LET Z=12+Y
760 PRINT AT  Z,1:MENGE(Y)
770 PRINT AT  Z,8:ART(Y)$
780 PRINT AT Z,20:  PREIS(Y)
790 PRINT AT Z,30:GPREIS(Y)
800 NEXT Y
810 PRINT AT 19,30:"---------"
820 PRINT AT 20,30:ZPREIS
830 PRINT AT 21,9:"zuzüglich 14% MWSt"
840 LET MWST=ZPREIS/100*14
850 PRINT AT 21,31:MWST
860 PRINT AT 22,30:"--------"
870 PRINT AT 23,9:"Rechnungsbetrag DM"
880 LET RBETRAG=ZPREIS+MWST
890 PRINT AT 23,30:RBETRAG
900 PRINT AT 24,30:"========"
910 PRINT AT 4,1:FIRMA$
920 PRINT AT 5,1:NAME$
930 PRINT AT 6,1:STRASSE$
```

```
 940 PRINT AT 8,1:PLZ;"  ";ORT$
 950 FOR I=1 TO 1000
 960 PRINT AT 5,30:"Weiter"
 970 PRINT AT 6,30:"Taste"
 980 PRINT AT 7,30:"drücken"
 990 GET I$
1000 IF I$="" THEN GOTO 990
1010 GOTO 10
1100 CLEAR SCREEN
1110 PRINT "Eingabe der Rechnungsadresse"
1120 PRINT "============================"
1130 INPUT AT 5,1,PROMPT "Gesuchte Firma  ":GFIRMA$
1140 FOR O=1 TO 10
1150 READ FIRMA$,NAME$,STR$,PLZ,ORT$,TEL$
1160 IF GFIRMA$=FIRMA$ THEN GOTO 1250
1170 NEXT O
1180 PRINT "Firma nicht gefunden, bitte neu eingeben"
1190 RESTORE
1200 GOTO 1100
1250 RETURN
1500 CLEAR SCREEM
1510 PRINT AT 12,10:"Das Programm endet hier."
1520 END
2000 DATA Müller,Kornmühle,Kornweg 13,8000,München,12345
2010 DATA Bäcker,Brotfabrik,Semmelweg 3,2000,Hamburg,123
     789
2020 DATA Lüfter,Heizungsbau,Oelweg 5,2800,Bremen,123456
2030 DATA Baumholz,Möbelfabrik,Kiefernweg,6000,Frankfurt,
     349876
```

Für die Zeilen 2040, 2050, 2060, 2070, 2080, 2090 und 2100 suchen Sie sich bitte eigene Namen.

Wenn Sie nun das ganze Programm abgetippt haben, können Sie es direkt starten. Sie haben nun ein sehr langes Programm geschrieben. Wenn Sie alles verstanden haben, haben Sie bisher sehr gut mitgearbeitet. Sollten Sie den Ablauf des Programmes nicht ganz verstehen, macht nichts.

Im nächsten Kapitel werden wir dieses Programm als Grundlage für die Programmentwicklung nehmen.

Sie sollten nun eine längere Pause einlegen; Sie haben Sie sich redlich verdient.

3. Programmentwicklung

Bevor wir uns nun den einzelnen Themen der Programmentwicklung zuwenden, möchte ich Ihnen zunächst eine kleine Geschichte erzählen.

Es war einmal eine große Firma. Die Manager dieser Firma suchten nach immer neuen Wegen, interne Abläufe zu optimieren. So kam es denn, daß in dieser Firma eine der ersten EDV - Anlagen installiert wurde. Doch was war schon eine EDV - Anlage ohne Programme. So zogen also die Spezialisten der Firma aus, um die richtige Software zu finden.

Doch welch eine Entäuschung. Auf der ganzen Welt gab es kein Programm, welches den Anforderungen der Manager genügt hätte. Einige Firmen boten nun dieser Firma entsprechende Programmpakete an. Diese allerdings mußten erst entwickelt werden. Warum aber eine fremde Firma mit der Entwicklung beauftragen, wenn es auch im eigenen Haus getan werden kann.

So also wurde der Personalchef in die oberste Etage gerufen. Er erhielt den Auftrag, geeignetes Personal einzustellen. Geld spiele keine Rolle, wurde ihm noch mit auf dem Weg gegeben. Anzeigen wurden geschaltet und die ersten Bewerbungen trafen ein. So nahm das Verhängnis seinen Lauf. Es trafen nur wenige Bewerbungen ein, da es zu dieser Zeit, so gut wie keine Programmierer gab.

Für viel Geld wurde nun ein Mann von der Herstellerfirma der EDV - Anlage abgeworben. Er erhielt den Auftrag, die komplette Software für die Firma zu entwickeln. Gesagt, getan. Mit Elan geht der neue Mann ans Werk. Schon nach drei Jahren präsentiert er der Geschäftsleitung eine gut durchdachte Software. Die Software ist so gut, daß man sich entschließt weitere Anwendungen der EDV zu übertragen.

Wenige Jahre später gibt es kaum noch einen Bereich in der Verwaltung, der nicht vom Computer abhängt. Unser Programmierer arbeitet unermüdlich. Er verdient mittlerweile mehr als die Manager der Firma. Die Jahre ziehen ins Land und unser Programmierer übersteht drei Herzattacken. Immer wieder verbessert er seine Programme. Die Firmenleitung ist mit der Entwicklung sehr zufrieden.

Eines Tages aber wird es zuviel für unseren Programmierer. Nach seiner vierten Herzattacke verstirbt er im Krankenhaus. Aber seine Software ist so gut, daß es nun auch ohne Ihn geht. Es vergehen zwei weitere Jahre ohne große Probleme. Die Frima blüht und gedeiht. Die Umsätze steigen und neue Mitarbeiter werden eingestellt.

Eines Tages aber zieht Gefahr herauf. Die EDV - Anlage, mittlerweile schon gut zehn Jahre alt, wird zu klein. Durch immer mehr Arbeit werden die Arbeitsabläufe zu langsam. Die Manager versuchen eine Lösung zu finden. Es dauert auch nicht lange und es steht fest, eine größere Anlage muß her. "Kein Problem",sagt der Hersteller. Da die Programme alle in der eigenen Firma entwickelt worden sein, müsse man sie nur an die veränderte Hardware anpassen.

Zu diesem Zeitpunkt wurden vier Programmierer im Unternehmen beschäftigt. Sie erhielten nun den Auftrag die vorhandenen Programme an die neue EDV - Anlage anzupassen. Kein Problem war die Antwort aus der EDV - Abteilung. So wurde also die neue Anlage gekauft.

Es vergingen drei Monate. Die neue Anlage stand immer noch wie am ersten Tag. Keine Anwendung lief. Die Manager beauftragten nun fremde Spezialisten.

Diese stellten nach weiteren drei Monaten fest, es sei billiger, eine neue Software zu entwickeln, als die alte anzupassen. Die Begründung: Es seien keinerlei Unterlagen zu finden, die ein sinnvolles Arbeiten ermöglichen.

Also wurden nun alle Programme neu entwickelt. Diese Neuentwicklung aber zog sich länger hin als erwartet. Nur mit letzter Kraft konnte dann ein Konkurs abgewendet werden.

