VOSS

DAS BASIC-TRAININGSBUCH ZU ATARI® 600XL/800XL

ABBUC EDITION 2022 (p) GBXL

Bearbeitungsstand: 6. Juni 2Ø22

Erstellt mit Open Office 4.1.12 unter Linux Mint 2Ø.3.

Danksagung

Mit Ablauf des Jahres 2Ø13 wurden der Zeitschriften- und der Buchverlag von DATA Becker geschlossen; zum 31. März 2Ø14 wurde der gesamte Geschäftsbetrieb aufgegeben. Es gibt keinen Rechtsnachfolger.

Der Autor *Prof. Dr. Werner Voß* gab auf Bitte des ABBUC e.V. im März 2Ø22 die Zustimmung zur digitalen Aufarbeitung und Veröffentlichung.

Dafür recht herzlichen Dank.

Anmerkung

Cover und Layout des Buches entsprechen weitestgehend dem des Originaldrucks von 1984. Durch notwendige Korrekturen und Anpassungen konnte das Werk nicht immer zeilengleich digitalisiert werden, aber die Seiteninhalte wurden gewahrt und erlauben so einen Abgleich zum originalen Buch aus 1984.

Das Buch erschien ursprünglich bei DATA Becker:

ISBN 3-89Ø11-Ø57-6

Copyright (C) 1984 DATA BECKER GmbH
Merowingerstr. 3Ø
4ØØØ Düsseldorf

Re-Edit by GBXL for ABBUC e. V. (p) 2Ø22

Wichtiger Hinweis!

Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technische Angaben und Programme in diesem Buch wurden von den Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, dass weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

Gliederung

	Seite
Einleitung	1
Kap. 1 : Grundlagen	5
1. Vorbemerkung	5
2. Grundbegriffe	6
3. Zur Funktionsweise von Rechner-systemen	12
4. Die Kommunikation mit dem Rechner	15
5. Gerätekonfigurationen	18
6. Das Arbeiten am Kleinrechner .	22
7. EXKURS : Zahlensysteme und Codes	24
Übungsaufgaben	30
Kap. 2 : Der Direktmodus	31
1. Begriffsklärungen	31
2. Die Tastatur	33
3. Das Arbeiten im Direktmodus ..	38
Übungsaufgaben	44
Kap. 3 : Die Vorbereitung der Programmierung	45
1. Vorbemerkung	47
2. Problemanalyse	33
3. BASIC	51
Übungsaufgaben	52

Seite

Kap. 4 : Einfaches Rechnen mit dem ATARI .. 45

1. Vorbemerkung 55
2. Erste BASIC-Anweisungen : Ein- und Ausgabe von Informationen 33

ERSTES TRAINING 67

o Kreisberechnungen 68
o Gewinnberechnungen 73
o Quadratzahlen 77

3. Das Korrigieren von Sätzen ... 8Ø

Übungsaufgaben 83

Kap. 5 : Größere Rechenprogramme 85

1. Vorbemerkung 85
2. Weitere BASIC-Anweisungen : Funktionen,Verzweigungen, Schleifen 86

ZWEITES TRAINING 95

o Quadrate und Wurzeln 95
o Großes Einmaleins 1Ø1
o Hypothekentilgung 1Ø5
o Primzahlenprüfung 111
o Mittelwertberechnung 116

Übungsaufgaben 12Ø

Seite

Kap. 6 : Datenverarbeitung mit BASIC-Programmen 123

1. Vorbemerkung 123
2. Zusätzliche BASIC-Anweisungen : Das Lesen von Daten und die Nutzung von Unterprogrammen .. 124

DRITTES TRAINING 133

o Sortieren von Zahlen 133
o Mittelwerte 141
o Häufigkeitsverteilung 146
o Lotto 154

Übungsaufgaben 159

Kap. 7 : Grundelemente der Textverarbeitung 161

1. Vorbemerkung 161
2. BASIC-Sprachelemente zur Behandlung von Strings 162

VIERTES TRAINING 166

o Übersetzung 166
o Aufsuchen eines Buchstabens 174
o Vokabeltest 178

Übungsaufgaben 19Ø

Seite

Kap. 8 : Die Benutzung der Graphik-Symbole 191

1. Vorbemerkung 191
2. BASIC-Elemente 192

FÜNFTES TRAINING 198

o Sinuslinie 198
o Ballspiel 2Ø4

EXKURS : Farbgebung von Bildschirm-rahmen und Bildschirm-hintergrund 21Ø

Übungsaufgaben 213

Kap. 9 : Blockgraphik 215

1. BASIC-Anweisungen 215

SECHSTES TRAINING 222

o Zufallsmuster 222
o Zufallslinie 228
o Beschäftigtenstatistik ... 233

Übungsaufgaben 24Ø

Kap.1Ø : Hochauflösende Graphik 241

1. Vorbemerkung 241
2. BASIC-Anweisungen 243

Seite

SIEBENTES TRAINING 245

o Gerade 245
o Kreis 251
o Sinuslinie 258

Übungsaufgaben 263

Kap.11 : Ergänzungen 265

1. Generelles 265
2. BASIC-Statements 269

A N H Ä N G E Übungsaufgaben 275

Anhang 1 : Übungsaufgaben 276

zu Kapitel 1 : 279
zu Kapitel 2 : 283
zu Kapitel 3 : 286
zu Kapitel 4 : 292
zu Kapitel 5 : 301
zu Kapitel 6 : 313
zu Kapitel 7 : 326
zu Kapitel 8 : 336
zu Kapitel 9 : 346
zu Kapitel 10 : 351

Anhang 2 : Fehlermeldungen 359

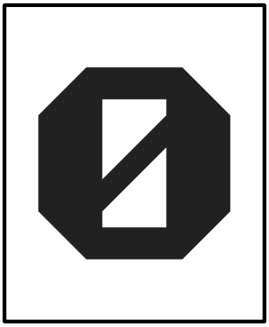
Seite

Anhang 3 : Wichtige EDV-Stichworte
Englisch-Deutsch 359

Anhang 4 : Kommandos 369

Anhang 5 : BASIC-Statements 370

Anhang 6 : Stichwortverzeichnis 373



Einleitung

Computer - wie zum Beispiel der ATARI - sind nicht nur zum Spielen da. So faszinierend fertige Programme auch sein mögen, noch aufregender und interessanter ist es, selbst zu programmieren.

Wer selbst Probleme so aufbereiten kann, daß das in den Rechner eingegebene Programm solche Probleme dann in Sekundenschnelle und fehlerfrei löst, der hat einen wichtigen Schritt in die Richtung der Nutzung neuer Technologien vollzogen. Er erkennt, daß Computer nützliche Arbeit leisten können, d.h. daß sie uns von der Last bestimmter mühsamer "Kopfarbeiten" befreien können - und dies ist auch der wesentliche Verwendungszweck der sich so rasch verbreitenden Computer; dies macht die zunehmende Bedeutung dieser Geräte aus.

Allerdings ist es erforderlich, daß man den Umgang mit einem Computer erst erlernt, wenn man alle seine Fähigkeiten wirklich ausnutzen möchte. Sowohl der Überblick über die generelle Funktionsweise eines Rechners wie auch seine Programmierung machen Kenntnisse erforderlich, die man erlernen und üben muß.

Dieses Buch dient nun in erster Linie dazu, dieses notwendige Üben zu erleichtern. Es ist ein Trainingsbuch, d.h. der Einsatz des ATARI-Rechners soll mit den hier vorgeführten Programmbeispielen trainiert werden. Denn wie so oft im Leben, gilt auch hier:

Übung macht den Meister !

Durch Üben und Training wird der Computerbenutzer entdecken, was ein solches Gerät alles vermag - und dies gilt nicht nur für den ATARI-Rechner, sondern ganz allgemein.

Allerdings kann der ATARI immer noch nicht ganz seine Herkunft von den "Spielcomputern" früherer Jahre verleugnen, aber wir betonen noch einmal : Zum Spielen allein ist er viel zu schade !

Diese Herkunft des ATARI erkennt man vor allem daran, daß derjenige ATARI-Benutzer, der selbst programmieren möchte, von den schriftlichen Unterlagen, die dem Rechner beigefügt sind, quasi im Stich gelassen wird: Die Möglichkei-

ten der BASIC-Programmierung sind so außerordentlich kurz und knapp dargestellt, daß insbesondere der Anfänger wahrscheinlich kaum etwas damit anfangen kann.

Auch Hinweise darauf, was mit dem ATARI vielleicht nicht geht, die Grenzen der Programmierbarkeit also, die vor allem für diejenigen Benutzer wichtig sind, die schon einige Erfahrungen mit anderen Rechnern haben, fehlen in der Regel.

So stellt der Benutzer dann verdutzt fest, daß bestimmte Programmierschritte, die er von anderen Rechnern als sehr nützlich schon kennt, beim ATARI so nicht funktionieren.

Deshalb ist es ganz besonders wichtig, diejenigen BASIC-Möglichkeiten, die der ATARI bietet, sorgfältig einzuüben und zu trainieren.

> Dieses Trainingsbuch verfolgt also insbesondere den Zweck, nicht einfach nur BASIC-Sprachelemente vorzustellen, sondern zm zeigen, in welcher Weise sie speziell beim ATARI verwendet werden.

Diesem Trainingsgesichtspunkt entspricht es auch, daß jedes Kapitel mit einer Reihe von Übungsaufgaben beschlossen wird, deren Lösungen sich im Anhang finden. Natürlich sollte der Leser erst dann im Lösungsanhang nachblättern, wenn er mit genügend Einsatz eigene Problemlösungen oder Antworten versucht hat oder diese nur kontrollieren will.

Einleitung

1

Kapitel 1 : Grundlagen

Abschnitt 1 : Vorbemerkung

Wer mit Rechnern umgehen möchte, steht nicht nur vor der Notwendigkeit, eine Programmiersprache zu erlernen, sondern er sollte auch über die wichtigsten Grundbegriffe aus dem generellen Bereich der Datenverarbeitung und der Computertechnologie Bescheid wissen.

Deshalb sollen in diesem Kapitel einige wenige der wichtigsten Grundbegriffe knapp und stichwortartig skizziert werden. Es wird auch kurz über die generelle Funktionsweise von Rechnern und über die unterschiedlichen Geräte eines Rechnersystems gesprochen.

Abschnitt 2 : Grundbegriffe

Computer :

Das Wort "Computer" bedeutet wörtlich übersetzt "Rechner". Ein moderner Computer kann aber mehr als bloß rechnen. Er kann beispielsweise auch mit Buchstaben, mit Worten und Texten umgehen; er kann graphische Darstellungen erzeugen, er kann Musik produzieren, er kann Steuerungsaufgaben übernehmen u.ä. Dies bedeutet also, daß die Rechner, die heute schon für ein paar Hundert Mark erworben werden können, nicht nur die Rechenkapazitäten bereitstellen wie z.B. die Großrechner der sechziger Jahre, die damals unerschwinglich teuer waren, sondern sie sind viel universeller einsetzbar.

Daten :

Unter "Daten" kann man im weitesten Wortsinn "Informationen" verstehen. Im einzelnen können dies zum Beispiel sein :

- Ziffern und Zahlen
- Werte
- Buchstaben
- Symbole (Sonderzeichen)
- Worte und Texte

Datenverarbeitung :

Unter "Datenverarbeitung" versteht man alle Prozeduren (häufig rechnerischer Art), Daten zu erfassen, zu speichern, auszuwerten oder zu analysieren und Ergebnisse auszugeben.

Führt man Datenverarbeitung im obigen Sinn nicht mehr per Hand, sondern mit Hilfe von Computern durch, dann ist es sinnvoll, auch die folgenden Begriffe zu kennen :

Bit :

Unter einem "Bit" (binary information digit = binäre Informationseinheit) versteht man die kleinste Informationseinheit. Von der "kleinsten" Informationseinheit spricht man deshalb, weil in einem Bit nur zwei Informationsinhalte vorstellbar sind. Aus der Sicht des Computers her gesprochen, heißt dies :

> Ein Bit kann nur einen von maximal zwei verschiedenen Informationsinhalten speichern.

Diese beiden Informationsinhalte werden üblicherweise mit Ø und 1 bezeichnet.

Byte :

Ein Byte ist eine Zusammenfassung einer Folge von Bits. Üblicherweise bilden acht Bits ein Byte.

Ein Byte ist somit in der Lage, eine Folge von acht Nullen und/oder Einsen zu speichern.

Symbol :

Wir unterscheiden in der Datenverarbeitung drei Gruppen von Symbolen :

- numerische Symbole = Ziffern
- alphabetische Symbole = Buchstaben
- sonstige Symbole = Sonderzeichen

Jedes dieser Symbole wird im Rechner dargestellt als Folge von acht Nullen und/oder Einsen, d.h. ein Byte (s.o.) ist in der Lage, ein Symbol aufzunehmen.

Feld :

Eine Folge von Symbolen (z. B. Worte oder Zahlen) bilden ein Feld.

Beispielsweise besteht das Wort PRINT (= drucke) aus fünf Symbolen, belegt mithin fünf Byte im Rechner, und da diese fünf Symbole zusammengehören, bilden sie ein Feld.

Entsprechendes gilt z. B. für die Zahl 178 : Sie wird in einem dreistelligen Feld gespeichert (man sieht an diesem Beispiel, daß es auch einstellige Felder geben kann).

Variable :

Eine Variable ist eine veränderliche Größe oder - präziser gesprochen - ein Name, unter dem unterschiedliche Werte (s.u.) angesprochen werden können.

Beispielsweise stehen hinter dem Variablennamen "Körpergröße" die Werte 178, 185, 167 usw., hinter dem Variablennamen "Kinderzahl" die Werte Ø, 1, 2 usw. und hinter dem Variablennamen "Wohnort" die Werte "Dortmund", "Berlin", "Hamburg" usw.

Wert :

Wie aus der vorangegangenen Begriffsklärung schon entnommen werden kann, verstehen wir unter dem Begriff "Wert" eine bestimmte "Ausprägung" einer gegebenen Variablen. Es muß sich dabei nicht notwendigerweise um Zahlen handeln, wie man zunächst vielleicht vermuten könnte. Das obige Beispiel der Variablen "Wohnort" zeigt, daß als Werte auch Worte oder Texte auftreten können.

Die Werte belegen jeweils ein Feld im Rechner (s.o.). Man spricht in diesem Zusammenhang auch von einer "Speicherstelle".

String :

Einen Text, der z.B. als Variablenausprägung auftreten kann (s.o.), nennt man "String". Allgemein gesprochen handelt es sich um eine Zeichenkette.

Array :

Die Gesamtheit aller Ausprägungen einer Variablen (Zahlen oder Worte) nennt man "Array".

Datensatz :

Mehrere zusammengehörige Felder nennt man einen Datensatz.

Beispielsweise bilden alle vorhandenen Angaben über eine Person (Name, Geschlecht, Alter, Wohnort usw.) einen Datensatz. In diesem Beispiel wäre ein Datensatz also eine Folge zusammengehöriger Zahlen und/oder Worte.

Aber auch eine Folge nur von Worten kann als Datensatz bezeichnet werden, so zum Beispiel eine Programmzeile in einem BASIC-Programm.

Der Datensatz wird auch als "Record", "Case" oder "Set" bezeichnet.

Datei :

Mehrere zusammengehörige Datensätze bilden eine Datei. Beispielsweise erhalten wir eine Datei, wenn wir die Datensätze aller Beschäftigten eines Unternehmens zusammenstellen. Genauso würden wir eine Datei erhalten, wenn wir alle Programmzeilen (alle Programmanweisungen) eines BASIC-Programms insgesamt betrachten.

Eine Datei wird auch als "File" bezeichnet.

Abschnitt 3 : Zur Funktionsweise von Rechnersystemen

Es wurde schon erläutert, was man unter "Datenverarbeitung" versteht. Aus dieser Erläuterung geht hervor, daß eine Datenverarbeitungsanlage, also ein Rechnersystem, die folgenden Tätigkeiten (Funktionen) ausführen muß :

- Informationsaufnahme	E	Eingabe
- Informationsverarbeitung	V	Verarbeitung
- Ausgabe von Ergebnissen	A	Ausgabe

An einem sehr einfachen Beispiel soll dieses sog. EVA-Prinzip illustriert werden :

Eingegeben werden sollen die Zahlen 3 und 4 (1. Schritt) und aus beiden soll der Durchschnitt berechnet werden (2. Schritt) - wir wissen schon auch ohne Rechner, daß dabei der Wert 3.5 als Ergebnis herauskommen muß - und dieses Ergebnis soll schließlich auch ausgegeben werden (3. Schritt).

Diese simple Aufgabe, die wir notfalls im Kopf oder unter Benutzung eines Taschenrechners lösen können (wobei allerdings ein Taschenrechner im Grunde schon viele

Funktionselemente eines Computers aufweist), macht beim Rechnereinsatz eine Reihe von Überlegungen erforderlich, die dann in entsprechender Weise auch für beliebig kompliziertere Probleme gelten :

Wenn wir dem Rechner die Input-Daten (die Zahlen 3 und 4) übergeben, so muß dieser mehreres leisten :

- er muß die Informationen aufnehmen,
- er muß die Symbole "3" und "4" als die Zahlenwerte 3 und 4 erkennen.

Auch die notwendige Informationsspeicherung macht die Erledigung bestimmter organisatorischer Aufgaben erforderlich :

- der Rechner muß Speicherplätze bereitstellen
- den Speicherplätzen müssen Namen zugeteilt werden
- es muß Sorge dafür getragen werden, daß diese Speicherplätze bei Bedarf auch wieder gefunden werden, d.h. der Rechner muß ich diese Speicherplätze "merken".

Ähnliche organisatorische Aufgaben, auf die im einzelnen hier nicht eingegangen werden soll, weil sie bei modernen Rechnern alle quasi automatisch erledigt werden, sind auch bei den folgenden Arbeitsschritten zu erledigen.

Beim Rechnen geht es beispielsweise um das Erkennen des Pluszeichens, um die Ausführung der Rechnung ("Heranholen" der gespeicherten Werte und additives "Zusammenfügen" usw.); bei der Ergebnisausgabe schließlich sind ebenfalls organisatorische Aufgaben zu erledigen, die denen bei der Eingabe gleichen.

Alle diese Aufgaben werden im allgemeinen vom Rechner selbst erledigt, ohne daß wir als Benutzer uns darüber allzu viele Gedanken machen müßten, Verantwortlich für diese organisatorische Aufgaben ist das Betriebssystem des Rechners.

Weil im Vordergrund dieses Buchs die Frage steht, wie mit Hilfe der Programmiersprache BASIC der ATARI-Rechner eingesetzt werden kann und nicht die Frage, wie er im Detail funktioniert, sollen an dieser Stelle diese einführenden Bemerkungen genügen.

Abschnitt 4 : Die Kommunikation mit dem Rechner

Der Rechner muß von uns Informationen erhalten, damit ein Datenverarbeitungsprozeß in Gang kommen kann. Er gibt seinerseits Informationen zurück, beispielsweise errechnete Ergebnisse.

Diejenigen Informationen, die wir ihm geben müssen, lassen sich in drei Informationsgruppen einteilen :

1. Zu verarbeitende Daten
2. Programmanweisungen
   (Arbeitsschritte, die notwendig sind, um die eingegebenen Daten den gewünschten Auswertungsprozeduren zu unterziehen)
3. Kommandos
   (Mitteilungen und Anweisungen an das Betriebssystem des Rechners)

An dem einfachen Beispiel aus Abschnitt 1.3 und unter Vorgriff auf die vielleicht jetzt noch nicht ausreichenden BASIC-Kenntnisse des Lesers läßt sich mit einem einfachen BASIC-Programm zeigen, was es mit dieser Unterteilung auf sich hat :

Beispiel :

Programm zur Berechnung des Durchschnitts aus zwei eingegebenen Zahlen :

```
10 LET A=3
20 LET B=4
30 LET C=(A+B)/2
40 PRINT C
50 END
```

Daten

Anweisungen

```
RUN
```

Kommando

In den Zeilen 1Ø und 2Ø der Programmanweisungen werden dem Rechner Input-Informationen (Daten, nämlich zwei Zahlen) mitgeteilt (Informationsgruppe 1).

Die Zeilen 1Ø bis 5Ø sind Programmanweisungen (Informationsgruppe 2; in den Zeilen 1Ø und 2Ø überschneiden sich in diesem einfachen Beispiel die Informationsgruppen 1 und 2).

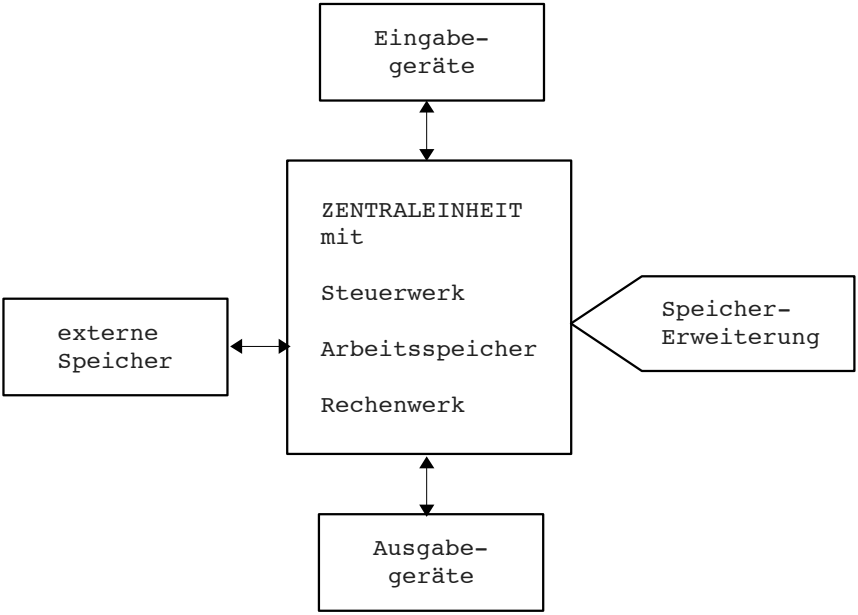
Der Befehl RUN ist ein Kommando (Informationsgruppe 3), also keine Programmanweisung mehr und damit eigentlich kein Bestandteil der Programmiersprache BASIC mehr; allerdings werden wir uns im folgenden auch mit den wichtigsten dieser Kommandos beschäftigen.

Dieses Kommando RUN richtet sich an das Betriebssystem des Rechners und fordert ihn auf, das eingegebene Programm

nun auch abzuarbeiten. Ist diese Abarbeitung erfolgt, so liefert der Rechner als Ergebnis den Wert 3.5.

Abschnitt 5 : Konfiguration

Jede Rechenanlage ist im Prinzip so aufgebaut wie es die folgende Skizze verdeutlicht :



Die Informationseingabe bei einer Kleinrechenanlage erfolgt üblicherweise über die Tastatur; die Informationsspeicherung und die Informationsverarbeitung erfolgt in der Zentraleinheit; zur Informationsausgabe dient normalerweise der Bildschirm oder ein Drucker.

Die Speichermöglichkeiten der Zentraleinheit können häufig durch Speichererweiterungen vergrößert werden.

Die Speicherkapazität bemißt man durch die Anzahl der Byte, die gleichzeitig im Rechner gespeichert werden können. Dabei gelten die folgenden Maßeinheiten :

1 Kilobyte (KB oder K) = 2^{10} = 1Ø24 Byte

Moderne Kleinrechner haben Speicherkapazitäten in der Zentraleinheit von 16 bis 128 Kb (und auch noch darüber hinaus), so daß hier teilweise Geräte zur Verfügung stehen, die mit den Großrechenanlagen der frühen siebziger Jahre durchaus vergleichbar sind.

Man kann sich den internen Speicher eines Rechners auch anders aufgebaut denken als dies in der obigen Skizze zum Ausdruck kam, nämlich in die folgenden beiden Bereiche :

- ROM-Speicher
- RAM-Speicher

ROM-Speicher sind solche Speicherbereiche, aus denen nur gelesen werden kann :

ROM = Read Only Memory
(Speicher nur zum Lesen)

In diesen Speicherbereichen finden sich also Informationen, die uns zur Verfügung stehen (dies ist gemeint, wenn man sagt, sie können gelesen werden). Diese Speicherbe

reiche können aber nicht beschrieben werden (d.h. es kann dort nichts hineingeschrieben werden, es kann in ihnen nichts gespeichert werden). Dies bedeutet, daß diese Speicherbereiche gegen Überschreibungen oder Löschen geschützt sind.

Der Inhalt solcher ROM-Speicher bleibt übrigens auch erhalten, wenn der Rechner ausgeschaltet wird, was beim RAM-Speicher gerade nicht der Fall ist : Sein Inhalt geht beim Ausschalten des Rechners verloren.

Dieser Schutz für den ROM-Speicher ist deshalb so wichtig, weil sich dort Informationen befinden, die der Rechner immer bei allen Datenverarbeitungsprozessen zur Verfügung haben muß : Beispielsweise befinden sich dort die Organisationsprogramme des Betriebssystems, Übersetzungscodes zum Erkennen der von uns benutzten Symbole u. ä., Informationen also, die der Rechner immer braucht und die vom Benutzer nicht überschrieben werden dürfen - deshalb sind sie geschützt.

Der RAM-Speicher hingegen kann gelesen aber auch beschrieben werden :

RAM = Random Access Memory
(Speicher mit Zufallszugriffsmöglichkeit)

Als "Eselsbrücke" kann bedeutet man sich auch merken, daß RAM bedeutet :

Read And Write Memory
(Speicher zum Lesen und Schreiben)

Dieser RAM-Speicherbereich ist also derjenige Speicherbereich, mit den wir bei der Datenverarbeitung direkt in Kontakt stehen, mit dem wir direkt arbeiten können. Deshalb wird die Speicherkapazität eines Rechners auch häufig nur darauf bezogen.

Vom Speicher der Zentraleinheit zu unterscheiden sind die sog. externen Speicher, die man auch periphere Speicher nennt. Sie dienen dazu, Datenbestände und/oder Programme, die im Moment nicht benötigt werden, gleichwohl aber aufbewahrt werden sollen, zu speichern.

Bei Kleinrechenanlagen sind die wichtigsten derartigen externen Speicher Magnetbandkassetten und Disketten.

Abschnitt 6 : Das Arbeiten am Kleinrechner

Einen Kleinrechner einzusetzen und alle seine Möglichkeiten weitestgehend auszunutzen, macht die Erfüllung der folgenden Voraussetzungen erforderlich :

- Erlernen der Programmiersprache BASIC oder einer anderen für Kleinrechner geeigneten Sprache und Üben bzw. Trainieren dieser Sprache
- Grober Überblick über das Betriebssystem des Rechners
- Kenntnisse der wichtigsten Kommandos, um die Möglichkeiten, die das Betriebssystem bietet, so weit als notwendig nutzen zu können.

Die wichtigste Voraussetzung aber, die hier genannt werden muß, hat mit der Rechenanlage, mit den Geräten und mit der Programmierung gar nicht viel zu tun :

> Die Lösung von Problemen mit Hilfe von Rechnern ist nur möglich, wenn diese Lösungen gedanklich korrekt vorbereitet werden, d.h. wenn die Logik eines Problemlösungsweges bekannt ist.

Erst wenn ein Problem gedanklich schon gelöst ist, kann es auch der Rechner mit Hilfe eines korrekten Programms lösen. Erst wenn diese gedanklichen Vorarbeiten geleistet sind, lohnt es sich, den Rechner einzusetzen.

Erst dann lohnt es sich, ihn einzuschalten und darauf zu warten, daß er mit dem Stichwort READY (fertig) anzeigt, daß er in der Lage ist, von uns Informationen (Daten, Programmanweisungen oder Kommandos) zu empfangen.

Diese READY-Meldung wird ergänzt durch ein helles kleines Quadrat am Anfang der folgenden Bildschirmzeile, das man Cursor nennt.

Der Cursor gibt die stelle auf dem Bildschirm an, die als nächste von uns beschrieben werden kann. Während der Informationseingabe wandert er nach rechts und wechselt automatisch zum Anfang der nächsten Bildschirmzeile, wenn er den rechten Bildschirmrand nach 4Ø Anschlägen erreicht hat.

Wenn der Rechner eine bestimmte Arbeit durchführt, kann vom Benutzer nichts eingegeben werden.

Abschnitt 7 : EXKURS : Zahlensysteme und Codes

> Dieser EXKURS stellt Hintergrundinformationen bereit. Derjenige Leser, der nur am Programmiertraining interessiert ist, kann ihn überblättern.

Bei allen Computern, bei Großrechenanlagen wie bei Kleinrechnern, spielt das sog. Binärsystem (oder Dualsystem) eine entscheidende Rolle.

Es handelt sich dabei um ein Zahlensystem, das nur zwei Ziffern aufweist (Ø und 1) und sich dadurch vom uns geläufigen Zahlensystem, dem Dezimalsystem mit seinen 1Ø Ziffern (Ø, 1, 2, ... , 9) unterscheidet.

Demnach kann eine Binärzahl zum Beispiel folgendermaßen aussehen :

11Ø1Ø1

Schauen wir uns zunächst einmal eine beliebige Dezimalzahl an :

2438

Woher wissen wir, daß dies zweitausendvierhundertachtund-

dreißig bedeutet ? Dies wissen wir einfach durch Aufteilen der Zahl in Tausender, Hunderter, Zehner und Einer. Es gilt nämlich :

$$2438 = 2\ T + 4\ H + 3\ Z + 8\ E$$

$$= 2*1ØØØ + 4*1ØØ + 3*1Ø + 8*1$$

$$= 2*1Ø^3 + 4*1Ø^2 + 3*1Ø^1 + 8*1Ø^Ø$$

Anhand dieser Zerlegung erkennen wir :

1. Eine Dezimalzahl läßt sich zerlegen in eine Summe, die so viele Glieder wie die Zahl Ziffern hat (hier also vier Glieder);
2. Jedes Glied ist ein Produkt aus zwei Faktoren;
3. Der jeweils erste Faktor ist die Ziffer der zu zerlegenden Zahl (von links nach rechts);
4. Der jeweils zweite Faktor ist immer eine Potenz zur <u>Basis 1Ø</u> (deshalb <u>Zehner</u>system = Dezimalsystem);
5. Die Hochzahlen (Exponenten) dieser Potenzen sind (bei ganzen Zahlen) die ganzen Zahlen n-1, n-2, ... 2, 1, Ø, wenn n die Anzahl der Ziffern der zu zerlegenden Zahl ist (hier vier Ziffern : Hochzahlen also 3, 2, 1 und Ø).

Diese Überlegung gilt nun genauso für das Binärsystem, allerdings mit <u>einem</u> Unterschied in Satz 4. Dieser lautet jetzt :

4. Der jeweils zweite Faktor ist immer eine Potenz zur Basis 2 (deshalb Zweiersystem oder Binär- oder Dualsystem);

Die Zweierpotenzen sind die folgenden :

...	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
...	256	128	64	32	16	8	4	2	1

Nach diesen Erklärungen können wir auch die Binärzahl 11Ø1Ø1 erkennen. Offenbar muß gelten :

11Ø1Ø1 =

$1*2^5 + 1*2^4 + Ø*2^3 + 1*2^2 + Ø*2^1 + 1*2^0 =$

1*32 + 1*16 + Ø*8 + 1*4 + Ø*2 + 1*1 =

32 + 16 + Ø + 4 + Ø + 1 =

53

Die binäre 11Ø1Ø1 entspricht also der Dezimalzahl 53.

Es wurde schon an anderer stelle erwähnt, daß der Rechner für jedes der von uns benutzten Symbole ein <u>Byte</u> reserviert, also eine Folge von acht <u>Bit</u>. Ein solches Byte kann also mit einer Folge von acht Nullen und/oder Einsen belegt sein.

Da es 256 verschiedene Möglichkeiten gibt, solche Folgen aufzubauen ($256 = 2^8$), können in einem Byte also 256 verschiedene Symbole als Null/Eins-Kombinationen verschlüsselt werden.

Eine solche Kombination ist zum Beispiel die folgende :

Ø	1	Ø	Ø	Ø	Ø	Ø	1

Wie oben dargestellt wurde, kann man ausrechnen, daß dieser Binärzahl die Dezimalzahl 65 entspricht.

Wohlgemerkt, dieses Byte ist nicht mit der Zahl 65 belegt - dazu bräuchte man in jedem Fall ja mindestens zwei Byte, weil jedes unserer Symbole ein Byte verbraucht und die Zahl 65 aus zwei Symbolen, nämlich den Ziffern 6 und 5 besteht.

Dieses Byte enthält eine Null/Eins-Kombination, welche, als Binärzahl betrachtet, der Dezimalzahl 65 entspricht.

Bei den modernen Kleinrechnern ist nun die folgende Regelung getroffen worden : Für jedes der von uns benutzten Symbole wurde eine ganz bestimmte Null/Eins-Kombination festgelegt. Diese Festlegung ist nichts anderes als ein Code, der angibt, in welcher Weise unsere Symbole im Rechner verschlüsselt werden.

Der hier maßgebliche Code ist der sog. ASCII-Code.

(American Standard Code for Information Interchange = amerikanischer Standardcode für den Informationsaustausch)

Dieser Code sieht vor, daß der Großbuchstabe A wie folgt verschlüsselt wird :

A ———> Ø1ØØØØØ1 (= 65 dezimal)

Wir können diese Verschlüsselungsvorschrift überprüfen, indem wir unter Vorgriff auf die noch zu besprechenden BASIC-Elemente einmal in unseren ATARI eintippen :

```
PRINT CHR$(65)
```

Dies heißt : Drucke das Symbol auf den Bildschirm, welches unter der (dezimalen) Codezahl 65 im ASCII-Code vorgesehen ist (PRINT heißt "drucke", bzw. "gib als Ergebnis aus"; CHR$ steht als Abkürzung für character = Symbol).

Wenn wir die obige Anweisung per RETURN-Taste beschließen, antwortet der Rechner mit :

```
A
```

Er bestätigt also, daß hinter der Codezahl 65 (=Ø1ØØØØØ1) sich das Symbol "A" "versteckt".

Auch der umgekehrte Weg ist beschreitbar : Wenn wir z. B. wissen wollen, unter welcher Codezahl der Buchstabe E gespeichert wird, so tippen wir ein :

```
PRINT ASC("E")
```

(ASC steht als Abkürzung für "Verschlüsselung gemäß des ASCII-Codes").

Betätigen wir die RETURN-Taste, so antwortet der Rechner:

```
69
```

Das Bitmuster dieser Dezimalzahl 69 sieht folgendermaßen aus :

```
Ø1ØØØ1Ø1
```

In dieser Weise wird also in einem bestimmten Byte der Buchstabe E gespeichert.

Glücklicherweise braucht man sich all dies nicht zu merken, wenn man nicht will. Das Betriebssystem des Rechners sorgt dafür, daß diese Codierungen bzw. Umwandlungen automatisch vorgenommen werden, so daß wir beim Computergebrauch gar nichts davon merken.

Gleichwohl scheint es nicht ganz falsch, zum Verständnis der Funktionsweise eines Rechners einige derartige Hintergrundinformationen einmal kurz vorzustellen.

Übungsaufgaben

1. Gegeben ist der folgende Datenbestand :

Name	Alter	Größe	Geschlecht
MUELLER	42	178	m
MEIER	38	165	w
SCHULZ	29	169	w
ARNOLD	54	174	m
...	...	...	...

   Zeigen Sie anhand dieser sog. Datenmatrix, was man unter den folgenden Begriffen versteht :

   Feld, Array, String, Variable, Wert, Datei, Symbol.