Eine schöne Geschichte werden Sie jetzt sicherlich denken. Sie ist aber so ähnlich wirklich passiert. Seit dieser Zeit ist es auch eine der Grundpflichten eines Programmierers, alle seine Schritte, von der Aufgabenstellung über die Idee bis zum fertigen Programm, zu dokumentieren. Auch Sie sollten alle Ihre eigenen Programme dokumentieren. So wissen Sie dann auch noch nach Jahren warum, wieso und wie Sie ein Programm erstellt haben.

3.1 Der erste Schritt

Bevor wir ein Programm schreiben können, muß erst einmal ein Problem vorhanden sein, welches mit Hilfe eines Computers gelöst werden soll. Wir werden uns im Folgenden mit den einzelnen Phasen der Programmentwicklung beschäftigen. Als Vorlage dient uns das letzte Programm aus dem vorigen Kapitel. Die Aufgabenstellung könnte wie folgt gelautet haben:

Entwickeln Sie ein Programm, daß auf einen festen Adressenbestand zurückgreifen kann. Die Adressen müssen jederzeit aufgelistet werden können. Weiterhin muß das Programm in der Lage sein, eine Direktfakturierung durchführen zu können. Dabei sollen automatisch die Gesamtpreise ermittelt werden. Am Ende soll die Mehrwertsteuer errechnet und ausgegeben werden. Alle Einzelpreise sind Nettopreise.

So etwa könnte die Anforderung lauten. Im ersten Schritt müßen wir nun die für uns wichtigen Fakten heraussuchen.

1. Fester Adressenbestand
2. Adressenliste
3. Direktfakturierung
4. Automatische Ermittlung der Gesamtpreise
5. Automatische Ermittlung der MWSt
6. MWST ausgeben
7. Erstellung des Rechnungsbetrages

Damit haben wir uns nun einen Forderungskatalog mit acht Punkten erstellt. Da wir es hier nur mit einem kleinen Programm zu tun haben, bleibt dieser Katalog übersichtlich. Trotzdem müssen wir Ihn nocheinmal mit der Aufgabenstellung vergleichen. Erst wenn wir völlig sicher sind, daß wir nichts übersehen haben, können wir uns dem nächsten Schritt zuwenden. Bevor wir dies tun, sei noch erwähnt, daß dieser Katalog das erste Arbeitsmittel des Programmierers ist. Mit diesem geht er zu seinem Auftraggeber und spricht noch einmal alles durch. Jetzt, wo das Problem in Einzelteile aufgeschlüsselt ist, kann der Auftraggeber noch korrigieren ohne viel Mehrarbeit zu ver-

ursachen.

3.2 Der zweite Schritt

Der zweite Schritt besteht darin, sich Gedanken zu machen, in welcher Programmiersprache gearbeitet werden soll. Außerdem muß ein Konzept für das Programm entworfen werden. Die Programmiersprache ist für uns keine Frage, da wir ja BASIC lernen. Wir aber wollen uns mit dem Konzept beschäftigen.

Dazu nehmen wir den Katalog, den wir vorher erstellt haben.

Als Punkt 1 steht dort ein fester Adressenbestand. Wenn diese Adressen wie in unserem Fall nur wenig Speicherplatz benötigen, können wir sie als Betandteil des Programmes verwenden. Wir beginnen nun ein Programmkonzept zu entwickeln. Unser Programm soll von jedem Laien zu bedienen sein. Inklusiv des Programmendes benötigen wir also 3 Programmteile. Wir benötigen also erst einmal einen Bildschirmaufbau, der dem Anwender sagt, was er mit unserem Programm alles bearbeiten kann.

Wir suchen uns nun für jeden der drei Programmteile einen funktionellen Namen:

1. Adressenliste
2. Rechnungserstellung
3. Programmende

Damit haben wir unser Programm in drei Blöcke unterteilt. Als nächstes müssen wir ermitteln, wieviel Variablen wir benötigen. Gleichzeitig suchen wir auch nach sinnvollen Variablennamen. Wir benötigen:

Wir arbeiten nun alle drei Programmteile chronologisch durch und erstellen eine Variablenliste. Die Liste für unser Programm finden Sie auf der nächsten Seite.

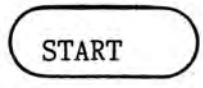
Lfd. Nr.	Bezeichnung	Variablenname
1. Adressenliste		
1	Firma	FIRMA$
2	Name	NAME$
3	Straße	STR$
4	Postleitzahl	PLZ
5	Ort	ORT$
6	Telefon	TEL$
2. Rechnungserstellung		
7	Menge	MENGE
8	Artikel	ART$
9	Preis	PREIS
10	Einzelgesamtpreis	GPREIS
11	Gesamt Netto	ZPREIS
12	Mehrwertsteuer	MWST
13	Rechnungsbetrag	RBETRAG
14	Gesuchte Firma	GFIRMA

Soweit unsere Variablenliste. Als nächstes müssen wir uns nun Gedanken über den Programmaublauf machen. Hierzu müssen wir dann einen Programmablaufplan erstellen. Das aber ist schon der nächste Schritt.

3.3 Der Dritte Schritt

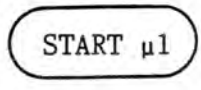
Nun kommen wir also zum zentralen Punkt der Programmentwicklung. Der Programmablaufplan. Dieser Plan besteht aus einer Reihe von Symbolen. Bevor wir nun unseren Plan erstellen, müssen wir erst einmal die Symbole kennenlernen.

Symbole für den Programmablaufplan

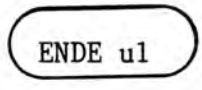


Start des Programmes

Ende des Programmes



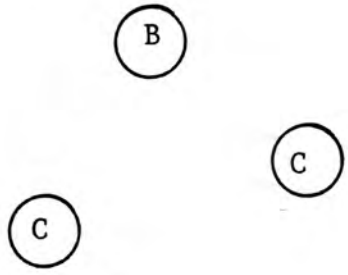
Beginn eines Unterprogrammes



Ende eines Unterprogrammes

Connection

Verbindungspunkte jeder Punkt enthält



eine Bezeichnung. Die gleiche Bezeichnung muß an einer anderen Stelle des Planes ebenfalls vorhanden sein.
Die Punkte werden auch für Routinen eingesetzt, die eine feste Rücksprung-adresse haben.

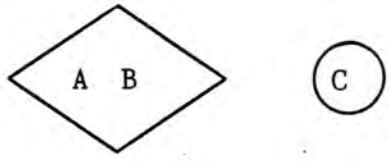
Ein- Ausgabe Symbole

Eingabe:

Der Text im Symbol gibt Auskunft über die Art der Eingabe.

Ausgabe

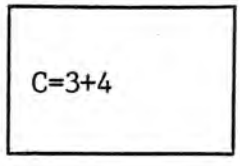
Auch hier gibt der enthaltene Text Auskunft über die Art der Ausgabe



Entscheidungs-Symbol

Wenn A kleiner B ist geht das Programm den Weg über Ja

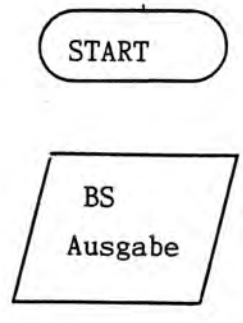
Ist A gleich oder größer B wird das Programm über Nein fortgesetzt

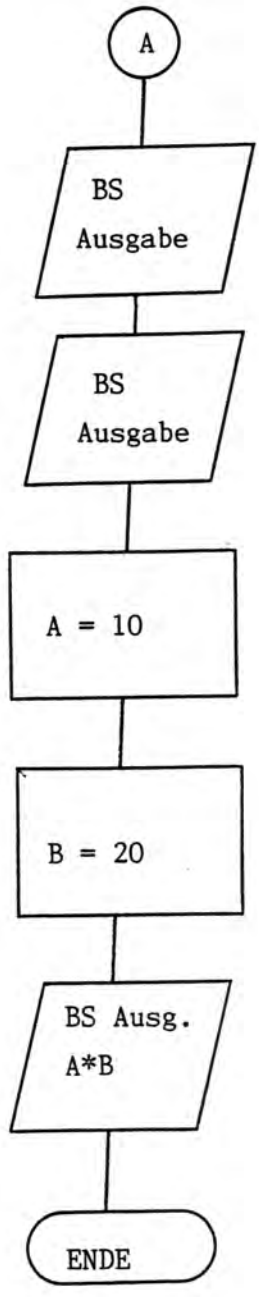


Allgemeine Operation

Hier werden interne Verarbeitungen dargestellt.
z.B. eine LET - Anweisung.

Soweit nun unsere Symbole. Zum besseren Verständnis wollen wir zunächst einen sehr kurzen Ablaufplan mit dem fertigen Programm erstellen. Wir verwenden als Beispiel unser "PROG1".



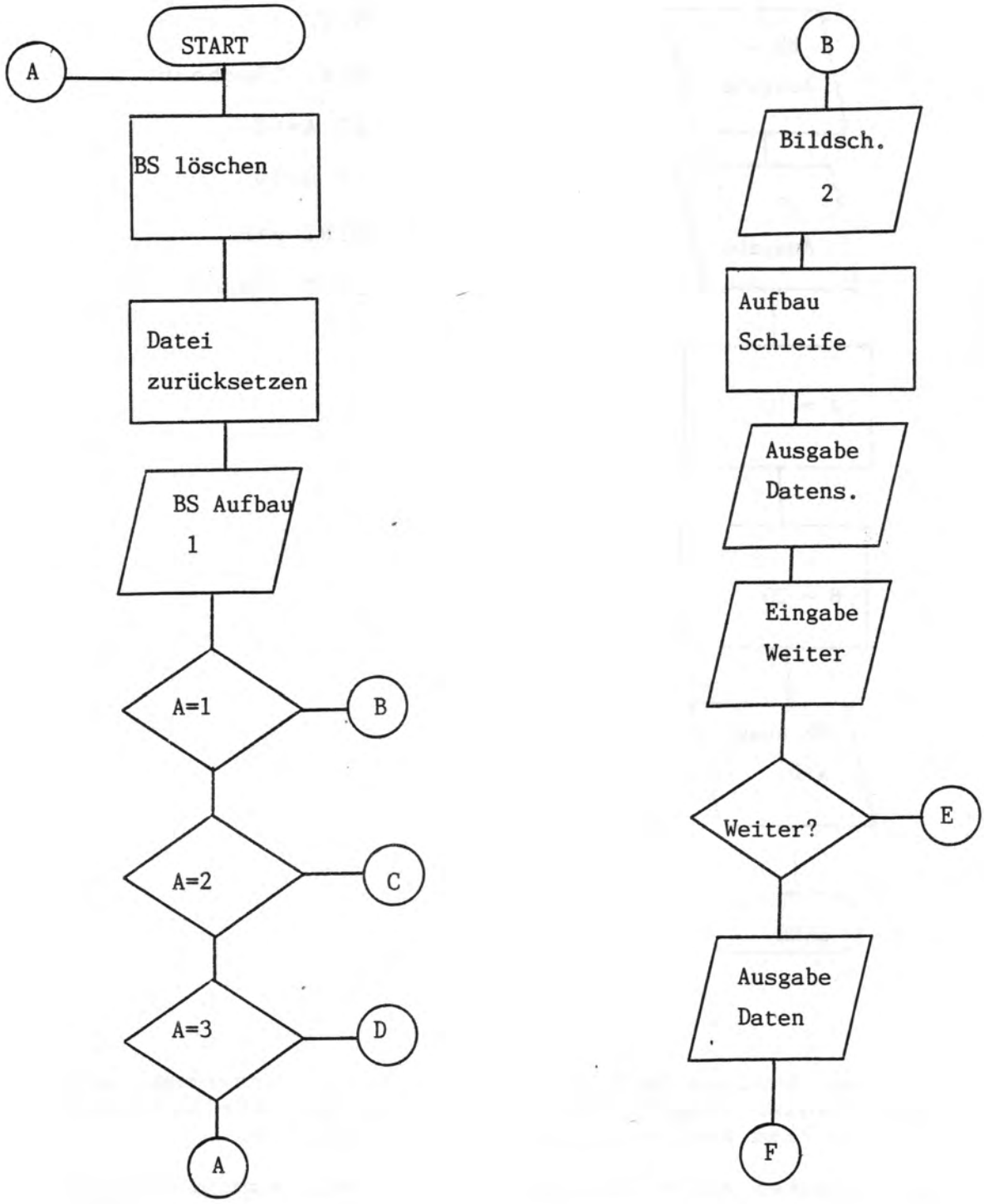


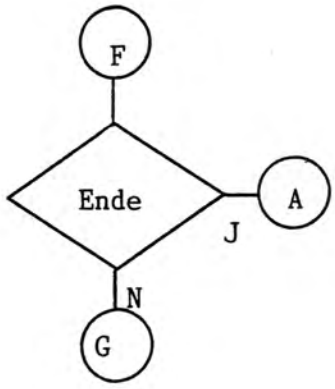

```
1 PRINT "PROG1"
2 PRINT "Dieses Pr.."
3 PRINT
4 PRINT "Rechenb.."
5 LET A=10
6 LET B=20
7 PRINT A*B
8 PRINT "Ende"
```

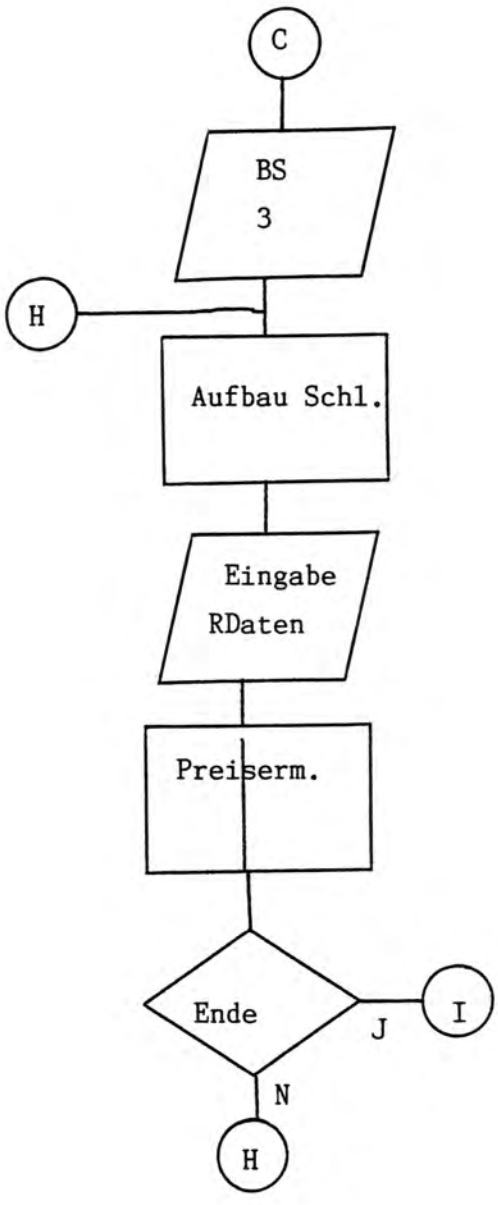
Soweit unser kleines Beispiel. Wenn Sie nun Programm und Plan miteinander vergleichen, werden Sie feststellen, daß der Weg vom Plan zum Programm nur sehr kurz ist.

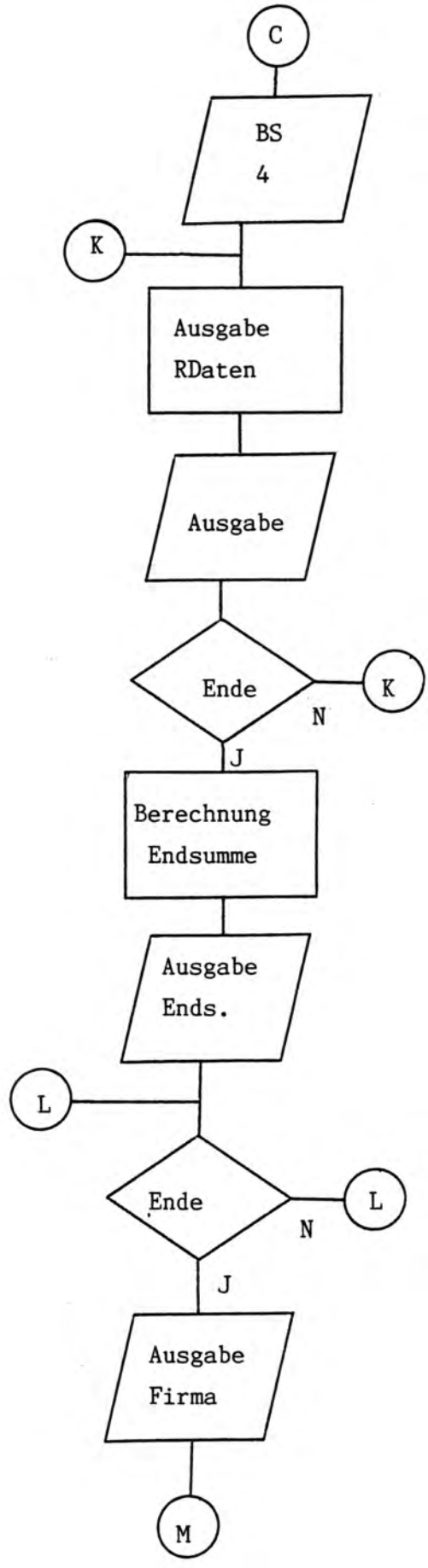
Ab der nächsten Seite finden Sie den Programmablaufplan unseres letzten Programmes. Dieser ist dort erst einmal kommentarlos dargestellt. Versuchen Sie einmal diesen Plan unter Zuhilfenahme des Programmes zu verstehen. Im Anschluß werden wir den Plan dann gemeinsam analysieren.

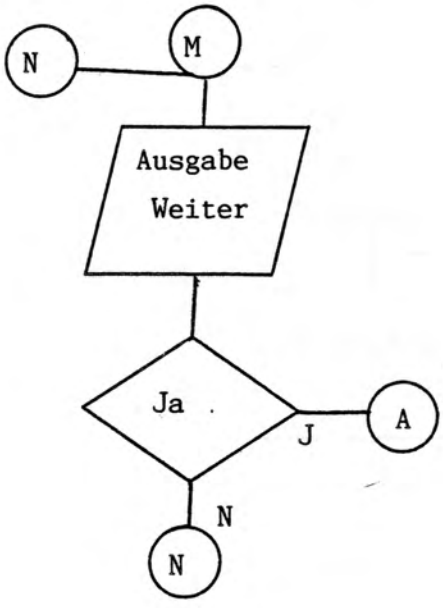
Programmablaufplan

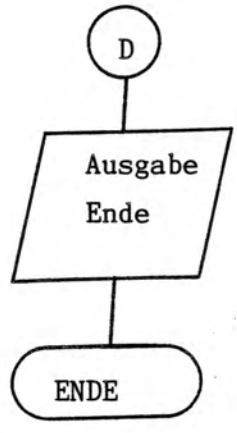


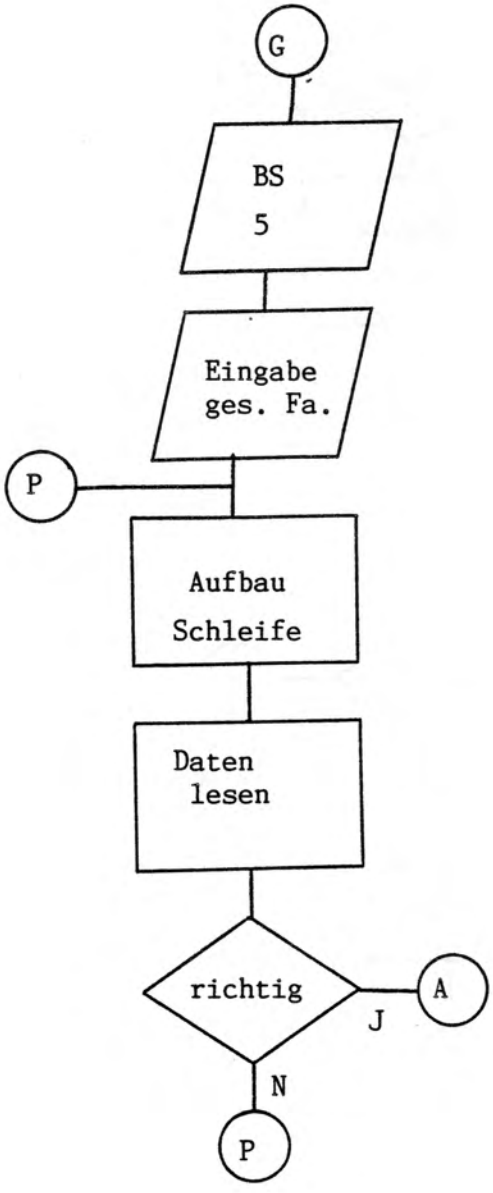

F
Ende
A
J
N
G


C
BS
3
H
Aufbau Schl.
Eingabe
RDaten
Preiserm.
Ende
J
I
N
H


C
BS
4
K
Ausgabe
RDaten
Ausgabe
Ende
K
N
J
Berechnung
Endsumme
Ausgabe
Ends.
L
Ende
L
N
J
Ausgabe
Firma
M


N
M
Ausgabe
Weiter
Ja
J
A
N
N


D
Ausgabe
Ende
ENDE


G
BS
5
Eingabe
ges. Fa.
P
Aufbau
Schleife
Daten
lesen
richtig
A
J
N
P

Soweit unser Programmablaufplan. Sie können mit diesem Plan das ganze Programm auf dem Papier testen. Außerdem ist der Plan nicht von einer bestimmten Programmiersprache abhängig. Trotzdem ist es nun möglich, den Plan ohne große Umwege in ein lauffähiges Programm umzusetzen.

Denken Sie daran, der Programmablaufplan dient als erste Testgrundlage für das spätere Programm. Wir wollen nun ein wenig üben.

Ü. 3.1 Aufgabe:

Erstellen Sie einen Programmablaufplan, danach Testen Sie den Programmablauf. Wenn der Test ohne Probleme erfolgt ist, schreiben Sie ein Programm.

Problemstellung:

Entwickeln Sie ein Programm, daß eine Zufallszahl zwischen 0 und 49 ermittelt. Dieser Vorgang soll sechs mal wiederholt werden. Erst nachdem die 6. Zahl ermittelt wurde, soll eine Ausgabe aller Zahlen auf den Bildschirm erfolgen.
Das Programm soll sich auf dem Bildschirm als "Lotto - Tip der Woche" melden.

Da es jedem Programmierer freigestellt ist, eine Aufgabe nach seinen eigenen Vorstellungen zu lösen, gibt es für dieses Programm keine einzig richtige Lösung. Wichtig ist, daß alle Anforderungen erfüllt werden.

Sollten Sie Schwierigkeiten haben, dieses Programm zu realisieren, finden Sie im Anhang eine Beispiellösung.

Sie können diese Aufgabe mit den bisher behandelten Anweisungen komplett lösen. Scheuen Sie sich nicht, noch einmal alle Anweisungen zu studieren.. Nur Übung macht den Meister. Zum besseren Verständnis werden wir alle nachfolgenden Programme erst einmal mit einem Programmablaufplan entwickeln.

3.4 Vierter und Fünfter Schritt

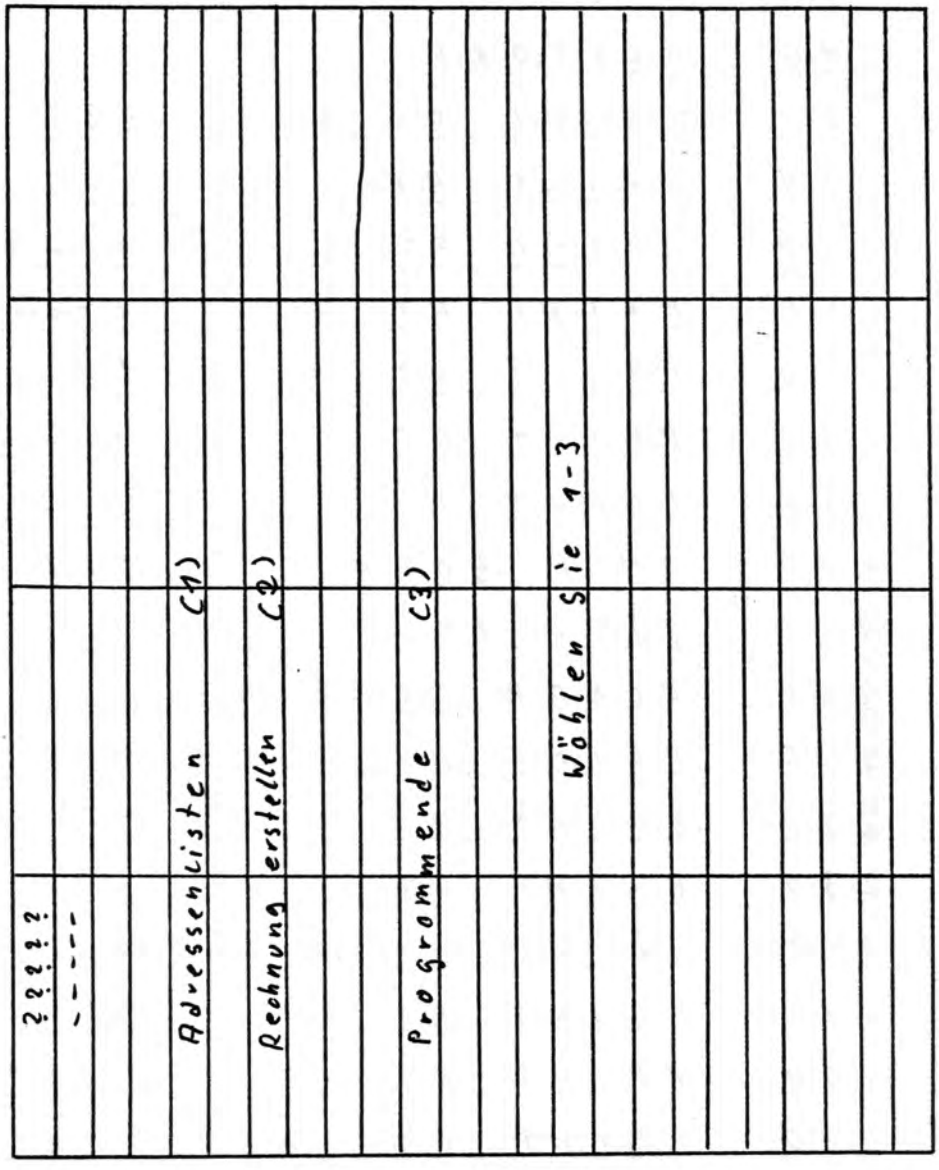
Diese beiden Schritte sind schnell erläutert. Nachdem unser Programm auf dem Papier fehlerfrei läuft, setzen wir es Schritt für Schritt in BASIC-Anweisungen um. Diese Umsetzung erfolgt auch noch auf dem Papier. Im nachfolgenden Beispiel sehen Sie einen solchen handgeschriebenen Aufbau.

Damit auch Sie mit diesen Formularen arbeiten können, befindet sich im Anhang ein DIN A4 Formular. Trennen Sie das Formular aus dem Buch und kopieren Sie es sich in ausreichender Anzahl.

Die seitlich angebrachten Kommentare werden später nicht ins Programm übernommen. Das Feld für die Programmieranweisungen ist auf ein Bildschirmformat von 40 Zeichen abgestimmt, wobei Sie aber immer den Platz für die Zeilennummer abziehen müssen.

Bevor Sie nun mit dem Programmieren beginnen, müssen Sie sich zunächst einmal Gedanken über den Bildschirmaufbau machen. Auch hierzu finden Sie im Anhang zwei entsprechende Formulare.

Bildschirmaufbau



Für jeden neuen Bildschirm wird ein neues Formular benötigt. Wir belassen es hier bei dem einen Beispiel. Wenn Sie Lust haben, können Sie ja die anderen nachvollziehen. Es gibt Ihnen dann auch gleich eine gewisse Fertigkeit, Ihren eigenen Bildschirmaufbau zu entwickeln.

Programmumsetzung