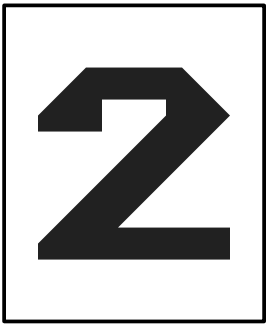
2. Was besagt das sog. EVA-Prinzip ?

3. Welches ist der Unterschied zwischen Programmanweisungen und Kommandos ?

4. Was sind ROM-Speicher und was sind RAM-Speicher ?

5. Was ist ein Byte ?

6. Welcher Dezimalzahl entspricht die Binärzahl 11Ø1Ø11Ø1 und welcher Binärzahl entspricht die Dezimalzahl 16Ø ?



Kapitel 2 : Der Direktmodus

Abschnitt 1 : Begriffsklärungen

BASIC-Rechner wie den ATARI kann man in unterschiedlichen Arten benutzen. Die beiden wichtigsten Arten sind:

- Direktmodus (auch Kommandomodus genannt)
- Programm-Modus

Im Direktmodus wird jede Anweisung, die wir eingeben, als Mitteilung an das Betriebssystem des Rechners, also als direkt ausführbares Kommando interpretiert. Dies bedeutet, daß der Rechner eine eingegebene Anweisung in diesem Modus direkt ausführt, ohne auf weitere Anweisungen zu warten.

Insoweit funktioniert er dann ähnlich wie ein Taschenrechner.

Im Programm-Modus hingegen sammelt der Rechner erst eine Folge von Anweisungen (ein Programm also) auf und arbeitet dieses Programm dann von Anfang bis Ende in einem Arbeitsgang ab. Dieser Programm-Modus ist der für uns viel interessantere Einsatzfall des Rechners und die späteren Kapitel werden sich ausschließlich mit diesem Fall beschäftigen.

Gleichwohl ist aber auch das Arbeiten im Direktmodus nicht uninteressant und zwar aus den folgenden Gründen :

1. Der Direktmodus eignet sich sehr gut dazu, mit der Tastatur des Rechners vertraut zu werden;
2. Im Direktmodus können wir schon einige Regeln der Programmiersprache BASIC kennenlernen und trainieren;
3. Im Direktmodus können wir den ATARI - wie schon erwähnt wurde - quasi als Taschenrechner-Ersatz benutzen;
4. Der Direktmodus wird später beim Austesten von Programmen gute Dienste leisten, wenn man logische Programmfehler suchen muß.

Deshalb nun also einige Einzelheiten zum Betrieb des ATARI im Direktmodus.

Abschnitt 2 : Die Tastatur

Schauen wir uns zunächst einmal die Tastatur des ATARI an:

Wir erkennen Symboltasten und eine Reihe von Sondertasten.

Zu den Symboltasten ist nicht viel zu sagen; sie funktionieren im Prinzip wie die auf einer Schreibmaschine. Neu hingegen sind die meisten der Sondertasten. Diejenigen von ihnen, die für den Einsteiger besonders wichtig sind, sollen hier kurz vorgestellt werden.

Betrachten wir deshalb zunächst kurz die sog. Funktionstasten am rechten Rand der Tastatur :

RESET	Diese Taste unterbricht den Programmablauf und setzt den Rechner in den Grundzustand zurück.
OPTION	Mit dieser Taste können Variationen innerhalb eines Programms gewählt werden.
START	Mit dieser Taste können Spiele oder bestimmte Programmteile gestartet werden.
SELECT	Diese Taste dient zur Auswahl von Anwendungen innerhalb eines Programms.
HELP	Mit dieser Taste können in einigen Programmen Bedienungshinweise erfragt werden.

Diese Funktionstasten sind für denjenigen ATARI-Nutzer, der meistens selbst programmiert (und an den wenden wir uns ja mit diesem Buch) nicht sonderlich wichtig: er kann auf sie völlig verzichten. Interessant ist allenfalls die RESET-Taste, die spätestens dann betätigt werden kann, wenn auf dem Rechner "gar nichts mehr geht".

Nun zu den wichtigeren der übrigen Sondertasten :

BREAK

Diese Taste dient zum Abbruch selbst geschriebener Programme.

CONTROL + ← → ↑ ↓

Wenn man die CONTROL-Taste zusammen mit einer der Pfeiltasten betätigt, dann kann der Cursor über den Bildschirm gesteuert werden, ohne daß das Programm oder eine sonstige Bildschirmanzeige verändert wird.

Auf diese Weise kann also z. B. sehr gut die Korrektur fehlerhafter Programmsätze vorbereitet werden, indem zunächst der Cursor an die fehlerhafte Stelle gesteuert wird. Die Korrektur selbst wird dann ausgeführt wie im folgenden beschrieben wird.

CONTROL + INSERT

Wenn man diese beiden Tasten gemeinsam betätigt, können in einen Satz Leerzeichen bzw. Zwischenräume eingefügt werden.

Müssen andere Zeichen bei einer Korrektur eingefügt werden (z. B. der Buchstabe A), so ist nach Betätigung dieser beiden Tasten einfach noch die Taste A zu drücken.

DELETE BACK SPACE

Die Betätigung dieser Taste rückt den Cursor eine Stelle nach links und löscht das Zeichen unter dem Cursor.

CONTROL + DELETE BACK SPACE	Die Betätigung dieser beiden Tasten zusammen löscht das jeweilige Zeichen unter dem Cursor. Die übrigen Zeichen rechts davon rücken nach.
CONTROL + ↓	Die Betätigung dieser beiden Tasten zusammen unterbricht die Ausgabe auf dem Bildschirm (z. B. beim LISTen von Programmen). Drückt man danach diese beiden Tasten erneut zusammen, so wird die Ausgabe fortgesetzt.
SHIFT	Diese Taste schaltet um zur zweiten Belegung der Symboltasten.
SHIFT + INSERT	Betätigt man diese beiden Tasten zusammen, so wird oberhalb des Cursors eine Leerzeile eingefügt.
SHIFT + DELETE BACK SPACE	Die Betätigung dieser beiden Tasten zusammen löscht die Zeile auf dem Bildschirm, in der sich der Cursor gerade befindet.
CAPS	Diese Taste dient zum Umschalten zwischen Groß- und Kleinschreibung.

SHIFT + CAPS

Betätigt man diese beiden Tasten, so wird der Computer im Großbuchstaben-Modus "verriegelt". Für die Sonderzeichen der numerischen Tasten muß allerdings weiterhin die SHIFT-Taste gedrückt werden.

RETURN

Mit Betätigung dieser Taste wird dem Rechner das Ende einer Zeile (einer Eingabe) angezeigt. Der Cursor rückt an den Anfang der nächsten Zeile ("Wagenrücklauf").

Generell gilt, daß jedes Zeichen, jede Tastenfunktion bzw. jede Funktion von Tastenkombinationen automatisch wiederholt wird, wenn die jeweilige Taste (bzw. Tastenkombination) länger gedrückt wird.

Abschnitt 3 : Arbeiten im Direktmodus

Für den Anfang unserer Arbeiten im Direktmodus sind die folgenden Feststellungen wichtig :

1. Wenn der Rechner auf dem Bildschirm sichtbare Ergebnisse im Direktmodus erzeugen soll, so müssen die entsprechenden Anweisungen mit dem BASIC-Wort

   PRINT

   eingeleitet werden. Dies hatten wir auch schon in Abschnitt 1.7 gesehen.

2. Die jeweils eingetippte Anweisung muß durch Betätigen der RETURN-Taste abgeschlossen werden. Dadurch erkennt der Rechner das Ende der Anweisung: schickt sie in den Arbeitsspeicher, schickt den Cursor an den Anfang der nächsten Zeile, führt die Anweisung aus und bringt das Ergebnis der Anweisung auf den Bildschirm.

3. Es ist darauf zu achten, daß man nicht die Ziffer Ø mit dem Buchstaben O (oder o) und die Ziffer 1 nicht mit dem Buchstaben l oder I verwechselt.

Geben wir nun beispielsweise die folgende Anweisung ein :

PRINT 3 und RETURN-Taste

Der Rechner antwortet mit

```
3
READY
```

d.h. er bringt das vorgesehene Ergebnis auf den Bildschirm und meldet mit dem Wort READY (fertig), daß er bereit ist, eine weitere Aufgabe auszuführen.

```
PRINT 3+50
```

Auf den Hinweis, daß am Ende der Anweisung die RETURN-Taste zu betätigen ist, soll hinfort verzichtet werden.

Der Rechner antwortet :

```
8
READY
```

Auch die sich immer wiederholende READY-Meldung soll im folgenden nicht mehr genannt werden.

```
PRINT 3*6/2
```

Der Rechner antwortet :

```
9
```

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

Ergebnis :

```
3.5
```

An diesen kleinen Beispielen erkennen wir einige wichtige Details :

1. Der Rechner verwendet einen Punkt, um Dezimalstellen abzutrennen (nicht etwa ein Komma), und so muß es auch der Benutzer halten, wenn er Dezimalstellen eingehen will.

2. Wird eine Klammer verwendet, so wird der Inhalt dieser Klammer zuerst gerechnet.

3. Es gelten die üblichen mathematischen Regeln, wie z. B.: Punktrechnung (Multiplikation und Division) gehen vor Strichrechnung (Addition und Subtraktion).

4. Die arithmetischen Operatoren sind die folgenden:

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
^	Potenzierung

Im Direktmodus kann man auch mehr als nur eine Rechenaufgabe in einer Anweisung ausführen, wie die folgenden beiden Beispiele zeigen :

```
PRINT 3+5,7-2
```

Das Ergebnis ist :

```
8           5
```

```
PRINT 3+5;7-2
```

Hier lautet das Ergebnis :

```
85
```

Wir erkennen : Trennt man zwei (oder mehrere) Aufgaben durch Komma, so werden die Ergebnisse auf dem Bildschirm auseinander gerückt, trennt man hingegen durch Strichpunkt (Semikolon), rücken die Ergebnisse eng zusammen.

Nützlicher HINWEIS :

> Das Wort PRINT darf durch das Fragezeichen ? abgekürzt werden.

Im Direktmodus kann man aber noch mehr als nur rechnen :

```
? "SUSI"
```

Der Rechner antwortet mit :

```
SUSI
```

Wir erkennen : Wird hinter dem Befehlswort PRINT (bzw. hinter dem Fragezeichen, welches ja PRINT ersetzen darf) eine Zeichenkette (ein String) in Anführungszeichen eingeschlossen, so wiederholt der Rechner diese Zeichenkette ohne die Anführungszeichen.

```
? "3+5 = ";3+5
```

Ergebnis :

```
3+5 = 8
```

Schließlich ist anzumerken, daß im Direktmodus auch Variablen benutzt werden können (über den Begriff der Variablen haben wir ja schon kurz gesprochen) :

Geben wir zum Beispiel ein :

```
X=5
```

so meldet der Rechner :

```
READY
```

Geben wir jetzt aber weiter ein :

```
PRINT X,X*X
```

so meldet der Rechner :

```
5          25
```

Wir erkennen daran, daß der Rechner mit dem Wert 5: den wir mit dem ersten Kommando eingegeben haben, in der zweiten Anweisung dann arbeiten kann, weil er den Wert 5 im Speicherfeld mit dem Namen X gespeichert und nicht vergessen hat.

Geben wir abschließend ein :

```
? Y
```

so antwortet der Rechner mit :

```
0
```

weil im Feld mit dem Namen Y nichts gespeichert wurde.

Übungsaufgaben

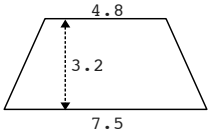
1. Mit dem ATARI soll die Fläche eines Rechtecks ausgerechnet werden, das 7.4 cm lang und 2.Ø8 cm breit ist.

2. Zu berechnen ist die Fläche eines Kreises mit dem Radius 5 cm.

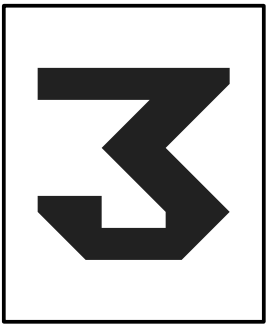
3. Zu berechnen ist die Mehrwertsteuer (14 %) bei einem Netto-Rechnungsbetrag von DM 45Ø.--

4. Zu berechnen ist die Mehrwertsteuer, die in einem Brutto-Rechnungsbetrag von DM 45Ø.-- enthalten ist.

5. Zu berechnen ist die Fläche eines Trapezes, dessen Maße der nebenstehenden Skizze zu entnehmen sind.



6. Zu berechnen ist der Rauminhalt eines Würfels, der eine Kantenlänge von 3.5 cm aufweist.



Kapitel 3 : Die Vorbereitung der Programmierung

Abschnitt 1 : Vorbemerkung

Will man einen Rechner zur Lösung bestimmter Aufgaben einsetzen, so ist es nicht damit getan, daß man eine Programmiersprache erlernt oder trainiert. Noch wichtiger ist es, daß man es lernt, Probleme, die durch einen Computer gelöst werden sollen, so weit und so detailliert vorzubereiten, daß man das entsprechende Programm schreiben und dem Rechner übergeben kann.

Kurz gesagt : Wer nicht weiß, wie beim Ausrechnen von Prozentwerten die Dreisatzrechnung eingesetzt wird und wie sie funktioniert, der kann auch das entsprechende Programm nicht schreiben, d.h. dem nützt der Computer nichts.

Mehr noch : Der Computereinsatz erfordert eine ganz spezielle Denkweise. Wenn der Rechner für uns ein Problem lösen soll, so muß es vorher in solche Teilschritte zerlegt werden, die der Rechner bewältigen kann.

Man spricht in diesem Zusammenhang von der sog. Problemanalyse und in den Trainingsabschnitten der folgenden Kapitel wollen wir gerade auf diesen vorbereitenden Schritt beim Computereinsatz besondere Aufmerksamkeit legen - er entscheidet nämlich letztlich darüber. Ob der Computereinsatz erfolgreich zum Ziel führt oder nicht.

Abschnitt 2 : Problemanalyse

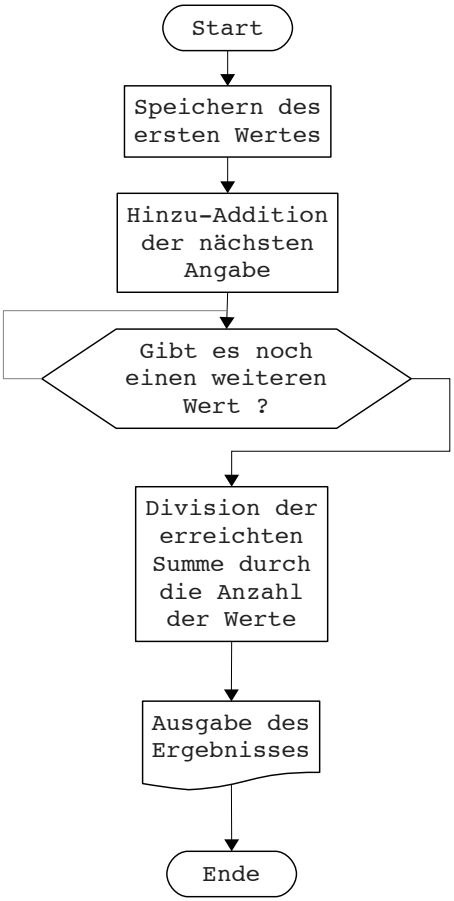
Das Stichwort "Problemanalyse" soll anhand einer konkreten Aufgabenstellung erläutert werden :

Bei einer solchen Problemanalyse geht es darum, die Abfolge der für die Problemlösung notwendigen Arbeitsschritte zu skizzieren. Diese Arbeitsschritte werden üblicherweise in einem Flußdiagramm (Programmablaufplan) aufgezeichnet, wie das folgende Beispiel zeigt.

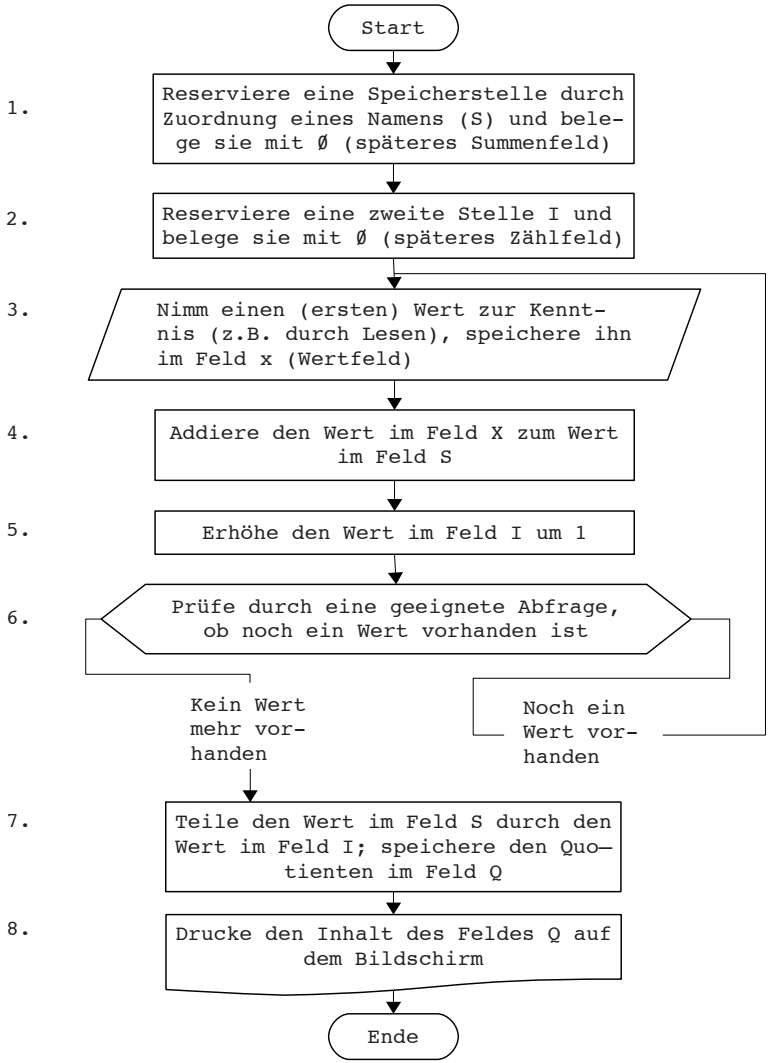
Das Beispiel, welches wir hier zur Illustration verwenden, geht davon aus, daß aus einer bestimmten Anzahl von Einkommenswerten (beispielsweise aus einer statistischen Umfrage stammend) das Durchschnittseinkommen berechnet werden soll.

Der Problemlösungsweg in der Darstellungsform eines Flußdiagramms könnte folgendermaßen aussehen :

Flußdiagramm Nr. 1 : Allgemeine Darstellung

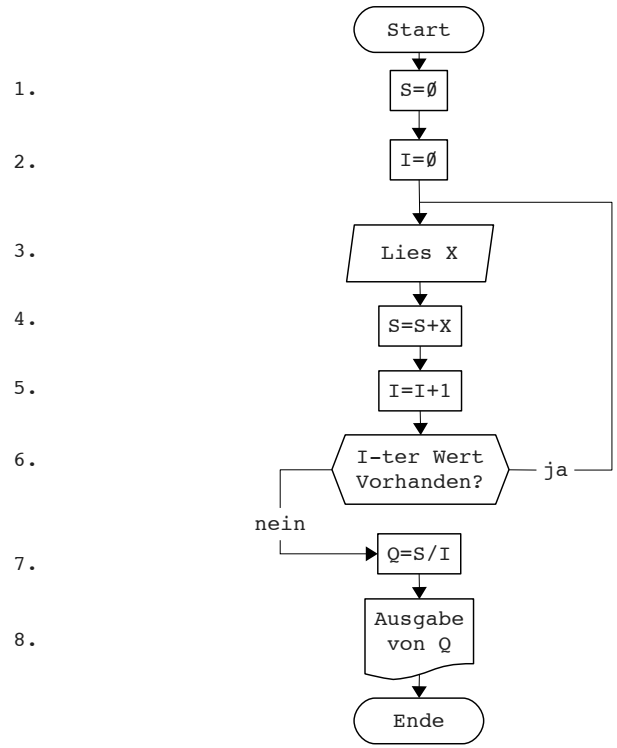


Flußdiagramm Nr. 2 : Computeradäquate Darstellung



Die vorangegangene etwas umständliche Formulierung eines Flußdiagramms läßt sich wesentlich verkürzen, wenn man schon ein bißchen Erfahrung in der Entwicklung solcher Flußdiagramme gesammelt hat :

Flußdiagramm Nr. 3 : Kurzfassung



Abschnitt 3 : BASIC

Wenn man ein Flußdiagramm, das bei komplizierten Problemstellungen recht umfangreich werden kann und gerade dann sehr wichtig für die vorbereitende Klarstellung des Problemlösungsweges ist, erstellt hat, dann kann man sich an die Programmierung des Problems machen.

Als Programmiersprache wählen wir in diesem Buch BASIC.

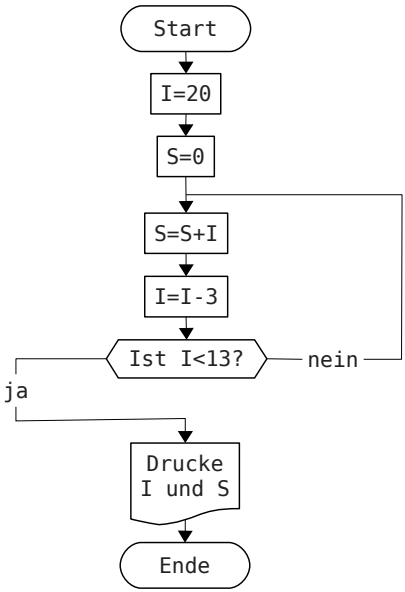
BASIC =	
Beginner's All-Purpose Symbolic Instruction Code	(Symbolischer Allzweck-Anweisungs-Code für Anfänger)

Wichtig ist in diesem Zusammenhang, daß eine Vielzahl unterschiedlicher BASIC-Dialekte existieren, d.h. diese Sprache ist nicht hinreichend genormt - und gerade bei den ATARI-Rechnern macht sich dies wieder bemerkbar.

Allerdings haben diese Dialekte große gemeinsame Bereiche und zwar glücklicherweise gerade dort, wo es für Einsteiger wichtig ist.

Übungsaufgaben

1. Was ist das Ergebnis des folgenden Flußdiagramms ?



2. Aus einem Datenbestand sollen der kleinste und der größte Wert herausgesucht werden. Entwerfen Sie ein dafür geeignetes Flußdiagramm.

3. In einen Rechner werden 1ØØØ Daten eingegeben. Dabei steht Ø für "ledig", 1 für "verheiratet", 2 für "geschieden" und 3 für "verwitwet". Die Ziffer 9 wird eingegeben, wenn bei einer bestimmten Person

keine Angabe zum Familienstand vorliegt.

Es soll ein Flußdiagramm entworfen werden, das dazu führt, daß wir am Ende wissen, wieviele Ledige, wieviele Verheiratete etc. in dem Datenbestand vorhanden sind.

4

Kapitel 4 : Einfaches Rechnen mit dem ATARI

Abschnitt 1 : Vorbemerkung

Dieses und die folgenden Kapitel dienen dazu, die Programmiersprache BASIC am ATARI-Rechner zu trainieren.

Wir gehen dabei so vor, daß wir eine Reihe von wichtigen BASIC-Statements (also von BASIC-Sprachelementen) vorstellen, um mit diesen dann beim Entwickeln und Austesten von Programmen diese Sprache zu üben.

Es leuchtet ein, daß wir dabei mit einfachen Problemstellungen beginnen müssen, weil wir ja nicht gleich das erste Trainingskapitel, um das es sich hier handelt, mit BASIC-Statements überfrachten wollen.

Kapitel 4 : Einfaches Rechnen
Abschnitt 1 : Vorbemerkung

Mit nur wenigen ersten Statements können aber schon eine Reihe unterschiedlicher Rechenaufgaben in Angriff genommen werden.

Nach der Besprechung der im jeweiligen Kapitel benutzten BASIC-Statements folgt dann der wichtige Trainingsteil, wo anhand ausgewählter Problemstellungen der Einsatz dieser Statements geübt wird.

In den Trainingsbeispielen orientieren wir uns immer gemäß der folgenden Arbeitsschritte :

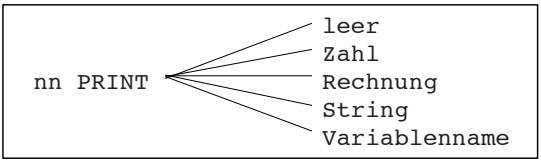
- 1. Vorstellung des Problems
- 2. Problemanalyse
- 3. Programm
- 4. Liste der im Programm verwendeten Variablen
- 5. Programmbeschreibung
- 6. Programmergebnisse

Abschnitt 2 : Erste BASIC-Anweisungen : Ein- und Ausgabe von Informationen

Wenden wir uns also zunächst den ersten BASIC-Statements zu, die wir benötigen, um in den Rechner zu verarbeitende Daten einzugeben, bzw. um die errechneten Ergebnisse auf dem Bildschirm auszugeben.

Beginnen wir mit dem PRINT-Statement :

Statement 1 :



Das "nn" in dieser allgemeinen Schreibweise bedeutet, daß vor dem Schlüsselwort PRINT in einem BASIC-Programm eine Nummer, die sog. Satznummer stehen muß - im Gegensatz zum Direktmodus, wo gerade das nicht der Fall war.

Jeder Satz in einem BASIC-Programm wird durch eine Satznummer eingeleitet.

Üblicherweise nummeriert man dabei in Zehnerschritten, weil man dann im Nachhinein die Möglichkeit hat, zusätzliche Sätze noch einzuschieben.

> Jeder Satz wird mit der RETURN-Taste abgeschlossen.

Wenn wir uns das obige PRINT-Statement etwas näher anschauen, so sehen wir, daß es fünf Möglichkeiten bietet :

1. "leer" bedeutet, daß hinter dem Schlüsselwort PRINT nichts steht. Der Effekt eines solchen "nackten" PRINT-Statements besteht in der Ausgabe einer Leerzeile bei der Programmabarbeitung.

2. Wenn hinter dem Wort PRINT nur eine Zahl steht, wird diese Zahl einfach ausgegeben.

3. Ist eine Rechnung angefügt (ein sog. arithmetischer Ausdruck), so wird das Rechenergebnis bestimmt und dieses dann ausgegeben.

4. Steht hinter dem PRINT ein String (eine Zeichenkette, die in Anführungszeichen einzuschließen ist), so wird diese Zeichenkette ausgegeben.

5. Bei Angabe eines Variablennamens wird der Inhalt des Speicherfeldes ausgegeben, das diesen Namen trägt.

Diese Möglichkeiten (mit Ausnahme der ersten) haben wir schon im Direktmodus kennengelernt, so daß hier auf Beispiele verzichtet werden kann.

Wie dort schon festgestellt wurde, können die Möglichkeiten 2. bis 5. auch mehrfach (und/oder gemischt) verwendet werden, wobei sie dann durch Komma oder durch Strichpunkte voneinander getrennt werden müssen.

Wichtig sind noch die folgenden Hinweise :

> Jede PRINT-Anweisung führt bei der Programmabarbeitung in eine neue Zeile.
>
> Diese Regelung gilt für ein bestimmtes PRINT-Statement dann nicht, wenn das vorhergehende PRINT—Statement auf , oder ; endet; der Zeilenvorschub wird dann aufgehoben.
>
> Mehrere BASIC-Statements können in einem Satz stehen. Sie müssen dann durch Doppelpunkte voneinander getrennt werden.
>
> Die Anweisung PRINT CHR$(125) löscht bei der Programmabarbeitung den Bildschirm.

Bei der Ausgabe von Programmergebnissen ist dieser letzte Hinweis besonders hilfreich, weil das Löschen des Bildschirms vor der Ergebnisausgabe sich sehr hübsch macht.

Das nächste Statement, das wir benötigen,ist das END-Statement :

Statement 2 :

```
nn END
```

Mit diesem Statement wird ein BASIC-Programm beendet. Nähere Erläuterungen dazu sind entbehrlich.

Mit diesen beiden Statements zusammen kann man schon erste kleine Programme erstellen, wie das folgende Beispiel zeigt :

```
10 PRINT CHR$(125)
20 PRINT "MEIN ERSTES PROGRAMM"
30 PRINT:PRINT
40 PRINT "3 + 5 = ";3+5
50 PRINT:PRINT
60 PRINT "ENDE":END
```

Dieses Programm druckt auf den Bildschirm: nachdem dieser freigemacht wurde (Satz 1Ø) den Text MEIN ERSTES PROGRAMM, läßt danach zwei Zeilen frei (Satz 3Ø mit zwei Statements) und druckt dann 3 + 5 = 8. Danach werden wieder zwei Zeilen frei gehalten (Satz 5Ø) und schließlich wird das Wort ENDE gedruckt und das Programm beendet (beides in Satz 6Ø).

Dieses Programm (und jedes andere Programm auch) wird aber erst dann abgearbeitet, wenn wir das folgende Kommando eingeben :

Kommando 1 :

```
RUN
```

Wir merken uns in diesem Zusammenhang :

Kommandos erhalten keine Satznummer !

Während das PRINT-Statement also zum Beispiel Rechenaufgaben erledigen kann und zur Ausgabe von Informationen dient, benutzen wir die folgenden beiden Statements zur Informationseingabe :

Statement 3 :

```
                                  Zahl
                                  Rechnung
nn LET Variablenname   =          String
                                  Variablenname
```

Mit diesem Statement wird dem Variablennamen, der links vom Gleichheitszeichen steht, der Wert zugewiesen: der sich rechts ergibt, bzw. der rechts steht. Physikalisch

bedeutet dies, daß der Speicherstelle im Rechner, die mit dem Namen links vom Gleichheitszeichen benannt ist, ein Wert zugewiesen wird, wobei ein "Wert" nicht notwendigerweise eine Zahl sein muß, sondern es kann sich beispielsweise auch um einen String handeln.

Also auch hier wieder - ähnlich wie beim PRINT-Statement - mehrere Möglichkeiten, wobei jetzt aber Mischungen oder mehrfaches Auftreten einzelner Möglichkeiten in nur einem Statement ausgeschlossen sind.

Zu diesem Statement ein paar einfache Illustrationsbeispiele :

```
10 LET X = 3
20 LET Y = 3+5
30 LET N$ = "OTTO"
40 LET Z = X
50 LET I = I + 1
```

Es ist in diesem Zusammenhang anzumerken, daß das Schlüsselwort LET im LET-Statement auch entfallen darf. Der obige Satz 5Ø zum Beispiel könnte also auch lauten :

```
50 I = I + 1
```

Bei der Verwendung von Variablennamen links vom Gleichheitszeichen (aber gegebenenfalls natürlich auch rechts) ist zu beachten, daß generell zwei Typen von Variablen unterschieden werden :

- Numerische Variablen
- String-Variablen

Numerische Variablen bezeichnen Speicherfelder im Rechner, die mit Zahlen belegt werden, String-Variablen hingegen solche, die nicht mit Zahlen belegt werden, sondern beispielsweise mit Worten oder sonstigen Texten.

> Der Rechner unterscheidet Stringvariablen von numerischen Variablen dadurch, daß an Stringvariablennamen das $-Zeichen angehängt wird.
> Das erste Symbol eines Variablennamens muß ein Buchstabe sein.
> Als Symbole sind generell nur Buchstaben und Ziffern erlaubt.

Erlaubte Namen für numerische Variablen sind also zum Beispiel :

X XX NETTO A5 PP17 ....

Stringvariablennnamen könnten folgendermaßen aussehen :

A$ N$ ORT$ NAME$ A15$ ....

Maximal können beim ATARI bis zu 11Ø Symbole verwendet werden, um einen Namen zu bilden. Es empfiehlt sich aber,

Namen nicht zu lang werden zu lassen, um die Gefahr von Tippfehlern zu vermeiden. Vor allem bei kleineren Programmen erfüllen durchaus auch einstellige Namen die Zwecke, die erfüllt werden müssen.

Bei der Namensvergabe muß darauf geachtet werden, daß kein BASIC-Schlüsselwort als Name verwendet wird : Beispielsweise können Worte wie PRINT oder END nicht als Variablennamen verwendet werden.

Eleganter als mit dem LET-Statement erfolgt die Informationseingabe im Dialog mit Hilfe des folgenden Statements :

Statement 4 :

```
nn INPUT Variablenliste
```

Erreicht der Rechner bei der Programmabarbeitung ein solches INPUT-Statement, so hält er in der Programmabarbeitung inne, produziert auf dem Bildschirm ein Fragezeichen und fordert vom Benutzer damit die Eingabe so vieler Werte an, wie Variablen in der Variablenliste genannt

sind : Besteht diese Liste nur aus einem Namen, so ist nur ein Wert einzugeben; besteht sie hingegen aus mehreren Namen, die dann durch Kommata voneinander getrennt werden müssen, so sind auch mehrere Werte einzugehen, die der Benutzer ebenfalls durch Kommata voneinander trennen muß.

```
10 INPUT X
```

erwartet also einen numerischen Wert,

```
10 INPUT A$
```

erwartet einen String,

```
10 INPUT X,Y,N$
```

erwartet zwei numerische Werte und dann einen String.

Mit diesem Statement ist es also offenbar möglich: ein einziges Programm, das für eine bestimmte Aufgabe entwickelt worden ist, immer wieder zu verwenden, indem man es mit unterschiedlichen Input-Informationen versorgt.

Es ist in diesem Zusammenhang zu beachten, daß es zu Fehlermeldungen kommt, wenn der Benutzer zu viele oder zu wenige Daten im Vergleich zur Variablenliste des INPUT-Statements eingibt.

Betrachten wir abschließend noch zwei wichtige Kommandos :

Kommando 2 :

```
LIST
```

Mit diesem Kommando kann ein Programm wieder auf dem Bildschirm aufgelistet werden, was beispielsweise dann notwendig ist, wenn es schon über den oberen Bildschirmrand hinausgewandert ist, man es aber wieder sehen möchte.

Wir haben schon darauf hingewiesen, daß das Auflisten unterbrochen werden kann, wenn man die Tasten

CONTROL und 1

zusammen drückt, und daß durch nochmaliges Betätigen dieser beiden Tasten der Listprozeß fortgesetzt werden kann.

Kommando 3 :

```
NEW
```

Dieses Kommando löscht den Inhalt des Programmspeichers (also VORSICHT !), was immer dann sinnvoll ist, wenn man ein neues Programm eingeben möchte und das alte nicht mehr benötigt.

ERSTES TRAINING

Mit den Statements und Kommandos, die wir bisher besprochen haben, soll nun die erste "Trainingssitzung" bestritten werden. Wir behandeln in dieser Sitzung ein paar einfache Rechenaufgaben, wobei gemäß des Gliederungsschemas vorgegangen wird, das an anderer Stelle schon vorgestellt wurde :

- Vorstellung des Problems
- Problemanalyse
- Programm
- Variablenliste
- Programmbeschreibung
- Ergebnisse

TRAININGSBEISPIEL 1 : Kreisberechnungen

1. Vorstellung des Problems

Es soll ein BASIC-Programm entwickelt werden, welches für beliebige Kreise Kreisumfang und Kreisfläche berechnet. Mehr ist zur Problemstellung bei dieser noch sehr einfachen Aufgabe nicht zu sagen.