```
10   CLEAR SCREEN
15   RESTORE
20   PRINT AT 1,5:"???
30   PRINT AT 2,5:"---
40   PRINT AT 5,5:"Adress---
50   PRINT AT 7,5:"Rechnung
70   PRINT AT 11,5:"Ende
80   PRINT AT 15,5:"Wählen
90   INPUT A
100  ON A GOTO 200,500,1000
110  GOTO 10
200  CLEAR SCREEN
210  PRINT AT 1,10:"Adressen
220  PRINT AT 2,10:"-------
230  PRINT AT 5,1:"Firma
240  PRINT AT 8,1:"Name
250  PRINT AT 11,1:"Strasse
260  PRINT AT 14,1:"PLZ.Ort
270  PRINT AT 17,1:"Telefon
280  FOR X=1 TO 10
290  READ FIRMA$,NAME$
300  PRINT AT 5,20:FIRMA$
310  PRINT AT 8,20:NAME$
320  PRINT AT 14,20:PLZ;ORT$
```

Programm: Blatt: 1

```
330 PRINT AT 17,20:TEL$
350 PRINT AT 20,15:"Weiter
360 GET W$
370 IF W$="" THEN GOTO 360
380 PRINT AT 5,20:"
390 PRINT AT 8,20:"
400 PRINT AT 11,20:"
410 PRINT AT 14,20:"
420 PRINT AT 17,20:"
430 NEXT X
500 CLEAR SCREEN
505 ZPREIS=Ø
510 PRINT A 1,10:"Rechnung
520 PRINT AT 2,10:"- - - -
530 PRINT AT 5,1:"Menge
540 PRINT AT 8,1:"Artikel
550 PRINT AT 11,1:"Preis
560 PRINT AT
570 INPUT AT 5,10:MENGE(X)
580 INPUT AT 8,10:ART$(X)
590 INPUT AT 11,15:PREIS(X)
600 LET SPREIS=MENGE(X)*PR
610 LET ZPREIS=ZPREIS+SPREI
620 IF Z$<>"J" THEN GOTO
```

Programm: Blatt: 2

```
630 NEXT X
640 GOSUB 1100
650 CLEAR SCREEN
660 PRINT AT 1,10:"Rechnung
670 PRINT AT 2,10:"---
690 PRINT AT 10,1:"Menge"
700 PRINT AT 10,8:"Artikel"
710 PRINT AT 10,20:"EP
720 PRINT AT 10,30:"Gesamt
730 PRINT AT 11,1:"---
740 FOR Y=1 TO X
750 LET Z=12+Y
760 PRINT AT Z,1:MENGE(Y)
770 PRINT AT Z,8:ART$(Y)
780 PRINT AT Z,20:PREIS(Y)
790 PRINT AT Z,30:GPREIS(Y)
800 NEXT Y
810 PRINT AT 19,30:"--
820 PRINT AT 20,30:ZPREIS
830 PRINT AT 21,9:"zuzuegl
840 LET MWST=ZPREIS/100*14
850 PRINT AT 21,30:MWST
860 PRINT AT 22,30:---
870 PRINT AT 23,9:"Rechn
```

Programm: Blatt: 3

```
: _:8:8:0:    :L:E:T: _:R:B:E:T:R: A$:=: 2P:R:E:I:S: +:M:W:S:T:
: _:8:9:0:    :P:R:I:N:T: _:A:T: _:2:3:,:3:0:;:R:B:E:T:R:A:S: _:
: _:9:0:0:    :P:R:I:N:T: _:A:T: _:2:4:,:30:;:": =:=: _: _: _: _: _:
: _:9:1:0:    :P:R:I:N:T: _:A:T: _:4:,:1:;:F:I:R:M:A:$: _: _: _: _:
: _:9:2:0:    :P:R:I:N:T: _:A:T: _:5:,:1:;:N:A:M:E:$: _: _: _: _: _:
: _:9:3:0:    :P:R:I:N:T: _:A:T: _:6:,:1:;:S:T:R:A:S:S:E:$: _: _:
: _:9:4:0:    :P:R:I:N:T: _:A:T: _:8:,:1:;:P:L:Z:;:": _:": ;: _: _:
: _:9:5:0:    :F:O:R: _:I:=:1: _:T:O:10:O:O: _: _: _: _: _: _: _: _:
: _:9:6:0:    :P:R:I:N:T: _:A:T: _:5:,:30:;:": W:e:i:t:e:r: _: _:
: _:9:7:0:    :P:R:I:N:T: _:A:T: _:6:,:30:;:": T:a:s:t:e: _: _: _:
: _:9:8:0:    :P:R:I:N:T: _:A:T: _:7:,:30:;:": d:r:ü:c:K:e:n: _:
: _:9:9:0:    :S:E:T: _:I:$: _: _: _: _: _: _: _: _: _: _: _: _: _: _:
:1:0:0:0:     :L:F: _:L:$:=:": ": _: _: _: _: _: _: _: _: _: _: _: _:
:1:0:1:0:     :S:O:T:O: _:1:0: _: _: _: _: _: _: _: _: _: _: _: _: _:
:1:1:0:0:     :C:L:E:A:R: _:S:C:R:E:E:N: _: _: _: _: _: _: _: _: _:
:1:1:1:0:     :P:R:I:N:T: _:": E:i:n:g:a:b:e: _:R:e:c:h:n:U:n:g:
:1:1:2:0:     :P:R:I:N:T: _:": =: _: _: _: _: _: _: _: _: _: _: _: _:
:1:1:3:0:     :I:N:P:U:T: _:A:T: _:5:,:1:;:P:R:O:M:P:T:": _: _: _:
:1:1:4:0:     :F:O:R: _:O:=:1:T:O: _:10: _: _: _: _: _: _: _: _: _:
:1:1:5:0:     :R:E:A:D: _:F:I:R:M:A:$:,:-: _: _: _: _: _: _: _: _:
:1:1:6:0:     :I:F: _:S:F:I:R:M:A:$:=:F:R:M:A:$: _:T:H:E:N: _:
:1:1:7:0:     :N:E:X:T: _:O: _: _: _: _: _: _: _: _: _: _: _: _: _:
:1:1:8:0:     :P:R:I:N:T: _:": F:i:r:m:o: _:n:i:c:h:t: _: _: _: _:
:1:1:9:0:     :R:E:S:T:O:R:E: _: _: _: _: _: _: _: _: _: _: _: _: _:
```

Programm: Blatt: 4

```
:1:2:0:0:   :S:0:T:0: :1:1:0:0:
:1:2:5:0:   :R:E:T:U:R:N:
:1:5:0:0:   :C:L:E:A:R: :S:C:R:E:E:N:
:1:5:1:0:   :P:R:I:N:T: :A:T: :1:2:,:1:0:: ":D:0:5:
:1:5:2:0:   :E:N:D:
:2:0:0:0:   :0:A:T:A:
:2:0:1:0:   :D:A:T:A:
:2:0:2:0:   :D:A:T:A:
:2:0:3:0:   :D:A:T:A:
:2:0:4:0:   :D:A:T:A:
:2:0:5:0:   :D:A:T:A:
:2:0:6:0:   :D:A:T:A:
:2:0:7:0:   :D:A:T:A:
:2:0:8:0:   :D:A:T:A:
:2:0:9:0:   :D:A:T:A:
```

Programm: Blatt: 5

Ich habe absichtlich das ganze Programm aufgenommen, damit Sie es Schritt für Schritt nachvollziehen können. Und hier nun eine weitere Übungsaufgabe:

Aufgabenstellung:

Entwickeln Sie ein Programm, das nach Eingabe von 10 Zahlen diese sortiert und in der richtigen Reihenfolge (aufsteigend) auf dem Bildschirm ausgibt. Der Bildschirm sollte dabei so aufgebaut werden, daß nur 2 Zahlen in einer Zeile stehen. Außerdem soll der Bildschirm optimal ausgenutzt werden. Das heißt, daß Sie die Ausgabe so organisieren, daß der Betrachter das Gefühl hat, der ganze Bildschirm wird benutzt.

Arbeitsanweisungen:

Erstellen Sie einen Katalog sowie eine Variablenliste. Danach entwickeln Sie einen Programmablaufplan. Mit dem Plan testen Sie dann Ihr Programm. Erst wenn Sie sicher sind, daß Ihr Programm richtig entwickelt ist, setzen Sie Ihr Programm in BASIC - Anweisungen um. Auch diese testen Sie wieder auf dem Papier. Danach geben Sie Ihr Programm in den Computer ein.

Sollte das Programm nicht gleich beim ersten Versuch laufen, grämen Sie sich nicht. Auch alte Hasen bringen Ihre Programme in den seltensten Fällen beim ersten Versuch zum Laufen.

Damit haben wir auch gleich den letzten Schritt angesprochen. Nachdem das Programm in den Computer eingetippt wurde, beginnt die schwierigste Arbeit. Das Programm muß nun so lange getestet und verändert werden, bis es fehlerlos läuft.

4. Externe Datenträger

Nachdem wir uns nun reichlich mit der Theorie beschäftigt haben, wollen wir wieder zur Praxis schreiten. In diesem Abschnitt werden wir uns mit dem Cassettenrecorder und dem Diskettenlaufwerk beschäftigen. Im ersten Teil kommen alle Anwender auf Ihre Kosten, die noch kein Diskettenlaufwerk besitzen. Im zweiten Teil werden wir uns dann mit dem Diskettenlaufwerk beschäftigen. Auch wenn Sie noch kein Diskettenlaufwerk haben, sollten Sie sich den zweiten Teil durchlesen.

Bevor wir aber nun in die Materie einsteigen, wollen wir uns noch einmal vor Augen führen, wozu diese beiden Geräte eigentlich gebraucht werden.

Beide Geräte dienen zu Speicherung von Daten und Programmen. Wie Sie wissen, verliert der Computer nach dem Abschalten alle Programme und Daten. Damit Sie nun Ihre Programme immer wieder verwenden können, benötigen wir einen externen Speicher, der seine Daten und Programme auch dann behält, wenn Sie Ihren Computer über längere Zeit nicht benutzen. Der Vorteil von beiden Geräten, ist, daß der Speicher ausgetauscht werden kann. Die Speichermedien sind heute so preiswert, daß Sie sich für wenig Geld eine beachtliche Sammlung von Programmen und Daten zulegen können. Als Speichermedien dienen Cassetten und Disketten.

Vorab noch ein wichtiger Hinweis:

Wenn Sie ein wichtiges Programm geschrieben haben, speichern Sie es immer auf zwei unterschiedliche Datenträger ab. Das gleiche gilt auch für wichtige Daten. Es kann immer mal ein Datenträger beschädigt werden. Wenn Sie dann keine Kopie haben, kann das < sehr > ärgerlich sein. Im günstigsten Fall brauchen Sie Ihre Programme dann nur neu zu schreiben. Wehe aber, Sie verdienen Ihr Geld mit dem Schreiben von Programmen.

4.1 Der Cassettenrecorder

Mit dem Enterprise Computer haben Sie einen der wenigen Computer erstanden, für den Sie keinen speziellen Cassettenrecorder kaufen müssen.

Sie können jeden preiswerten Recorder verwenden. Die Kabel für die Verbindung mit dem Computer haben Sie ja auch schon mit dem Enterprise gekauft. Den Anschluß des Recorders können Sie im Handbuch auf der Seite 10 nachschlagen. Für die Besitzer der englischen Version will ich es hier aber noch einmal kurz erklären.

Auf der Rückseite des Computers befinden sich vier runde Buchsen. An den Buchsen finden Sie folgende Bezeichnungen:

1. REM 1
2. IN
3. OUT
4. REM 2

Die beiden Buchsen mit der Bezeichnung REM 1 und REM 2 haben einen Durchmesser von 2,5 mm. Sie dienen als Start und Stop - Leitungen. Wenn Sie an Ihrem Recorder eine REM Buchse haben, können Sie damit den Recorder vom Computer aus starten und stoppen.

Die Buchse IN dient zum Einlesen von Daten in den Computer. Diese Buchse verbinden Sie mit der EAR - Buchse an Ihrem Recorder.

Die Buchse OUT dient zur Übertragung von Daten aus dem Computer in Ihren Recorder. Verbinden Sie diese Buchse mit der Buchse MIC (Mikrofon). Wenn Sie nur einen Recorder verwenden, benötigen Sie die Buchse REM 2 nicht.

Wenn Sie nun den Computer mit Ihrem Cassettenrecorder verbunden haben, müssen wir noch die Lautstärke regulieren. Da Sie mit dem Enterprise Computer so gut wie jeden Cassettenrecorder verwenden können, ist diese Arbeit leider nicht zu vermeiden. Sie brauchen nun etwas Geduld.

Stellen Sie zunächst den Lautstärkeregler in die mittlere Position. Nun legen Sie am besten die Demonstrations-

cassette in den Recorder ein. Nun tippen Sie LOAD ein. Nachdem Sie die <ENTER> - Taste betätigt haben, verändert sich der obere Bildschirmteil. Es erscheint:

searching...

und dazu ein roter Balken.

Wenn Sie nun großes Glück haben, verändert der Balken seine Farbe und aus "searching" wird

LOADING XXXXXXXXX

Das X steht für den Programmnamen der gefunden wurde. Sollte Ihnen also dieses Glück beschieden sein, ist die Einstellung des Recorders abgeschlossen.

Der Normalfall sieht aber anders aus. Sie müssen jetzt so lange die Lautstärke verändern, bis sich die Farbe des Balkens verändert. Danach spulen Sie das Band zurück, und laden noch einmal. Wenn es dann funktioniert sind Sie fertig. Sollte es nicht gleich klappen. Haben Sie bitte Geduld, ich habe fast eine Stunde benötigt.

Nachdem Sie Ihren Cassettenrecorder eingestellt haben, sollten Sie den Lautstärkeregler in irgend einer Form feststellen.

Wie Sie Programme laden oder speichern, haben Sie ja schon gelernt. Wir wollen uns nun mit der Funktion des Cassettenrecorders als Datenträger beschäftigen.

Dazu erinnern wir uns noch einmal an unser Adressenprogramm. Wie Sie sich erinnern, haben wir dort unsere Daten in Form von DATA - Anweisungen im Programm untergebracht. Diese Art ist aber nur dann sinnvoll, wenn es sich um sehr wenige Daten handelt. Wir wollen aber nun ein Programm schaffen, mit dem wir beliebig viele Adressen speichern können. Bevor wir uns nun wieder das Programm ansehen, noch ein Hinweis:

Wenn Sie mit einem Recorder arbeiten, können Sie eine Datei nur einmal zu schreiben öffnen. Wenn Sie also zu einem späteren Zeitpunkt weitere Adressen eingeben wollen, benötigen Sie einen zweiten Recorder. Doch dazu später mehr.

Nachfolgend nun unser neues Adressenprogramm. Wir werden dieses Programm in zwei Teilen entwickeln. Im ersten Teil werden wir uns um die Dateneingabe kümmern. Im zweiten Teil werden wir den Programmteil entwickeln, mit dem wir die Daten vom Band lesen können. Doch zunächst befassen wir uns nun mit dem ersten Teil. Da wir das Programm richtig entwickeln wollen zunächst einmal unseren Forderungskatalog.

Katalog:

1. Grundbild mit Auswahlmöglichkeiten

   Dateneingabe

   Datenausgabe

   Programmende

2. Speicherung der Daten auf Cassette

3. Ausgabe der Daten von Band

Nachdem wir nun diesen recht einfachen Forderungskatalog erstellt haben, müssen wir uns Gedanken über die zu belegenden Variablen machen.

Variablenliste:

Zunächst einmal müssen wir feststellen, wieviel Variablen wir für die Adresse benötigen.

Wir benötigen folgende Felder: Name, Vorname, Straße, Postleitzahl, Ort und Telefon.

Getreu unserem Muster verwenden wir natürlich aussagekräftige Namen. Diese tragen wir dann in unsere Variablenliste ein. Für die Auswahl der Teilprogramme benötigen wir eine weitere Variable. Nachdem wir diese in unsere Liste eingetragen haben, können wir uns den Ablauf weiter vor Augen führen. Wenn wir die Namen eingeben, müssen wir ja eine Möglichkeit haben, die Eingabe abzubrechen. Also eine weitere Variable. Damit hätten wir dann erst einmal alle Variablen zusammen.

Variablenliste

Lfd.Nr.	Bezeichnung	Variablenname
Adresse		
1	Name	NAME$
2	Vorname	VNAME$
3	Straße	STR$
4	Postleitzahl	PLZ
5	Ort	ORT$
6	Telefon	TEL$
Funktionen		
7	Menu - Auswahl	A
8	Weiter	W$

Den Rest der Liste lassen wir erst einmal offen. Es könnte ja sein, daß wir das Programm noch erweitern wollen. Sie sollten generell solche Listen offen gestalten.

Als nächstes sollte nun ein Programmablaufplan auf unserer Checkliste stehen. Also frisch ans Werk.

Hier folgt nun der Ablaufplan für den ersten Teil unseres Programmes (den Sie bitte selbst erstellen).

Soweit nun der Ablaufplan. Gehen wir davon aus, daß der Papiertest gelaufen ist. Auf das handgeschriebene Programmlisting verzichten wir auch an dieser Stelle. Sie sollten jedoch nicht darauf verzichten, wenn Sie eigene Programme entwickeln. Sicher ist sicher!

Programmlisting Teil 1