2. Problemanalyse

Bei jeder Problemanalyse ist es sinnvoll, sich an das schon erwähnte EVA-Prinzip zu erinnern. Dies bedeutet, daß zunächst zu überlegen ist, welche Input-Informationen der Rechner braucht, welche Verarbeitungsschritte dann notwendig sind und welche Ausgaben durch das Programm schließlich produziert werden sollen.

Bei dieser Aufgabenstellung ist dies noch sehr einfach :

Als Input-Information benötigt das Programm den Radius des Kreises, dessen Umfang und Fläche berechnet werden sollen.

Die Verarbeitungsschritte bestehen dann in der Anwendung der entsprechenden Berechnungsformeln wie sie aus der

Schulmathematik bekannt sind :

Umfang = 2 * r π

r = radius

Fläche = r2 * π

Wir benutzen im folgenden Programm für die Kreiskonstante π den Näherungswert 22/7.

Nach den Verarbeitungsschritten (hier Rechenschritte) ist ausschließlich die Ergebnisausgabe zu programmieren.

Hier liegt also ein recht einfaches Problem vor, das relativ leicht zu programmieren ist. Gleichwohl ist das folgende Programm umfangreich geraten, weil wir häufig von PRINT-Statements Gebrauch gemacht haben, um Erläuterungen und dergl. auf dem Bildschirm auszugeben. Auch bei den später folgenden Programmen werden wir diese Art der Programmierung beibehalten.

Überdies leiten wir alle Programme mit einem REM-Statement ein, über das erst später gesprochen wird. Es dient in diesem Zusammenhang lediglich dazu, den Programmen eine Überschrift zu geben.

• 3. Programm

```
10 REM T1-KREISBERECHNUNGEN
20 PRINT CHR$(125)
30 PRINT "          KREISBERECHNUNGEN."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET FUER BELIE- "
80 PRINT " BIGE KREISE UMFANG UND FLAECHE."
90 PRINT :PRINT :PRINT
100 PRINT "RADIUS DES KREISES : ";
110 INPUT R
120 LET PI=22/7
130 LET U=2*PI*R
140 LET F=PI*R*R
150 PRINT :PRINT :PRINT
160 PRINT "EIN KREIS MIT DEM RADIUS ";R
170 PRINT "HAT DEN UMFANG   ";U
180 PRINT "UND DIE FLAECHE  ";F
190 PRINT :PRINT :PRINT
200 PRINT "ENDE DER BERECHNUNGEN"
210 PRINT "NEUSTART NUR MIT  RUN"
220 END
```

Es wurde oben schon erwähnt, daß ein solches Programm auch viel kürzer abgefaßt werden kann, wenn man z. B. auf alle überflüssigen PRINT- oder REM-Statements verzichtet, und wenn man sich daran erinnert, daß ja auch mehrere Statements in einem Satz stehen dürfen.

Für dieses Kreisbeispiel soll eine solche kurze Version einmal vorgestellt werden, damitder Leser den Unterschied studieren kann. Bei den folgenden Programmen soll darauf aber dann verzichtet werden :

Kurzversion des vorangegangenen Programms :

```
10 ? CHR$(125):? "KREISBERECHNUNGEN."
20 ? "RADIUS ";:INPUT R
30 PI=22/7:U=2*PI*R:F=PI*R*R
40 ?: ?: ? "RADIUS = ";R
50 ? "UMFANG = ";U:? "FLAECHE = ";F
60 END
```

Diese Programmversion ist also offenbar ganz wesentlich verkürzt und schöpft immer noch nicht alle Verkürzungsmöglichkeiten aus. Der Leser sollte versuchen, noch weitere Verkürzungen übungshalber vorzunehmen.

• 4. Variablenliste

F = Fläche des Kreises
PI = Kreiskonstante (Näherung durch 22/7)
R = Kreisradius
U = Umfang des Kreises

• 5. Programmbeschreibung

Bei der Programmbeschreibung orientieren wir uns an der ersten Programmversion :

Satz 1Ø	:	Programmüberschrift
Satz 2Ø	:	Löschen des Bildschirms
Satz 3Ø-8Ø	:	Ausgabe von Überschrift und Erläuterung
Satz 9Ø	:	Drei leere Zeilen
Satz 1ØØ-11Ø	:	Anforderung der Input-Information mit erläuterndem Text
Satz 12Ø-14Ø	:	Berechnungen
Satz 15Ø	:	Drei leere Zeilen
Satz 16Ø-18Ø	:	Ausgabe der Ergebnisse
Satz 19Ø	:	Drei leere Zeilen
Satz 2ØØ-21Ø	:	Erläuternde Schlußangaben
Satz 22Ø	:	Beendigung des Programms

• 6. Ergebnisse

Wenn wir auf die Anforderung des Programms (Satz 1ØØ-11Ø) beispielsweise den Radius 5 eingeben, so erhalten wir :

```
EIN KREIS MIT DEM RADIUS 5
HAT DEN UMFANG   31.4285714
UND DIE FLAECHE 78.5714285



ENDE DER BERECHNUNGEN
NEUSTART NUR MIT RUN
```

TRAININGSBEISPIEL 2 : Gewinnberechnungen

• 1. Vorstellung des Problems

Wenn bei einem bestimmten Geschäftsvorgang Kosten und Erträge in bestimmter Höhe anfallen, dann ist es häufig von Interesse, auszurechnen, wie hoch der Gewinn (bzw. der eventuelle Verlust) absolut und in Prozent ausfällt.

Es soll nun ein BASIC-Programm entwickelt werden: das derartige Berechnungen durchführt.

• 2. Problemanalyse

Auch hier ist die Problemanalyse wieder sehr einfach, so daß wir uns kurz fassen können :

Der notwendige Programminput sind die jeweiligen Kosten und Erträge; die Verarbeitungsschritte bestehen in der Berechnung der Differenz aus beiden Größen. Darüber hinaus ist auszurechnen, wieviel Prozent diese Differenz bezogen auf die Kosten ausmacht. Die berechneten Werte sind dann als Ergebnisse auf dem Bildschirm auszugeben.

3. Programm

```
10 REM T2-GEWINNBERECHNUNGEN
20 PRINT CHR$(125)
30 PRINT "          GEWINNBERECHNUNGEN."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET FUER BELIE-"
80 PRINT "BIG EINZUGEBENDE KOSTEN UND ERTRAEGE"
90 PRINT "DEN GEWINN IN DM UND IN PROZENT."
100 PRINT :PRINT :PRINT
110 PRINT "NETTODETRAG IN DM  :  ";
120 INPUT NB
130 PRINT "BRUTTOBETRAG IN DM :  ";
140 INPUT BB
150 G=BB-NB
160 P=(G/NB)*100
170 PRINT :PRINT :PRINT
180 PRINT "NETTO             :  ";NB
190 PRINT "BRUTTO            :  ";BB
200 PRINT :PRINT
210 PRINT "GEWINN IN DM      :  ";G
220 PRINT "GEWINN IN PROZENT:  ";P
230 PRINT :PRINT :PRINT
240 PRINT "ENDE DER BERECHNUNGEN"
250 PRINT "NEUSTART NUR MIT RUN"
260 END
```

• 4. Variablenliste

BB = Bruttobetrag (Ertrag)
G = Gewinn (Differenz zwischen Netto- und Bruttobetrag)
NB = Nettobetrag (Kosten)
P = Gewinn in Prozent

• 5. Programmbeschreibung

Satz 1Ø-1ØØ	:	Überschrift, Löschen des Bildschirms, Ausgabe von Überschrift und Erläuterungen und Leerzeilen
Satz 11Ø-14Ø	:	Anforderung von Netto- und Bruttobetrag
Satz 15Ø	:	Berechnung der Differenz
Satz 16Ø	:	Berechnung der prozentualen Differenz (Dreisatzrechnung)
Satz 17Ø-23Ø	:	Ausgabe der Ergebnisse und Leerzeilen
Satz 24Ø-26Ø	:	Beendigung des Programms

Der Leser erkennt, daß wir uns in dieser Programmbeschreibung kürzer gefaßt haben als in dem ersten Beispiel. So wollen wir es auch in Zukunft halten, weil es sicherlich nicht notwendig ist, in der Programmbeschreibung auf jede einzelne Leerzeile oder auf jeden einzelnen Erläuterungstext gesondert einzugehen.

• 6. Ergebnisse

Gehen wir auf die Anforderung des Programms als Nettobetrag z. B. 4ØØ, als Bruttobetrag 5ØØ ein, so erzeugt das Programm die folgenden Ergebnisse :

```
NETTO            : 400
BRUTTO           : 500


GEWINN IN DM     : 100
GEWINN IN PROZENT: 25



ENDE DER BERECHNUNGEN
NEUSTART NUR MIT RUN
```

TRAININGSBEISPIEL 3 : Quadratzahlen

• 1. Vorstellung des Problems

Es soll ein BASIC-Programm entwickelt werden, welches für eine beliebige einzugebende Zahl die Quadratzahl bestimmt.

• 2. Problemanalyse

Die Problemanalyse bereitet bei diesem sehr einfachen Beispiel überhaupt keine Schwierigkeiten :

Das Programm benötigt als Input-Information die zu quadrierende Zahl; der Verarbeitungsschritt besteht in der Quadrierung selbst und der Ausgabeschritt stellt die Ergebnisse auf dem Bildschirm bereit.

Entsprechend einfach wie die Problemanalyse ist auch das Programm. Es fällt nur deshalb wieder etwas länger aus, weil wir eine Reihe erläuternder Angaben über PRINT-Statements mit ausgeben.

3. Programm

```
10 REM T3-ZAHLENTAFEL
20 PRINT CHR$(125)
30 PRINT "              ZAHLENTAFEL"
40 PRINT :PRINT: PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT: PRINT
70 PRINT "DIESES PROGRAMM BERECHNET FUER EINE"
80 PRINT "BELIEBIGE EINZUGEBENDE ZAHL DIE QUA-"
90 PRINT "              DRATZAHL."
110 PRINT :PRINT: PRINT
120 PRINT "BITTE EINE ZAHL : ";:INPUT X
140 Q=X*X
150 PRINT :PRINT: PRINT
160 PRINT "DAS QUADRAT VON ";X;" IST ";Q
170 PRINT :PRINT: PRINT
180 PRINT "ENDE DER BERECHNUNGEN"
190 PRINT "NEUSTART NUR MIT RUN"
200 END
```

4. Variablenliste

Q = Quadratzahl
X = zu quadrierende Zahl

• 5. Programmbeschreibung

Satz 1Ø-1ØØ	:	Überschrift, Löschen des Bildschirms, Ausgabe von Informationen zum Programm, Leerzeilen
Satz 12Ø	:	Anforderung der zu quadrierenden Zahl
Satz 14Ø	:	Quadrierung
Satz 15Ø-17Ø	:	Ausgabe der Ergebnisse
Satz 18Ø-2ØØ	:	Schlußmitteilungen und Beendigung des Programms

• 6. Ergebnisse

Geben wir beispielsweise auf die Anforderung durch das Programm als zu quadrierende Zahl die Zahl 5 ein, so ergibt sich auf dem Bildschirm:

```
DAS QUADRAT VON 5 IST 25


ENDE DER BERECHNUNGEN
NEUSTART NUR MIT RUN
```

Abschnitt 3 : Das Korrigieren von Sätzen

Es ist leider nicht zu umgehen, daß man bei der Eingabe von Programmen Fehler begeht, die dann korrigiert werden müssen. Bevor nun im folgenden Kapitel längere Programme vorgestellt werden, ist es deshalb ganz sinnvoll, sich kurz anzuschauen, wie man in geschickter Weise Eingabefehler korrigieren kann.

Prinzipiell sind dabei drei Fälle zu unterscheiden.

1. Der fehlerhafte Satz ist noch nicht per RETURN-Taste abgeschickt, d.h. man hat sofort gemerkt, daß man sich vertippt hat.
2. Der fehlerhafte Satz ist schon per RETURN-Taste abgeschickt und der Rechner reagiert daraufhin mit einer Fehlermeldung.
3. Der Fehler wird erst bei der Programmabarbeitung, also erst nach Eingabe des Kommandos RUN festgestellt und der Rechner stellt die weitere Programmabarbeitung mit einer Fehlermeldung ein.

Wie ist vorzugehen ?

Zu 1.: Wenn der Fehler bei der Eingabe sofort bemerkt wird, dann kann man mit der DELETE/BACKSPACE-Taste den Cursor an die fehlerhafte Stelle zurücksteuern und von da an den Satz korrekt komplettieren.

Zu 2.: Gibt man beispielsweise den fehlerhaften BASIC-Satz ein :

```
10 PRUNT "AHA"
```

so meldet der Rechner nach Betätigen der RETURN-Taste :

```
10 ERROR-   PRUNT "AHA"
```

und zeigt mit den Wort ERROR (=Fehler) an, daß dieser Satz fehlerhaft eingegeben wurde.

Die einfachste Möglichkeit der Korrektur besteht nun darin, diesen Satz mit der gleichen Satznummer einfach neu zu schreiben und abzuschicken.

Wir können aber auch folgendermaßen vorgehen :

a) Das Programm wird mit dem Kommando LIST gelistet;
b) Wir steuern mit den Pfeiltasten, die zusammen mit der Taste CONTROL zu betätigen sind, den Cursor an die fehlerhafte Stelle;
c) Korrektur des Fehlers;
d) Betätigen der RETURN-Taste.

Wie vorzugehen ist, wenn an der zu korrigierenden Stelle Symbole eingefügt oder gelöscht werden sollen, wurde bei der Tastaturbeschreibung schon erläutert.

Zu 3.: Wird der Fehler erst bei der Programmabarbeitung festgestellt, so gibt der Rechner eine Fehlermeldung aus. Betrachten wir beispielsweise das folgende kleine Programm :

```
10 LET A=-1
20 LET B=A/0
30 PRINT B
40 END
```

Wenn wir dieses Programm per RUN-Kommando starten, so meldet der Rechner unverzüglich :

```
ERROR-   11 AT LINE 20
```

Schlagen wir im Fehleranhang nach, so finden wir unter der Fehlernummer 11 die Mitteilung, daß der (unzulässige) Versuch unternommen wurde, durch Null zu dividieren. Da dies nicht möglich ist, muß das Programm korrigiert werden. Es ist dabei so zu verfahren, wie auf der zweiten Hälfte der Seite 81 beschrieben wurde.

Wie die Korrektur auszusehen hat, hängt vom jeweiligen Sachproblem und von der jeweiligen Fehlermeldung ab (die wichtigsten ATARI-Fehlermeldungen sind im Anhang aufgeführt und kurz erläutert).

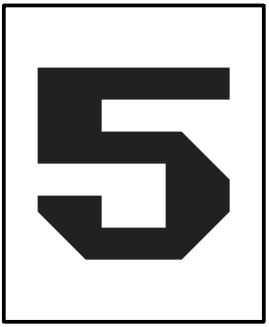
Übungsaufgaben

1. Zu entwerfen ist ein BASIC-Programm, welches für beliebige Dreiecke, deren Grundseite und Höhe gegeben sind, die Fläche ausrechnet.

2. Gegeben sei der Bevölkerungsstand einer Nation zu einem bestimmten Zeitpunkt. Es soll ein Programm erstellt werden, welches für eine beliebig einzugebende Zeitdauer den endgültigen Bevölkerungsstand ausrechnet, wenn von einer beliebigen jährlichen Bevölkerungswachstumsrate ausgegangen wird.

3. Es soll ein allgemeines Prozentberechnungsprogramm geschrieben werden. Dieses soll ausrechnen können, welchen Anteil in Prozent eine beliebige Zahl bezogen auf eine andere beliebige Zahl ausmacht.

Kapitel 4 : Einfaches Rechnen
Übungsaufgaben



Kapitel 5 : Größere Rechenprogramme

Abschnitt 1 : Vorbemerkung

In diesem Kapitel werden zusätzliche BASIC-Statements und wichtige der sog. BASIC-Funktionen vorgestellt, mit deren Hilfe es dann möglich ist, anspruchsvollere Aufgaben in Angriff zu nehmen.

Im Aufbau auch dieses Kapitels folgen wir dem Beispiel des vorangegangenen :

1. Vorstellung zusätzlicher BASIC-Elemente
2. ZWEITES TRAINING
3. Übungsaufgaben

Abschnitt 2 : Weitere BASIC-Anweisungen : Funktionen, Verzweigungen und Schleifen

Viele Rechenaufgaben, wie sie zum Beispiel im PRINT- oder im LET-Statement ausgeführt werden, können durch Funktionen erleichtert werden, auf die der Rechner zurückgreifen kann.

Die allgemeine Form eines solchen Funktionsaufrufs lautet:

```
Variablenname = Funktionsname(Argument)
```

An den konkreten Beispielen, die jetzt genannt werden, erklärt sich diese allgemeine Form sehr schnell :

Wichtige arithmetische Funktionen sind die folgenden :

Funktions-name	**Aufgabe der Funktion**
ABS	Diese Funktion liefert den Absolutwert des Wertes, der als Argument angegeben wird.
CLOG	Gibt den Zehnerlogarithmus des Arguments an.

Funktions-name	**Aufgabe der Funktion**
EXP	Gibt den Wert der Euler'schen Zahl e an, erhöht um die Potenz, die als Argument angegeben ist (e ist ca. 2.718...).
INT	Gibt die größte ganze Zahl an, die im Argument enthalten ist.
LOG	Gibt den natürlichen Logarithmus (Basis e) des Arguments an (Umkehrfunktion zur Funktion EXP).
RND	Gibt eine beliebige Zahl zwischen Ø bis kleiner 1 an (Zufallszahl). Es ist dabei gleichgültig, welcher Wert als Argument benutzt wird.
SQR	Gibt die positive Quadratwurzel eines positiven Wertes an, der als Argument eingegeben wird.

Interessant sind auch die folgenden beiden trigonometrischen Funktionen :

SIN	Gibt den Sinus des Argumente an, das in Einheiten des Kreisparameters anzugeben ist.
COS	Gibt den Cosinus an (Argument siehe oben).

Auf Beispiele zum Gebrauch dieser Funktionen soll hier verzichtet werden. Sie tauchen ohnehin in der nächsten Trainingssitzung auf.

Wenden wir uns nun den Möglichkeiten der Programmverzweigungen zu, die beim Programmieren komplizierterer Probleme außerordentlich wichtig sind. Man spricht in diesem Zusammenhang auch von Programmsprüngen und unterscheidet zwei Typen von Sprunganweisungen :

1. Unbedingter Sprung
2. Bedingter Sprung

Der unbedingte Sprung benötigt das folgende Statement :

Statement 5 :

```
nn GOTO mm
```

Wird dieses Statement bei der Programmabarbeitung erreicht, so erfolgt ein Programmsprung zum Satz mit der Nummer mm.

Das folgende sehr einfache Programmbeispiel erläutert die Verwendung dieses Statements :

```
10 PRINT "GUTEN TAG"
20 GOTO 10
```

Dies ist - wie wir beim Starten des Programms unschwer erkennen - ein sog. Endlosprogramm, weil es unendlich oft den Text

```
GUTEN TAG
```

untereinander ausgibt. Immer wenn das Programm diesen Text gedruckt hat, wird es durch das Statement 2Ø wieder zum Statement 1Ø geschickt und muß erneut drucken usw.

Solche Endlosprogramme sind natürlich nicht sehr sinnvoll (den sinnvolleren Gebrauch des Statements GOTO werden wir noch kennenlernen). Sollte uns aber doch einmal ein solches Programm "durchgegangen" sein, so hilft nur die "Notbremse", d.h. das Betätigen der BREAK-Taste (BREAK = Unterbrechung).

Ein Programm, das per BREAK-Taste unterbrochen wurde, kann durch das folgende Kommando an der Stelle, an der es unterbrochen.wurde, wieder in Gang gesetzt werden :

Kommando 4 :

```
CONT
```

CONT steht als Abkürzung für continue = fahre fort.

Der bedingte Sprung wird mit dem folgenden Statement ausgeführt :

Statement 6 :

```
nn IF logische Bedingung THEN Anweisung
```

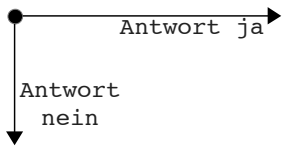
Entscheidend ist in diesem Statement die sog. logische Bedingung, die man auch als logischen Ausdruck oder einfach nur als eine Frage auffassen kann.

Beispiel : Logische Bedingung : 5>3
Frage : Ist 5 größer als 3?

Prinzipiell muß gelten :

Eine logische Bedingung kann sein :	erfüllt	nicht erfüllt
Dann ist der logische Ausdruck :	wahr	falsch
Die Antwort auf die entsprechende Frage lautet dann :	ja	nein

Ganz generell muß man sich die in nebenstehender Skizze angegebene Verzweigungsrichtung Merken :



Dies bedeutet :

Ist die logische Bedingung erfüllt (ist der logische Ausdruck wahr; lautet die Antwort "ja"), so wird die Anweisung ausgeführt, die hinter dem Schlüsselwort THEN (=dann) des IF...THEN-Statements steht.

Ist dies hingegen nicht der Fall (logische Bedingung nicht erfüllt, logischer Ausdruck falsch, Antwort "nein"), dann geht das Programm zum nächstfolgenden Satz über, in der Programmabarbeitung also nach unten.

Anzumerken ist, daß hinter dem THEN auch mehrere Anweisungen stehen können, die dann durch Doppelpunkte voneinander zu trennen sind. Alle diese Anweisungen werden hintereinander nur dann ausgeführt, wenn die logische Bedingung erfüllt ist.

Das folgende einfache Programmbeispiel illustriert die Verwendung dieses Statements in anschaulicher Weise :

```
10 I=1
20 Q=I*I:W=SQR(I)
30 PRINT I,Q,W
40 I=I+1
50 IF I<=15 THEN GOTO 20
60 PRINT :PRINT "ENDE":END
```

Dieses Programm bildet für alle ganzen Zahlen von 1 bis 15 die Quadratzahlen und die Quadratwurzeln und druckt dies aus. Der Programmabschnitt zwischen Satz 2Ø und Satz 5Ø wird so lange wiederholt wie die Variable I einen Wert aufweist, der kleiner als 16 ist. Erst wenn nach der Erhöhung von I um 1 in Satz 4Ø der Wert 16 erreicht wird, geht das Programm zum Satz 6Ø über und wird damit beendet.

Die Programmschritte zwischen Satz 2Ø und Satz 5Ø nennt man eine Schleife.

Mit den folgenden Statements kann eine solche Schleife noch einfacher und allgemeiner nutzbar programmiert werden :

Statement 7 :

```
nn FOR Laufvariablenname = Anfangswert TO
   Endwert (STEP Schrittweite)
```

Notwendig dazu gehört das folgende Statement :

Statement 8 :

```
mm NEXT Laufvariablenname
```

Wenn man mit Hilfe dieser beiden Statements die gleiche Aufgabe programmiert, wie sie mit dem vorangegangenen Programm vorgestellt wurde, so gelangt man zu dem folgenden Programm :

```
10 FOR I=1 TO 15
20 PRINT I,I*I,SQR(I)
30 NEXT I
40 PRINT :PRINT "ENDE":END
```

An diesem Beispiel sieht man sehr deutlich, wie diese beiden Statements zusammenwirken.

Zu erwähnen ist in diesem Zusammenhang, daß das FOR...TO-Statement mit einer Schrittweite (STEP) versehen werden kann, so daß dann die Laufvariable gemäß dieser Schrittweitenangabe verändert wird. Wird diese Angabe weggelassen (so wie in dem obigen Programm), dann ist die Schrittweite automatisch immer gleich 1.

Mit zwei zusätzlichen Statements wollen wir diesen Abschnitt beschließen :

Statement 9 :

```
nn STOP
```

Wird dieses Statement erreicht, so wird die Programmabarbeitung automatisch unterbrochen. Die Verwendung dieses STOP-Statements empfiehlt sich also vor allem dann, wenn größere Ergebnisausgaben erzeugt werden. Bevor die ersten Ergebnisse über den oberen Bildschirmrand verschwinden, weil unten neue Ergebnisse nachgeschoben werden, ist es sinnvoll,ein STOP-Statement einzubauen.

Die Fortsetzung des per STOP unterbrochenen Programms ist durch Eingabe des Kommandos CONT möglich.

Statement 1Ø :

```
nn REM Text
```

Dieses Statement, das wir in vielen der vorangegangenen Programme schon benutzt haben, dient dazu, Texte (also z. B. Erläuterungen, Überschriften, Zwischenüberschriften und dergl.) in ein Programm aufzunehmen. Immer wenn ein solches Programm gelistet wird, tauchen diese Texte wieder mit auf, ohne daß die REM-Statements einen Einfluß auf die Programmabarbeitung ausüben.

ZWEITES TRAINING

TRAININGSBEISPIEL 1 : Quadrate und Wurzeln

1. Vorstellung des Problems

In einem ersten Rechenbeispiel in dieser zweiten Trainingssitzung wollen wir uns noch einmal mit Quadratzahlen und Quadratwurzeln beschäftigen. Es soll ein BASIC-Programm entwickelt werden, welches für alle ganzen Zahlen von 1 bis 1ØØ die Quadratzahlen und die Quadratwurzeln ausgibt.

2. Problemanalyse

Die Problemanalyse bereitet hier keine Schwierigkeiten. Es ist lediglich bei der Ausgabe der Ergebnisse darauf zu achten, daß sie nicht zu rasch über den oberen Bildschirmrand hinaus verschwinden. Deshalb soll vorgesehen werden, daß das Programm nach je 15 Angaben sich selbst unterbricht.

Dies kann man mit dem STOP-Statement erreichen, wenn man dieses an die Erfüllung einer bestimmten logischen Bedin-

gung knüpft. Wie muß diese Bedingung aussehen ?

Nach jeder 15. Runde der zu entwickelnden Programmschleife soll gestoppt werden. Wenn wir die Schleifen-Laufvariable I nennen, so könnte das entsprechende IF...THEN-Statement lauten :

```
IF I=15 OR I=30 OR I=45 OR I=60 OR I=75
OR I=90 THEN STOP
```

Dies ist natürlich eine etwas umständliche Programmierung.

Man sieht übrigens an diesem (nicht empfehlenswerten) Beispiel des IF...THEN-Statements, daß in einem solchen Statement auch mehrere logische Einzelbedingungen zu einer logischen Gesamtbedingung verknüpft werden können. Der hier benutzte Verknüpfungsoperator lautet OR (=oder).

Weitere Verknüpfungsoperatoren sind AND (=und) und NOT (=nicht).

Im obigen Beispiel wäre die Gesamtbedingung erfüllt, wenn auch nur eine der Teilbedingungen erfüllt ist.

Die gewünschten Programmunterbrechungen lassen sich aber eleganter programmieren, wenn man die INT-Funktion benutzt : Das Programm soll ja dann unterbrochen werden, wenn I=15 oder I=3Ø oder I=45 usw. ist. Anders aus-gedrückt heißt dies, daß eine Unterbrechung dann gewünscht ist, wenn die Division von I durch 15 ohne Rest möglich ist.

Wenn dies der Fall ist, muß folgende Be-ziehung gelten :

I/15 = INT(I/15)

Nur wenn I den Wert 15, 3Ø, 45 ... usw. hat, dann ergibt die Division von I durch 15 den gleichen Wert: wie die auf dieses Divisionsergebnis angewandte INT-Funktion, die ja den größtmöglichen Wert, der in diesem Divisionsergebnis drinsteckt, aufsucht.

Deshalb können wir also programmieren :

```
IF I/15 = INT(I/15) THEN STOP
```

Damit der auch ungeübte Programmbenutzer weiß, daß er nach der erfolgten Programmunterbrechung zur Fortsetzung Programms CONT eingeben muß, versehen wir dieses Statement noch mit einer PRINT-Anweisung, aus der dieser Tatbestand klar wird.

3. Programm

```
10 REM T7-QUADRATE UND WURZELN
20 PRINT CHR$(125)
30 PRINT "              ZAHLENTAFEL2"
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET FUER ALLE"
80 PRINT "GANZE ZAHLEN VON 1 BIS 100 DIE QUA-"
90 PRINT "DRATZAHLEN UND DIE QUADRATWURZELN."
110 PRINT :PRINT :PRINT
115 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN"
116 STOP
120 PRINT CHR$(125)
125 REM TABELLENUEBERSCHRIFT
130 PRINT "ZAHL","QUADRAT","WURZEL":PRINT
140 FOR I=1 TO 100
150 Q=I*I:W=SQR(I)
160 W=INT(W*1000+0.5)/1000
170 PRINT I,Q,W
175 REM UNTERBRECHUNG DER AUSGABE
180 IF I/15=INT(I/15) THEN PRINT "BITTE CONT EINGEBEN":STOP
200 NEXT I
210 PRINT :PRINT :PRINT
220 PRINT "ENDE DER AUSGABE"
230 END
```

4. Variablenliste

I = Laufvariable für die benötigte Programmschleife
Q = Quadratzahl
W = Quadratwurzel

5. Programmbeschreibung

Satz 1Ø-11Ø	:	Überschrift, Erläuterungen, Leerzeilen
Satz 115	:	Mitteilung, daß zur Programmfortsetzung CONT einzugeben ist
Satz 116	:	Unterbrechung des Programms per STOP
Satz 12Ø-13Ø	:	Löschen des Bildschirms und Ausgabe einer Tabellenüberschrift
Satz 14Ø	:	Beginn der Rechen- und Ausgabeschleife
Satz 15Ø	:	Berechnung der jeweiligen Quadratzahl und der jeweiligen Quadratwurzel
Satz 16Ø	:	Aufrunden der Wurzel auf drei Dezimalstellen

Dies ist ein recht wichtiges Statement, das uns in dieser Form noch öfter begegnen wird :

Zunächst wird der Wert im Feld W mit 1666 multipliziert. Ist W zum Beispiel 1.41421356 (= $\sqrt{2}$), so ergibt sich :

1414.21356

Dann wird Ø.5 dazu addiert, also

1414.71356

Die Anwendung der INT-Funktion ergibt

1414

Dividieren wir dieses Ergebnis dann wieder durch 1ØØØ, so erhalten wir

1.414

d.h. wir haben eine korrekte Rundung (hier auf drei Dezimalstellen) durchgeführt.

Satz 17Ø : Ausgabe der drei Werte I, Q und W
Satz 175-18Ø : Unterbrechung des Programms nach jeder 15. Schleifenrunde, wie in der Problemanalyse ausführlich beschrieben
Satz 2ØØ : Ende der Programmschleife
Satz 2ØØ-23Ø : Beendigung des Programms

• 6. Ergebnisse

Wenn wir dieses Programm per Kommando RUN starten, so erhalten wir nach den Titelausdrucken die folgende Tabelle (Ausschnitt) :

ZAHL	QUADRAT	WURZEL
1	1	1
2	4	1.414
3	9	1.732
4	16	2
...	...	...
15	225	3.873

BITTE CONT EINGEBEN

STOPPED AT LINE 18Ø

TRAININGSBEISPIEL 2 : Großes Einmaleins

• 1. Vorstellung des Problems

Es soll ein Programm erstellt werden, welches in tabellarischer Form die Ergebnisse des sog. großen Einmaleins (von 1Ø*1Ø bis 17*17) ausgibt.

• 2. Problemanalyse

Auch hier treten bei der Problemanalyse keine besonderen Schwierigkeiten auf. Es ist lediglich darauf zu achten, daß die gewünschten Rechenergebnisse nicht alle untereinander, sondern zeilenweise auf dem Bildschirm ausgegeben werden. Dies kann man durch geeignete Verwendung des Semikolons erreichen (siehe Satz 16Ø des folgenden Programms).

3. Programm

```
10 REM T8-GROSSES EINMALEINS
20 PRINT CHR$(125)
30 PRINT "         GROSSES EINMALEINS."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET DAS GROSSE"
80 PRINT "   EINMALEINS VON 10*10 BIS 17*17"
110 PRINT :PRINT :PRINT
130 FOR I=10 TO 17
140 FOR J=10 TO 17
150 P=I*J
160 PRINT P;" ";
170 NEXT J
180 PRINT
190 NEXT I
210 PRINT :PRINT :PRINT
220 PRINT "ENDE DER AUSGABE"
230 END
```

4. Variablenliste

I = Laufindex (erster Faktor bei der Produktbildung)
J = Laufindex (zweiter Faktor)
P = Produkt

5. Programmbeschreibung

Satz 1Ø-11Ø : Überschrift, Erläuterungen, Leerzeilen

Satz 13Ø : Beginn der I-Schleife (äußere Schleife)

Satz 14Ø : Beginn der J-Schleife (innere Schleife)

Satz 15Ø : Berechnung des jeweiligen Produkts

Satz 16Ø : Ausgabe auf dem Bildschirm; dahinter ein Leerzeichen, damit der nächste Wert nicht nahtlos angehängt wird.

Dieses PRINT-Statement wird mit einem Semikolon abgeschlossen, damit das nächste PRINT-Statement (es ist das gleiche wieder, weil es im nächsten J-Schleifendurchlauf sofort wieder berührt wird) sein Ergebnis nicht in die neue Bildschirmzeile schickt.

Ein Komma oder ein Semikolon am Ende eines PRINT-Statements unterdrückt den automatischen Zeilenvorschub.

Satz 17Ø : Ende der inneren Schleife

Satz 18Ø : "Nacktes" PRINT-Statement, um den in Satz 16Ø unterbundenen Zeilenvorschub wieder in Gang zu setzen. Wenn nämlich der äußere Schleifenindex (I) erhöht wird (erster Faktor dann also 11), dann soll bei der Ausgabe eine neue Bildschirmzeile begonnen werden.

Satz 19Ø : Ende der äußeren Programmschleife
Satz 21Ø-23Ø : Beendigung des Programms

Aus dieser Programmbeschreibung wird deutlich, daß hier zwei geschachtelte Schleifen verwendet wurden : Die J-Schleife befindet sich innerhalb der I-Schleife. Generell gilt :

> Schleifen können geschachtelt werden. Der Anfang der inneren muß hinter dem Anfang der äußeren, ihr Ende vor dem Ende der äußeren liegen.

• 6. Ergebnisse

Nach den Titelausdrucken erzeugt dieses Programm eine Tabelle nach dem folgenden Muster (Ausschnitt) :

```
100 110 120 130 140 150 160 170
110 121 132 143 154 165 176 187

...

170 187 204 221 238 255 272 289


ENDE DER AUSGABE
```

TRAININGSBEISPIEL 3 : Hypothekentilgung

• 1. Vorstellung des Problems

Es soll ein BASIC-Programm entwickelt werden: welches ausrechnen kann, wie lange die Tilgung einer Hypothek dauert. Dieses Programm soll unterschiedliche Hypothekenbeträge und unterschiedliche Rückzahlungskonditionen gleichermaßen berücksichtigen können.

Zusätzlich soll der jeweilige Schuldenstand Jahr für Jahr ausgegeben werden.

• 2. Problemanalyse

Als Input-Informationen benötigt das Programm die folgenden Angaben :

- Aktueller Schuldenstand der Hypothek
- monatlicher Rückzahlungsbetrag (wir unterstellen dabei, daß monatlich ein konstanter Betrag zu zahlen ist, der Schuldzinsen und Tilgungsanteil umfaßt)
- Jährlicher Schuldzinssatz in Prozent (wir unterstellen dabei monatliche Zinsberechnung).

Wenn das Programm diese Input-Informationen zur Kenntnis genommen hat, dann kann als erstes der im ersten Monat des ersten Jahres anfallende Zins berechnet werden.