```
 10 CLEAR SCREEN
 20 PRINT AT 1,10:"A d r e s s e n"
 30 PRINT AT 2,10:"---------------"
 40 PRINT AT 5,1:"Adressen erfassen (1) "
 50 PRINT AT 8,1:"Adresse suchen    (2) "
 60 PRINT AT 11,1:"Ende              (3) "
 70 INPUT AT 13,1,PROMPT "Wählen Sie (1-3) ":A
 80 IF A>3 THEN GOTO 10
 90 ON A GOTO 100,1500,2500
100 CLEAR SCREEN
110 PRINT AT 1,10:"Neue Adressendatei"
120 PRINT AT 2,10:"------------------"
130 OPEN #8:"Adressen" ACCESS OUTPUT
140 PRINT AT 5,1:"Name (0=End) :"
150 PRINT AT 7,1:"Vorname       :"
160 PRINT AT 9,1:"Strasse       :"
170 PRINT AT 11,1:"Postleitzahl  :"
180 PRINT AT 13,1:"Ort           :"
190 PRINT AT 15,1:"Telefon       :"
200 For X=1 TO 200
210 INPUT AT 5,17:NAME$
```

```
220 IF NAME$="0" THEN GOTO 400
230 INPUT AT 7,17:VNAME$
240 INPUT AT 9,17:STR$
250 INPUT AT 11,17:PLZ
260 INPUT AT 13,17:ORT$
270 INPUT AT 15,17:TEL$
280 INPUT AT 18,10,PROMPT "Alle Daten richtig? (j) ":R$
290 IF R$ <> "j" THEN GOTO 210
300 PRINT #8,NAME$,VNAME$,STR$,PLZ,ORT$,TEL$
310 PRINT AT 5,17:"                    "
320 PRINT AT 7,17:"                    "
330 PRINT AT 9,17:"                    "
340 PRINT AT 11,17:"                    "
350 PRINT AT 13,17:"                    "
360 PRINT AT 15,17:"                    "
370 NEXT X
400 CLEAR SCREEN
410 PRINT AT 1,10:"Ende Adresseneingabe"
420 PRINT AT 2,10:"--------------------"
430 PRINT AT 7,5:"Es wurden ";X;" Adressen erfasst."
440 PRINT AT 10,5:"Bitte für neue Adressen neues Band
    verwenden"
450 For Z=1 TO 2000
460 NEXT Z
470 CLOSE #8
480 GOTO 10
```

Soweit der erste Teil unseres Programmes. Es enthält zwei für uns neue Anweisungen.

1. Zeile 130 OPEN #8:"Adressen" ACCESS OUTPUT

2. Zeile 470 CLOSE #8

In Zeile 130 wird ein Kanal geöffnet, damit wir unsere Daten auf eine Cassette bringen können. ACCESS OUTPUT legt fest, daß wir auf diesen Kanal nur schreiben können. Der Name "Adressen" bezeichnet den Dateinamen. Wenn Sie nun alle Adressen eingegeben haben, wird dieser Kanal mit der Anweisung in Zeile 470 wieder geschlossen.

Mit der Anweisung CLOSE wird aber nicht nur der Kanal geschlossen, sondern auch noch im Rechner befindliche Daten auf unser Band geschrieben.

Nachdem wir unsere Datei geschlossen haben, können wir sie nicht mehr zum Schreiben öffnen. Für unser Programm bedeutet dies, daß alle erfassten Adressen eine neue geschlossene Datei bilden. Wollen wir neue Adressen erfassen, empfiehlt es sich, ein neues Band zu verwenden. Wie in allen Bereichen des Lebens gibt es auch hier einen Ausweg. Diesen werden wir uns aber erst nach Fertigstellung unseres Programmes ansehen.

Uns fehlen noch zwei Programmteile. Als nächstes müssen wir uns ein Programm schaffen, mit dem wir die erfassten Daten wieder lesen können. Dazu sehen wir uns erst einmal den zweiten Teil des Ablaufplanes an. Wer es sich schon zutraut, der kann selber einen Ablaufplan entwickeln und diesen dann mit dem hier aufgezeigten vergleichen.

Beachten Sie dabei, daß das Grundbild bereits im ersten Teil unseres Planes enthalten ist.

Da der dritte Teil unseres Programmes sehr kurz ist, werden wir ihn in den zweiten Teil integrieren.

Programmablaufplan Teil 2

(Sie dürfen wieder zeichnen)

Soweit unser Ablaufplan. Beide Teile ergeben nun unser Gesamtprogramm. Es fehlt uns nur noch das Programmlisting.

Programmlisting Teil 2

```
1500 CLEAR SCREEN
1510 PRINT AT 1,10:"Adressenliste"
1520 PRINT AT 2,10:"-------------"
1530 OPEN #8:"Adressen" ACCESS INPUT
1540 PRINT AT 5,1:"Name          : "
1550 PRINT AT 7,1:"Vorname       : "
1560 PRINT AT 9,1:"Straße        : "
1570 PRINT AT 11,1:"Postleitzahl : "
1580 PRINT AT 13,1:"Ort           : "
1580 PRINT AT 15,1:"Telefon       : "
1600 FOR X=1 TO ? **** ? = Anzahl Ihrer Datensätze ****
1610 INPUT #8,NAME$,VNAME$,STR$,PLZ,ORT$,TEL$
1620 PRINT AT 5,17:NAME$
1630 PRINT AT 7,17:VNAME$
1640 PRINT AT 9,17:STR$
1650 PRINT AT 11,17:PLZ
1660 PRINT AT 13,17:ORT$
1670 PRINT AT 15,17:TEL$
1680 PRINT AT 18,10:"Weiter mit beliebiger Taste ";
1690 GET W$
1700 IF W$="" THEN GOTO 1690
1710 PRINT W$
1720 PRINT AT 5,17:"                    "
```

```
1730 PRINT AT 7,17:"                          "
1740 PRINT AT 9,17:"                          "
1750 PRINT AT 11,17:"                          "
1760 PRINT AT 13,17:"                          "
1770 PRINT AT 15,17:"                          "
1780 NEXT X
1790 CLEAR SCREEN
1800 PRINT AT 1,10:"Ende Adressenliste"
1810 PRINT AT 2,10:"------------------"
1820 PRINT AT 12,10:"Es wurden ";X;" Adressen gelistet."
1830 FOR Z=1 TO 2000
1840 NEXT Z
1850 CLOSE #8
1860 GOTO 10
2500 CLEAR SCREEN
2510 PRINT AT 12,15:"ENDE Adressenprogramm"
2520 END
```

Soweit unser komplettes Programm.

Dieses Programm läßt sich nun beliebig erweitern. Wir haben bisher nur zwei Funktionen eingebaut. Im zweiten Teil des Kurses werden wir dieses Programm als Grundlage für eine komfortable Adressenverwaltung verwenden.

Erinnern Sie sich noch an die GOSUB - Anweisung?

Wenn Sie unser Programm genau betrachten, werden Sie zwei identische Programmteile finden. Es handelt sich um den Bildschirmaufbau. Diesen Verwenden wir bei der Eingabe unserer Daten sowie bei der Ausgabe. Wir können diesen Aufbau in einem Unterprogramm erfassen und an den ent-

sprechenden Stellen jederzeit aufrufen. Versuchen Sie nun, das Programm so zu ändern, daß der Bildschirmaufbau nur einmal im Programm vorhanden ist. Im Anhang finden Sie eine geänderte Version.

Im ersten Programmteil haben wir erfahren, daß wir unsere Datei nur einmal zum Schreiben öffnen können. Es gibt nun einen Weg, wie wir unsere Datei jederzeit verändern können.

Um diesen Weg zu gehen, benötigen Sie allerdings zwei Cassettenrecorder. Den einen verbinden Sie mit der Buchse IN sowie REM 1. Dieser Recorder dient zum Lesen unserer Daten. Wir benötigen ihn für die Cassette, auf der sich unsere bisherige Datei befindet.

Den zweiten Recorder verbinden Sie mit der Buchse OUT und REM 2. Diesen Recorder benötigen wir für unsere neue Datei. In diesen Recorder legen Sie dann eine neue Cassette.

Nachdem wir nun die beiden Recorder angeschlossen haben, müssen wir noch einen entsprechenden Programmteil entwickeln. Mit diesem Programmteil können wir dann unsere Datei jederzeit um neue Adressen ergänzen. Wir verzichten an dieser Stelle auf einen neuen Ablaufplan. Sie sollten ihn jedoch erstellen. Unser Grundbild werde ich hier auch nicht verändern. Es ist Ihre Aufgabe, diesen Programmteil selbstständig zu entwickeln. Da Sie jedoch einige wichtige Dinge beachten müssen, werde ich die neuen Anweisungen hier erklären.

Da wir nun mit zwei Recordern arbeiten, müssen wir auch zwei Kanäle öffnen. Hierbei ist es wichtig, daß wir den Massenspreicher kennzeichnen. Die Kennzeichnung erreichen wir mit einer veränderten OPEN - Anweisung

```
OPEN #8:"TAPE-1:Adressen" ACCESS INPUT
```

Mit dieser Anweisung haben wir unseren ersten Recorder zum Einlesen der Daten definiert. Nun wollen wir noch unseren zweiten Recorder definieren.

```
OPEN #9:"TAPE-2:Adressen" ACCESS OUTPUT
```

Damit haben wir unsere beiden Recorder definiert. Vergessen Sie bitte nicht, beide Kanäle nach getaner Arbeit wieder zu schließen. Nun benötigen wir nur noch einen Programmteil, der die Daten von Recorder 1 ließt und auf das Band in Recorder 2 schreibt.

Damit wir nicht immer wissen müssen, wieviel Datensätze auf dem Band sind, betrachten wir uns nun eine veränderte INPUT - Anweisung.

```
INPUT #8,IF MISSING THEN GOTO xxx:NAME$,VNAME$,.....
```

Mit dem Zusatz IF MISSING THEN GOTO xxx verhindern wir, daß das Programm nach dem letzten Satz auf dem Band abgebrochen wird. Mit dieser Funktion benötigen wir die Anzahl unserer Adressen für die FOR NEXT Schleife nicht mehr.

Nachdem nun der letzte Satz vom Band gelesen wurde, können wir neue Daten über die Tastatur eingeben. Der neue Programmteil muß diese Eingabe auch enthalten. Erst wenn wir den Kanal wieder schließen, ist unsere neue Datei fertig.

Verändern Sie nun Ihr Programm entsprechend. Auf dem Datenträger finden Sie dieses neue Programm unter der Bezeichnung "ADR3".

Soweit der Cassettenrecorder. Als nächstes betrachten wir uns nun das Diskettenlaufwerk.

4.2 Das Diskettenlaufwerk

Das Diskettenlaufwerk ist ein Massenspeicher, der sich durch seine Schnelligkeit besonders auszeichnet. Daten und Programme werden mit der mehrfachen Geschwindigkeit eines Cassettenrecorders übertragen.

Da aber ein Diskettenlaufwerk etwa zehnmal soviel wie ein Cassettenrecorder kostet, müssen noch andere Vorteile vorhanden sein, um die Geldausgabe zu rechtfertigen.

Vorteile einer Diskette

1. Schnellere Datenübertragung
2. Schnellerer Zugriff auf die Programme
3. Systemvorteile durch DOS

Die schnellere Datenübertragung haben wir ja schon angesprochen. Betrachten wir uns also gleich den nächsten Punkt.

2. Schnellerer Zugriff auf die Programme

Erinnern wir uns an die Cassette. Hier werden die Programme nacheinander auf das Band geschrieben. Wenn wir nun ein Programm suchen, wird der Computer ein Programm nach dem anderen überprüfen, ob es das gesuchte ist. Wird das Programm gefunden, beginnt der Ladevorgang. Je mehr Programme nun auf der Cassette stehen, desto länger kann das Laden dauern.

Auf der Disketten werden die Programme nun ganz anders gespeichert. Eine Diskette verfügt über ein Inhaltsverzeichnis. In diesem Verzeichnis stehen alle Programmnamen und die Stellen, an denen sich die einzelnen Programme auf

der Diskette befinden. Wenn Sie nun ein Programm aufrufen, überprüft das Disketten - Betriebssystem im Inhaltsverzeichnis, ob das Programm auf der Diskette vorhanden ist und an welcher Stelle es steht.

Ist das Programm auf der Diskette, wird der Schreib-Lesekopf direkt an die richtige Stelle gebracht. Das Programm wird also direkt gelesen.

Nachteil: Sie müssen immer einen Programmnamen angeben.

Wenn Sie von der Cassette das erste Programm einlesen wollen, benötigen Sie keinen Namen. Bei der Diskette können Sie dieses Verfahren nicht anwenden. Alle Programme können nur mit einem Namen gespeichert oder gelesen werden.

Systemvorteile

Es gibt professionelle Programme, die nur mit dem Diskettenlaufwerk arbeiten können. Diese Programme sind so umfangreich, daß es unmöglich ist, diese von der Cassette zu "fahren". Zum Beispiel:

DBASE II: DBASE II ist ein Datenbanksystem mit direktem Zugriff auf große Datenmengen. Das Programm kann nach verschiedenen Suchbegriffen jeden Datensatz in kürzester Zeit anzeigen, egal wo dieser Satz innerhalb der Datei steht. Jeder Datensatz kann beliebig verändert werden, ohne dabei die Datei neu organisieren zu müssen.

Dieses ist nur mit einem Diskettenlaufwerk möglich.

WORDSTAR: WORDSTAR ist ein sehr komfortables Textverarbeitungsprogramm. Es können Texte beliebiger Länge erfasst werden. Ist der Arbeitsspeicher voll, wird der Text auf der Diskette abgelegt. . Wenn ich nun eine Textstelle vom Anfang suche, wird der bereits gespeicherte Text automatisch wieder geladen.

Diese Liste von Beispielen ließe sich nun beliebig erweitern. Wichtig ist für uns nur:

Alle professionelle Programme arbeiten

grundsätzlich mit einer Diskette.

Betrachten wir uns nun einen weiteren Systemvorteil.

Wir können bis zu 4 Diskettenlaufwerke gleichzeitig anschließen, d. h. wir können mit riesigen Datenmengen gleichzeitig arbeiten. Dieser Vorteil wirkt sich bei größeren Anwendungen sehr positiv aus.

Da wir aber meistens mit einem Laufwerk anfangen, betrachten wir uns nun den für uns wichtigsten Vorteil:

Wenn das erste Laufwerk an den Rechner angeschlossen ist, haben Sie die Möglichkeit, sich ein zweite Laufwerk selbst zu schaffen. Der Enterprise bietet hier einen sehr einfachen und effektiven Weg. Sie können im Computer ein Diskettenlaufwerk beliebiger Größe simulieren. Die Größe des internen Laufwerkes ist abhängig von der Größe des RAM - Speichers. Die Speichergröße des internen Laufwerkes wird immer in 16 KB - Blöcken definiert.

Bevor wir nun ein solches Laufwerk definieren, müssen wir uns erst einmal mit der Diskette beschäftigen.

Arbeiten mit der Diskette

Bevor wir mit einer neuen Diskette arbeiten können, müssen wir diese erst vorbereiten. Nehmen Sie nun eine neue Diskette zur Hand. Legen Sie diese in das Laufwerk (Wenn Sie den Anhang über die Diskette schon gelesen haben, wissen Sie ja wie es geht. Ansonsten lesen Sie bitte erst den Anhang).

Als nächstes müssen wir nun unsere Diskette formatieren. Wir wollen unsere Diskette beidseitig und mit doppelter Dichte formatieren. Geben Sie nun bitte ein:

```
:FORMAT      <ENTER>
```

Wenn Sie aus dem BASIC heraus einen Diskettenbefehl eingeben wollen, müßen Sie grundsätzlich mit dem Doppelpunkt arbeiten. Der Computer wird Sie nun fragen:

```
Destroy data on drive A: (Y/N)?
```

Bei Enterprise 128 sieht die Frage so aus:

```
Daten im Laufwerk A: löschen (J/N)?
```

Da wir ja eine neue Diskette verwenden, können dort noch keine Daten vorhanden sein. Wir können also die Frage mit Y oder J beantworten. Sollten Sie einmal eine gebrauchte Diskette verwenden, können Sie den Vorgang durch die Eingabe von N verhindern.

Nachdem wir nun unseren Wunsch nach einer Formatierung bestätigt haben, beginnt unser Diskettenlaufwerk zu arbeiten. Unsere Diskette wird nun beidseitig mit doppelter Dichte formatiert. Nach der Formatierung haben wir Platz für 720 KB Daten und Programme.

Wenn wir nun wissen wollen, was auf unserer Diskette steht, geben wir folgenden Befehl ein:

```
:DIR        <ENTER>
```

Nun wird uns das Inhaltsverzeichnis der Diskette angezeigt. Da auf unserer Diskette noch nichts steht, bekommen wir auch keine Inhaltsangabe. Legen Sie nun Ihre Programmdiskette ein. Wenn Sie nun den Befehl wiederholen, werden alle auf der Diskette vorhandenen Programme angezeigt.

Als nächstes werden wir uns nun ein internes Diskettenlaufwerk definieren. Die Größe unseres Laufwerkes soll 16 KB betragen. Geben Sie nun folgenden Befehl ein:

```
:RAMDISK 1        <ENTER>
```

Damit hätten wir also "2" Diskettenlaufwerke. Wir wollen es gleich ausprobieren. Laden Sie nun das Programm "ADR1" in Ihren Computer. Als nächstes müssen wir dafür sorgen, daß unser Computer nur noch mit dem internen Laufwerk arbeitet. Der Wechsel eines Laufwerkes wird wie folgt durchgeführt:

```
:E:               <ENTER>
```

Ein sehr kurzer Befehl mit weitreichender Wirkung. Wir können zwar mit 4 Laufwerken arbeiten, aber nur ein Laufwerk gilt als Standardlaufwerk. Alle Befehle, die ohne Angabe einer Laufwerksbezeichnung eingegeben werden, verwenden grundsätzlich das angemeldete Standardlaufwerk. Normalerweise ist dies das Laufwerk A.

Das interne Laufwerk hat grundsätzlich die Bezeichnung E. Mit dem Befehl E: haben wir nun das Laufwerk E zum

Standardlaufwerk erklärt. Alle Eingaben werden sich nun auf dieses Laufwerk beziehen.

Wir wollen nun unser Programm auf unser neues Diskettenlaufwerk speichern.