Zieht man diesen Zinsbetrag vom monatlichen Rückzahlungsbetrag ab, so bleibt der Betrag übrig, der zur Tilgung der Hypothek verwendet werden kann. Wird dieser Restbetrag null oder sogar negativ, so kann nichts getilgt werden, d.h. die Schuld wächst anstatt abzunehmen. Das Programm muß dann den Rechengang mit einer entsprechenden Meldung abbrechen, weil es andernfalls in eine Endlosschleife geraten würde.

Kann hingegen mit positivem Restbetrag getilgt werden, so sinkt die Schuld. Wenn sie den Wert null erreicht (oder rechnerisch unterschreitet), dann ist die Hypothek getilgt. Auch dann ist das Programm mit einer entsprechenden Meldung zu beenden.

Solange die Hypothek noch nicht getilgt ist: muß das Programm zur nächsten Monatsberechnung übergehen. Wenn 12 Monate vollendet sind, soll der jährliche Schuldenausdruck erfolgen und das Programm geht zum nächsten Jahr über, das wiederum mit dem Monat Nr. 1 zu beginnen hat ("geschachtelte Schleifen"!).

3. Programm

```
10 REM T9-HYPOTHEKENTILGUNG
20 PRINT CHR$(125)
30 PRINT "          HYPOTHEKENTILGUNG."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET DIE DAUER"
80 PRINT "     EINER HYPOTHEKENTILGUNG."
110 PRINT :PRINT :PRINT
120 PRINT "DAS PROGRAMM BENOETIGT DIE FOLGENDEN"
130 PRINT "INPUT-INFORMATIONEN :":PRINT
140 PRINT "     AKTUELLE SCHULD    : ";
150 INPUT S
160 PRINT "     JAEHRL.ZINSSATZ (%): ";
170 INPUT P
180 PRINT "     MONATLICHE RATE    : ";
190 INPUT B
200 PRINT CHR$(125)
210 PRINT "NACH...JAHREN","SCHULD":PRINT
220 J=1
230 M=1
240 Z=((P/100)*S)/12:T=B-Z
250 IF T<=0 THEN 400
260 S=INT((S-T)*100+0.5)/100
270 IF S<=6 THEN 310
280 M=M+1:IF M<13 THEN 240
290 PRINT J,,S
300 J=J+1:GOTO 230
310 PRINT :PRINT :PRINT
320 PRINT "TILGUNGSDAUER : ":PRINT
330 PRINT J-1;" JAHRE UND ";M-1;" MONATE."
340 GOTO 500
400 PRINT :PRINT :PRINT
410 PRINT "DIE KONDITIONEN SIND SO, DASS NICHTS"
420 PRINT "GETILGT WERDEN KANN."
430 PRINT :PRINT "BITTE NEUEINGABE":GOTO 110
500 PRINT :PRINT :PRINT "ENDE":END
```

• 4. Variablenliste

B = Monatlicher Rückzahlungsbetrag
J = Laufindex ("Jahresschleife")
M = Laufindex ("Monatsschleife")
P = Jährlicher Schuldzinssatz in Prozent
S = Aktueller Schuldenstand
T = Monatlicher Tilgungsbetrag
Z = Monatlicher Zinsbetrag

• 5. Programmbeschreibung

Satz 1Ø-11Ø : Überschrift, Erläuterungen, Leerzeilen
Satz 12Ø-19Ø : Anforderung der Input-Informationen
Satz 2ØØ-21Ø : Löschen des Bildschirms und Ausgabe einer Tabellenüberschrift
Satz 22Ø : Beginn der "Jahresschleife"
Satz 23Ø : Beginn der "Monatsschleife"
Satz 24Ø : Berechnung des monatlichen Zinsbetrags und der Monatstilgung
Satz 25Ø : Wenn der Tilgungsbetrag kleiner oder gleich null wird, erfolgt ein Programmsprung zum Satz 4ØØ
Satz 26Ø : Ist dies nicht der Fall, wird die neue Schuld berechnet und auf zwei Dezimalstellen gerundet
Satz 27Ø : Wenn diese neue, nun kleinere Schuld kleiner oder gleich null wird, geht es weiter bei Satz Nr. 31Ø

Satz 28Ø : Ist dies hingegen noch nicht der Fall, so wird der Monatslaufindex M um 1 erhöht;
solange dabei M kleiner als 13 geblieben ist, geht es zurück zum Satz 24Ø

Satz 29Ø : Hat hingegen M den Wert 13 erreicht, so ist ein Jahr vergangen und deshalb wird der aktuelle Schuldenstand ausgegeben

Satz 3ØØ : Der Jahresindex J wird um 1 erhöht und es erfolgt ein Rücksprung zum Satz 23Ø, d.h. die Monate werden wieder ab 1 gezählt.

Satz 31Ø-33Ø : Satz 31Ø wird nur erreicht, wenn die Schuld getilgt ist (siehe Satz 27Ø); dann ist die Tilgungsdauer auszugeben.

Der Leser überlege genau, warum wir von J und von M jeweils den Wert 1 abgezogen haben.

Satz 34Ø : Nach der Ausgabe Sprung zum Satz 5ØØ

Satz 4ØØ-43Ø : Der Satz 4ØØ wird nur erreicht, wenn die Rückzahlungskonditionen nicht in Ordnung sind (siehe Satz 25Ø);dann ist eine entsprechende Meldung auszugehen und es wird eine Neueingabe durch Sprung nach 11Ø verlangt

Satz 5ØØ : Beendigung des Programms.

• 6. Ergebnisse

Geben wir auf die Anforderungen des Rechners zum Beispiel die folgenden Input-Informationen ein :

- Aktueller Schuldenstand : 4ØØØØ
- Jährlicher Zinssatz in % : 5.5
- Monatliche Rate : 7ØØ

so. ergibt sich die folgende Tabelle :

NACH...JAHREN	SCHULD
1	33641.31
2	26923.93
3	19827.65
4	12331.Ø6
5	4411.63

TILGUNGSDAUER :

5 JAHRE UND 6 MONATE.

ENDE

TRAININGSBEISPIEL 4 : Primzahlenprüfung

• 1. Vorstellung des Problems

Es soll ein BASIC-Programm entwickelt werden, welches für eine beliebige einzugebende ganze Zahl prüft, ob sie eine Primzahl ist oder nicht.

• 2. Problemanalyse

Bekanntlich handelt es sich bei einer Primzahl um eine Zahl, die nur durch 1 oder durch sich selbst ohne Rest teilbar ist.

Um zu prüfen, ob eine beliebige Zahl eine Primzahl ist, muß sie durch alle Teiler, die größer als 1, aber kleiner als sie selbst sind, geteilt werden. Wenn bei diesen Divisionen ein Rest auftritt, bzw. wenn das Divisionsergebnis nicht ganzzahlig ist, muß durch den nächsten Teiler dividiert werden.

Erzielt man bei keiner dieser Divisionen ein Ergebnis, das ganzzahlig ist, also kein Ergebnis ohne Rest, dann ist schließlich festzustellen, daß der zu prüfende Ausgangswert eine Primzahl darstellt.

Der Teiler 2 kann dabei gesondert behandelt werden :
Ist die Zahl durch 2 ohne Rest teilbar, so handelt es

sich um eine gerade Zahl, die nie eine Primzahl sein kann.

Es brauchen übrigens nicht alle Teiler probiert zu werden, die kleiner als die zu prüfende Zahl sind: sondern es genügt, die Teilungsversuche bei der Quadratwurzel aus dieser Zahl zu beenden. Bei allen größeren Teilern kann kein Divisionsergebnis mehr ohne Rest auftreten (der Leser überlege sorgfältig, warum das so sein muß).

- 3. Programm

```
10 REM T10-PRIMZAHLENPRUEFUNG
20 PRINT CHR$(125)
30 PRINT "        PRIMZAHLENPRUEFUNG."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM PRUEFT, OB EINE BE-"
80 PRINT "LIEBIGE GANZE POSITIVE ZAHL EINE "
90 PRINT "    PRIMZAHL IST ODER NICHT."
100 DIM A$(1)
110 PRINT :PRINT :PRINT
120 PRINT "ZU PRUEFENDE ZAHL : ";:INPUT X
125 T=2
130 IF X/2=INT(X/2) THEN 200
140 FOR T=3 TO SQR(X)
150 IF X/T=INT(X/T) THEN 200
160 NEXT T
170 PRINT :PRINT :PRINT
180 PRINT X;" IST EINE PRIMZAHL."
190 GOTO 250
200 PRINT :PRINT :PRINT
210 PRINT X;" IST KEINE PRIMZAHL, SONDERN"
220 PRINT "DURCH ";T;" TEILBAR."
250 PRINT :PRINT :PRINT
260 PRINT "NOCH EINE RECHNUNG (J/N) ";
270 INPUT A$
280 IF A$="J" THEN PRINT CHR$(125):GOTO 120
290 PRINT :PRINT "ENDE":END
```

4. Variablenliste

A$ = Stringvariable für die Antworten J (für ja) oder N (für nein)
T = Teiler
X = Zu prüfende Zahl

5. Programmbeschreibung

Satz 1Ø-9Ø : Überschrift, Erläuterungen, Leerzeilen

Satz 1ØØ : Dimensionierung der Stringvariablen A$ (über das DIM-Statement zur Dimensionierung von Variablen wird später gesprochen)

Generell gilt für den ATARI-Rechner :

Für jede Stringvariable muß per DIM-Statement Platz freigehalten werden.

Da hier nur jeweils ein Symbol einzugeben ist (siehe Satz 27Ø), findet sich im entsprechenden DIM-Statement der Wert 1.

Satz 1ØØ-12Ø : Eingabe der zu prüfenden Zahl

Satz 13Ø	:	Wenn X durch 2 ohne Rest teilbar ist (dann nämlich ist X/2 = INT(X/2)), Sprung zum Satz 2ØØ
Satz 14Ø	:	Ist dies nicht der Fall, so beginnt die Divisionsschleife mit T=3 (sie endet, wie oben ausgeführt, bei $\sqrt{X}$)
Satz 15Ø	:	Ergibt die Division ein Ergebnis ohne Rest, Sprung zum Satz 2ØØ
Satz 16Ø	:	Ist dies nicht der Fall, so ist der nächste Teiler zu probieren
Satz 17Ø-18Ø	:	Sind alle Teiler ohne Erfolg ausprobiert worden (beim Erfolg hätte ja ein Sprung zum Satz 2ØØ erfolgen müssen; siehe Satz 15Ø), so ist auszugeben, daß es sich bei der Zahl im Feld X um eine Primzahl handelt
Satz 19Ø	:	Danach Sprung zum Satz 25Ø
Satz 2ØØ-24Ø	:	Satz 2ØØ wird nur erreicht, wenn eine Division ohne Rest möglich war; dann ist auszugeben, daß es sich um keine Primzahl handelt, weil die zu prüfende Zahl zumindest durch den gerade benutzten Teiler T teilbar ist ohne Rest
Satz 25Ø-26Ø	:	Das Programm fragt den Benutzer, ob er eine erneute Überprüfung wünscht. Wenn ja: soll er J eingeben, wenn nein ein N
Satz 27Ø	:	Anforderung von J oder N
Satz 28Ø	:	Hat der Benutzer ein J eingegeben, das im Feld A$ gespeichert wird, so wird

der Bildschirm gelöscht und es erfolgt ein Rücksprung zum Satz 12Ø, d.h. es wird erneut eine zu prüfende Zahl angefordert

Satz 29Ø : Steht im Feld A$ nicht der String J, d.h. wünscht der Programmbenutzer keine erneute Rechnung, so wird das Programm beendet.

• 6. Ergebnisse

Geben wir auf die Anforderung des Rechners als zu prüfende Zahl zum Beispiel den Wert 143 ein: so erhalten wir als Ergebnis :

```
143 IST KEINE PRIMZAHL, SONDERN
DURCH 11 TEILBAR.


NOCH EINE RECHNUNG (J/N) ?
```

TRAININGSBEISPIEL 5 : Mittelwertberechnung

1. Vorstellung des Problems

Es soll ein BASIC-Programm vorgestellt werden: welches aus einer beliebigen Anzahl von Zahlenwerten den Durchschnitt (arithmetisches Mittel) ausrechnet.

2. Problemanalyse

Unter problemanalytischen Gesichtspunkten ist festzustellen, daß zunächst dafür gesorgt werden muß, daß der Rechner weiß, wie viele Daten zu mitteln sind. Als erstes muß ihm also die Anzahl der Werte zur Kenntnis gegeben werden.

Danach sind die Werte selbst einzugehen, die sofort aufaddiert werden können.

Das arithmetische Mittel ist ja nichts anderes als die Summe von Werten geteilt durch deren Anzahl. Dieses Divisionsergebnis kann dann als Ergebnis des Programmlaufs ausgegeben werden.

3. Programm

```
10 REM T11-MITTELWERTBERECHNUNG
20 PRINT CHR$(125)
30 PRINT "        MITTELWERTBERECHNUNG."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET DAS ARITH-"
80 PRINT "METISCHE MITTEL (DURCHSCHNITT) AUS"
90 PRINT "BELIEBIG VIELEN EINZUGEBENDEN WERTEN."
100 FOR I=1 TO 5:PRINT :NEXT I
110 PRINT "WIEVIELE WERTE : ";:INPUT N
120 PRINT CHR$(125)
130 FOR I=1 TO N
140 PRINT I;".WERT : ";:INPUT X
150 S=S+X
160 NEXT I
170 AM=S/N
180 PRINT :PRINT :PRINT
190 PRINT "DURCHSCHNITT = ";AM
200 PRINT :PRINT "ENDE":END
```

4. Variablenliste

AM = Arithmetisches Mittel (Durchschnitt)
I = Laufindex
N = Anzahl der zu mittelnden Werte
S = Summe der zu mittelnden Werte
X = Zahlenwert, der in die Mittelwertberechnung eingeht.

• 5. Programmbeschreibung

Satz 1Ø-1ØØ	:	Überschrift, Erläuterungen, Leerzeilen
Satz 11Ø	:	Anforderung der Anzahl der zu mittelnden Werte
Satz 12Ø	:	Löschen des Bildschirms
Satz 13Ø	:	Beginn der Eingabe- und Additionsschleife
Satz 14Ø	:	Anforderung des I-ten Wertes
Satz 15Ø	:	Hinzu-Addition des gerade eingegebenen Wertes X zu der schon erreichten Summe im Feld 5 (Erhöhung von S um X)
Satz 16Ø	:	Beendigung der Eingabe- und Additionsschleife
Satz 17Ø	:	Berechnung des arithmetischen Mittels
Satz 18Ø-19Ø	:	Ausgabe des Ergebnisses
Satz 2ØØ	:	Beendigung des Programms.

• 6. Ergebnisse

Wenn wir dieses Programm starten, so meldet der Rechner nach den Titelausdrucken als erstes :

```
WIEVIELE WERTE ?
```

Gehen wir daraufhin zum Beispiel die Zahl 4 ein, so ergibt sich :

```
1.WERT : ?
```

Geben wir daraufhin eine Zahl ein, so wird ein zweiter Wert angefordert usw.; in diesem Beispiel würden insgesamt vier Werte angefordert.

Wenn dies die Zahlen 11, 13, 12 und 14 sind, so erhalten wir auf dem Bildschirm :

```
DURCHSCHNITT = 12.5
ENDE
```

Übungsaufgaben

1. Es soll ein BASIC-Programm geschrieben werden, welches für beliebige rechtwinklige Dreiecke die Hypotenuse berechnet.

2. Mit einem BASIC-Programm soll das sog. Gauß-Problem gelöst werden :

   Es geht dabei darum, die Summe aller ganzen Zahlen von 1 bis 1ØØ zu berechnen.

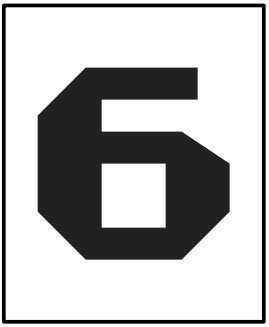
3. Zu bestimmen sind per BASIC-Programm beliebige Fakultäten :

   Unter einer Fakultät aus einer beliebigen Zahl x versteht man das Produkt aller ganzen Zahlen von 1 bis K, also :

   Fakultät von x = x! = x*(x-1)*(x-2)* ... *3*2*1

   Es ist dabei zu beachten, daß diese Produkte rasch sehr groß werden, deshalb muß für x größer als 33 das Programm abgebrochen werden, weil die Rechenkapazitäten nicht ausreichen.

4. Ein BASIC-Programm soll 1ØØ Würfelwürfe simulieren, den Durchschnitt all dieser Augenzahlen berechnen und eine Häufigkeitstabelle erstellen, aus der hervorgeht, wie oft die Eins, wie oft die Zwei ... usw., wie oft die Sechs gewürfelt wurde.



Kapitel 6 : Datenverarbeitung mit BASIC-Programmen

Abschnitt 1 : Vorbemerkung

In den bisher besprochenen Aufgabenstellungen wurden nur solche Rechenprobleme angesprochen, die vornehmlich aus dem schulischen Bereich stammen. Im kommerziellen Einsatz von Datenverarbeitungsanlagen findet sich hingegen häufig ein anderer Typ von Aufgabenstellungen, der sich vor allem dadurch auszeichnet, daß größere Datenmengen zu verwalten und zu verarbeiten sind.

Dabei sind zusätzliche Statements hilfreich, die wieder zuerst besprochen werden sollen.

Abschnitt 2 : Zusätzliche BASIC-Anweisungen : Das Lesen von Daten und Nutzung von Unterprogrammen

Die Datenbereitstellung kann in der Programmiersprache BASIC nicht nur über das LET- und über das INPUT-Statement erfolgen, sondern auch mit Hilfe der folgenden beiden Statements :

Statement 11 :

```
nn DATA Wert1, Wert2, Wert3, ...
```

Statement 12 :

```
mm READ Name1, Name2, Name3 ...
```

Diese beiden Statements gehören notwendigerweise zusammen.

Das DATA-Statement dient dazu, dem Rechner größere Datenbestände bereitzustellen. Reicht ein Satz für die Angabe

aller einzugebenden Daten nicht aus (sie werden; durch Kommata voneinander getrennt, hinter dem Schlüsselwort DATA aufgeführt), so können durchaus mehrere DATA-Statements aufeinander folgen.

Die mit dem DATA-Statement bereitgestellten Werte werden mit Hilfe des READ-Statements in den Rechner eingelesen.

Das folgende einfache Beispiel zeigt das Zusammenwirken dieser beiden Statements :

```
10 DATA 12,13,14
20 READ A,B,C
30 PRINT A,B,C
40 END
```

Ergebnis :

```
12        13        14
```

Man sieht an diesem Beispiel sehr deutlich, daß im READ-Statement genauso viele Variablen genannt werden müssen, wie im DATA-Statement Daten angegeben sind. Der erste Wert wird in das Feld mit dem ersten Namen eingelesen, der zweite Wert in das zweite Feld usw.

Der Versuch, mehr Daten zu lesen, als in den DATA-Statements vorhanden sind, führt zu Fehlermeldungen und Programmabbruch. Allerdings können weniger gelesen werden. Eventuell folgende READ-Statements fahren in den DATA-

Statements dann dort fort zu lesen, wo das vorhergehende READ-Statement aufgehört hat.

Beim Zusammenwirken von DATA- und READ-Statements ist auf die Gleichartigkeit von Variablen- und Werttypen zu achten : Zahlen werden in numerische Felder eingelesen, Zeichenketten (Strings) benötigen in den READ-Statements entsprechend Stringvariablen.

Bei manchen Problemstellungen ist es notwendig, daß auf einen einmal gelesenen Datenbestand erneut zugegriffen werden muß. Dann muß dafür gesorgt werden: daß ein folgendes READ-Statement wieder vorn anfangen kann zu lesen, d.h. der Datenbestand muB wieder "restauriert" werden. Dies leistet das folgende Statement :

Statement 13 :

```
nn RESTORE
```

Mit den bisher besprochenen Statements gerät man aber immer noch recht rasch in Schwierigkeiten, wenn man sehr große Datenbestände eingeben möchte, weil man dann im READ-Statement zu viele Variablennamen benötigen würde. Deshalb hat man das Konzept der indizierten Variablen eingeführt.

Bisher wurden nur Variablen benutzt, die nur je einen Wert repräsentieren konnten. Wurde dieser Wert geändert, so war der vorhergehende verloren.

Indizierte Variablen hingegen können unter einem Namen mehrere Werte speichern. Damit man diese verschiedenen Werte einzeln ansprechen kann, werden sie mit einem Index indiziert. Dieser Index wird in der Programmiersprache BASIC in Klammern hinter dem Variablennamen genannt.

Beispiel : Y(Ø), Y(1), Y(2), ... Y(99), Y(1ØØ)

Wenn man also beispielsweise bei einer Umfrage die Altersangaben von 4ØØ Personen erfaßt, so kann man diese in den indizierten Feldern A(Ø), A(1), ..., A(399) ablegen. Die Gesamtheit dieser 4ØØ Speicherplätze nennt man Array.

In diesem Beispiel ist dann A(17) die Altersangabe der 18. Person (natürlich könnte man der Übersichtlichkeit halber auch von 1 bis 4ØØ nummerieren und läßt dann einfach das Feld A(Ø) frei).

Der in Klammern stehende Index kann seinerseits eine Variable sein, z. B. die Variable I : A(I) ist dann allgemein die I-te Altersangabe (bzw. die I+1-te Angabe: falls die Zählung bei I-1 und nicht bei I=Ø beginnt).

Es ist auch möglich, Variablen mehrfach zu indizieren, zum Beispiel doppelt :

X(Ø,Ø)	X(Ø,1)	X(Ø,2)	X(Ø,3)
X(1,Ø)	X(1,1)	X(1,2)	X(1,3)
X(2,Ø)	X(2,1)	X(2,2)	X(2,3)

Dies ist eine doppelt indizierte Variable X, die abgekürzt z. B. X(I,J) geschrieben werden kann, wobei die Laufvariable I von Ø bis 2, die Laufvariable J von Ø bis 3 läuft (I ist in diesem Beispiel der sog. Zeilenindex, J der Spaltenindex).

Wenn man in einem BASIC-Programm indizierte Variablen benutzt, dann muß dem Rechner am Anfang des Programms mitgeteilt werden, wie groß die einzelnen Arrays sind, d.h. wieviele Speicherplätze er frei halten muß. Dies gelingt mit dem folgenden Statement :

Statement 14 :

```
nn DIM Name1(Wert1), Name2(Wert2), ...
```

Beispiele :

Das Statement DIM X(2Ø) hält für die Variable X 21 Plätze frei.

Das Statement DIM X(15),Y(2Ø) hält für die Variable X 16 und für die Variable Y 21 Plätze frei.

Das Statement DIM Z(3,4) hält für die Variable Z 4*5=2Ø Plätze frei usw.

Jetzt ist beispielsweise das Lesen großer Datenbestände sehr einfach, wie im folgenden schematisch gezeigt wird :

```
10 DIM X(1000)
20 DATA Wert1,Wert2,Wert3,...
.
.
.
.
500 FOR I = 1 TO 1000
510 READ X
520 X(I) = X
530 NEXT I
.
.
.
weitere Verarbeitung
```

Hier sind 1ØØØ Werte per DATA anzugeben (Fortsetzungs-Statements !)

Besondere Aufmerksamkeit ist in diesem Zusammenhang beim ATARI-Rechner den Stringvariablen zu widmen. Generell gilt die folgende Regel :

Jede benutzte Stringvariable muß zu Beginn des Programms dimensioniert werden.
Die Dimensionierungszahl bestimmt die Anzahl der Symbole, die in das betreffende Stringfeld aufgenommen werden können.

Auf das Einhalten dieser Regel kann nur dann verzichtet werden, wenn Strings direkt per PRINT ausgegeben werden.

Beispiel :

Das Statement DIM A$(1) ermöglicht, daß bei der folgenden Programmabarbeitung in das Feld A$ ein String von der Länge 1 aufgenommen werden kann, d.h. eine Zeichenkette, die aus nur einem Symbol besteht.

Entsprechend können wir beispielsweise einen Namen, der aus maximal 8 Zeichen besteht, eingeben, wenn wir dimensionieren : DIM N$(8). Versucht man, einen längeren String einzugehen, so wird dieser "abgeschnitten", d.h. es gehen Symbole verloren.

Eine Dimensionierung von Stringvariablen in dem Sinn, wie wir oben numerische Variablen dimensioniert haben, also Freihalten eines Arrays, in dem mehrere Werte gespeichert werden können, ist beim ATARI nicht möglich.

Es gibt Probleme, bei denen bestimmte Teilaufgaben innerhalb einer Programmabarbeitung mehrfach erledigt werden müssen (z. B. das Zeichnen waagrechter Striche bei der Erstellung längerer Tabellen, mehrfache Mittelwertberechnungen, wenn mehrere Variablen gleichzeitig zu verarbeiten sind u. ä.).

Wenn dies der Fall ist, empfiehlt es sich, den betreffenden Programmteil nur einmal zu programmieren, ihn aber

dann mehrfach zu benutzen.

Einen solchen Programmteil nennt man Unterprogramm (=Subroutine). Um vom jeweiligen Hauptprogramm in dieses Unterprogramm übergehen zu können, benötigen wir das folgende Statement :

Statement 15 :

```
nn GOSUB mm
```

Wird dieses Statement bei der Programmabarbeitunq erreicht, so wird die Programmabarbeitung in dem Unterprogramm fortgesetzt, das mit der Satznummer mm beginnt.

Ist dieses Unterprogramm abgearbeitet, so muß ein Rücksprung erfolgen, d.h. notwendigerweise muß am Ende eines Unterprogramms das folgende Statement stehen :

Statement 16 :

```
nn RETURN
```

Dieses Statement bewirkt den Rücksprung an das Statement des Hauptprogramms, das dem GOSUB-Statement folgt.

Das folgende kleine Beispiel zeigt recht anschaulich, wie ein solches Unterprogramm eingesetzt werden kann :

```
10 PRINT "DIESES PROGRAMM ZEIGT,"
20 GOSUB 500
30 PRINT "WIE MAN JEDE ZEILE"
40 GOSUB 500
50 PRINT "EINFACH UNTERSTREICHEN KANN."
60 GOSUB 500
70 PRINT :PRINT "ENDE" : END
500 REM UP STRICH
510 FOR I = 1 TO 28 : PRINT "-";: NEXT I : PRINT
520 RETURN
```

Das Ergebnis dieses Programms ist der folgende Ausdruck :

```
DIESES PROGRAMM ZEIGT,......
----------------------------
WIE MAN JEDE ZEILE
----------------------------
EINFACH UNTERSTREICHEN KANN.
----------------------------
```

Der Leser überlege sich, warum die Programmsprünge, die durch das GOSUB- und das RETURN-Statement im obigen Programm erzeugt werden, durch die Verwendung von GOTO-Statements nicht erledigt werden können.

DRITTES TRAINING

TRAININGSBEISPIEL 1 : Sortieren von Zahlen

1. Vorstellung des Problems

Es soll ein BASIC-Programm entwickelt werden, das einen beliebigen Datenbestand der Größe nach sortiert, in der Weise, daß am Ende der kleinste Wert zuerst und der größte Wert zuletzt genannt werden.

2. Problemanalyse

Das Sortieren von Datenbeständen ist eines der wichtig sten Aufgabengebiete der Datenverarbeitung. Wir stehen vor einer ganzen Reihe unterschiedlicher Sortieralgorithmen, die sich vor allem letztlich in der Sortiergeschwindigkeit voneinander unterscheiden.

Dieser Aspekt der Rechengeschwindigkeit soll hier nicht interessieren. Vielmehr wollen wir einen Sortieralgo-rithmus entwickeln, der besonders klar den wesentlichen Grundgedanken, dem jeder Sortierprozeß folgen muß, zum Ausdruck bringt.

"Sortieren" bedeutet, daß wir je zwei Zahlen aus einem gegebenen Datenbestand miteinander vergleichen :

Wir vergleichen die erste Zahl Schritt für Schritt mit allen anderen Zahlen. Ist dabei die erste Zahl größer als die zweite, so müssen die beiden Zahlen, die gerade miteinander verglichen werden, vertauscht werden.

Dadurch wird erreicht, daß dann, wenn die erste Zahl mit allen übrigen verglichen wurde (nach der ersten Vergleichsrunde also), an der ersten Stelle die Zahl steht, die insgesamt die kleinste ist.

Dann vergleichen wir die jetzt an zweiter Stelle stehende Zahl mit allen übrigen außer mit der ersten. Auch hier wird wieder getauscht, wenn dies erforderlich ist. Am Schluß der zweiten Vergleichsrunde haben wir dann die zweitkleinste Zahl gefunden.

Dann vergleichen wir die jetzt an dritter Stelle stehende Zahl mit allen übrigen, außer mit der ersten und der zweiten ... usw.

Es werden insgesamt so viele Vergleichsrunden durchlaufen, wie überhaupt Vergleiche möglich sind. Haben wir beispielsweise fünf Zahlen zu sortieren, so müssen vier derartige Runden durchlaufen werden : In der ersten Runde gibt es vier Vergleiche, in der zweiten Runde gibt es drei Vergleiche, in der dritten Runde noch zwei Vergleiche und in der vierten Runde ist nur noch ein Vergleich möglich.

Schematisch.sieht dies folgendermaßen aus :

1. Runde : Vergleich Feld 1 mit Feld 2
Vergleich Feld 1 mit Feld 3
Vergleich Feld 1 mit Feld 4
Vergleich Feld 1 mit Feld 5

2. Runde : Vergleich Feld 2 mit Feld 3
Vergleich Feld 2 mit Feld 4
Vergleich Feld 2 mit Feld 5

3. Runde : Vergleich Feld 3 mit Feld 4
Vergleich Feld 3 mit Feld 5

4. Runde : Vergleich Feld 4 mit Feld 5

Es ist dabei zu beachten, daß nach den ersten Vergleichen die Inhalte der Felder immer andere sein können, je nachdem ob und an welchen Stellen getauscht werden mußte.

Zur nochmaligen Verdeutlichung dieser sehr wichtigen Vorgehensweise soll die folgende Übersicht dienen, die von einem "Spieldatenbestand" ausgeht.

Dieser Datenbestand besteht aus den folgenden Zahlen :

6 3 5 9 7

Gehen wir nun das obige Schema von Arbeitsschritten entlang, so ergibt sich :

1. Runde :

Vergleich Feld 1 mit Feld 2 : Die Zahl 6 ist mit der Zahl 3 zu tauschen :

3 6 5 9 7

Vergleich Feld 1 mit Feld 3 : Kein Tausch von 3 und 5
Vergleich Feld 1 mit Feld 4 : Kein Tausch von 3 und 9
Vergleich Feld 1 mit Feld 5 : Kein Tausch von 3 und 7

2. Runde :

Vergleich Feld 2 mit Feld 3 : Tausch erforderlich :

3 5 6 9 7

Vergleich Feld 2 mit Feld 4 : Kein Tausch
Vergleich Feld 2 mit Feld 5 : Kein Tausch

3. Runde :

Vergleich Feld 3 mit Feld 4 : Kein Tausch
Vergleich Feld 3 mit Feld 5 : Kein Tausch

4. Runde :

Vergleich Feld 4 mit Feld 5 : Tausch erforderlich :

3 5 6 7 9

Damit sind alle Vergleiche durchgeführt und der Ausgangsdatenbestand liegt in der Tat sortiert vor.

Wie das Vergleichen und eventuelle Vertauschen unter programmlogischen Gesichtspunkten vor sich geht, soll nun kurz skizziert werden :

Offenbar müssen wir mit zwei geschachtelten Schleifen arbeiten : Eine "Rundenschleife", die bei 5 Zahlen von 1 bis 4 laufen muß, und eine "Vergleichsschleife".

Bezeichnen wir den Rundenschleifenindex mit L, so läuft dieser also bei fünf Zahlen von 1 bis 4 oder allgemein bei N Zahlen von 1 bis N-1.

Der Vergleichsschleifenindex (er soll mit R bezeichnet werden) läuft offenbar, wie aus dem Schema von Seite 136 deutlich hervorgeht, immer von L+1 bis N.

Verglichen werden dann jeweils die Werte X(L) und X(R). Wenn X(L) kleiner als X(R) ist, so braucht nicht getauscht zu werden, d.h. wir können zum nächsten Vergleich übergehen (NEXT R). Muß hingegen getauscht werden, so ist es notwendig, ein Hilfsfeld (Name H) einzurichten, in das der Wert X(L) hineingepackt wird. Das Feld X(L) kann dann mit dem Inhalt von X(R) belegt werden und das Feld X(R) kann sich seinen (neuen) Wert aus dem Hilfsfeld H holen.

Nach jeder abgeschlossenen Runde kann der Inhalt des Feldes X(L) ausgedruckt werden, weil dies dann der jeweils kleinste Wert ist. Nach der letzten Runde muß dann auch noch der allerletzte Wert ausgegeben werden.

- 3. Programm

```
10 REM T12-SORTIEREN
20 PRINT CHR$(125)
30 PRINT "         SORTIEREN VON ZAHLEN."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM SORTIERT EINEN VORGE-"
80 PRINT "         GEBENEN DATENBESTAND."
110 PRINT :PRINT :PRINT
120 PRINT "WIEVIELE DATEN ";:INPUT N
130 DIM X(N)
140 REM DATENEINGABE
150 PRINT CHR$(125)
160 FOR I=1 TO N
170 PRINT I;".WERT :  ";:INPUT X
180 X(I)=X:NEXT I
190 REM KONTROLLE
200 PRINT :PRINT :PRINT
210 PRINT CHR$(125)
220 PRINT "UNSORTIERTE DATEN":PRINT
230 PRINT "NR.","WERT":PRINT
240 FOR I=1 TO N
250 PRINT I,X(I)
260 NEXT I
270 GOSUB 1000:REM WARTEN
280 PRINT "SORTIERTE DATEN":PRINT
290 PRINT "NR.","WERT":PRINT
300 FOR L=1 TO N-1
310 FOR R=L+1 TO N
320 IF X(L)<=X(R) THEN 340
330 H=X(L):X(L)=X(R):X(R)=H
340 NEXT R
350 PRINT L,X(L)
360 NEXT L
370 PRINT L,X(L)
380 PRINT :PRINT "ENDE":END
1000 REM UP WARTEN
1010 PRINT :PRINT :PRINT
1020 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !"
1030 STOP
1040 PRINT CHR$(125)
1050 RETURN
```

4. Variablenliste

I = Laufindex
L = Laufindex
N = Anzahl der zu sortierenden Werte
R = Laufindex (Vergleichszähler innerhalb der Runden)
X = Zu sortierende Zahlen

5. Programmbeschreibung

Satz 1Ø-11Ø	:	Überschrift, Erläuterungen, Leerzeilen
Satz 12Ø	:	Anforderung der Anzahl der zu sortierenden Zahlen
Satz 13Ø	:	Dimensionierung des Zahlen-Arrays
Satz 14Ø-18Ø	:	Eingabe der zu sortierenden Zahlen über INPUT-Statements
Satz 19Ø-26Ø	:	Kontrollausgabe der unsortierten Zahlen
Satz 27Ø	:	Sprung ins Unterprogramm 1ØØØ zum Warten
Satz 28Ø-29Ø	:	Ausgabe einer Überschrift für den sortierten Datenbestand
Satz 3ØØ	:	Beginn der Rundenschleife
Satz 31Ø	:	Beginn der (inneren) Vergleichsschleife
Satz 32Ø	:	Wenn X(L) kleiner oder gleich X(R) ist, braucht nicht getauscht zu werden, deshalb weiter bei Satz 34Ø
Satz 33Ø	:	In diesem Satz wird der Inhalt von X(L) mit dem von X(R) ausgetauscht

Satz 34Ø : Ende der Vergleichsschleife
Satz 35Ø : Wenn die innere Schleife beendet ist, d.h. wenn eine Runde beendet ist, kann der Wert im Feld X(L) ausgegeben werden; es ist der im (Rest-)Datenbestand vorhandene kleinste Wert
Satz 36Ø : Ende der äußeren Schleife, d.h. Übergang zur nächsten Runde
Satz 37Ø : Ist auch diese Schleife abgearbeitet, muß noch der allerletzte Wert ausgegeben werden
Satz 38Ø : Beendigung des Hauptprogramms
Satz 1ØØØ : Beginn des Unterprogramms zum Warten
Satz 1Ø1Ø-1Ø2Ø : Ausgabe einer Meldung
Satz 1Ø3Ø : Programmunterbrechung mit dem STOP-statement
Satz 1Ø4Ø : Löschen des Bildschirms
Satz 1Ø5Ø : Rücksprung in das Hauptprogramm und Beendigung des Unterprogramms

• 6. Ergebnisse

Aufgrund der detaillierten Beschreibung dessen, was mit einem Spieldatenbestand geschieht (siehe 2.: Problemanalyse), dürfte die Darstellung von Programmergebnissen an dieser Stelle entbehrlich sein.

TRAININGSAUFGABE 2 : Mittelwerte

1. Vorstellung des Problems

In einer Betriebsstatistik finden sich Mengen- und Preisangaben für ein bestimmtes Produkt und zwar je 12 Werte für die Monate Januar bis Dezember eines bestimmten Jahres. Mit Hilfe eines BASIC-Programms sollen aus diesen Angaben der jahresdurchschnittliche Preis, die jahresdurchschnittliche Menge und der jahresdurchschnittliche Umsatz bestimmt werden.

2. Problemanalyse

Aus der Aufgabenstellung geht hervor, daß drei Mittelwerte (arithmetische Mittel) berechnet werden sollen. Es empfiehlt sich deshalb, diese Teilaufgabe nur einmal in einem Unterprogramm zu programmieren und dieses dann dreimal "anzuspringen".

Allerdings muß dabei berücksichtigt werden, daß die Variable, die im Unterprogramm benutzt wird (es handelt sich ja nur um eine Variable), neu belegt werden muß, bevor der zweite (...dritte) Sprung ins Unterprogramm erfolgt.

Die Daten selbst geben wir zweckmäßigerweise über DATA- und READ-Statements dem Rechner bekannt.

3. Programm

```
10 REM T13-MITTELWERTE
20 PRINT CHR$(125)
30 PRINT "   BERECHNUNG VON MITTELWERTEN."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET MEHRERE"
80 PRINT "MITTELWERTE AUS EINGEGEBENEN DATEN."
110 PRINT :PRINT :PRINT
115 N=12
120 DIM P(N),M(N),U(N)
130 DATA 5,6,4,4,5,7,6,4,4,6,7,5
140 DATA 10,11,12,10,12,13,11,14,11,10,11,12
150 FOR I=1 TO N:READ P
160 P(I)=P:NEXT I
170 GOSUB 1000:REM MITTELWERT
180 PRINT
190 PRINT "DURCHSCHNITTSPREIS    = ";AM
200 FOR I=1 TO N:READ M
210 M(I)=M
220 U(I)=P(I)*M(I):P(I)=M(I):NEXT I
230 GOSUB 1000:REM MITTELWERT
240 PRINT
250 PRINT "DURCHSCHNITTSMENGE    = ";AM
260 FOR I=1 TO N:P(I)=U(I):NEXT I
270 GOSUB 1000:REM MITTELWERT
280 PRINT
290 PRINT "DURCHSCHNITTSUMSATZ   = ";AM
300 PRINT :PRINT "ENDE":END

1000 REM UP MITTELWERT
1010 REM IM P-ARRAY SIND DIE ZU MITTELNDEN WERTE
1020 S=0
1030 FOR I=1 TO N:S=S+P(I):NEXT I
1040 AM=S/N
1050 AM=INT(AM*100+0.5)/100
1060 RETURN
```

4. Variablenliste

AM = Arithmetisches Mittel (Durchschnitt)
I = Laufindex
M = Mengenangaben
N = Anzahl der jeweils zu berücksichtigenden Werte
P = Preisangaben (gleichzeitig ist dies die Variable, die im Unterprogramm zur Mittelwertberechnung die zu mittelnden Werte generell enthält)
S = Summe der zu mittelnden Werte
U = Umsatzwerte (Umsatz = Menge * Preis)

5. Programmbeschreibung

Satz 10-110 : Überschrift, Erläuterungen, Leerzeilen
Satz 115 : Angabe der Zahl jeweils zu mittelnder Werte
Satz 120 : Dimensionierungen
Satz 130 : Bereitstellung von 12 Preisangaben
Satz 140 : Bereitstellung von 12 Mengenangaben
Satz 150-160 : Einlesen der Preise
Satz 170 : Sprung ins Unterprogramm 1000 zur Mittelwertberechnung

Satz 180-190	:	Ausgabe des ersten Mittelwerte (Durchschnittspreis)
Satz 200-210	:	Einlesen der 12 Mengenangaben
Satz 220	:	Berechnung der Monatsumsätze und Neubelegung des P-Arrays; Ende der Schleife
Satz 230	:	Sprung ins Unterprogramm 1000
Satz 240-250	:	Ausgabe des zweiten Mittelwerts (Durchschnittsmenge)
Satz 260	:	Neu-Belegung des P-Arrays
Satz 270	:	Sprung ins Unterprogramm 1000
Satz 280-290	:	Ausgabe des dritten Mittelwerte (Durchschnittsumsatz)
Satz 300	:	Beendigung des Hauptprogramms
Satz 1000	:	Beginn des Unterprogramms zur Mittelwertberechnung
Satz 1010	:	Erläuterung
Satz 1020	:	Belegung des Summenfeldes mit null, damit beim erneuten "Ansprung" nicht die vorher schon erreichte Summe noch "mitgeschleppt" wird
Satz 1040-1050	:	Berechnung des arithmetischen Mittels und Rundung auf zwei Dezimalstellen
Satz 1060	:	Rücksprung ins Hauptprogramm

• 6. Ergebnisse

Wenn wir dieses Programm mit dem RUN-Kommando starten, so erhalten wir nach den nun schon gewohnten Titelausgaben und Erläuterungen auf dem Bildschirm die folgenden Ergebnisse :

```
DURCHSCHNITTSPREIS  = 5.25

DURCHSCHNITTSMENGE  = 11.42

DURCHSCHNITTSUMSATZ = 59.83

ENDE
```

TRAINININGSAUFGABE 3 : Häufigkeitsverteilung

1. Vorstellung des Problems

Gegeben ist ein Datenbestand, der aus 20 in einer Umfrage erhobenen Körpergrößenangaben besteht. Über diese Daten soll eine Häufigkeitstabelle erstellt werden, die der Statistiker eine Häufigkeitsverteilung nennt.

Es ist leicht einzusehen, daß das BASIC-Programm, das hier erstellt werden soll, im Prinzip genauso funktioniert, wenn man anstatt 20 Körpergrößenangaben 200 oder 2000 Angaben hat.

2. Problemanalyse

Unter problemanalytischen Gesichtspunkten ist diese Aufgabenstellung wieder recht einfach : Die Körpergrößenangaben müssen bereitgestellt werden, ein Klassifizierungsraster ist vorzugeben (wir wählen in dem folgenden Programm 10-cm-Klassen, also 150 bis unter 160, 160 bis unter 170 usw.) und für jeden einzelnen Wert ist zu prüfen, in welche Klasse er hineingehört.

Die zuletzt genannte Teilaufgabe erledigen wir wieder mittels geschachtelter Schleifen : Die äußere Schleife

behandelt eine Körpergröße nach der anderen, die innere Schleife untersucht für jede dieser Körpergrößen alle Körpergrößenklassen daraufhin, ob die betrachtete Größe in die jeweilige Klasse hineinfällt. Ist dies der Fall, so muß ein Zähler um 1 erhöht werden und die innere Schleife kann vorzeitig abgebrochen werden, d.h. die nächste Körpergröße (äußere Schleife) kann betrachtet werden.

Das folgende Programm hat wieder eher Demonstrationscharakter, weil es bei konkreten Aufgabenstellungen - wie schon erwähnt wurde - erweitert werden muß.

3. Programm

```
10 REM T14-HAEUFIGKEITSVERTEILUNG
20 PRINT CHR$(125)
30 PRINT "        HAEUFIGKEITSVERTEILUNG."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM ERSTELLT EINE STATI-"
80 PRINT "  STISCHE HAEUFIGKEITSVERTEILUNG."
90 PRINT :PRINT :PRINT
100 PRINT "AUSGANGSDATEN SIND 20 KOERPERGROESSEN."
110 PRINT :PRINT :PRINT
115 N=20
120 DIM X(N),H(7)
130 FOR I=1 TO N:READ X
140 X(I)=X:NEXT I
150 GOSUB 1000:REM WARTEN
160 PRINT "KONTROLLE DER DATEN":PRINT
170 PRINT "NR.","WERT":PRINT
180 FOR I=1 TO N:PRINT I,X(I)
190 IF I/12=INT(I/12) THEN GOSUB 1000
200 NEXT I
210 REM HAEUFIGKEITSVERTEILUNG
220 GOSUB 1000:REM WARTEN
230 PRINT "KLASSE",,"ANZAHL"
240 FOR I=1 TO 26:PRINT "-";:NEXT I:PRINT
245 FOR K=1 TO 7:H(K)=0:NEXT K
250 FOR I=1 TO N
260 FOR K=1 TO 6
270 KE=K*10+140
280 IF X(I) < KE THEN H(K)=H(K)+1:GOTO 310
290 NEXT K
300 H(7)=H(7)+1
310 NEXT I
320 REM AUSGABE
330 FOR K=1 TO 7
340 KA=K*10+130
350 KE=KA+10
360 PRINT KA;" BIS UNTER ";KE,H(K)
370 NEXT K
380 PRINT :PRINT :PRINT "ENDE":END
```

```
500 DATA 172,188,175,178,192,168,173
510 DATA 184,171,165,198,156,177,181
520 DATA 174,179,181,166,168,173
1000 REM UP WARTEN
1010 PRINT :PRINT
1020 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !"
1030 STOP
1040 PRINT CHR$(125)
1050 RETURN
```

- 4. Variablenliste

H = Häufigkeiten
I = Laufindex (über alle Körpergrößen)
K = Laufindex (über alle Körpergrößenklassen)
KA = Klassenanfang (Untergrenze)
KE = Klassenende (Obergrenze)
N = Anzahl der Merkmalswerte (Körpergrößen)
X = Werte (Körpergrößen

• 5. Programmbeschreibung

Satz 10-110	:	Überschrift, Erläuterungen, Leerzeilen
Satz 115	:	Vorgabe der Anzahl der Werte
Satz 120	:	Dimensionierungen (N Werte, 7 Häufigkeiten)
Satz 130-140	:	Einlesen der Ausgangswerte
Satz 150	:	Sprung ins Unterprogramm 1000 zum Warten
Satz 160-200	:	Kontrollausdruck der eingegebenen Daten (Überschrift, Tabellenüberschrift, Ausdruck und Warten durch Sprung ins Unterprogramm 1000 nach jedem 12. Wert
Satz 210-220	:	Programmerläuterung und Warten (Unterprogramm 1000)
Satz 230	:	Tabellenüberschrift
Satz 240	:	Unterstreichung der Tabellenüberschrift
Satz 245	:	Belegung aller Häufigkeiten (Zählfelder) mit null
Satz 250	:	Beginn der "Körpergrößen-Schleife"
Satz 260	:	Beginn der "Klassen-Schleife"
Satz 270	:	Bestimmung des jeweiligen Klassenendpunktes (Klassenobergrenze)
Satz 280	:	Ist die jeweils betrachtete Körpergröße kleiner als der jeweilige Klassenendpunkt, so ist dieser Klassenzähler (Häufigkeit) um 1 zu erhöhen und die innere Schleife wird durch Sprung zum Satz 318 verlassen (nächste Körpergröße

Satz 29Ø : Ist dies nicht der Fall, so ist die nächste Klasse zu bearbeiten

Satz 3ØØ : Dieser Satz wird nur erreicht, wenn die innere Schleife "normal" verlassen wird, d.h. wenn kein Sprung zum Satz 31Ø im Satz 28Ø erfolgt war.

Dies ist nur dann der Fall, wenn die betrachtete Körpergröße in keine der ersten sechs Körpergrößenklassen hineingehört. In diesem Fall wird eine siebte "Restklasse" um 1 erhöht.

Satz 31Ø : Nächste Körpergröße

Satz 32Ø : Beginn der Ausgabe der Häufigkeitstabelle

Satz 33Ø : Beginn der Klassenschleife

Satz 34Ø-35Ø : Bestimmung von Klassenunter- und Klassenobergrenze

Satz 36Ø : Ausgabe der Klasse und der Klassenhäufigkeit

Satz 37Ø : Nächste Klasse

Satz 38Ø : Beendigung des Hauptprogramms

Satz 5ØØ-52Ø : Daten

Satz 1ØØØ-1Ø5Ø : Unterprogramm zum Warten (Detailbeschreibung siehe vorhergehendes Trainingsbeispiel)

• 6. Ergebnisse

Wenn wir das Programm mit diesem "Spieldatenbestand" starten, so erhalten wir nach dem üblichen Titelausdruck die folgenden Ergebnisse :

```
KONTROLLE DER DATEN

NR.        WERT

1          172
2          188
.           .
.           .
.           .
12         156

ZUR FORTSETZUNG BITTE CONT EINGEBEN !

STOPPED AT LINE 1030

.           .
.           .
.           .
```

Als Häufigkeitstabelle erhalten wir schließlich :

```
KLASSE                ANZAHL
-----------------------------
140 BIS UNTER 150     0
150 BIS UNTER 160     1
160 BIS UNTER 170     4
170 BIS UNTER 180     9
180 BIS UNTER 190     4
190 BIS UNTER 200     2
200 BIS UNTER 210     0

ENDE
```

TRAININGSAUFGABE 4 : Lotto

1. Vorstellung des Problems

Es soll ein BASIC-Programm entwickelt Werden, welches die Wahrscheinlichkeit dafür ausrechnet, im Lotto (6 aus 49) 6 Richtige zu erzielen.

2. Problemanalyse

Die Wahrscheinlichkeit für sechs Richtige im Lotto berechnet sich nach der folgenden Formel :

$$W = 1/\binom{49}{6}$$

Der Ausdruck $\binom{49}{6}$ (lies : 49 über 6) ist ein sog. Binomialkoeffizient, wie er sich in der Schulmathematik beim Rechnen mit sog. Binomen ergibt. Ein solcher Binomialkoeffizient kann folgendermaßen ausgerechnet werden :

$$\binom{49}{6} = \frac{49!}{43!*6!}$$

Die in dieser Berechnungsformel auftretenden Ausdrücke, die man Fakultäten nennt, sind uns an anderer Stelle

schon begegnet. Es gilt die folgende Berechnungsformel in allgemeiner Schreibweise :

$$n! = n*(n-1)*(n-2) \ldots 3*2*1$$

Also erhalten wir in diesem Fall :

$$\binom{49}{6} = \frac{49*48*\ldots*44*43*42\ldots*3*2*1}{6*5*4*3*2*1*43*42*\ldots*3*2*1}$$

$$= \frac{49*48*47*46*45*44}{6*\ 5*\ 4*\ 3*\ 2*\ 1}$$

Den gesuchten Binomialkoeffizienten erhalten wir also, indem wir zwei Produkte bilden, die aus je 6 Faktoren bestehen (Sechserschleife !). Aus beiden Produkten ist der Quotient zu bilden und der Kehrwert dieses Quotienten ist die gesuchte Wahrscheinlichkeit.

3. Programm

```
10 REM T15-LOTTO
20 PRINT CHR$(125)
30 PRINT "       6 RICHTIGE IM LOTTO."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET DIE WAHR-"
80 PRINT "SCHEINLICHKEIT FUER 6 RICHTIGE IM "
85 PRINT "        LOTTO (6 AUS 49)."
90 PRINT :PRINT :PRINT
100 B1=1:B2=1
110 P1=49:P2=6
120 FOR I=1 TO 5
130 P1=P1*(49-I)
140 P2=P2*(6-I)
150 NEXT I
160 B3=P1/P2
170 W=(B1*B2)/B3
180 PRINT "WAHRSCHEINLICHKEIT FUER 6 RICHTIGE IM"
190 PRINT "LOTTO : ":PRINT
200 PRINT "          ";W
210 PRINT :PRINT "ENDE":END
```

4. Variablenliste

B1 = Hilfsgröße für den Zähler des Binomialkoeffizienten (hier gleich 1; dieser Wert verändert sich, wenn es nicht um 6, sondern z.B. um 5, 4 oder 3 Richtige geht)

B2 = entsprechend wie B1

B3 = Binomialkoeffizient $\binom{49}{6}$ = Quotient aus den beiden Produkten P1 und P2

I = Laufindex
P1 = Produkt im Zähler des Binomialkoeffizienten
P2 = Produkt im Nenner
W = Wahrscheinlichkeit

• 5. Programmbeschreibung

Satz 10-90	:	Überschrift, Erläuterungen, Leerzeilen
Satz 100	:	Vorgabe von B1 und B2 (beide 1 bei dieser Problemstellung)
Satz 110	:	Vorgabe der Startwerte für die Produktschleife
Satz 120	:	Beginn der Schleife, in der die beiden Produkte bestimmt werden
Satz 130-140	:	Bestimmung der beiden Produkte
Satz 150	:	Ende der Produktschleife
Satz 160	:	Bestimmung der Größe B3, deren Kehrwert die gesuchte Wahrscheinlichkeit W ist
Satz 170	:	Berechnung der gesuchten Wahrscheinlichkeit
Satz 180-200	:	Ausgabe
Satz 210	:	Beendigung des Programms

• 6. Ergebnisse

Wenn wir dieses Programm starten, so erhalten wir die folgenden Bildschirmausgaben :

```
            6 RICHTIGE IM LOTTO

             PROF. DR. W. VOSS, 1984

     DIESES PROGRAMM BERECHNET DIE WAHR-
     SCHEINLICHKEIT FUER 6 RICHTIGE IM
             LOTTO (6 AUS 49).

     WAHRSCHEINLICHKEIT FUER 6 RICHTIGE IM
     LOTTO :

             7.15112385E-08

     ENDE
```

Es ergibt sich also der Wert 0.0000000715..., also eine sehr geringe Wahrscheinlichkeit (ungefähr 1 zu 14 Millionen).

Übungsaufgaben

1. Aus einem einzugebenden, vom Benutzer beliebig zu bestimmenden Datenbestand, soll per BASIC-Programm der kleinste und der größte Wert herausgesucht und ausgegeben werden.

2. Ein BASIC-Programm soll 188 Würfelwürfe simulieren und aus den sich ergebenden Augenzahlen den Durchschnitt (arithmetisches Mittel AM) und als Streuungsmaß die Standardabweichung (S) berechnen.

   Die Berechnungsformel für die Standardabweichung, die man auch die mittlere quadratische Abweichung nennt, lautet :

$$S = \sqrt{\frac{1}{N} \sum (x_i - AM)^2}$$

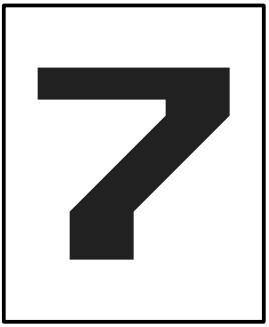
   Von allen Merkmalswerten ist also das arithmetische Mittel abzuziehen, die entstehenden Differenzen werden quadriert, diese Quadrate werden aufaddiert, die Summe wird durch die Anzahl der Werte dividiert und aus dem entstehenden Quotienten wird die Wurzel gezogen.

3. Gegeben sei ein Datenbestand, der Auskunft gibt über den Familienstand von beliebig vielen befragten Per-

sonen. Dabei gilt :

Ø = ledig
1 = verheiratet
2 = geschieden
3 = verwitwet

Per BASIC-Programm soll eine Häufigkeitsverteilung über diese Angaben erstellt werden (Häufigkeitstabelle).



Kapitel 7 : Grundelemente der Textverarbeitung

Abschnitt 1 : Vorbemerkung

In den bisherigen Trainingskapiteln wurden nur Rechenaufgaben behandelt. Wir haben aber schon einleitend darauf hingewiesen, daß Rechner wie zum Beispiel der ATARI nicht nur zum Rechnen geeignet sind, sondern auch mit Buchstaben, Worten und Texten, allgemein also mit Zeichenketten (Strings) umgehen können. Nicht zuletzt deshalb haben wir ja auch schon in den vorangegangenen Programmen mit Strings und Stringvariablen arbeiten können.

In diesem Kapitel soll nun gezeigt werden, was man mit Strings noch anfangen kann.

Abschnitt 2 : BASIC-Sprachelemente zur Behandlung von Strings

Auch in diesem Kapitel sollen zunächst diejenigen BASIC-Sprachelemente kurz vorgestellt werden, die wir im folgenden benötigen.

Ähnlich wie wir das Kapitel 5 mit einer Reihe von Funktionen eingeleitet haben (arithmetische und trigonometrische Funktionen) sind hier zunächst die sog. String-Funktionen zu besprechen, Funktionen also zur Be- und Verarbeitung gegebener Strings.

Einige dieser Funktionen, nämlich die ersten beiden in der folgenden Liste sind uns schon aus anderem Zusammenhang bekannt :

Funktions-name	**Aufgabe**
CHR$	Diese Funktion ermittelt das Symbol, welches einer bestimmten ASCII-Code-Zahl entspricht : PRINT CHR$(65) ergibt : A

Funktions-name	**Aufgabe**
ASC	Diese Funktion ermittelt zu einem gegebenen Symbol, das in Anführungszeichen eingeschlossen als Funktionsargument angegeben werden muß, die entsprechende ASCII-Code-Zahl : PRINT ASC("A") ergibt : 65
LEN	Diese Funktion ermittelt die Länge eines als Argument angegebenen Strings, d.h. die Anzahl der Symbole, aus denen er sich zusammensetzt (einschließlich eventueller Leerzeichen) PRINT LEN("ATARI IST GUT") ergibt : :13
STR$	Diese Funktion formt eine als Argument eingegebene Zahl in einen String um : STR$(123) = "123" Dies ist beispielsweise dann sinnvoll, wenn die Anzahl der Ziffern einer Zahl mit Hilfe der LEN-Funktion bestimmt werden soll, die ihrerseits als Argument ja einen String benötigt (s.o.) : PRINT LEN(STR$(789)) ergibt : 3

Funktions-name	**Aufgabe**
VAL	Diese Funktion schneidet von einem String, der als Funktionsargument vorzugehen ist, führende Ziffern ab, sofern solche vorhanden sind : PRINT VAL("46ØØ DORTMUND") ergibt : 46ØØ

Von besonderer Bedeutung ist beim ATARI die folgende Regel, auf die in anderem Zusammenhang schon einmal aufmerksam gemacht wurde :

> Jede in einem BASIC-Programm benutzte String-Variable muß zuvor dimensioniert werden.

Diese Dimensionierung muß sich auf die Maximalzahl der Symbole eines Strings beziehen, der unter dem betreffenden Stringvariablennamen gespeichert werden soll.

> Wird die Dimensionierung zu klein gewählt, so werden Symbole abgeschnitten und gehen verloren.

Wichtig ist auch die folgende Möglichkeit der Stringbehandlung, die gleich an einem Beispiel erläutert werden soll :

```
10 DIM A$(7)
20 LET A$="COMPUTER"
30 ? A$
40 ? A$(4,7)
50 END
```

Dieses kleine Programm erzeugt auf dem Bildschirm den folgenden Ergebnisausdruck :

```
COMPUTER
PUTE
```

Dies bedeutet, daß in dem Satz 40 aus dem eingegebenen String ("COMPUTER") ein Teil "herausgebrochen" und ausgegeben wird, nämlich der Teil vom vierten bis zum siebten Symbol (einschließlich dieser beiden). Dies ist der Teilstring "PUTE".

Auf diese Möglichkeit des Herausgreifens von Teilstrings aus einem gegebenen String werden wir bei den Trainingsbeispielen wieder zurückkommen.

VIERTES TRAINING

TRAININGSAUFGABE 1 : Übersetzung

1. Vorstellung des Problems

Es soll ein BASIC-Programm entwickelt werden, das quasi als Vokabelheft-Ersatz dient, weil es in der Lage ist, deutsch-französische oder französisch-deutsche Vokabelübersetzungen vorzunehmen.

Auf ein einzugebendes deutsches Wort soll das Programm also mit dem entsprechenden französischen Wort antworten und umgekehrt.

Derartige Übersetzungen sind natürlich nur dann möglich, wenn das Programm über die entsprechenden Wortpaare verfügt; sie müssen beispielsweise über DATA-Statements bereitgestellt werden.

Um das Programm, um das es hier geht, nicht zu groß werden zu lassen, beschränken wir uns auf die Vorgabe von 1Ø Vokabelpaaren. Insoweit hat das Programm also wieder nur demonstrativen Charakter. Wenn es konkret genutzt

werden soll, muß es erweitert werden, indem weitere DATA-Statements hinzugefügt werden. An den Funktionsprinzipien des Programms ändert sich dadurch aber natürlich nichts.

• 2. Problemanalyse

Bei dieser Problemstellung können wir uns relativ kurz fassen, auch wenn das Programm selbst recht lang wird :

Gibt der Benutzer ein deutsches Wort ein, so muß das Programm durch Vergleich mit allen ihm zur Verfügung stehenden deutschen und damit paarweise verbundenen französischen Worten das entsprechende deutsch-französische Wortpaar aus seinem Datenbestand heraussuchen und ausgeben. Entsprechend ist auch im umgekehrten Fall zu verfahren.

Damit beide Übersetzungsrichtungen in nur einem Programm bewältigt werden können, muß der Benutzer zunächst mitteilen, ob er deutsch-französische oder französisch-deutsche Übersetzung wünscht. Je nachdem, was eingegeben wird, werden unterschiedliche Programmteile angesprungen.

Weiterhin ist Vorsorge für den Fall zu treffen, daß die gewünschte Übersetzung durch das Programm nicht durchgeführt werden kann, weil das entsprechende Vokabelpaar im Datenbestand (noch) nicht vorhanden ist. Dann ist eine entsprechende Meldung auszugeben.

Schließlich soll das Programm den Benutzer die Gelegenheit bieten, bei Bedarf eine weitere Vokabel zu übersetzen.

- 3. Programm

```
10 REM T17-VOKABELN DEUTSCH-FRANZ.
20 PRINT CHR$(125)
30 PRINT "PROGRAMM ZUM NACHSCHLAGEN VON VOKABELN"
40 PRINT :PRINT "DEUTSCH-FRANZOESISCH ODER UMGEKEHRT."
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "IM DATENBESTAND DIESES PROGRRMMS BE-"
80 PRINT "FINDEN SICH NUR 10 VOKABELN.":PRINT
90 PRINT "SOLL DAS PROGRAMM AUSGEWEITET WERDEN,"
100 PRINT "SO MUESSEN IN 500 FF. WEITERE DATA-"
110 PRINT "STATEMENTS ANGEFUEGT UND DER WERT"
120 PRINT "FUER N IN SATZ 160 GEAENDERT WERDEN."
125 PRINT :PRINT :PRINT
130 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !":STOP
150 PRINT CHR$(125)
160 N=10
170 DIM A$(1),W$(15),D$(15),F$(15)
200 PRINT "WELCHE UEBERSETZUNO WIRD GEWUENSCHT ?"
205 PRINT :PRINT :PRINT
210 PRINT "     DEUTSCH-FRANZOESISCH  (1)"
220 PRINT :PRINT "     ODER":PRINT
230 PRINT "     FRANZOESISCH-DEUTSCH  (2)"
240 PRINT :PRINT "BITTE 1 ODER 2 EINGEBEN  :  ";
242 INPUT Z
245 PRINT CHR$(125)
247 RESTORE
250 IF Z=2 THEN 350
255 PRINT :PRINT
```

```
260 PRINT "DEUTSCHES WORT BITTE ";:INPUT W$
270 FOR I=1 TO N
280 READ D$,F$:IF D$=W$ THEN 320
290 NEXT I
295 PRINT :PRINT
300 PRINT "GESUCHTES WORT IM DATENBESTAND NICHT"
310 PRINT "VORHANDEN.":GOTO 450
320 PRINT CHR$(125)
325 PRINT "DEUTSCH          :  ";D$:PRINT
330 PRINT "FRANZOESISCH     :  ";F$
340 GOTO 450
350 PRINT :PRINT :PRINT
355 PRINT "FRANZOESISCHES WORT BITTE ";:INPUT W$
357 RESTORE
360 FOR I=1 TO N:READ D$,F$:IF F$=w$ THEN 400
370 NEXT I
375 PRINT :PRINT
380 PRINT "GESUCHTES WORT IM DATENBESTAND NICHT"
390 PRINT "VORHANDEN.":GOTO 450
400 PRINT CHR$(125)
405 PRINT "FRANZOESISCH   :  ";F$:PRINT
410 PRINT "DEUTSCH        :  ";D$
450 PRINT :PRINT :PRINT "NOCHMAL (J/N) ";:INPUT A$
460 IF A$="J" THEN PRINT CHR$(125):GOTO 200
470 PRINT :PRINT "ENDE DER AUSGABE":END
500 DATA KAUFEN,ACHETER
502 DATA HABEN,AVOIR
504 DATA GEHEN,ALLER
506 DATA SEHEN,VOIR
508 DATA TRINKEN,BOIRE
510 DATA MACHEN,FAIRE
512 DATA LESEN,LIRE
514 DATA ESSEN,MANGER
516 DATA NEHMEN,PRENDRE
518 DATA RAUCHEN,FUMER
```

• 4. Variablenliste

A$ = Antwortvariable für J/N (für Ja oder Nein)
D$ = Deutsche Worte
F$ = Französische Worte
I = Laufindex
N = Anzahl der im Datenbestand vorhandenen Vokabelpaare
W$ = Vom Benutzer gewünschtes Wort
Z = Variable, die die Übersetzungsrichtung bestimmt

• 5. Programmbeschreibung

Satz 1Ø-125	:	Überschrift, Erläuterungen, Leerzeilen
Satz 13Ø	:	Programmunterbrechung
Satz 15Ø	:	Löschen des Bildschirms
Satz 16Ø	:	Vorgabe der Anzahl der Vokabelpaare
Satz 17Ø	:	Dimensionierungen (für die Worte werden maximal 15 Symbole vereinbart; die Variable A$ braucht nur mit 1 dimensioniert zu werden, da sie nur J oder N zu speichern haben wird)
Satz 2ØØ-242	:	Anforderung der gewünschten Übersetzungsrichtung (für deutsch-französisch wird Z=1, umgekehrt wird Z=2)
Satz 245	:	Löschen des Bildschirms
Satz 247	:	Restaurieren des Datenbestandes :

Diese Restaurierung ist notwendig, damit bei einer eventuell gewünschten weiteren Übersetzung dem Programm der Ausgangsdatenbestand wieder zur Verfügung steht.

Wird nämlich eine solche erneute Übersetzung gewünscht (siehe Satz 45Ø), so springt das Programm zurück zum Satz 2ØØ und muß ab da natürlich wieder alle Vokabelpaare durchsuchen, d.h. Lesen können.

Satz 25Ø : Wird französisch-deutsche Übersetzung gewünscht (Z=2), so erfolgt ein Sprung zum Satz 35Ø

Satz 255-26Ø : Ist dies hingegen nicht der Fall, wird also deutsch-französische Übersetzung gewünscht, so wird der Benutzer (nach einigen Leerzeilen) gebeten, das gewünschte deutsche Wort einzugeben

Satz 27Ø : Beginn der Suchschleife

Satz 28Ø : Lesen eines Vokabelpaares; entspricht das gelesene deutsche Wort dem gewünschten deutschen Wort, so erfolgt ein Programmsprung zum Satz 32Ø

Satz 29Ø : Ist dies nicht der Fall, so wird das nächste Vokabelpaar gelesen

Satz 295 : Zwei Leerzeilen

Satz 3ØØ-31Ø : Der Satz 3ØØ wird nur dann erreicht, wenn die Such-/Leseschleife völlig abgearbeitet worden ist, ohne daß das gewünschte Wort gefunden worden wäre (dann wäre ja ein Sprung nach 32Ø erfolgt - siehe Satz 28Ø - und der Satz 3ØØ wäre gar nicht berührt worden)

Dies bedeutet, daß das gewünschte Wort im Datenbestand nicht vorhanden ist. Deshalb wird eine entsprechende Meldung ausgegeben und es erfolgt ein Sprung zum Satz 45Ø.

Satz 32Ø-33Ø : Satz 32Ø wird erreicht, wenn das gewünschte Wort gefunden wurde (siehe Satz 28Ø); deshalb können jetzt das gewünschte Wort und seine Übersetzung nach Löschen des Bildschirms ausgegeben werden.

Satz 34Ø : Sprung zum Satz 45Ø

Satz 35Ø-41Ø : Der Satz 35Ø wird nur dann erreicht, wenn eine französisch-deutsche Übersetzung gewünscht wird.

Es läuft dann im Prinzip der gleiche Prozeß ab wie er in Satz 255-34Ø oben beschrieben wurde. Details sind des-

halb jetzt entbehrlich.

Satz 45Ø : Abfrage, ob noch eine Übersetzung gewünscht wird
Satz 46Ø : Wenn ja, Löschen des Bildschirms und Rücksprung zum Satz 2ØØ, d.h. Erneute Vorgabe der Wahl der Übersetzungsrichtung
Satz 47Ø : Wenn nein, Beendigung des Programms
Satz 5ØØ-518 : Datenbestand (Vokabelpaare)

Es ist leicht einzusehen, daß dieses Programm in entsprechender Weise zum Beispiel auch für deutsch-englisch oder für italienisch-englisch oder dergl. verwendet werden kann, wenn der Datenbestand in den Statements 5ØØ ff. entsprechend verändert wird.

• 6. Ergebnisse

Auf die Ausgabe von Programmergebnissen kann angesichts dieser sehr einfachen Problemstellung Sicherlich verzichtet werden. Am sinnvollsten ist es, der Leser erprobt das Programm, um zu sehen, was auf dem Bildschirm erscheint.

TRAINININGSBEISPIEL 2 : Aufsuchen eines Buchstabens

1. Vorstellung des Problems

Es soll ein BASIC-Programm entwickelt werden, welches untersucht, ob ein bestimmter, vom Benutzer vorzugebender Buchstabe in einem einzugebenden String vorkommt oder nicht. Wenn dieser gesuchte Buchstabe auftauchen sollte, soll auch ausgezählt werden, wie oft er in dem untersuchten String auftritt.

Diese Aufgabenstellung kann als ein erster Schritt für die Entwicklung solcher Programme interpretiert werden, die mit dem buchstabenweisen Zerlegen von Strings etwa zum Zwecke des Sortierens von Worten oder dergl. zu tun haben.

2. Problemanalyse

Die Problemanalyse bei dieser Aufgabenstellung ist nicht allzu schwierig :

Zunächst sind der zu untersuchende String und der Buchstabe, für den sich der Benutzer interessiert, vorzugeben.

Der entscheidende Schritt in diesem Programm besteht

darin, aus dem eingegebenen String nun einen Buchstaben nach dem anderen herauszugreifen, um ihn mit dem gesuchten Buchstaben.jeweils vergleichen zu können. Zu diesem Zweck benötigen wir eine Programmschleife, die so viele "Runden" aufweist, wie der zu prüfende String Symbole hat.

Wir erinnern in diesem Zusammenhang daran, daß auch das Leerzeichen ein Symbol darstellt und würde man beispielsweise mit diesem Programm gerade auf Leerzeichen hin untersuchen, so können auf diese Weise etwa Sätze in Worte zerlegt werden, weil ja Wortende des einen und Wortanfang des anderen Wortes üblicherweise durch ein Leerzeichen getrennt sind.

Die oben angesprochene Programmschleife muß also laufen von 1 bis zur Länge des zu untersuchenden Strings, also bis LEN(String). Das Herausgreifen je eines Symbols, das ja nichts anderes als einen (sehr kurzen) Teilstring darstellt, gelingt dann, indem wir einer weiteren Stringvariablen die Größe String(I,I) zuweisen (bzw. diese direkt benutzen), wobei I der Laufindex der Programmschleife sein soll :

In der ersten Runde wird also ein Teilstring bestimmt, der von Symbol 1 bis Symbol 1 reicht (dies ist eindeutig das erste Symbol), in der zweiten Runde reicht der herausgegriffene Teilstring von Symbol 2 bis Symbol 2 (das ist der zweite Buchstabe) usw.

Dieses jeweils herausgegriffene Symbol wird mit dem Suchsymbol verglichen. Stimmen die beiden überein, so ist ein Zählindex um 1 zu erhöhen; ist dies hingegen nicht der Fall,kann das nächste Symbol untersucht werden.

Das Programm wird mit der Ergebnisausgabe und der Möglichkeit, einen weiteren Programmlauf durchzuführen, abgeschlossen.

3. Programm

```
10 REM T18-BUCHSTABE
20 PRINT CHR$(125)
30 PRINT "         BUCHSTABENSUCHE."
40 PRINT :PRINT :PRINT
50 PRINT "PROF.DR.W.VOSS,1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM DIENT DAZU, HERAUSZU-"
80 PRINT "FINDEN, OB EIN BELIEBIGER EINZUGEBEN-"
90 PRINT "DER BUCHSTABE IN EINEM STRING VORHAN-"
100 PRINT "         DEN IST UND WIE OFT."
110 DIM S$(200),B$(1),A$(1)
120 PRINT :PRINT :PRINT
130 PRINT "BITTE DEN ZU PRUEFENDEN STRING :"
140 PRINT :PRINT :INPUT S$
150 PRINT :PRINT
160 PRINT "WELCHER BUCHSTABE INTERESSIERT : ";
170 INPUT B$
180 Z=0
190 FOR I=1 TO LEN(S$)
200 IF S$(I,I)=B$ THEN Z=Z+1
210 NEXT I
220 PRINT :PRINT :PRINT
230 PRINT "DER BUCHSTABE ";B$;" KOMMT IN DEM"
240 PRINT "VORGEGEBENEN STRING ";Z;" MAL VOR."
250 PRINT :PRINT :PRINT
260 PRINT "NOCHMAL (J/N) ";:INPUT A$
270 IF A$="J" THEN PRINT CHR$(125):GOTO 130
280 PRINT :PRINT "ENDE":END
```

• 4. Variablenliste

A$ = Antwortstring (Ja/Nein, bzw. J/N)
B$ = Stringvariable, die das Symbol aufnimmt, das gesucht werden soll
I = Laufindex
S$ = Zu untersuchender String
Z = Variable

• 5. Programmbeschreibung

Satz 1Ø-1ØØ : Überschrift, Erläuterungen, Leerzeilen
Satz 11Ø : Dimensionierung der verwendeten Stringvariablen
Satz 12Ø : Drei Leerzeilen
Satz 13Ø-17Ø : Eingabe der Input-Informationen : Zu untersuchender String und gewünschtes Symbol
Satz 18Ø : Belegung der Zählvariablen mit null
Satz 19Ø : Beginn der Suchschleife
Satz 2ØØ : Herausgreifen eines Symbols und Vergleich mit dem gewünschten Symbol; stimmen die beiden überein: so ist der Zähler Z um 1 zu erhöhen
Satz 21Ø : Beendigung der Suchschleife
Satz 22Ø-25Ø : Ausgabe der Ergebnisse
Satz 26Ø-27Ø : Abfrage: ob ein weiterer Programmlauf gewünscht wird;

wenn ja, Löschen des Bildschirms und Rücksprung zum Satz 13Ø

Satz 28Ø : Wird kein erneuter Programmlauf gewünscht, so wird das Programm beendet.

6. Ergebnisse

Starten wir dieses Programm mit RUN, so erscheint nach dem üblichen Titelausdruck die Anforderung des zu untersuchenden Strings. Wenn wir nun beispielsweise eingeben:

```
HEUTE IST SCHLECHTES WETTER
```

so fordert das Programm den zu suchenden Buchstaben an.

Geben wir beispielsweise ein E ein, so erhalten wir das folgende Ergebnis :

```
DER BUCHSTABE E KOMMT IN DEM
VORGEGEBENEN STRING 6 MAL VOR.

NOCHMAL (J/N) ?
```

TRAINININGSBEISPIEL 3 : Vokabeltest

• 1. Vorstellung des Problems

Ähnlich wie in dem ersten Programmbeispiel dieser Trainingssitzung sollen in dem folgenden Programm wieder Übersetzungen bereitgestellt werden - ausgehend von einem bestimmten Vokabelvorrat. Im Gegensatz aber zu dem obigen Programm soll nun der Benutzer durch das Programm aufgefordert Werden, zu einem vom Rechner zufällig ausgewählten deutschen Wort, das korrekte englische Wort einzusetzen oder umgekehrt.

Es handelt sich hier also gewissermaßen um ein Abfrage- oder Testprogramm, das dazu benutzt werden kann: Vokabeln zu lernen. Auch dabei gilt wieder der Hinweis: daß ein solches Programm für deutsch-englische Abfragen (oder umgekehrt) natürlich auch für andere Sprachen verwendet werden kann, wenn die DATA-Statements: in denen der benutzte Vokabelvorrat steht, verändert werden.

Im Interesse der Reduzierung des Programmieraufwands beschränken wir uns hier wieder auf die exemplarische Darstellung anhand von nur zehn Vokabelpaaren. Bei der konkreten Programmbenutzung empfiehlt es sich also wieder, weitere Vokabelpaare zunächst vorzugeben.

• 2. Problemanalyse

Auch wenn das Programm recht umfangreich wird: ist die Problemanalyse doch wieder vergleichsweise einfach :

Wir lassen den Benutzer zunächst über eine geeignete Abfrage entscheiden, ob er einen deutsch-englischen oder einen englisch-deutschen Vokabeltest wünscht. Weiterhin lassen wir ihn darüber entscheiden, wievielen Abfragen er sich unterziehen lassen möchte, wieviele Vokabeln also in dem bevorstehenden Programmlauf getestet werden sollen.

Über die Zufallsfunktion RND suchen wir dann aus dem gegebenen Vokabelvorrat ein deutsches (englisches) Wort heraus und lassen den Benutzer das entsprechende englische (deutsche) Wort eingeben.

Ist die Antwort des Benutzers richtig, so wird er vom Programm gelobt und ein Zählindex, der am Schluß die insgesamt erreichte Anzahl richtiger Antworten ausgibt, wird um 1 erhöht. Ist die Antwort des Benutzers hingegen falsch, so wird dies vom Programm mitgeteilt und zudem wird die korrekte Antwort ausgegeben.

Die Programmierung solcher Schritte ist sehr einfach, weil ja nur das vom Benutzer als Antwort eingegebene Wort mit dem "Partnerwort" desjenigen Vokabelpaares, von dem das Programm ein Wort vorgegeben hat, verglichen werden muß.

Wird bei dem Vergleich Identität festgestellt, so hat der Benutzer richtig geantwortet: andernfalls eben nicht.

Wenn diese eine Vokabelüberprüfung erledigt ist, so folgt die nächste Zufallsauswahl (je nachdem, wieviele Abfragen der Benutzer gewünscht hat). Dies erfordert, daß das Programm den Datenbestand, also die Vokabelpaare erneut liest, und dies wiederum erfordert das Restaurieren des Datenbestands mit dem RESTORE-Statement.

Ist die gewünschte Zahl von Tests durchgeführt worden, so kann der prozentuale Anteil korrekter Antworten ausgerechnet und ausgegeben werden und das Programm wird daraufhin beendet.

3. Programm

```
10 REM T19-VOKABELTEST
20 PRINT CHR$(125)
30 PRINT "PROGRAMM ZUM ABFRAGEN VON VOKABELN."
35 PRINT
40 PRINT "HIER : "
45 PRINT "DEUTSCH/ENGLISCH ODER UMGEKEHRT"
47 PRINT :PRINT
50 PRINT "PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT
70 PRINT "IM DATENBESTAND DIESES PROGRAMMS BE-"
80 PRINT "FINDEN SICH NUR 10 VOKABELN.":PRINT
90 PRINT "SOLL DAS PROGRAMM AUSGEWEITET WERDEN,"
100 PRINT "SO MUESSEN IN 1000 FF. WEITERE DATA-"
110 PRINT "STATEMENTS ANGEFUEGT UND DER WERT"
120 PRINT "FUER N IN STATEMENT 160 GEAENDERT"
125 PRINT "WERDEN."
130 PRINT :PRINT :PRINT
140 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN":STOP
150 PRINT CHR$(125)
160 N=10
170 DIM D$(15),E$(15),W$(15),H$(15)
185 ZZ=1
200 PRINT "DEUTSCH/ENGLISCH    (1)"
210 PRINT :PRINT :PRINT "ODER"
215 PRINT :PRINT
220 PRINT "ENGLISCH/DEUTSCH    (2)"
225 PRINT :PRINT
230 PRINT "BITTE 1 ODER 2 EINGEBEN ";:INPUT Z
247 PRINT :PRINT :PRINT
250 PRINT "WIEVIELE VOKABELN SOLLEN GEPRUEFT"
260 PRINT "WERDEN ";:INPUT A
270 GOSUB 2000:REM PRUEFUNG
420 PRINT :PRINT "ENDE DER AUSGABE":END
```

```
1000 DATA LAUFEN,RUN,SCHREIBEN,WRITE
1010 DATA DRUCKEN,PRINT,GEHEN,GO
1020 DATA WENN,IF,UNTERPROGRAMM,SUBROUTINE
1030 DATA RECHNER,COMPUTER,ZEICHEN,CHARACTER
1040 DATA BILDSCHIRM,SCREEN,TASTE,KEY
2000 PRINT CHR$(125)
2002 RESTORE
2005 K=0
2010 I=1
2020 R=INT(RND(1)*N+1)
2022 FOR J=1 TO R:READ D$,E$:NEXT J
2025 IF Z<>ZZ THEN GOSUB 3000:REM TAUSCH
2030 ? D$
2035 PRINT :PRINT "ANTWORT : ":PRINT
2037 INPUT W$
2038 PRINT :PRINT :PRINT
2040 IF W$=E$ THEN K=K+1:PRINT "SEHR GUT !":GOTO 2070
2045 PRINT :PRINT :PRINT
2050 PRINT "LEIDER FALSCH !":PRINT
2060 PRINT "RICHTIG MUSS ES HEISSEN : ";E$
2070 PRINT :PRINT :PRINT :I=I+1
2080 IF I<=A THEN RESTORE :GOTO 2020
2090 KA=(K/A)*100:KA=INT(KA*100+0.5)/100:PRINT
2100 PRINT "ANTEIL KORREKTER ANTWORTEN : ";KA;" %"
2110 RETURN
3000 REM UP TAUSCH
3010 H$=D$:D$=E$:E$=H$
3020 RETURN
```

• 4. Variablenliste

A = Anzahl gewünschter Abfragen
D$ = Deutsche Worte
E$ = Englische Worte
H$ = Hilfsfeld, das für den eventuellen Tausch von deutschem mit englischem Wort benötigt wird
I = Laufindex (Nummerierung der Abfragen)
J = Laufindex (Einleseschleife)
K = Zählindex für korrekte Antworten
KA = Prozentualer Anteil korrekter Antworten
N = Anzahl der im Datenbestand vorhandenen Vokabelpaare
W$ = Vom Benutzer eingegebene Antwort
Z = Menüvariable, die die Übersetzungsrichtung bestimmt
ZZ = Hilfsvariable, die gegebenenfalls den Vokabelaustausch veranlaßt, wenn die Übersetzungsrichtung vom Benutzer geändert wird.

• 5. Programmbeschreibung

Satz 10-130	:	Überschrift, Erläuterungen, Leerzeilen
Satz 140	:	Programmunterbrechung
Satz 150	:	Drei Leerzeilen
Satz 160	:	Angabe der Zahl der Vokabelpaare
Satz 170	:	Dimensionierungen (es wird dabei unterstellt, daß die einzelnen Worte maximal 15 Symbole umfassen
Satz 185	:	Vorgabe der Hilfsgröße ZZ (vergleiche Satz 2025)
Satz 200-247	:	Erfragen der Übersetzungsrichtung; bei deutsch-englisch wird Z mit 1, umgekehrt wird Z mit 2 belegt.
Satz 250-260	:	Abfrage, wieviele Vokabeln getestet werden sollen; diese Anzahl wird in A gespeichert
Satz 270	:	Sprung ins Unterprogramm 2000, in dem der eigentliche Vokabeltest durchgeführt wird
Satz 420	:	Beendigung des Programms
Satz 1000-1040	:	Datenbestand (Vokabelpaare)
Satz 2000	:	Beginn des Unterprogramme, in dem die Vokabeln getestet werden, mit Löschen des Bildschirms
Satz 2002	:	Restaurieren des Datenbestands (dies ist notwendig, damit bei der zweiten

und allen eventuell noch folgenden Abfragen der Datenbestand immer wieder von vorn zur zufälligen Auswahl eines Vokabelpaares gelesen werden kann)

Satz 2005 : Belegung der Variablen K mit dem Wert null (Zahl der korrekten Antworten)

Satz 2010 : Beginn der ersten (der I-ten) Abfrage (I=1)

Satz 2020 : Bestimmung einer Zufallszahl R zwischen 1 und N (N ist die Anzahl der vorhandenen Vokabelpaare; siehe Satz 160)

Satz 2022 : Lesen aller Vokabelpaare bis zum zufällig ausgewählten Paar (das R-te Paar); in den Feldern D$ und E$ steht dann das R-te Vokabelpaar

Satz 2025 : Wenn Z ungleich ZZ ist, dann wünscht der Benutzer die andere Übersetzungsrichtung (vergleiche Sätze 185 und 240); deshalb erfolgt dann ein Sprung in das Unterprogramm 3000, in dem der Inhalt von D$ mit dem von E$ getauscht wird

Satz 2030 : Das deutsche Wort, welches das Programm zufällig ausgewählt hatte: wird ausgegeben (bzw. das englische Wort: falls der Sprung ins Unterprogramm 3000 erfolgt war, also getauscht wurde - siehe Satz 2025)

Satz 2Ø35-2Ø37 : Anforderung der Antwort des Benutzers, die im Feld W$ gespeichert wird

Satz 2Ø38-2Ø4Ø : War die Antwort des Benutzers korrekt, so muß W$ mit E$ übereinstimmen und das Programm meldet

SEHR GUT !

und erhöht zudem den Zählindex K (Anzahl korrekter Antworten) um 1; danach erfolgt ein Sprung zum Satz 2Ø7Ø

Satz 2Ø45-2Ø5Ø : War die Antwort hingegen nicht korrekt, so meldet der Rechner :

LEIDER FALSCH !

Satz 2Ø6Ø : Weiterhin gibt das Programm aus :

RICHTIG MUSS ES HEISSEN :

und präsentiert dahinter die korrekte Antwort

Satz 2Ø7Ø : Drei Leerzeilen und Erhöhung des Laufindex I um 1, d.h. Übergang zur nächsten Abfrage

Satz 2Ø8Ø : So lange nach dieser Erhöhung I kleiner oder gleich A ist, so lange werden weitere Abfragen gewünscht; deshalb Restaurierung des Datenbestands und zurück zum Satz 2Ø2Ø

Satz 2090 : Berechnung des Prozentanteils korrekter Antworten und Rundung auf zwei Dezimalstellen
Satz 2100 : Ausgabe dieses prozentualen Anteils
Satz 2110 : Rücksprung ins Hauptprogramm
Satz 3000 : Beginn des Unterprogramms, das D$ mit E$ vertauscht
Satz 3010 : Tausch : In ein Hilfsfeld (H$) wird der Inhalt von D$ gepackt; D$ wird (neu) mit dem Inhalt von E$ belegt; E$ wird (neu) mit dem Inhalt von H$ belegt, womit der Tausch vollzogen ist
Satz 3020 : Rücksprung

• 6. Ergebnisse

Die Ergebnisse eines Laufs dieses Programms brauchen hier nicht vorgeführt zu werden. Aus der Programmbeschreibung wird hinreichend klar, was geschieht.

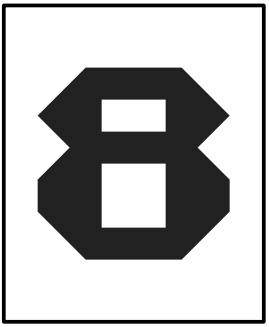
Übungsaufgaben

1. Es soll ein BASIC-Programm entwickelt werden: welches die folgende Bildschirmausgabe erzeugt :

```
C
CO
COM
COMP
COMPU
COMPUT
COMPUTE
COMPUTER
```

2. In einem BASIC-Programm werden eine Reihe von Namen bereitgestellt. Das Programm soll nun aus dieser Namensliste alle diejenigen Namen heraussuchen, die mit einem bestimmten Buchstaben, der vom Benutzer frei gewählt werden kann, anfangen.

3. Es soll ein BASIC-Programm entwickelt werden; welches einen beliebig in Großbuchstaben einzugebenden String in Kleinbuchstaben verwandelt und in dieser Schreibweise auf dem Bildschirm auch ausgibt.



Kapitel 8 : Die Benutzung der Graphik-Symbole

Abschnitt 1 : Vorbemerkung

Der ATARI-Rechner zeichnet sich insbesondere dadurch aus, daß er im Bereich der Graphikprogrammierung, insbesondere auch bei der Benutzung von Farben vielfältige Möglichkeiten bietet. Die Farbmöglichkeiten sollen hier nun nicht sehr ausführlich besprochen werden, um auch demjenigen, der nur einen Schwarz-Weiß-Monitor angeschlossen hat, das weitere Training interessant zu gestalten.

Allerdings können auch im Schwarz-Weiß-Modus die Graphikmöglichkeiten genutzt werden, denen wir uns nun zuwenden wollen. In diesem Kapitel beschäftigen wir uns dabei ausschließlich mit den Möglichkeiten, die die Verwendung der vorhandenen Graphik-Symbole bieten.

Abschnitt 2 : BASIC-Elemente

Die einfachste Möglichkeit, mit Hilfe des ATARI-Rechners und einem BASIC-Programm eine Graphik zu erzeugen, besteht darin, auf den Graphikzeichen-Vorrat des ATARI mit Hilfe von PRINT-Statements zuzugreifen. Auf diese Weise können - wie wir noch sehen werden - an beliebige Stellen des Bildschirms bestimmte Graphikzeichen gesetzt und gegebenenfalls miteinander verbunden werden, wenn man sich das folgende BASIC-Statement zunutze macht :

Statement 17 :

```
nn POSITION S,Z
```

Dieses Statement führt dazu, daß folgende PRINT-Anweisungen ihre Ausgaben an derjenigen Bildschirmstelle beginnen, die mit der Spalte S und der Zeile Z bezeichnet wird.

Man muß sich dabei den Bildschirm aufgeteilt denken in 4Ø Spalten (Nr. Ø bis 39) und 24 Zeilen (Nr. Ø bis 23); die Zeile Nr. Ø befindet sich dabei ganz oben, die Spalte Ø ganz links auf dem Bildschirm.

Das folgende kleine Programm verdeutlicht den Gebrauch dieses Statements - noch völlig unabhängig von der Frage, wie mit seiner Hilfe und der Hilfe der Graphiksymbole graphische Darstellungen erzeugt werden können :

```
10 POSITION 19,10
20 PRINT "AHA"
30 END
```

Dieses Programm druckt das Wort AHA ungefähr in die Bildschirmmitte.

Nun zu den schon mehrfach erwähnten Graphiksymbolen :

Der Zugriff auf diese Symbole erfolgt mit der Funktion CHR$, wenn diese innerhalb von PRINT-Statements verwendet wird. Diese Funktion, die als Argument eine Zahl aufweisen muß, produziert das zu dieser Zahl gehörende ASCII-Code-Zeichen - und dazu gehören auch die Graphik-Symbole.

Einige der wichtigsten dieser Symbole sind beispielsweise die folgenden :

CHR$(160) : ■

CHR$(96) : ◆

CHR$(20) : ●

CHR$(18) : ▬

Eine komplette Liste der ASCII-Codezeichen (darunter auch der Graphik-Symbole) liefert das folgende Programm :

```
10 REM -ASCII-CODE
20 FOR I=0 TO 255
30 IF I=28 OR I=29 OR I=125 THEN 70
40 IF I/15=INT(I/15) THEN GOSUB 100
50 PRINT I,CHR$(I)
70 NEXT I
80 PRINT :PRINT "ENDE":END
100 REM UP KOPFZEILE
110 PRINT :PRINT "BITTE CONT EINGEBEN !"
120 STOP
130 PRINT CHR$(125)
140 PRINT "NR.","ZEICHEN":PRINT
150 RETURN
```

Betrachten wir nun ein Beispiel zum Gebrauch dieser Möglichkeiten :

Das folgende Programm zeigt, wie ein beliebig eingegebenes Wort mit Hilfe von CHR$(18) (siehe S. 193) unterstrichen werden kann, wenn man dies wünscht. Dieses Beispiel ist also auch noch kein reines Graphik-Programm, stellt aber ohne Zweifel eine Verbindung her zwischen Programmen zur Textverarbeitung einerseits und Graphikprogrammen andererseits.

```
10 REM T21-UNTERSTREICHUNG
20 PRINT CHR$(125)
30 PRINT "         UNTERSTREICHUNGEN."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM ZEIGT, WIE BELIEBIGE"
80 PRINT "EINZUGEBENDE WORTE KORREKT UNTERSTRI-"
90 PRINT "        CHEN WERDEN KOENNEN."
100 PRINT :PRINT :PRINT
110 DIM A$(1),S$(35)
120 PRINT "BITTE EIN WORT EINGEBEN : "
130 PRINT "     ";:INPUT S$
140 PRINT CHR$(125)
150 PRINT S$
160 FOR I=1 TO LEN(S$)
170 PRINT CHR$(18);
180 NEXT I
190 PRINT :PRINT :PRINT :PRINT
200 PRINT "NOCHMAL (J/N) ";:INPUT A$
210 IF A$="J" THEN PRINT CHR$(125): GOTO 120
220 PRINT :PRINT "ENDE":END
```

Erläuterungen zu diesem Programm sind an dieser Stelle entbehrlich : Der Leser erkennt, was geschieht, wenn er das Programm aufmerksam studiert, bzw. einen Probelauf startet.

Schließlich soll mit einem weiteren Beispiel ein reines Graphik-Programm vorgeführt werden :

Das folgende Programm zeichnet auf dem Bildschirm ein Rechteck, bzw. einen Kastenrahmen unter Verwendung von CHR$(16Ø) :

```
10 REM T22-RAHMEN
20 PRINT CHR$(125)
30 PRINT "              RAHMEN."
40 PRINT :PRINT :PRINT
50 PRINT "       PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM ZEICHNET EINEN RAHMEN."
100 PRINT :PRINT :PRINT
110 PRINT "BITTE CONT EINGEBEN ! ":STOP
120 PRINT CHR$(125)
130 FOR I=1 TO 30
140 PRINT CHR$(160);
150 NEXT I:PRINT
160 FOR I=1 TO 11
165 POSITION 2,I
170 PRINT CHR$(160);
175 PRINT
180 NEXT I
190 FOR I=1 TO 30
200 PRINT CHR$(160);
210 NEXT I
220 FOR I=1 TO 11
230 POSITION 31,I
240 PRINT CHR$(160);
250 NEXT I
255 PRINT :PRINT
260 PRINT :PRINT "ENDE":END
```

Wenn man dieses Programm mit dem Kommando RUN startet, erkennt man, daß die entstehende Zeichnung relativ grob ist. Dies liegt natürlich daran, daß bei einer Bildschirmaufteilung in 4Ø Spalten und in 24 Zeilen keine sehr hohe "Auflösung" erreicht werden kann. Deshalb sind nur grobe "Skizzen" möglich.

Wir werden im Kapitel über die sog. hochauflösende Graphik erkennen, welche Alternativen sich bieten, wenn man mit derart groben Darstellungen nichts anfangen kann.

FÜNFTES TRAINING

TRAININGSAUFGABE 1 : Sinuslinie

• 1. Vorstellung des Problems

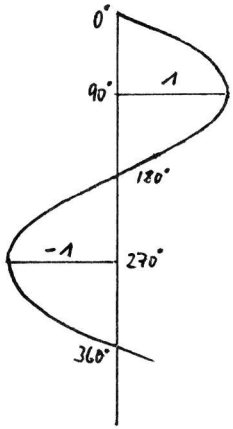
Es soll ein Programm vorgestellt werden, welches mit Hilfe derjenigen BASIC-Elemente, die im vorangegangenen Abschnitt besprochen wurden, graphisch eine Sinus-Schwingung auf dem Bildschirm produziert.

• 2. Problemanalyse

Es handelt sich hier um ein vergleichsweise einfaches Problem : Das zu entwickelnde BASIC-Programm muß für eine fortlaufende Zahl von Werten die dazugehörigen Sinuswerte bestimmen, wobei wir uns daran erinnern müssen, daß das Argument der Sinusfunktion SIN in Einheiten des Kreisparameters (und nicht etwa in Winkelgraden) angegeben werden muß.

In dem folgenden Programm ist diese Informationsbereitstellung so gelöst, daß ein Laufindex (J) immer um 1/3 erhöht wird; dieser Laufindex bzw. seine Werte werden als Argumente der Sinusfunktion benutzt. Dies bedeutet, daß nach ca. 1Ø Runden (J = 3 1/3 = 3.33...) eine Einheit des Kreisparameters (= 3.14...) überschritten wird, was ja bekanntlich einem Winkel von 18Ø° entspricht.

In diesem Bereich bewegt sich also SIN(J) von Ø bis auf 1 und wieder zum Wert Ø zurück. In den folgenden 1Ø Runden bewegt sich dann SIN(J) von Ø über -1 wieder zum Wert Ø zurück (vergleiche die nebenstehende Skizze) und bei weiteren Programmrunden beginnt dieser Prozeß wieder von vorne.



Baut man nach diesem Muster ein Endlosprogramm, so entsteht eine kontinuierliche Sinusschwingung, die immer zwischen den Werten 1 und -1 hin und herpendelt.

Wir haben in dem folgenden Programm diese Schwingung allerdings verstärkt, um sie deutlicher optisch sichtbar zu machen, indem wir die Werte SIN(J) im Wertebereich zwischen 1 und -1 mit 1Ø multipliziert haben, so daß nun die Werte sich zwischen 1Ø und -1Ø bewegen.

Addiert man zu diesen Werten nun noch jeweils den Wert 11 hinzu, so erhält man Werte die sich zwischen 1 und 21 bewegen.

Die Zeichnung selbst kann man jetzt in der Weise erzeugen, daß man in jeder Bildschirmzeile von Spalte 1 beginnend bis zur Spalte, deren Nummer sich jeweils gemäß der eben beschriebenen Rechenprozedur ergibt, Graphik-Symbole (beispielsweise "Kügelchen" per CHR$(2Ø)) einträgt. Auf diese Weise entstehen kürzere oder längere, aus Kügelchen zusammengesetzte waagrechte Stäbe, je nachdem ob gerade ein kleiner oder ein größerer Sinuswert berechnet wurde.

Das folgende Programm ist ein Endlosprogramm, d.h. es wiederholt die zu zeichnende Sinusschwingung beliebig oft. Um das Programm zu stoppen, muß der Benutzer die BREAK-Taste des ATARI betätigen.

- 3. Programm

```
10 REM T23-SINUS
20 PRINT CHR$(125)
30 PRINT "              SINUS."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM ZEICHNET EINE SINUS-"
80 PRINT "                LINIE."
100 PRINT :PRINT :PRINT
110 PRINT "BITTE CONT EINGEBEN ! ":STOP
120 PRINT CHR$(125)
130 I=1
135 J=I/3
140 S=SIN(J):S=S*10
150 FOR K=1 TO 11+S
160 PRINT CHR$(20);
170 NEXT K:PRINT
180 I=I+1
190 GOTO 135
```

• 4. Variablenliste

I = Laufindex (Rundenzähler)
J = Laufindex (= I/3; wird als Argument für die Sinusfunktion benutzt)
K = Laufindex, der angibt, wie weit die jeweilige Bildschirmzeile schon gefüllt ist
S = Sinuswert (und transformierter Sinuswert)

• 5. Programmbeschreibung

Satz 1Ø-1ØØ : Überschrift, Erläuterungen, Leerzeilen
Satz 11Ø : Programmunterbrechung
Satz 12Ø : Löschen des Bildschirms
Satz 13Ø : Beginn der Sinus-Berechnungs-Runden
Satz 135 : Bestimmung des Argumente der Sinusfunktion
Satz 14Ø : Bestimmung des Funktionswerts und Vergrößerung der Amplitude der Schwingung von 1 auf 1Ø
Satz 15Ø-17Ø : Zeichen-Schleife für eine Bildschirmzeile
15Ø : Schleifenbeginn; gezeichnet wird von Spalte 1 bis zur Spalte 11+S (der kleinste Wert, den S annehmen kann, ist -1Ø, d.h. dann wird nur in Spalte 1 gezeichnet. Der größte Wert ist 1Ø, d.h. Dann wird von Spalte 1 bis Spalte 21 gezeichnet)

160 : Zeichnen eines Kügelchens per CHR$(20);
damit das nächste in der gleichen Bildschirmzeile gezeichnet werden kann, muß das entsprechende PRINT-Statement mit einem Strichpunkt abgeschlossen werden

170 : Beendigung der Zeilen-Zeichenschleife und Zeilenvorschub durch PRINT

Satz 180 : Erhöhung des Laufindex I um 1, d.h. Vorbereitung der nächsten Sinus-Berechnungs-Runde

Satz 190 : Rücksprung zum Satz 135, d.h. Beginn der nächsten Runde

ACHTUNG : ENDLOSPROGRAMM !

6. Ergebnisse

Die Ergebnisse dieses Programms sollen hier nicht vorgeführt werden. Dem Leser wird empfohlen, einfach das Programm zu starten und sich anzuschauen, was geschieht.

TRAINININGSAUFGABE 2 : Ballspiel

• 1. Vorstellung des Problems

Wenn man sich daran erinnert, wie die ersten Tele- und Videospiele aussahen, die auf den deutschen Markt kamen, so fällt einem wieder ein, daß es zunächst sehr einfache und schematische Ballspiele waren, die vorgestellt wurden. Im einfachsten Fall sah das so aus, daß ein Punkt über den Bildschirm flog (einen Ball repräsentierend), der am Bildschirmrand oder an geraden Strichen, die der Benutzer mittels eines Knopfes auf dem Bildschirm bewegen konnte, zurückgeworfen wurden.

Auf diese Weise entstanden beispielsweise sehr simple "Tennis-Spiele" u. ä.; und im Grunde finden wir auch in den heutigen Spielprogrammen noch häufig diesen "Grundbaustein" des sich bewegenden Punktes, der an irgendwelchen Hindernissen zurückgeworfen wird oder plötzlich seine Flugbahn ändert.

Es soll nun ein Programm vorgestellt werden, welches einen solchen "fliegenden Ball" darstellt, der, jedesmal, wenn er an den Rand eines vorgegebenen "Spielfeldes" anstößt, dort zurückgeworfen wird ("abprallt").

• 2. Problemanalyse

Den "Spielfeldrand", den wir als erstes benötigen, bevor wir unseren Ball "fliegen" lassen können, erzeugen wir genauso wie den Rahmen in einem der vorangegangenen Beispielprogranme, d.h. unter Benutzung von CHR$(16Ø).

Den "Ball" selbst können wir als Kügelchen (per CHR$(2Ø)) darstellen und entscheidend ist es nun, wie vorzugehen ist, damit dieses Kügelchen sich auf dem Bildschirm bewegt, um so den Eindruck eines fliegenden Balles im Beschauer zu erwecken.

In einem ersten Problemlösungsversuch könnte man folgendermaßen vorgehen :

Wir setzen an eine beliebige Bildschirmstelle innerhalb des durch den Rahmen abgesteckten Spielfeldes ein Kügelchen. Die Bildschirmkoordinaten dieses Kügelchens (Spalte x, Zeile 1) werden in einem nächsten Schritt verändert : Neuer X-Wert = X+VX; neuer Y-Wert = Y+VY).

Nun wird auch an die neue Bildschirmstelle ein Kügelchen gezeichnet und gleichzeitig das Kügelchen an der alten Stelle wieder gelöscht (z. B. durch "Überzeichnen" mit einem inversen Cursor-Zeichen, d.h. einem kleinen Quadrat in der Farbe des Bildschirmhintergrunds). Dieses Zeichnen an der neuen Stelle bei gleichzeitigem Löschen an der alten Stelle erweckt, wenn es sich ständig wiederholt,

den Eindruck einer mehr oder weniger kontinuierlichen Bewegung.

Allerdings muß nun bei der Bestimmung der neuen Koordinaten jedesmal geprüft werden, ob diese noch innerhalb des vorgegebenen "Spielfelds" sich befinden. Ist dies nicht der Fall, muß der Ball gewissermaßen zurückgeschickt werden : Wir können dies erreichen, indem wir in den Fällen, wo der Rand erreicht oder überschritten wird, die Veränderungsgrößen VX und VY einfach ihr Vorzeichen wechseln lassen.

Damit ist das Problem des fliegenden Balls schon gelöst, wobei wieder darauf hinzuweisen ist, daß das folgende Programm ein Endlosprogramm ist. Soll es abgebrochen werden, muß der Benutzer die BREAK-Taste betätigen.

- 3. Programm

```
10 REM T24-BALL
20 PRINT CHR$(125)
30 PRINT "              BALL."
40 PRINT :PRINT :PRINT
50 PRINT "       PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM SIMULIERT EIN EINFA-"
80 PRINT "          CHES BALLSPIEL."
90 PRINT :PRINT :PRINT
92 PRINT "ZUM ABBRECHEN BITTE DIE BREAK-TASTE"
94 PRINT "             DRUECKEN"
100 PRINT :PRINT :PRINT
110 PRINT "BITTE CONT EINGEBEN ! ":STOP
120 PRINT CHR$(125)
130 REM RAHMEN
140 FOR S=2 TO 30
150 POSITION S,1:PRINT CHR$(160)
160 POSITION S,15:PRINT CHR$(160)
170 NEXT S
180 FOR Z=1 TO 15
190 POSITION 2,Z:PRINT CHR$(160)
200 POSITION 30,Z:PRINT CHR$(160)
210 NEXT Z
220 REM START
230 X=INT(RND(1)*29):Y=INT(RND(1)*14)
240 VX=1:VY=1
270 POSITION X,Y:PRINT CHR$(20)
275 AX=X:AY=Y
280 X=X+VX:Y=Y+VY
290 IF X<3 THEN X=3:VX=-VX
300 IF X>29 THEN X=29:VX=-VX
310 IF Y<2 THEN Y=2:VY=-VY
320 IF Y>14 THEN Y=14:VY=-VY
330 POSITION AX,AY:PRINT CHR$(32)
340 GOTO 270
```

4. Variablenliste

AX = Alte X-Koordinate (Spalte) des Balls
AY = Alte Y-Koordinate (Zeile) des Balls
S = Laufindex bei der Rahmenzeichnung (Spalte)
VX = Veränderung der X-Koordinate
VY = Veränderung der Y-Koordinate
X = Aktuelle X-Koordinate des Balls
Y = Aktuelle Y-Koordinate des Balls
Z = Laufindex bei der Rahmenzeichnung (Zeile)

5. Programmbeschreibung

Satz 1Ø-1ØØ : Überschrift, Erläuterungen, Leerzeilen
Satz 11Ø : Warten
Satz 12Ø : Löschen des Bildschirms
Satz 13Ø-21Ø : Zeichnen des Rahmens (Spielfeldrand)
13Ø : Kommentar
14Ø Zeichnen der waagrechten Begren-
-
17Ø : zungen
18Ø Zeichnen der senkrechten Begren-
-
21Ø : zungen
Satz 22Ø : Beginn des Spiels

Satz 23Ø	:	Vorgabe von Startkoordinaten innerhalb des Rahmens (X=Spalte, Z=Zeile)
Satz 24Ø	:	Vorgabe der Koordinaten-Veränderungsparameter (Schrittweiten; sie bestimmen die Bewegungsgeschwindigkeit)
Satz 27Ø	:	Zeichnen des Balls an die Stelle X,Y
Satz 275	:	Belegung der Hilfsfelder AX und AY mit X und Y
Satz 28Ø	:	Veränderung der Ballkoordinaten
Satz 29Ø-32Ø	:	Überprüfung, ob die veränderten Koordinaten den Spielfeldrand überschreiten; wenn ja, "Zurückschicken" des Balls und Umdrehung der Koordinaten-Veränderungsrichtung
Satz 33Ø	:	Löschen des vorher gezeichneten Balls
Satz 33Ø	:	Rücksprung zum Satz 27Ø

• 6. Ergebnisse

Auch bei diesem Programm soll auf die Darstellung von Ergebnissen verzichtet werden, die ja in diesem Fall auf dem Papier auch gar nicht vernünftig dargestellt werden können. Viel sinnvoller ist es, wenn der Leser das Programm einfach eingibt und startet.

Da es ein Endlosprogramm ist, muß es nach einiger Zeit mit der BREAK-Taste unterbrochen werden.

EXKURS : Farbgebung von Bildschirmrahmen und Bildschirmhintergrund

Unter dem Gesichtspunkt der Benutzung von Farben innerhalb laufender Programme oder unabhängig davon zeigt der ATARI vielfältige Möglichkeiten. Wir wollen in diesem Buch auf diese Möglichkeiten im Detail nicht eingehen, damit derjenige Leser, der nur einen Schwarz-Weiß-Monitor besitzt, nicht zu viele Seiten überblättern muß.

Auf einige wenige dieser Möglichkeiten soll nun aber doch hingewiesen werden, so daß zumindest im Prinzip erkennbar wird, was alles gemacht werden kann. Wir beschränken uns dabei auf die Farbgebung von Bildschirmrahmen und -hintergrund.

Zur Veränderung dieser Farben (die Voreinstellung nach Einschalten des Rechners ist schwarzer Rahmen und blauer Hintergrund) benötigen wir das folgende BASIC-Statement :

Statement 18 :

```
nn SETCOLOR Zahl1,Zahl2,Zahl3
```

Hinter dem Schlüsselwort SETCOLOR sind also in diesem Statement drei Zahlenwerte vorzugeben, die die folgende

Bedeutung haben :

Zahl1 : Steuergröße

Hat Zahl1 den Wert 2, so wird der Bildschirm-hintergrund angesprochen;
hat Zahl1 den Wert 4, so wird der Bildschirm-rahmen angesprochen.

Zahl2 : Steuerung der Farbauswahl

Mit dieser zweiten Größe können die Farben gemäß der folgenden Farbtabelle ausgewählt werden :

Zahl	Farbe	Zahl	Farbe
Ø	grau	8	blau
1	goldgelb	9	hellblau
2	orange	1Ø	türkis
3	rot-orange	11	grünblau
4	rosa	12	grün
5	lila	13	gelbgrün
6	flieder	14	orange-grün
7	blau	15	hellorange

Zahl3 : Steuerung der Farb-Brillianz

Hier können gerade Zahlen zwischen 8 und 14 angegeben werden : Je höher der Zahlenwert,

desto brillianter erscheint die jeweils gewählte Farbe.

Das folgende kleine Programm spielt die verschiedenen zur Verfügung stehenden Farben durch unterschiedliche Farbgebung des Bildschirmhintergrundes einmal durch :

```
10 REM T25
20 FOR F=0 TO 15
30 SETCOLOR 2,F,4
40 FOR I=1 TO 400:NEXT I
50 NEXT F
60 END
```

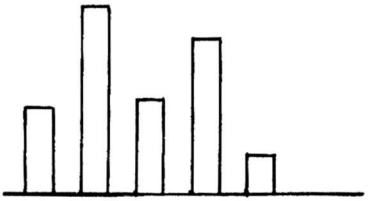
Das folgende Programm führt das gleiche Farbenspiel noch einmal vor, nun aber für den Bildschirmrahmen :

```
10 REM T26
20 FOR F=0 TO 15
30 SETCOLOR 4,F,4
40 FOR I=1 TO 400:NEXT I
50 NEXT F
60 END
```

Übungsaufgaben

1. Es soll ein BASIC-Programm entwickelt werden, welches auf dem Bildschirm drei zufällig angeordnete Rechtecke zufälliger Länge und Breite zeichnet.

2. Es soll ein BASIC-Programm entwickelt werden; welches die folgenden Daten einer einfachen Häufigkeitsverteilung graphisch in Form eines sog. Stabdiagramms ausgibt, wie es schematisch die nebenstehende Skizze veranschaulicht.

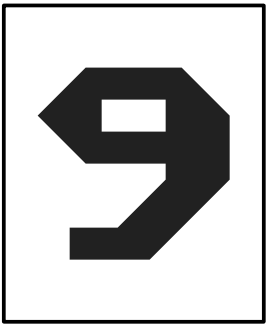


Datenbestand :

Klasse	1	2	3	4	5
Häufigkeit	15	33	17	28	7

3. Mit einem BASIC-Programm soll erreicht werden, daß Bildschirmhintergrund und Bildschirmrahmen gemeinsam alle zur Verfügung stehenden Farben "durchspielen".

Kapitel 8 : Graphik-Symbole
Übungsaufgaben



Kapitel 9 : Blockgraphik

Abschnitt 1 : BASIC-Anweisungen

Wir haben im vorangegangenen Kapitel gesehen, wie man die Graphik-Symbole des ATARI benutzen kann, um erste graphische Darstellungen erzeugen zu können. Beispielsweise war es möglich, mit Hilfe der Anweisung

```
PRINT CHR$(16Ø)
```

einen hellen quadratischen Punkt in der Größe des Cursors auf den Bildschirm zu zeichnen.

In Verbindung mit der POSITION-Anweisung und unter Benutzung von Anweisungen zur Erstellung von Programm-

schleifen, war man so in der Lage, graphische Bilder zu erzeugen. Dies haben die vorangegangenen Beispiele und die Übungsaufgaben zu Kapitel 8 hinreichend gezeigt.

Hier nun geht es darum, einige BASIC-Anweisungen kennenzulernen, die die Erzeugung solcher und ähnlicher graphischer Bilder wesentlich vereinfachen und damit erleichtern.

Das erste dieser neuen Statements ist die BASIC-Anweisung zur Erzeugung eines Punktes, also zur Ersetzung von PRINT CHR$(16Ø) in Verbindung mit der POSITION-Anweisung.

Statement 19 :

```
nn PLOT S,Z
```

Dieses Statement zeichnet an derjenigen Stelle des Bildschirms, die durch die Kreuzung der Spalte Nr. S mit der Zeile Nr. Z gebildet wird, ein kleines helles Quadrat.

Man muß sich dabei den Bildschirm aufgeteilt denken in 4Ø Spalten, die mit Ø bis 39 nummeriert sind, und in 2Ø Zeilen, die mit Ø bis 19 nummeriert sind. Die (gedachte) Spalte Nr. Ø befindet sich ganz links, die (gedachte) Zeile Nr. Ø befindet sich ganz oben auf dem Bildschirm.

Anzumerken ist in diesem Zusammenhang, daß zu den 2Ø für die Blockgraphik zur Verfügung stehenden Bildschirmzeilen noch weitere Zeilen am unteren Bildschirmrand für eventuelle Textausgaben bereitstehen.

Baut man dieses PLOT-Statement in Programmschleifen ein, so können ohne Schwierigkeiten Linien auf dem Bildschirm gezeichnet werden, wie sicherlich leicht einzusehen ist (dies wäre übrigens auch schon beispielsweise mit der Anweisung PRINT CHR$(16Ø) möglich gewesen). Derartiges Linienziehen wird aber durch das folgende Statement wesentlich vereinfacht :

Statement 2Ø :

```
nn DRAWTO S,Z
```

Dieses Statement zeichnet eine Linie vom jeweiligen Stand des Graphik-Cursors, wie er durch ein vorhergehendes PLOT- oder ein anderes DRAWTO-Statement festgelegt wurde, zum Koordinatenpunkt mit der Spalte S und der Zeile Z.

Die Benutzung dieser Statements setzt allerdings voraus, daß der Rechner vom sog. Textmodus, in dem wir uns bislang bei der Programmierung bewegten, in den Blockgraphik-Modus "umgeschaltet" wird. Diese Umschaltung leistet das folgende Statement, das also Blockgraphik-Programme einleiten muß :

Statement 21 :

```
nn GRAPHICS Zahl
```

Steht in diesem Statement an der Stelle "Zahl" der Wert 3, so erfolgt die gewünschte Umschaltung in den Blockgraphik-Modus.

Sinnvollerweise wird am Ende eines solchen Graphikprogramms wieder in den Textmodus zurückgeschaltet. Dies kann entweder dadurch geschehen, daß eine entsprechende Rückschalt-Anweisung an das Ende des Programms gesetzt wird, oder dadurch, daß der Benutzer nach Ablauf des Programms die entsprechende BASIC-Anweisung als Kommando eingibt.

Zum Zurückschalten in den Textmodus dient ebenfalls das Statement 21, wobei an der Stelle "Zahl" nun der Wert Ø einzufügen ist.

Das folgende kleine Programm, welches ein einfaches Rechteck auf den Bildschirm zeichnet und die unten zur Verfügung stehenden Bildschirmzeilen für eine Textausgabe nutzt, zeigt in anschaulicher Weise das Zusammenwirken der hier vorgeführten BASIC-Anweisungen :

```
10 REM T27-RECHTECK
20 GRAPHICS 3
25 COLOR 1
30 PLOT 5,5
40 DRAWTO 20,5
50 DRAWTO 20,12
60 DRAWTO 5,12
70 DRAWTO 5,5
76 PRINT "BITTE WARTEN"
80 REM WARTESCHLEIFE
90 FOR I=1 TO 3000:NEXT I
100 GRAPHICS 0
110 PRINT "ENDE":END
```

Auch in diesem Blockgraphik-Modus ist es möglich, Farben in den Programmen, die man entwirft, zu benutzen. Das "Einstellen"- einer bestimmten Farbe erfolgt durch dieses Statement :

Statement 22 :

nn COLOR Zahl

Es stehen im Blockgraphik-Modus vier Farben gemäß der folgenden Farbtabelle zur Verfügung. Die jeweilige Farbzahl ist an der Stelle "Zahl" des COLOR-Statements einzusetzen :

Zahl	Farbe
Ø	Hintergrundfarbe (unsichtbar)
1	orange
2	hellgrün
3	dunkelblau

In diesem Zusammenhang ist allerdings die folgende Anmerkung wichtig :

> Ändert man die Hintergrundfarbe des Bildschirms mit der SETCOLOR-Anweisung (siehe Statement 18), so ändern sich auch die Farbbelegungen gemäß obiger Farbtabelle.

Das folgende Programmbeispiel zeigt in Form eines sehr einfachen Balkendiagramms die im Blockgraphik-Modus zur Verfügung stehenden Farben :

```
10 REM T28-FARBBALKEN
20 GRAPHICS 3
30 FOR F=0 TO 3
40 COLOR F
50 PLOT 3*F,3
60 DRAWTO 3*F,15
70 NEXT F
80 PRINT "BITTE WARTEN"
90 FOR I=1 TO 3000:NEXT I
100 GRAPHICS 0
110 PRINT "ENDE":END
```

SECHSTES TRAINING

TRAININGSAUFGABE 1 : Zufallsmuster

1. Vorstellung des Problems

Es soll - die Farbmöglichkeiten des ATARI im Blockgraphik-Modus ausnutzend - ein BASIC-Programm vorgestellt werden, welches den Bildschirm mit einem Zufallsfarbmuster füllt.

2. Problemanalyse

Die programmlogische Vorbereitung dieses Programms bereitet keine besonderen Schwierigkeiten :

Wir wollen alle im Blockgraphik-Modus zur Verfügung stehenden Zeilen und Spalten des Bildschirms mit kleinen Quadraten füllen, die aber von Bildschirmposition zu Bildschirmposition ihre Farbe wechseln sollen und zwar in zufälliger Weise.

Diese Aufgabenstellung macht es zunächst erforderlich, daß in den Graphik-Modus umgeschaltet wird. Die einzelnen Bildschirmpositionen erreichen wir durch zwei geschachtelte Programmschleifen, wobei die äußere Schleife über alle Zeilen, die innere über alle Spalten des Bildschirms läuft.

Das Zeichnen selbst an der jeweiligen Bildschirmposition übernimmt die PLOT-Anweisung, die Farbgebung erreichen wir mit der COLOR-Anweisung. Der Parameter der COLOR-Anweisung muß dabei - jedes Mal, wenn diese Anweisung berührt wird während der Programmabarbeitung - eine Zufallszahl zwischen Ø und 3 (ganzzahlig) sein. Eine derartige Zufallszahl erhält man bekanntlich mit der folgenden BASIC-Anweisung :

```
PRINT INT(RND(1)*4)
```

Wenn die Graphik komplett erzeugt ist, ist es sinnvoll, in das Programm eine sog. Warteschleife einzufügen. Wir verstehen darunter einen Programmteil, in dem nichts geschieht, der Rechner sich also quasi in Warteposition befindet. Damit soll dem Benutzer Gelegenheit geboten werden, sich in Ruhe die Graphik anzuschauen, bevor per Programmanweisung in den Textmodus zurückgeschaltet wird und damit dann die Graphik vom Bildschirm verschwindet. Es versteht sich, daß derartige Warteschleifen auch in anderen Zusammenhängen sinnvoll verwendet werden können, zum Beispiel dann, wenn bestimmte Textausgaben eine vorgegebene Zeitspanne lang auf dem Bildschirm stehenbleiben

sollen, bevor die weitere Programmabarbeitung diese etwa mit Hilfe einer Löschanweisung vor weiteren Ausgaben verschwinden läßt.

Wenn man eine solche Warteschleife benutzt, sollte sinnvollerweise auf dem Bildschirm eine entsprechende Meldung ausgegeben werden, während die Programmabarbeitung unterbrochen ist, damit auch der ungeübte Programmbenutzer nicht den Eindruck hat, sein Rechner hätte einen Defekt, weil plötzlich nichts mehr geschieht.

Ist die Warteschleife in dem hier zu entwickelnden Programm abgelaufen, dann kann mit der Anweisung GRAPHICS Ø wieder in den Textmodus zurückgeschaltet werden und das Programm ist dann beendet.

- 3. Programm

```
10 REM T29-FARBGRAPHIK
20 PRINT CHR$(125)
30 PRINT :PRINT :PRINT
40 PRINT "              FARBGRAPHIK."
50 PRINT :PRINT :PRINT
60 PRINT "         PROF.DR.W.VOSS, 1984"
70 PRINT :PRINT :PRINT
80 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !"
90 STOP
100 GRAPHICS 3
130 FOR Z=0 TO 19
140 FOR S=0 TO 39
150 F=INT(RND(1)*4)
160 COLOR F
170 PLOT S,Z
180 NEXT S
190 NEXT Z
200 PRINT "BITTE WARTEN"
210 FOR I=1 TO 3000:NEXT I
220 GRAPHICS 0
230 PRINT "ENDE":END
```

• 4. Variablenliste

F = Zufallszahl zwischen Ø und 3 zur Auswahl einer Farbe im Blockgraphik-Modus
I = Laufindex in der Warteschleife
S = Bildschirmspalte, in der gezeichnet werden soll
Z = Bildschirmzeile, in der gezeichnet werden soll

• 5. Programmbeschreibung

```
Satz 1Ø-7Ø      :   Überschrift, Erläuterungen, Leerzeilen
Satz 8Ø-9Ø      :   Unterbrechung des Programms
Satz 1ØØ        :   Umschalten in den Blockgraphik-Modus
Satz 13Ø-19Ø    :   Schleife über alle Bildschirmzeilen
                    14Ø   Schleife über alle Bildschirm-
                     -
                    18Ø   spalten
                          15Ø : Zufallsauswahl einer Farbe
                          16Ø : Farbvorgabe
                          17Ø : Zeichnen eines Quadrats
                    18Ø : Ende der Spaltenschleife
                    19Ø : Ende der Zeilenschleife
Satz 2ØØ        :   Ausgabe einer Wartemeldung
Satz 21Ø        :   Warteschleife
Satz 22Ø        :   Zurückschalten in den Textmodus
Satz 23Ø        :   Beendigung des Programms
```

6. Ergebnisse

Wie bei allen Graphik-Programmen, insbesondere dann, wenn auch noch Farben eine Rolle spielen, ist die Darstellung der Programmergebnisse hier nicht sehr sinnvoll. Es soll darauf deshalb verzichtet werden. Stattdessen sollte der Leser das vorangegangene Programm mit einem angeschlossenen Farbmonitor oder Farbfernseher einmal ausprobieren.

TRAINININGSAUFGABE 2 : Zufallslinie

1. Vorstellung des Problems

Das folgende BASIC-Programm dient dazu, einen Linienzug auf dem Bildschirm zu erzeugen, der sich aus geraden Linienstücken unterschiedlicher und zufälliger Länge zusammensetzt und nach jedem Linienstück jeweils rechtwinklig abknickt.

Solche und ähnliche Programme können als Grundbausteine in Computerspielen verwendet werden, in denen zufallsabhängige, also nicht vorher berechenbare Bewegungsabläufe benötigt werden.

2. Problemanalyse

Bei dieser Aufgabenstellung ist die Problemanalyse ein wenig aufwendiger als dies bei dem vorangegangenen Beispiel der Fall war, obwohl wir von dort einige Programmelemente übernehmen können.

Wir starten das Programm damit, daß wir zunächst an eine beliebige Stelle des Bildschirms, die zufällig auszuwählen

ist (sowohl Bildschirmspalte wie auch Bildschirmzeile), einen Punkt zeichnen.

Von diesem Punkt aus wird dann ein gerader Linienzug gezeichnet, dessen Länge ebenfalls zufällig bestimmt wird. Aber nicht nur die Länge dieses Linienstücks, sondern auch die Entscheidung darüber, in welche Richtung von dem ersten Punkt ausgehend dieses Linienstück gezeichnet wird, wird vom Zufall abhängig gemacht. Man kann dies beispielsweise dadurch erreichen, daß eine Zufallszahl zwischen 1 und 2 (also entweder die Eins oder die Zwei) gewählt wird und je nach Ergebnis, entweder die eine oder eben die andere Bewegungsrichtung gewählt wird.

Auf diese Weise bestimmen wir die Spaltenkoordinate, die angibt, bis wohin das gerade Linienstück gezeichnet werden soll (und entsprechend später auch die Zeilenkoordinate).

Es muß dabei bei all diesen Zufallsauswahlen, bzw. bei der Fixierung der jeweiligen Koordinaten darauf geachtet werden, daß der zulässige Bildschirmbereich nicht verlassen wird, d.h. es darf keine Spalte bestimmt werden, die kleiner als Ø oder größer als 39 und keine Zeile, die kleiner als Ø oder größer als 19 ist. Entsprechende Abfragen können dies kontrollieren.

- 3. Programm

```
10 REM T30-LINIEN
20 PRINT CHR$(125)
30 PRINT :PRINT :PRINT
40 PRINT "              LINIEN."
50 PRINT :PRINT :PRINT
60 PRINT "        PROF.DR.W.VOSS, 1984"
70 PRINT :PRINT :PRINT
75 PRINT "DIESES PROGRAMM ZEICHNET AUS GERADEN"
76 PRINT "     LINIEN EIN ZUFALLSMUSTER."
77 PRINT :PRINT :PRINT
78 PRINT "ABBRECHEN MIT DER BREAK-TASTE"
79 PRINT "DANACH KOMMANDO GRAPHICS 0":PRINT
80 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !"
90 STOP
100 GRAPHICS 3
110 COLOR 1
120 S=INT(RND(1)*39):Z=INT(RND(1)*19)
130 PLOT S,Z
140 L=INT(RND(1)*19)
150 V=INT(RND(1)*2)
160 IF V=1 THEN L=-L
170 S=S+L
180 IF S<1 THEN S=1
190 IF S>38 THEN S=38
200 DRAWTO S,Z
210 B=INT(RND(1)*7)
220 V=INT(RND(1)*2)
230 IF V=1 THEN B=-B
240 Z=Z+B
250 IF Z<1 THEN Z=1
260 IF Z>18 THEN Z=18
270 DRAWTO S,Z
280 GOTO 140
```

4. Variablenliste

B = Anzahl der Bildschirmzeilen, über die sich ein gerades Linienstück erstrecken soll
L = Anzahl der Spalten, über die sich ein gerades Linienstück erstrecken soll
S = Spalte, in der gezeichnet wird
V = Markiervariable zur eventuellen Veränderung der Bewegungsrichtung
Z = Zeile, in der gezeichnet wird

5. Programmbeschreibung

Satz 1Ø-79 : Überschrift, Erläuterungen, Leerzeilen
Satz 8Ø-9Ø : Programmunterbrechung
Satz 1ØØ : Umschalten in den Blockgraphik-Modus
Satz 11Ø : Angabe einer Zeichenfarbe
Satz 12Ø : Zufallsauswahl der Startkoordinaten innerhalb der erlaubten Grenzen
Satz 13Ø : Zeichnen des Startpunktes
Satz 14Ø : Bestimmung der Länge eines waagrechten, geraden Linienstücks (Zufallsauswahl zwischen Ø und 9)
Satz 15Ø : Zufällige Belegung der Markiervariablen zur Bestimmung der Zeichenrichtung (links oder rechts) mit 1 oder 2
Satz 16Ø : Wenn die Markiervariable V zufällig den Wert 1 hat, wird die Bewegungs-

richtung umgedreht; hat hingegen V den Wert 2, so erfolgt keine Umdrehung

Satz 17Ø : Bestimmung der neuen Spaltenkoordinate

Satz 18Ø-19Ø : Prüfung, ob der neue S-Wert den zulässigen Bildschirmbereich überschreitet; wenn ja, wird S am Bildschirmrand quasi festgehalten

Satz 2ØØ : Das gerade waagrechte Linienstück wird zeichnet

Satz 21Ø-27Ø : Genauso wie in den Sätzen 14Ø-2ØØ wird nun verfahren, um ein senkrechtes gerades Linienstück zu zeichnen

Satz 28Ø : Rücksprung zum Satz 14Ø, d.h. Wiederholung der ganzen Prozedur.

ACHTUNG : Bei diesem Programm handelt es sich wieder um ein Endlosprogramm. Es kann durch Betätigung der BREAK-Taste unterbrochen werden.

Nach Abbruch des Programms muß das Kommando GRAPHICS Ø eingegeben werden, wenn der Benutzer in den üblichen Textmodus zurückkehren möchte.

• 6. Ergebnisse

Auch bei diesem Programm ist die Darstellung von Ergebnissen an dieser Stelle nicht sinnvoll.

TRAININGSAUFGABE 3 : Beschäftigtenstatistik

• 1. Vorstellung des Problems

Im folgenden Programm soll an einem sehr einfachen Beispiel gezeigt werden, wie die Blockgraphik-Möglichkeiten zusammen mit den Farbgestaltungsmöglichkeiten dazu benutzt werden können, statistische Sachverhalte anschaulich graphisch zu verdeutlichen.

Wir haben dazu den folgenden Datenbestand ausgesucht :

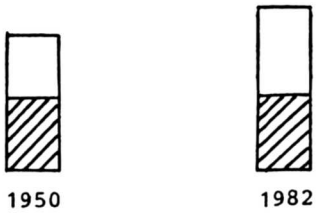
Betrachtet wird die Beschäftigtenstatistik in der Bundesrepublik Deutschland in den Jahren 195Ø und 1982 :
Dabei ergibt sich, daß 195Ø in der Bundesrepublik 21.2 Millionen Menschen beschäftigt waren, von denen 51 % Arbeiter waren. 1982 hingegen weist die amtliche Statistik 25.6 Millionen Beschäftigte aus, davon 41.3 % Arbeiter.

Diese Daten sollen nun so, wie es die Skizze auf der folgenden Seite andeutet, graphisch (in Form eines sog. Blockdiagramms) dargestellt werden.

Es handelt sich hier um ein Beispiel der sog. "Business Graphics", wo mit Hilfe geometrischer Figuren (häufig

Rechtecke, Kreise u.ä.) oder mit Hilfe von Kurven in Achsenkreuzen betriebliche oder auch gesamtwirtschaftliche statistische Daten dargestellt werden.

Graphische Veranschaulichung der oben genannten Daten :



• 2. Problemanalyse

Obwohl dieses Programm recht umfangreich gerät, ist die Problemanalyse doch nicht sehr schwer :

Zunächst müssen die statistischen Ausgangsinformationen dem Programm bekannt gegeben werden. Danach ist für die beiden Jahre getrennt in einer Farbe zunächst ein Rechteck zu zeichnen, dessen Fläche die Arbeiterzahl, dann in einer anderen Farbe jeweils ein Rechteck, dessen Fläche die Zahl der übrigen Beschäftigten repräsentiert.

Damit diese Graphik dann auch ins Auge fällt, dehnen wir

die zu zeichnenden Stäbe (Rechtecke) über vier Bildschirmspalten aus.

In die Bildschirm-Textzeilen geben wir dann noch unterhalb der beiden Rechtecke die beiden Jahreszahlen aus, hängen eine Warteschleife an das Programm an und kehren nach Ablauf dieser Schleife in den Textmodus zurück, um dann das Programm auch zu beenden.

- 3. Programm

```
10 REM T31-BUSINESS GRAPHICS
20 PRINT CHR$(125)
30 PRINT :PRINT :PRINT
40 PRINT "        BUSINESS GRAPHICS"
50 PRINT :PRINT :PRINT
60 PRINT "        PROF.DR.W.VOSS, 1984"
70 PRINT :PRINT :PRINT
75 PRINT "PROGRAMM ZUR GRAPHISCHEN DARSTELLUNG"
76 PRINT "        STATISTISCHER DATEN."
77 PRINT :PRINT :PRINT
80 DIM A$(1)
90 GOSUB 1000:REM WARTEN
110 PRINT " 1950 GAB ES IN DER BUNDESREPUBLIK"
120 PRINT "21.2 MILL. ERWERBSTAETIGE, DAVON 51%"
130 PRINT "ARBEITER.":PRINT
140 PRINT "1982 WAREN ES 25.6 MILL. ERWERBSTAE-"
150 PRINT "TIGE, DAVON 41.3% ARBEITER."
160 PRINT :PRINT :PRINT
170 PRINT "DIESE SACHVERHALTE SOLLEN NUN GRA-"
180 PRINT "PHISCH DARGESTELLT WERDEN."
190 GOSUB 1000:REM WARTEN
200 J1=1950:J2=1982
```

```
210 E1=21.2:E2=25.6
220 P1=51:P2=41.3
230 A1=E1*P1/100:A2=E2*P2/100
240 S1=E1-A1:S2=E2-A2
250 B=17
260 GRAPHICS 3
265 FOR I=1 TO 4
270 COLOR 1
280 PLOT 10+I,B
290 A=B-A1/2
300 DRAWTO 10+I,A
310 COLOR 2
320 A=B-E1/2
330 DRAWTO 10+I,A
332 NEXT I
335 FOR I=1 TO 4
340 COLOR 1
350 PLOT 20+I,B
360 A=B-A2/2
370 DRAWTO 20+I,A
380 COLOR 2
390 A=B-E2/2
400 DRAWTO 20+I,A
405 NEXT I
410 PRINT "          1950       1982"
500 FOR I=1 TO 3000:NEXT I
505 GRAPHICS 0
510 PRINT "ENDE":END
1000 REM UP WARTEN
1010 PRINT :PRINT :PRINT
1020 PRINT "ZUR FORTSETZUNG BITTE EINE TASTE"  *)
1030 PRINT "DRUECKEN UND RETURN-TASTE."
1040 INPUT A$
1050 IF A$="" THEN 1040
1060 PRINT CHR$(125)
1070 RETURN
```

*) Leertaste nicht zulässig

• 4. Variablenliste

A = Oberes Ende (Zeile) des jeweils zu zeichnenden Stabes (vier Stäbe nebeneinander ergeben eines der Rechtecke (s.Seite 235 oben))
A1 = Anteil der Arbeiter 195Ø in Millionen
A2 = Anteil der Arbeiter 1982 in Millionen
A$ = Stringvariable zur Aufnahme eines Symbols
B = Unteres Ende der zu zeichnenden Stäbe
E1 = Zahl der Erwerbstätigen 195Ø
E2 = Zahl der Erwerbstätigen 1982
I = Laufindex
J1 = Erstes Jahr
J2 = Zweites Jahr
P1 = Anteil der Arbeiter 195Ø in Prozent
P2 = Anteil der Arbeiter 1982 in Prozent
S1 = Zahl der anderen Erwerbstätigen 195Ø
S2 = Zahl der anderen Erwerbstätigen 1982

• 5. Programmbeschreibung

Satz 1Ø-77 : Überschrift, Erläuterungen, Leerzeilen
Satz 8Ø : Dimensionierung der Stringvariablen
Satz 9Ø : Sprung ins Unterprogramm 1ØØØ zum Warten
Satz 11Ø-18Ø : Erläuterungen

Kapitel 9 : Blockgraphik
SECHSTES TRAINING

Satz 19Ø : Sprung ins Unterprogramm 1ØØØ zum Warten

Satz 2ØØ-22Ø : Eingabe der Ausgangsdaten

Satz 23Ø : Berechnung der Anzahl der Arbeiter in Millionen

Satz 24Ø : Berechnung der Anzahl der übrigen Beschäftigten in Millionen

Satz 25Ø : Vorgabe der unteren Bildschirmzeile, ab der die beiden Rechtecke gezeichnet werden sollen

Satz 26Ø : Umschalten in den Graphikmodus

Satz 265-332 : Viererschleife zum Zeichnen eines vier Spalten breiten Rechtecks

265 : Schleifenbeginn

27Ø : Vorgabe einer Farbe

28Ø : Zeichnen eines Punktes in der 11. Spalte der untersten Zeile (Zeile B; B=17)

29Ø : Bestimmung des oberen Stabendes

3ØØ : Zeichnen eines ersten Stabes in der ersten Farbe

31Ø : Vorgabe der zweiten Farbe

32Ø : Neu-Bestimmung des oberen Stabendes

33Ø : Fortsetzung des Stabes, nun in der zweiten Farbe

332 : Schleifenende, d.h. viermalige Wiederholung dieser Prozedur

Satz 335-4Ø5 : Gleiche Prozedur wie in den Sätzen 265-332; nun aber mit den Daten für 1982

Satz 41Ø : Ausgabe einer Textzeile
Satz 5ØØ : Warteschleife
Satz 5Ø5 : Rückkehr in den Textmodus
Satz 51Ø : Beendigung des Programms

Satz 1ØØØ : Beginn des Unterprogramme zum Warten
Satz 1Ø1Ø-1Ø3Ø : Ausgabe einer Erläuterung
Satz 1Ø4Ø : Abwarten einer String-Eingabe
Satz 1Ø5Ø : Solange keine String-Eingabe erfolgt, d.h. solange der Programmbenutzer keine Taste betätigt, verharrt das Programm im Satz 1Ø5Ø, d.h. es wartet
Satz 1Ø6Ø : Ist hingegen die Tasteneingabe erfolgt, wird der Bildschirm gelöscht
Satz 1Ø7Ø : Rücksprung ins Hauptprogramm

Der Leser erkennt, daß wir hier eine andere Art gewählt haben, das Programm zu unterbrechen und es abwarten zu lassen. Anstelle des Statements STOP, das zu einer Programmunterbrechung führt, die nur mit dem Kommando CONT wieder aufgehoben werden kann, unterbrechen wir hier mit Hilfe des INPUT-Statements : Es wird eine Tasteneingabe angefordert und solange diese nicht erfolgt, bleibt das Programm gewissermaßen stehen.

• 6. Ergebnisse

Wie bei den anderen Graphikprogrammen verzichten wir auch hier auf die Darstellung der Ergebnisse (vergleiche Skizze auf Seite 234).

Übungsaufgaben

1. Es soll ein BASIC-Programm entwickelt werden, welches auf dem Bildschirm im Blockgraphik-Modus ein Gitter erzeugt. Dabei sollen die Abstände zwischen den senkrechten Gitterstrichen und die zwischen den waagrechten Gitterstrichen frei wählbar sein.

2. Zu Übungszwecken soll ein BASIC-Programm geschrieben werden, welches im Blockgraphik-Modus auf dem Bildschirm eine Diagonale erzeugt. Diese soll von links oben nach rechts unten über den Bildschirm verlaufen.

1 Ø

Kapitel 1Ø : Hochauflösende Graphik

Abschnitt 1 : Vorbemerkung

Die Graphiken, die mit den Programmen der beiden vorangegangenen Kapitel erzeugt werden konnten und entsprechende andere Programme, die man auf dieser Grundlage entwickeln könnte, haben den Nachteil, daß sie nur zu relativ groben Zeichnungen führen können.

Bei bestimmten Aufgabenstellungen, wie zum Beispiel bei der Erzeugung von Flächendiagrammen, von Stabdiagrammen u. ä. spielt dies keine besondere Rolle, bzw. die Blockgraphik kann dann sogar von Vorteil sein.

Es gibt aber eine ganze Reihe von Aufgabenstellungen, bei denen eine höhere Auflösung der graphischen Linien in einer Zeichnung unumgänglich ist. Man denke beispielsweise an geometrische Konstruktionszeichnungen, an architektonische Skizzen, an die Darstellung statistischer Zeitreihen, an die einzelnen graphischen Elemente von bestimmten Computerspielen usw.

Für diese und ähnliche Zwecke stellt der ATARI, wie viele andere Rechner auch, die sog. hochauflösende Graphik zur Verfügung, über die nun in diesem Kapitel gesprochen werden soll.

Abschnitt 2 : BASIC-Anweisungen

Im Gegensatz zum Blockgraphik-Modus hat man sich nun den Bildschirm aufgeteilt zu denken in 32Ø Spalten (Ø bis 319; Spalte Ø ist ganz links) und 16Ø Zeilen (Ø bis 159; Zeile Ø ist wieder ganz oben auf dem Bildschirm).

Damit sind nun 16Ø * 32Ø = 512ØØ verschiedene Punkte auf dem Bildschirm ansprechbar, so daß man zu sehr viel detailreicheren Darstellungen gelangen kann, als dies im Blockgraphik-Modus möglich war, wo ja nur 8ØØ verschiedene Bildschirmpositionen zur Verfügung standen.

Auch hier stehen zusätzlich Textzeilen am unteren Bildschirmrand bereit.

Das Umschalten vom Textmodus in den Graphikmodus mit höherer Auflösung erfolgt ebenfalls mit der GRAPHICS-Anweisung, die nun aber mit dem Parameterwert 8 versehen werden muß, also :

```
GRAPHICS 8
```

Die Zurückschaltung in den Textmodus erfolgt auch hier wieder mit der Anweisung GRAPHICS Ø.

Kapitel 1Ø : Hochauflösende Graphik
Abschnitt 1 : Vorbemerkung

Die BASIC-Anweisungen, die benötigt werden, um in der hochauflösenden Graphik Punkte oder Linien zu zeichnen, oder um Farben zu erzeugen, sind die gleichen wie die im Blockgraphik-Modus (vergleiche Kapitel 9, Abschnitt 1). Wir brauchen deshalb an dieser Stelle keine weiteren Einzelheiten zu besprechen und können stattdessen sofort die nächste Trainingssitzung in Angriff nehmen.

SIEBENTES TRAINING

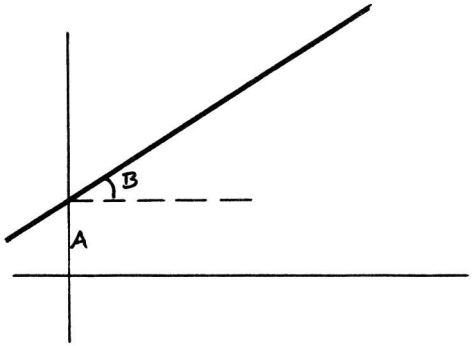
TRAININGSAUFGABE 1 : Gerade

1. Vorstellung des Problems

In einem ersten Programmbeispiel soll ein BASIC-Programm gezeigt werden, welches in der Lage ist, in ein Achsenkreuz eine beliebige Gerade einzuzeichnen. Dies ist also eine Aufgabe aus dem Geometriebereich, der sich besonders für den Einsatz der hochauflösenden Graphik eignet, wie man sich leicht vorstellen kann.

Mit diesem BASIC-Programm soll also auf dem Bildschirm eine Zeichnung erstellt werden, die schematisch der folgenden Skizze entspricht (in diese Skizze haben wir auch die relevanten Parameter eingetragen, die bei der geometrischen Konstruktion einer Geraden zu beachten sind).

Gerade :



• 2. Problemanalyse

Bei der Problemanalyse ist zunächst von der folgenden Überlegung auszugehen :

Eine hochauflösende Graphik, wie zum Beispiel eine Zeichnung einer Geraden, kann in der Weise erzeugt werden, daß man in horizontaler Richtung über den Bildschirm wandert, also alle 32Ø zur Verfügung stehenden Bildschirmspalten "entlangwandert", um dabei in jeder Spalte den entsprechenden Bildschirm-Zeilenwert zu berechnen: der sich gemäß der darzustellenden mathematischen Funktion (hier der Geraden) ergibt.

Bei jeder Kombination von Bildschirmspalte und Bildschirmzeile ist dabei darauf zu achten, daß der zulässige Bildschirmbereich nicht verlassen wird.

An jedem Koordinatenschnittpunkt (Bildschirmspalte/Bildschirmzeile) kann dann ein Punkt gezeichnet werden und auf diese Weise entsteht bei Bearbeitung aller Bildschirmspalten nach diesem Muster das gewünschte Bild.

In diesem Beispiel ist dabei Voraussetzung, daß zunächst die die Gestalt und Lage der Geraden bestimmenden Parameter als Input-Informationen vorgegeben werden. Es handelt sich dabei um den Ordinatenabschnitt der Geraden, den wir mit A bezeichnen, und um den Tangens des Steigungswinkels, der mit B bezeichnet wird.

Die Gerade selbst bestimmt sich dann bekanntlich gemäß der folgenden mathematischen Funktion :

$$Y_i = A + B * X_i$$

(vergl. auch Skizze auf Seite 246).

Bevor unter Benutzung dieser Funktionalbeziehung die Gerade auf dem Bildschirm gezeichnet wird, soll das Programm auch ein Achsenkreuz ausgeben, dessen Ursprung (Achsenschnittpunkt) in der Mitte des Bildschirms liegen soll.

- 3. Programm

```
10 REM T32-GERADE
20 PRINT CHR$(125)
30 PRINT "PROGRAMM ZUM ZEICHNEN EINER GERADEN."
35 PRINT :PRINT :PRINT
40 PRINT "        PROF.DR.W.VOSS, 1984"
45 FOR I=1 TO 8:PRINT :NEXT I
50 PRINT "ORDINATENABSCHNITT A: ";:INPUT A
60 PRINT "STEIGUNG            B: ";:INPUT B
70 GRAPHICS 8
75 COLOR 1
80 PLOT 0,80:DRAWTO 319,80
90 PLOT 160,0:DRAWTO 160,159
100 FOR X=0 TO 319
110 Y=80-A-B*(X-160)
120 IF Y<=0 OR Y>=159 THEN 140
130 PLOT X,Y
140 NEXT X
145 PRINT "BITTE WARTEN"
150 FOR I=1 TO 2000:NEXT I
160 GRAPHICS 0
170 PRINT "ENDE":END
```

• 4. Variablenliste

A = Ordinatenabschnitt
B = Tangens des Steigungswinkels
I = Laufindex
X = Spaltenkoordinate
Y = Zeilenkoordinate

• 5. Programmbeschreibung

Satz 1Ø-45 : Überschrift, Erläuterungen, Leerzeilen
Satz 5Ø-6Ø : Bekanntgabe der Input-Informationen (Ordinatenabschnitt und Steigung)
Satz 7Ø : Umschalten in den Graphik-Modus (hochauflösende Graphik)
Satz 8Ø-9Ø : Zeichnen eines Achsenkreuzes
Satz 1ØØ-14Ø : Schleife zum Zeichnen der Geraden
1ØØ : Schleifenbeginn (die Schleife läuft über alle Bildschirmspalten von Ø bis 319)
11Ø : Bestimmung des Y-Wertes gemäß der Geradengleichung (Bildschirmzeile); es muß dabei berücksichtigt werden, daß die Zeile Nr. 9 ganz oben auf dem Bildschirm ist.

12Ø : Überprüfung, ob Y den zulässigen Bildschirmbereich einhält; wenn nicht, wird nicht gezeichnet, sondern gleich zum nächsten x-Koordinatenwert übergegangen

13Ø : Zeichnen des Koordinatenpunktes X,Y

14Ø : Ende der Zeichenschleife (nächster X-Wert = nächste Bildschirmspalte)

Satz 145 : Wartemeldung

Satz 15Ø : Warteschleife

Satz 16Ø : Zurückschalten in den Textmodus

Satz 17Ø : Beendigung des Programms

• 6. Ergebnisse

Wie schon bei den Graphikprogrammen zuvor, soll hier auf die Ergebnisdarstellung verzichtet werden. Gibt man als Input-Informationen zum Beispiel den Ordinatenabschnitt A = 1Ø und die Steigung B = 1 ein, so erhält man ein Bild, wie es schematisch der Skizze auf Seite 246 entspricht.

TRAININGSAUFGABE 2 : Kreis

• 1. Vorstellung des Problems

Das folgende BASIC-Programm ist dazu geeignet, an beliebige Stellen des Bildschirms, die der Benutzer des Programms frei wählen kann, einen Kreis mit beliebigem Radius zu zeichnen.

• 2. Problemanalyse

Unter problemanalytischen Gesichtspunkten sind hier im Prinzip die gleichen Überlegungen anzustellen wie in der Trainingsaufgabe zuvor. Der einzige Unterschied besteht nun darin, daß mit einer anderen mathematischen Funktion zu arbeiten ist, nämlich mit der Kreisgleichung.

Bei Kreisen, deren Mittelpunkt nicht im Koordinatenursprung liegt, lautet diese Kreisgleichung folgendermassen :

$$(x-a)^2 + (y-b)^2 = r^2$$

Dabei ist r der Radius des Kreises (im Programm mit R bezeichnet), a ist eine Zahl, die die Verschiebung des Kreises gegenüber der Y-Achse bezeichnet (im Programm : T) und b ist eine Zahl, die die Verschiebung gegenüber der x-Achse kennzeichnet (im Programm : Z).

Wenn man die obige Kreisgleichung nach der Variablen Y^2 auflöst, so erhält man eine quadratische Gleichung, welche - dann nach Y aufgelöst - zur Berechnung von je zwei Y-Werten führt (oberer und unterer Halbkreis). Diese beiden Werte unterscheiden sich jeweils dadurch voneinander, daß bei der ersten Berechnung eine Quadratwurzel hinzugezählt, bei der zweiten hingegen subtrahiert wird (d.h. es wird mit dem positiven und dem negativen Wert der jeweiligen Quadratwurzel gearbeitet).

Die Zeichnung selbst erstellen wir wieder genauso, wie das auch im vorangegangenen Beispiel der Fall war, d.h. es werden alle X-Werte von T-R bis T+R abgearbeitet und jeweils die beiden dazugehörigen Y-Werte berechnet. Werte außerhalb dieses angegebenen Bereichs von T-R bis T+R brauchen (und dürfen) nicht überprüft werden, weil für solche X-Werte die Quadratwurzel ein negatives Argument hätte und der ATARI deshalb dann das Programm mit einer Fehlermeldung abbrechen würde.

Der Einfachheit halber werden wir allerdings die beiden Y-Werte in zwei getrennten Schleifen berechnen, so daß also erst der eine und danach dann der andere Halbkreis auf dem Bildschirm gezeichnet wird.

Nach jeder Y-Berechnung wird wieder überprüft, Ob der zulässige Bildschirmbereich nicht verlassen wird. Nur wenn der Wert im zulässigen Bereich liegt, wird der entsprechende Punkt mit den Koordinaten X und Y auch gezeichnet, sonst nicht.

- 3. Programm

```
10 REM T33-KREIS
20 PRINT CHR$(125)
30 PRINT "PROGRAMM ZUM ZEICHNEN EINES KREISES."
35 PRINT "         PROF.DR.W.VOSS, 1984"
40 PRINT :PRINT :PRINT: PRINT
50 PRINT "DIESES PROGRAMM BENOETIGT ALS INPUT-"
60 PRINT "INFORMATIONEN SPALTE UND ZEILE DES"
70 PRINT "KREISMITTELPUNKTES UND DEN RADIUS."
80 PRINT :PRINT :PRINT
90 PRINT "MITTELPUNKT-SPALTE :  ";:INPUT T
100 PRINT "MITTELPUNKT-ZEILE  :  ";:INPUT Z
105 PRINT :PRINT
110 PRINT "RADIUS             :  ";:INPUT R
120 GRAPHICS 8: COLOR 1
130 PLOT 0,0:DRAWTO 0,159
140 PLOT 0,159:DRAWTO 319,159
150 PLOT T,0:DRAWTO T,159
160 PLOT 0,Z:DRAWTO 319,Z
170 A=T-R:B=T+R
180 IF A<0 THEN A=0
190 IF B>319 THEN B=319
```

```
200 FOR X=A TO B
210 D=R*R-(X-T)^2
220 IF D<=Ø THEN Y=Z:GOTO 25Ø
230 Y=Z+SQR(D)
240 IF Y<Ø OR Y>159 THEN 26Ø
25Ø PLOT X,Y
26Ø NEXT X
27Ø FOR X=B TO A STEP -1
28Ø D=R*R-(X-T)^2
29Ø IF D<=Ø THEN Y=Z:GOTO 32Ø
3ØØ Y=Z-SQR(D)
31Ø IF Y<Ø OR Y>159 THEN 33Ø
32Ø PLOT X,Y
33Ø NEXT X
34Ø FOR I=1 TO 2ØØØ:NEXT I
35Ø GRAPHICS Ø
36Ø PRINT "ENDE":END
```

4. Variablenliste

A = Linker "Rand" des Kreises (A = T - R)
B = Rechter "Rand" des Kreises (B = T + R)
D = Argument der Quadratwurzel in der Kreisgleichung (D = Determinante = $R^2 - (X-T)^2$)
I = Laufindex
R = Radius des Kreises
T = Verschiebung des Kreismittelpunktes gegenüber der Y-Achse des Koordinatensystems
X = X-Koordinate des Punktes auf der Kreislinie
Y = Y-Koordinate des Punktes auf der Kreislinie
Z = Verschiebung des Kreismittelpunktes gegenüber der X-Achse des Koordinatensystems

5. Programmbeschreibung

Satz 1Ø-8Ø : Überschrift, Erläuterungen, Leerzeilen
Satz 9Ø-11Ø : Anforderung der Input-Informationen
Satz 12Ø : Umschalten in den Graphikmodus
Satz 13Ø-14Ø : Zeichnen eines Achsenkreuzes
Satz 15Ø-16Ø : Zeichnen eines Hilfsachsenkreuzes, dessen Ursprung im Kreismittelpunkt liegt
Satz 17Ø : Wartemeldung
Satz 15Ø : Warteschleife
Satz 16Ø : Zurückschalten in den Textmodus
Satz 17Ø : Bestimmung der Grenzen für die Variable X

Satz 18Ø-19Ø : Überprüfung, ob die Grenzen der Variablen x im zulässigen Bildschirmbereich liegen

Satz 2ØØ-26Ø : Zeichnen des unteren Halbkreises

- 2ØØ : Beginn der Zeichenschleife
- 21Ø : Berechnung der Determinanten D
- 22Ø : Verhinderung von unzulässigen Werten der Determinanten
- 23Ø : Berechnung der Y-Koordinaten
- 24Ø : Überprüfung, ob der Y-Wert im zulässigen Bildschirmbereich liegt
- 25Ø : Ist der Y-Wert zulässig, so wird der Punkt X,Y gezeichnet
- 26Ø : Schleifenende, d.h. Übergang zum nächsten X-Wert

Satz 27Ø-33Ø : Durchführung der gleichen Prozedur wie in Satz 2ØØ-26Ø nun für den oberen Halbkreis

Satz 34Ø : Warteschleife

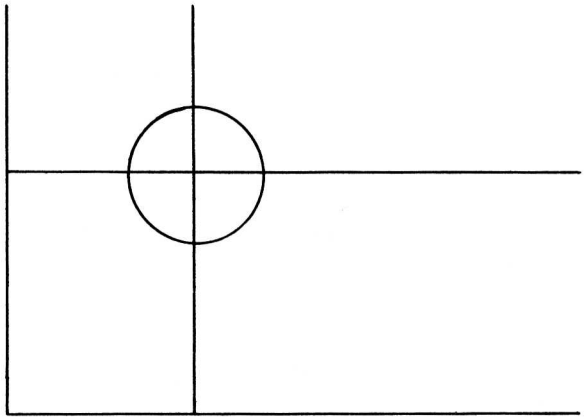
Satz 35Ø : Zurückschalten in den Textmodus

Satz 36Ø : Beendigung des Programms

• 6. Ergebnisse

Wenn wir nach dem Start dieses Programms zum Beispiel für T den Wert 1ØØ, für Z den Wert 8Ø und als Kreisradius R den Wert 4Ø eingeben, so erzeugt das Programm auf dem

Bildschirm eine Zeichnung, die ungefähr der folgenden Skizze entspricht :



TRAININGSAUFGABE 3 : Sinuslinie

1. Vorstellung des Problems

In einem letzten Beispiel zur hochauflösenden Graphik soll ein BASIC-Programm vorgestellt werden, welches auf dem Bildschirm eine Sinuslinie zeichnet.

2. Problemanalyse

Bei dieser Aufgabenstellung bieten sich unter problemanalytischen Gesichtspunkten keine neuen Probleme :

Als Input-Informationen benötigt das Programm die Amplitude der Schwingung, wobei wieder darauf zu achten ist, daß der zulässige Bildschirmbereich nicht verlassen wird, und die Frequenz der Schwingung. Diese Frequenz muß in Einheiten des Kreisparameters (3.14...) angegeben werden.

- 3. Programm

```
10 REM T34-SINUS
20 PRINT CHR$(125)
30 PRINT "PROGRAMM ZUM ZEICHNEN EINER SINUSLINIE"
35 PRINT :PRINT :PRINT
40 PRINT "        PROF.DR.W.VOSS, 1984"
50 PRINT :PRINT :PRINT :PRINT
60 PRINT "DAS PROGRAMM BENOETIGT DIE FOLGENDEN"
70 PRINT "INPUT-INFORMATIONEN :":PRINT :PRINT
80 PRINT "FREQUENZ (IN EINHEITEN VON PI) ";
85 INPUT P
90 PRINT "AMPLITUDE (Ø BIS 79)           ";
95 INPUT A
100 GRAPHICS 8: COLOR 1
110 PLOT Ø,8Ø:DRAWTO 319,8Ø
120 FOR X=Ø TO 319
130 J=X/5Ø:J=J*(1/P)
140 Y=SIN(J):Y=Y*A
150 Y=8Ø-Y
160 PLOT X,Y
170 NEXT X
180 FOR I=1 TO 2ØØØ:NEXT I
190 GRAPHICS Ø
200 PRINT "ENDE DER AUSGABE":END
```

4. Variablenliste

A = Amplitude der Schwingung
I = Laufindex
J = Normierter X-Wert (es wird so normiert, daß der auf dem Bildschirm zur Verfügung stehende Platz geschickt ausgenutzt wird :

$$J = \frac{X}{5Ø} * \frac{1}{P}$$

P = Frequenz der Schwingung (in Einheiten des Kreisparameters und nicht in Winkelgraden)
X = X-Koordinate der Sinuslinie
Y = Y-Koordinate der Sinuslinie

5. Programmbeschreibung

Satz 1Ø-7Ø : Überschrift, Erläuterungen, Leerzeilen
Satz 8Ø-95 : Anforderung der beiden Input-Informationen
Satz 1ØØ : Umschalten in den Graphikmodus und Auswahl einer Farbe
Satz 11Ø : Zeichnen einer horizontalen Mittellinie
Satz 12Ø-17Ø : Schleife zum Zeichnen der Sinuslinie
12Ø : Schleifenbeginn

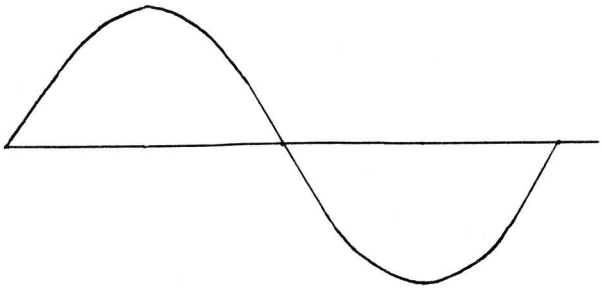
13Ø : Normierung von X, resultierend in der Variablen J
14Ø : Bestimmung des Sinuswertes von J und Veränderung dieses Wertes gemäß des Amplitudenparameters A
15Ø : Verschiebung in die Bildschirmmitte
16Ø : Zeichnen des Punktes mit den Koordinaten X und Y
17Ø : Schleifenende, d. h. Übergang zum nächsten X-Wert

Satz 18Ø : Warteschleife
Satz 19Ø : Zurückschalten in den Textmodus
Satz 2ØØ : Beendigung des Programms

• 6. Ergebnisse

Wenn wir dieses Programm starten und beispielsweise als Amplitudenwert den Wert 4Ø und als Frequenzwert den Wert Ø.5 eingeben, so erzeugt das Programm ein graphisches Bild, das ungefähr der folgenden Skizze entspricht :

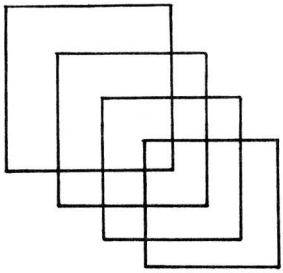
Sinuslinie mit A = 4Ø und P = Ø.5 :



Übungsaufgaben

1. Es soll ein BASIC-Programm entwickelt werden, welches nach dem Muster des vorangegangenen Trainingsbeispiels an eine beliebige Stelle des Bildschirms eine Ellipse zeichnet. Die beiden Halbachsen der Ellipse sollen vom Programmbenutzer frei gewählt werden können.

2. Es soll ein BASIC-Programm geschrieben werden, daß mehrere Quadrate in perspektivischer Darstellung zeichnet, so wie es schematisch die folgende Skizze zeigt :



Kapitel 1Ø : Hochauflösende Graphik
Übungsaufgaben

11

Kapitel 11 : Ergänzungen

Abschnitt 1 : Generelles

In einem abschließenden Kapitel sollen nun noch diejenigen Möglichkeiten des ATARI angesprochen werden, die in den vorangegangenen Kapiteln nicht besprochen und auch nicht benutzt wurden. Dabei werden wir uns aber sehr kurz fassen und auf die wichtigsten Stichworte beschränken.

Als erstes wollen wir in diesen ergänzenden Anmerkungen darauf hinweisen, daß man sich die Arbeit mit einem Rechner wie dem ATARI wesentlich vereinfachen kann, wenn man zum Speichern von Programmen (oder von anderen Informationen) ein Diskettenlaufwerk anschließt.

Ist dies der Fall, so muß zunächst das Disketten-Betriebssystem von der sog. Master-Diskette, die dem Laufwerk beigefügt ist, geladen werden. Man kann dabei folgendermaßen vorgehen :

1. Einschalten des Diskettenlaufwerks
2. Einlegen der Master-Diskette
3. Einschalten der Geräte (Monitor, Rechner)
4. Eingabe des Kommandos DOS, nachdem die READY-Meldung erschienen ist

   Es erscheint daraufhin auf dem Monitor der Inhalt der Master-Diskette. Will man nun beispielsweise den Inhalt der eigenen Diskette überprüfen, so braucht man nur F einzutippen (F = FILES = Dateien)

5. Einlegen der eigenen Diskette (diese muß natürlich formatiert sein. Wie man Disketten formatiert, ist im Handbuch des Diskettenlaufwerks ausführlich beschrieben)
6. Viermaliges Betätigen der RETURN-Taste

   Der Rechner listet nun den Inhalt der Diskette auf dem Bildschirm auf.

Will man ein Programm, das sich im Arbeitsspeicher des Rechners befindet, auf der eingelegten Diskette speichern, so benötigt man das folgende Kommando :

```
SAVE "D1:Name"
```

An der Stelle "Name" kann der Benutzer einen Namen für sein Programm frei wählen.

Will man ein Programm von der eingelegten Diskette in den Arbeitsspeicher des Rechners laden, so benötigt man das folgende Kommando :

```
LOAD "D1:Name"
```

Eine weitere generelle Ergänzung bezieht sich darauf, daß man Statements und Kommandos bei der Benutzung des ATARI-Rechners abkürzen kann und auf diese Weise beträchtlich an Eintipparbeit sparen kann.

Einige Beispiele für derartige Abkürzungen zeigt die folgende Tabelle :

Abkürzungen :

Schlüsselwort (BASIC)	Abkürzung
COLOR	C.
DATA	D.
DRAWTO	DR.
GOSUB	GOS.
GOTO	G.
INPUT	I.
LIST	L.
NEXT	N.
PLOT	PL.
POSITION	POS.
PRINT	PR. oder ?
RETURN	RET.
SAVE	S.
SETCOLOR	SE.

Diese Liste ist nicht vollständig. Weitere Abkürzungsmöglichkeiten sind dem ATARI BASIC Handbuch zu entnehmen.

Abschnitt 2 : BASIC-Statements

Ebenfalls ergänzend ist anzumerken, daß wir in den vorangegangenen Kapiteln nicht alle BASIC-Anweisungen besprochen oder benutzt haben, die uns der ATARI zur Verfügung stellt. Deshalb noch die folgenden stichwortartigen Hinweise :

```
ON ... GOTO
```

Mit diesem Statement können Mehrfachverzweigungen programmiert werden. Während zum Beispiel das Statement

```
80 GOTO 20
```

nur einen einzigen Sprung (eine einzige Verzweigung) ermöglicht, wie auch zum Beispiel

```
80 IF A = 16 THEN GOTO 20
```

ermöglicht das folgende Beispiel einer Programmzeile eine Mehrfachverzweigung :

```
80 ON A GOTO 20,300,500
```

Erreicht das Programm dieses Statement, so erfolgt

- ein Sprung zum Satz 2Ø, wenn A den Wert 1 aufweist,
- ein Sprung zum Satz 3ØØ, wenn A den Wert 2 aufweist,
- ein Sprung zum Satz 5ØØ, wenn A den Wert 3 aufweist,

andernfalls erfolgt kein Sprung.

ON ... GOSUB

Dieses Statement wirkt genauso wie das eben besprochene ON ... GOTO-Statement, mit dem Unterschied, daß nun unterschiedliche Unterprogramme angesprungen werden können. Ein Beispiel dürfte an dieser Stelle entbehrlich sein.

LPRINT

Dieses BASIC-Statement schickt Programmergebnisse nicht auf den Bildschirm, sondern auf einen angeschlossenen Drucker.

POKE

Mit dieser Anweisung kann eine beliebige auszuwählende Speicherstelle des Arbeitsspeichers des Rechners mit

einem ASCII-Codezeichen belegt werden, wenn dessen Codezahl (zwischen Ø und 255; vergl. ASCII-Code-Tabelle) angegeben wird. Die Speicheradressen liegen zwischen Ø und 65535.

PEEK

Mit dieser Anweisung kann der Inhalt einzelner Speicherstellen gelesen werden (also umgekehrt wie POKE)

SOUND

Diese Anweisung dient dazu, mit dem ATARI Töne zu erzeugen. Obwohl es mit dieser Anweisung möglich ist, ganze Musikstücke zu komponieren, haben wir ihr kein eigenes Trainingskapitel gewidmet, weil wir der Auffassung sind, daß derartige Programme schon eher in den Spielbereich gehören. Und zu zeigen, wie der ATARI als Spielcomputer eingesetzt werden kann, war nicht Hauptanliegen dieses Buches.

Die Anweisung SOUND benötigt vier verschiedene Parameterwerte, also schematisch :

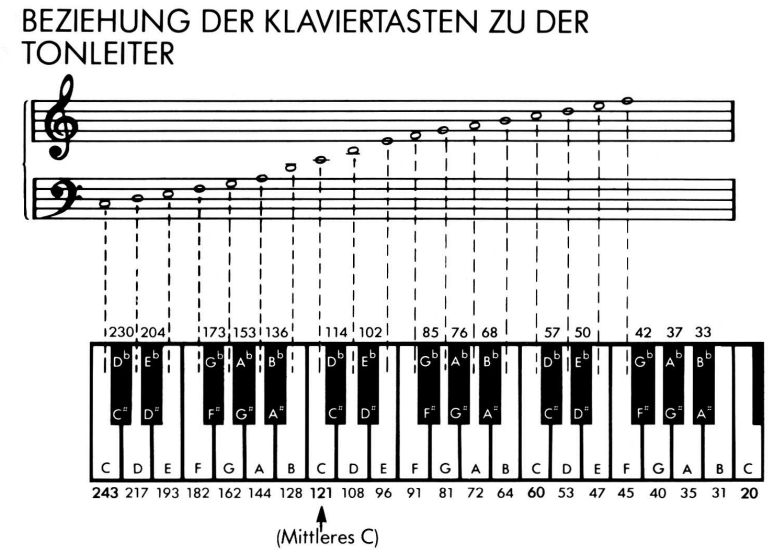
```
SOUND Zahl1, Zahl2, Zahl3, Zahl4
```

Die einzelnen Parameter in diesem Statement haben die folgende Bedeutung :

Zahl1 : Mit diesem Parameter, der Werte zwischen Ø und 3 annehmen kann, wird der jeweilige Tonkanal ausgewählt.

Zahl2 : Mit diesem Parameter wird die Tonhöhe bestimmt: Je größer der Zahlenwert, desto tiefer ist die Tonfrequenz.

Die folgende Übersicht gibt über die Tonhöhen und die dazugehörigen Parameterwerte Auskunft:



Quelle : ATARI BASIC Handbuch, Seite 15.

Zahl3 : Dieser Parameter sorgt für Verzerrungen der Töne. Es sind ganzzahlige Werte zwischen Ø und 14 zugelassen.

Nur die Parameterwerte 4 und 1Ø ergeben "reine" Töne.

Zahl4 : Dieser Parameter regelt die Lautstärke der erzeugten Töne : Je höher der Parameterwert ist, desto lauter ist der jeweilige Ton. Zugelassen sind die Werte Ø bis 15.

Das folgende Programm erzeugt beispielsweise einen sehr schönen C-Dur-Dreiklang (c-e-g) :

```
10 FOR I=1 TO 100
20 SOUND 1,126,10,10
30 SOUND 2, 96,10,10
40 SOUND 3, 81,10,10
50 NEXT I
```

In entsprechender Weise erzeugt das folgende Programm einen nicht mehr endenden Sirenenton (Abbruch durch die RESET-Taste !) :

```
10 FOR I=32 TO 251:SOUND 1,I,10,10:NEXT I
20 FOR I=251 TO 32 STEP-1:SOUND 1,I,10,10:NEXT I
30 GOTO 10
```

Auf weitere Beispiele zur Benutzung dieses interessanten Statements soll hier nicht eingegangen werden. Hier eröffnet sich für den ATARI-Benutzer ein breites Experimentierfeld. Beispielsweise kann er sich bei der Pro-

grammierung angewöhnen, an besonders wichtigen Stellen in die Programme Tonerzeugungen einzustreuen. So könnte etwa jede INPUT-Anweisung durch einen bestimmten Ton unterstrichen werden, der auch akustisch erkennen läßt, daß nun das Programm eine Eingabe erwartet. Weitere Anwendungen könnten die eventuell benutzten STOP-Statements sein, bei denen ein akustisches Signal den vielleicht vorm Rechner eingeschlafenen Benutzer aufweckt, um mitzuteilen, daß nun das Programm unterbrochen ist.

Ohne Schwierigkeiten wird man sich viele ähnliche oder auch ganz andere Anwendungsbeispiele des SOUND-Statements ausdenken können.

A N H Ä N G E

Anhang 1 : Übungsaufgaben

In diesem ersten Anhang sollen zunächst die Lösungen der Übungsaufgaben vorgestellt werden, die wir dem Leser am Ende der einzelnen Kapitel gestellt haben.

Eine wichtige Anmerkung ist vor der Präsentation dieser Ergebnisse allerdings erforderlich :

Zur Lösung eines bestimmten Problems mit Hilfe eines BASIC-Programms gibt es in der Regel nicht nur einen einzigen Lösungsweg. Vielmehr muß man davon ausgehen, daß verschiedene, unter Umständen sich sehr stark voneinander unterscheidende Wege zur Lösung eines Problems führen können.

Der Leser braucht sich deshalb nicht zu wundern, wenn ein Lösungsweg, den er sich in mühsamer gedanklicher Arbeit selbst ausgedacht hat, vielleicht nicht mit dem übereinstimmt, den wir hier vorführen. Entscheidend ist in diesem Zusammenhang nur, ob dieser, sein eigener Lösungsweg zum gleichen Endergebnis führt, wie diejenige Programmversion, die wir hier als Problemlösung vor-führen. Ist dies der Fall, so spricht nichts gegen die gefundene Lösung.

Allerdings dürfte es auch dann noch sinnvoll und lehrreich sein, die verschiedenen Lösungsmöglichkeiten miteinander zu vergleichen, um ihre jeweiligen Vorzüge und eventuellen Nachteile festzustellen und zu bewerten.

Auf diese Weise wird man dann auf Dauer zu ständig verbesserten Programmen und Programmstrukturen gelangen.

Eine weitere Anmerkung ist erforderlich :

Wie schon bei den Trainingsaufgaben in den vorangegangenen Kapiteln legen wir bei den folgenden Problemlösungen unser Hauptaugenmerk nicht auf größtmögliche Programmeleganz oder auf besonders hohe Rechengeschwindigkeit, sondern wieder darauf, daß die Programme leicht nachvollziehbar, d.h. lesbar und verständlich bleiben. Deshalb ist es durchaus denkbar, daß der Leser zu wesentlich besseren Programmen gelangt als sie hier vorgestellt werden. Insbesondere wird es häufig vielleicht so sein, daß er kürzere Programme schreibt als die hier vorgelegten. Aber auch in diesem Zusammenhang gilt wieder :

Ein bestimmtes Problem kann durch sehr unterschiedliche Programme gleich gut (wenn man das jeweilige Programmergebnis im Auge hat) gelöst werden.

Bei den Programmen, die hier als Problemlösungen vorgestellt werden, verfahren wir so ähnlich wie in den einzelnen Trainingskapiteln : Wir stellen nicht nur die

Anhang 1 : Übungsaufgaben

Programme selbst vor, sondern wir liefern dazu auch stichwortartige Programmbeschreibungen und fügen von Fall zu Fall erläuternde Erklärungen an, sofern dies erforderlich ist.

Die folgenden Lösungen der einzelnen Übungsaufgaben sind kapitelweise geordnet.

Anhänge
Anhang 1 : Übungsaufgaben

zu Kapitel 1

1. Ein Feld ist der Kreuzungsbereich von einer Spalte (Array) und einer Zeile (Record) der Datenmatrix; beispielsweise belegt in der angegebenen Datenmatrix die Körpergröße einer Person ein Feld.

   Ein Array ist die Gesamtheit aller Felder in einer Spalte der Datenmatrix; es handelt sich also dabei um die Gesamtheit der Werte (Ausprägungen) einer Variablen; beispielsweise bildet die Gesamtheit a11er angegebenen Körpergrößen einen Array.

   Ein String ist eine Zeichenkette, also eine Folge von Symbolen (,die in BASIC in Anführungszeichen eingeschlossen werden muß); beispielsweise ist "MEIER" ein String.

   Eine numerische Variable ist eine Variable, die als Werte (Ausprägungen) nur Zahlen annehmen kann; beispielsweise ist "Alter" eine numerische Variable.

   Ein Wert ist eine Ausprägung einer Variablen; beispielsweise ist 42 eine Ausprägung der Variablen "Alter", also ein Wert dieser Variablen; entsprechend ist "MEIER" ein Wert der Variablen "Namen".

Eine Datei (File) ist die Gesamtheit aller Datensätze bzw. die Gesamtheit aller Arrays; dies bedeutet, daß die gesamte Datenmatrix als Datei aufgefaßt werden kann.

Ein numerischer Wert ist eine Zahl, bzw. die Ausprägung einer numerischen Variablen.

Ein Symbol (Character) ist ein Zeichen, d.h. ein Element eines Feldes.

Symbole können Buchstaben, Ziffern oder Sonderzeichen sein; jedes dieser Symbole belegt im Rechner ein Byte.

2. Das EVA-Prinzip besagt, daß bei der Analyse eines Datenverarbeitungsprozesses zunächst zu untersuchen ist,

welche Eingabeinformationen das zu entwickelnde Programm zur Lösung des gestellten Problems benötigt,	E
welche Verarbeitungsschritte dann notwendig sind, um die Eingabeinformationen in angemessener Weise zu verarbeiten	V
und welche Ausgabeinformationen dann bereitgestellt werden sollen.	A

3. Der Unterschied zwischen Kommandos und Programmanweisungen besteht darin, daß Kommandos direkt ausführbare