```
SAVE "ADR1"      <ENTER>
```

Unser Computer meldet die Ausführung der Arbeit. Unser "richtiges" Diskettenlaufwerk hat sich aber nicht gemeldet. Um sicher zu gehen, entfernen Sie nun die Diskette aus dem Laufwerk. Als nächstes löschen wir unseren Speicher.

```
NEW              <ENTER>
```

Mit dem Befehl LIST überzeugen wir uns davon, daß der Speicher auch wirklich leer ist. Geben Sie nun

```
:DIR             <ENTER>
```

ein. Im Laufwerk E befindet sich ein Programm. Nun laden wir dieses Programm.

```
LOAD "ADR1"      <ENTER>
```

Sie können das Programm starten. Unser internes Laufwerk laßt sich wie jedes normale Laufwerk verwenden. Sie können mit dem internen Laufwerk arbeiten wie mit jedem "richtigen" Laufwerk.

W I C H T I G ! ! !

Wenn Sie den Computer abschalten werden alle Programme und Daten auf dem internen Laufwerk gelöscht. Es ist eben nur ein Teil des normalen Arbeitsspeichers.

Wenn wir unsere Arbeit beenden, müßen wir alle Programme und Daten auf eine richtige Diskette übertragen. Für diese Arbeit gibt es einen Befehl:

```
:COPY E: A:
```

Dieser Befehl überträgt alle Daten und Programme vom Laufwerk E nach Laufwerk A. Sollten die gleichen Programmnamen schon auf der Diskette A vorhanden sein, werden Sie überschrieben. Wenn Sie diesen Effekt verhindern wollen, verwenden Sie besser eine neu formatierte Diskette.

Wir können aber auch gezielt kopieren. Legen Sie bitte eine neu formatierte Diskette in Laufwerk A ein. Bevor wir ein Programm kopieren, sorgen wir dafür, daß auf dem Laufwerk E mehr als ein Programm steht. Unser Programm steht ja noch im Speicher. Wir speichern es unter einem neuen Namen ab.

```
SAVE "ADRX"        <ENTER>
```

Durch DIR können Sie sich davon überzeugen, daß zwei Programme auf der internen Diskette stehen. Jetzt können wir eines der zwei Programme kopieren.

```
:COPY E:ADRX. A:     <ENTER>
```

Nachdem der Computer fertig ist schalten wir wieder auf unser Laufwerk A um.

```
:A:          <ENTER>
```

Ab sofort wird wieder unser reales Diskettenlaufwerk verwendet. Durch DIR können Sie sich davon überzeugen, daß die Kopie gelungen ist.

Zum Schluß betrachten wir uns noch einen anderen Befehl: Legen Sie bitte die IS-DOS Diskette ein.

```
:ISDOS
```

Mit diesem Befehl schalten Sie um auf das Betriebssystem IS-DOS. Sie haben nun die Sprache BASIC nicht zu Verfügung. Dafür können Sie alle Diskettenbefehle ohne die

lästigen Doppelpunkte ausführen. Wenn Sie wieder im BASIC arbeiten wollen, genügt die Eingabe

BASIC <ENTER>

Soweit unsere Diskettenbefehle. Als nächstes wenden wir uns der praktischen Anwendung zu.

Das Adressenprogramm aus dem Teil "Cassettenrecorder" läßt sich in seiner Grundkonzeption direkt übernehmen. Die Erweiterung der Datei werden wir dann im zweiten Teil des Kurses behandeln. Wenden wir uns nun unserem letzten Thema zu. Wir werden uns auf den nächsten Seiten ein wenig mit dem Drucker beschäftigen.

5. Der Drucker

Um mit einem Computer universell arbeiten zu können, benötigen wir nur noch einen Drucker.

Das Prinzip der Datenverarbeitung beruht auf Eingabe - Verarbeitung - Ausgabe. Bisher haben wir unsere Ausgaben auf Diskette und Bildschirm beschränkt. Eine Diskette läßt sich zwar leicht transportieren, sie ist aber leider ohne Computer nicht zu lesen. Was aber tun, wenn wir Daten oder Briefe versenden wollen. Wenn der Empfänger einen Enterprise Computer besitzt, können wir ihm ja ein Band oder eine Diskette schicken. Was aber, wenn er keinen Computer besitzt.

In diesem Fall benötigen wir einen Drucker. Ein Drucker ist ein sogenanntes Datenendgerät. Ein Drucker dient immer als Ausgabegerät. Eine Dateneingabe in den Computer ist nicht möglich. Der Drucker bringt Daten und Programme in lesbarer Form auf Papier.

Doch genug mit der Vorrede. Wir wollen unseren Drucker arbeiten sehen.

Wenn wir längere Programme entwickeln, ist es sehr schwierig, diese über den Bildschirm zu lesen. Es ist daher sinnvoll, diese Programme auf Papier auszudrucken.

Laden Sie nun das Programm "ADR1" in den Computer. Überzeugen Sie sich vorher noch davon, daß der Drucker auch ordnungsgemäß mit dem Computer Verbunden ist. Wenn der Drucker eingeschaltet ist, müssen drei grüne Lampen aufleuchten (dies gilt für den Enterprise Drucker EP 80+). Jetzt geben wir den folgenden Befehl ein:

LLIST <ENTER>

Wenn alles richtig angeschlossen ist, beginnt der Drucker nun mit seiner Arbeit. Unser Programm wird auf das Papier gebracht. Der Befehl LLIST kann genauso angewendet werden, wie der normale LIST - Befehl, den Sie ja bereits kennengelernt haben.

Als nächstes werden wir unseren Drucker mit einem kleinen Programm näher kennenlernen.

Drucker - Programm

```
 10 CLEAR SCREEN
 20 PRINT AT 1,10:"Drucker - Demonstration"
 30 PRINT AT 2,10:"-----------------------"
 40 PRINT AT 5,1:"Dieses Programm arbeitet mit dem
    Enterprise "
 50 PRINT AT 7,1:"Drucker EP 80+.
 60 PRINT AT 9,1:"Wenn Sie den Drucker eingeschaltet
    haben, "
 70 PRINT AT 11,1:"drücken Sie eine bliebige Taste."
 80 GET A$
 90 IF A$="" THEN GOTO 80
100 LPRINT CHR$(14);"       Druckprogramm"
110 LPRINT:LPRINT:LPRINT
120 LPRINT CHR$(27);CHR$(27);CHR$(1);"Der folgende Text
    wird unterstrichen."
130 LPRINT "Auch diese Zeile wird noch automatisch unter
    strichen."
140 LPRINT CHR$(27);CHR$(45);CHR$(0)
150 LPRINT:LPRINT:LPRINT
160   LPRINT "Der folgende Text wird nicht mehr
      unterstrichen. Der Drucker beherrscht noch eine "
170 LPRINT "ganze Palette von Funktionen"
180 LPRINT:LPRINT:LPRINT
190 LPRINT CHR$(15) "Wir können auch noch mehr Text in
    einer Zeile unterbringen. Obwohl wir die Schrift noch
    gut lesen können"
```

```
200 LPRINT CHR$(18)

210 LPRINT:LPRINT

220 LPRINT "Und hier noch einmal der Zeichensatz:"

230 LPRINT:LPRINT:LPRINT

240 FOR X=33 TO 254

250 LPRINT CHR$(X);

260 NEXT X

270 CLEAR SCREEN

280 PRINT AT 12,10:"Programmende"

290 END
```

Soweit unser kleines Druckerprogramm.

Sie haben eben eine Reihe von Anweisungen und Funktionen neu kennengelernt. Wir wollen diese an dieser Stelle untersuchen.

LPRINT

Die Anweisung LPRINT wird genauso eingesetzt wir die PRINT - Anweisung. Nur das die Ausgabe hier nicht auf den Bildschirm erfolgt.

Der Zusatz CHR$(x) hat eine besondere Funktion. Zum einen werden dem Drucker Steueranweisungen übermittelt. Zum anderen drucken wir aber auch den Zeichensatz des Druckers mit dieser Funktion aus. Wir werden uns dies gleich näher betrachten.

CHR$(x) liefert das Zeichen oder den Wert, dessen ASCII-Code-Nummer x ist. Unser normaler Zeichensatz beginnt bei 33 und endet bei 254. Von 0 bis 32 befinden sich überwiegend Steuerzeichen.

Druckersteuerzeichen

CHR$(14) = Veranlasst den Drucker, in Breitschrift zu drucken. In diesem Format werden 40 Zeichen pro Zeile gedruckt.
Diese Funktion hat nur für die Zeile Gültigkeit, in der sie aufgerufen wird. Jede neue LPRINT - Anweisung druckt in der normalen Schrift weiter.

CHR$(15) = Veranlasst den Drucker, in komprimierter Schrift zu drucken. In diesem Modus können Sie 142 Zeichen pro Zeile drucken. Dieser Modus wird solange beibehalten, bis er wieder ausgeschaltet wird.

CHR$(18) = Durch diese Funktion wird der Drucker wieder auf Normalschrift umgeschaltet. Wenn Sie den Drucker auf komprimierter Schrift eingestellt haben, müssen Sie mit CHR$(18) wieder auf normale Schrift umstellen.

CHR$(27) = ist die Escape - Funktion. Der Drucker wird veranlasst, einen Steuerbefehl zu empfangen.

CHR$(45);
CHR$(1) = Veranlasst den Drucker, den folgenden Text zu unterstreichen. Dieser Steuerbefehl wird solange aufrecht erhalten, bis er widerrufen wird.

CHR$(45);
CHR$(0) = Veranlasst den Drucker, die Unterstreichung wieder aufzuheben.

CHR$(12) = Veranlasst den Drucker, einen Formularvorschub durchzuführen.

Soweit die Druckersteuerung. Wenn Sie einen großen Teil des Leistungsumfanges kennenlernen wollen, finden Sie auf dem Datenträger ein Programm mit dem Namen:

DRUCKER

Dieses Programm können Sie aber nur starten, wenn Sie einen Drucker angeschlossen haben. Für den Anfang soll dies ausreichen. Sie werden den Drucker überwiegend für Ihre Programmlisten benötigen.

Abschließend noch eine Aufgabe:

Integrieren Sie in Ihr Adressenprogramm eine Druckausgabe, mit der Sie eine Adressenliste ausdrucken können. Einen Lösungsvorschlag finden Sie auf dem Datenträger unter der Bezeichnung "ADR4".

A n h a n g:

Auf den nächsten Seiten finden Sie Lösungen Tabellen und andere Informationen.

Lesen Sie die Lösungen bitte erst, nachdem Sie die Aufgaben gelöst haben.

Schlußwort

Wenn Sie den ersten Teil bis hierher sorgfältig durchgearbeitet haben, verfügen Sie über ein solides Grundwissen. In den nächsten beiden Teilen geht es dann richtig zur Sache. Beide Teile werden vom Umfang wesentlich größer sein.

Außerdem wird das hier erarbeitete Grundwissen vorausgesetzt. Sie lernen im 2. Teil unter anderem folgende Dinge:

Neuaufbau unseres Adressenprogrammes mit den neuen Funktionen:

- Sortieren
- löschen
- ändern

Weiterhin werden wir verschiedene neue Anweisungen kennenlernen.

Wir werden uns mit der Grafik beschäftigen und diese sinnvoll in unsere Programme einbauen.

Sie werden auf jeden Fall eine Menge neuer Dinge lernen. Ich würde mich freuen wenn Sie dann beim 2. Teil auch wieder dabei sind.

Anzahl der Binärstellen	Maximaler Binärwert	Dezimales Äquivalent
1	1	1
2	11	3
3	111	7
4	1111	15
5	11111	31
6	111111	63
7	1111111	127
8	11111111	255
9	111111111	511
10	1111111111	1023
11	11111111111	2047
12	111111111111	4095
13	1111111111111	8191
14	11111111111111	16383
15	111111111111111	32767
16	1111111111111111	65535

Hexadezimal	Dezimal	Oktal	Binär
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111
10	16	20	10000

L ö s u n g e n

Ü 1.3.1

```
  1011 0000          0001 1011
+ 0001 1011        + 0111 1111
-----------        -----------

= 1100 1011        = 1001 1010
===========        ===========
```

```
  1100 0111          0010 1010
+ 0010 0111        + 1010 0111
-----------        -----------

= 1110 1110        = 1101 0001
===========        ===========
```

```
  0110 0000          0000 1111
+ 1001 1111        + 1111 0000
-----------        -----------

= 1111 1111        = 1111 1111
===========        ===========
```

Ü. 1.3.2

dezimal	25	=	dual	0001 1001
"	127	=	"	0111 1111
"	65	=	"	0100 0001

ú. 1.3.2

dezimal	139	=	dual	1000 1011
"	228	=	dual	1110 0100
"	254	=	"	1111 1110
"	89	=	"	0101 1001
"	111	=	"	0110 1111

ú. 1.3.3

23	0001 0111	44	0010 1100
+ 46	+ 0010 1110	+ 6	+ 0000 0110
= 69	= 0100 0101	= 52	= 0011 0010

128	1000 0000	71	0100 0111
+ 25	+ 0001 1001	+ 99	+ 0110 0011
= 153	= 1001 1001	= 170	= 1010 1010

99	0110 0011	144	1001 0000
+ 13	+ 0000 1101	+ 23	+ 0001 0111
= 112	= 0111 0000	= 167	= 1010 0111

ü 1.3.3

158	1001 1110	36	0010 0100
+ 77	+ 0100 1101	+ 201	+ 1100 1001
= 235	= 1110 1011	= 237	= 1110 1101

111	0110 1111	199	1100 0111
+ 132	+ 1000 0100	+ 22	+ 0001 0110
= 243	= 1111 0011	= 221	= 1101 1101

ü. 1.3.4

dezimal	8	=	Hex	8
"	23	=	"	17
"	75	=	"	4B
"	112	=	"	70
"	180	=	"	B4
"	224	=	"	E0
"	254	=	"	FE

Ü. 1.3.5

1010 0111	=	$ A7
1110 1011	=	$ EB
0101 1010	=	$ 5A
1101 1111	=	$ DF
1100 0011	=	$ C3
1011 0111	=	$ B7

Ü. 1.3.6

```
  $ 23        $ 0F        $ A0        $ B7
+ $ 17      + $ E9      + $ 11      + $ 05
------      ------      ------      ------

= $ 3A      = $ F8      = $ B1      = $ BC
======      ======      ======      ======
```

```
  $ AA
+ $ 11
------

= $ BB
======
```

Ü. 1.3.7

```
  65      $ 41       79      $ 4F       83     $ 53
+ 93    + $ 5D    + 120      $ 78     + 22   + $ 16
----    ------    -----      ----     ----   ------

=158    = $ 9E    = 199    = $ C7     =105   = $ 69
====    ======    =====    ======     ====   ======
```

ü. 1.3.7

159	$ 9F	128	$ 60	16	$ 10
+ 83	$ 53	+ 127	+ $ 7F	+ 15	+ $ 0F
= 242	$ F2	= 255	= $ FF	= 31	= $ 1F

Vorbereitung einer Diskette

Damit Sie mit einer Diskette arbeiten, können, muß diese vor der ersten Verwendung im Computer vorbereitet werden.

Die Diskette ist unterteilt in Spuren und Sektoren. Eine Spur ist reserviert für das Inhaltsverzeichnis der Diskette. Da aber alle Formate einer Diskette unterschiedlich sind, werden sie vom Hersteller unformatiert geliefert.

Wir wollen nun auf unserer Diskette diese Spuren und Sektoren anlegen. In unserem Fall geschieht das beidseitig. Die Kapazität unserer Diskette beträgt 1 MB.

Nachdem wir sie formatiert haben, ist die Kapazität nur noch 720 KB.

Schieben Sie eine Diskette in Ihr Laufwerk. Die runde Antriebsscheibe aus Metall muß dabei nach unten weisen.

Nun geben Sie folgenden Befehl ein:

```
:FORMAT      <ENTER>
```

Der Computer stellt nun folgende Frage:

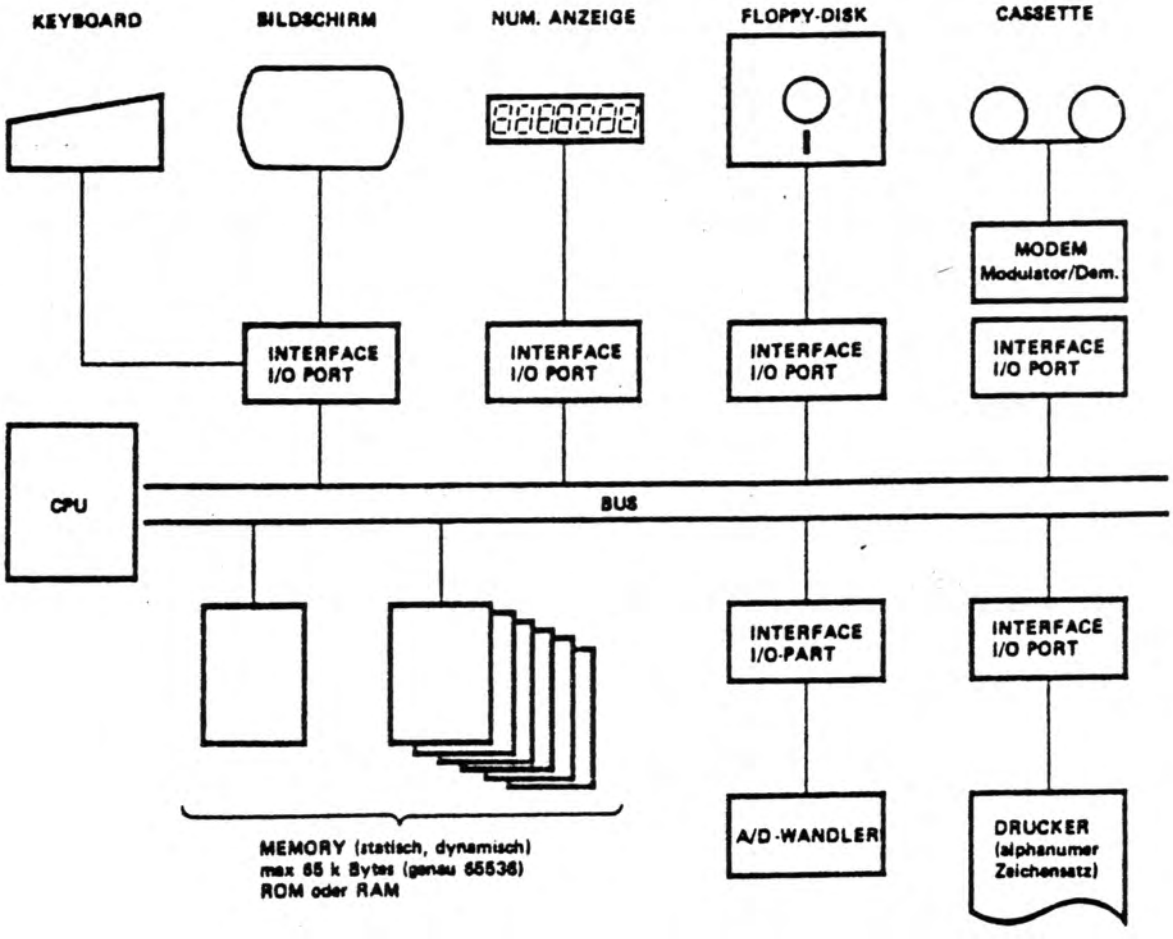
```
Destroy data on drive A: (Y/N)
```

Der Enterprise 128 fragt:

```
Daten im Laufwerk A. löschen (J/N)
```

Antworten Sie mit Y oder J je nach Modell.

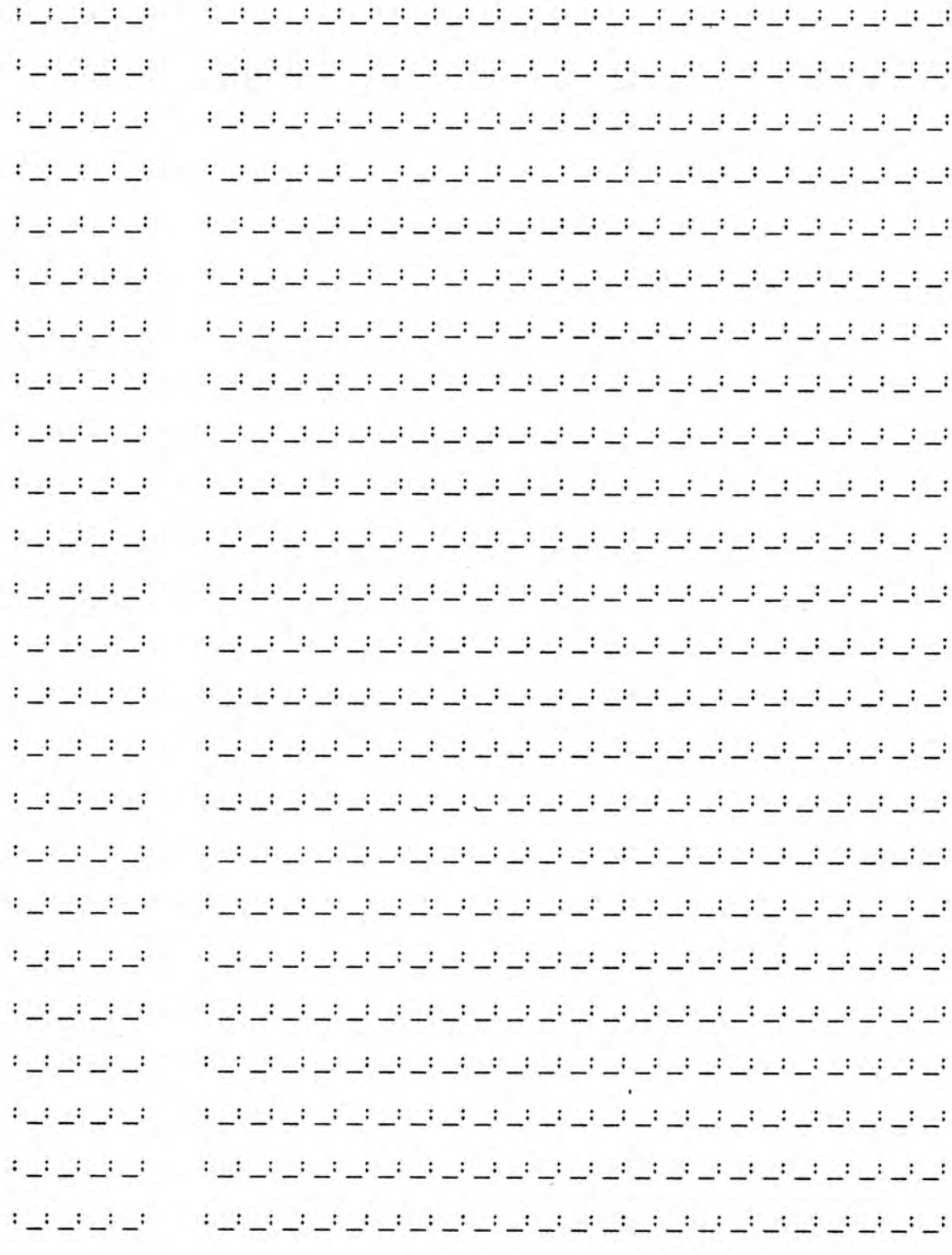
Damit haben wir unsere erste Diskette zur Arbeit mit unserem Computer vorbereitet.



Systemaufbau eines Mikrocomputers

Programm

```
10 PRINT AT 1,10:"Lotto - Tip  der Woche"
20 PRINT AT 2,10:"----------------------"
30 FOR X=1 TO 6
40 LET A(X) = RND(50)
50 NEXT X
60 FOR Y=1 TO 6
70 PRINT A(Y),
80 NEXT Y
90 END
```



Programm: Blatt: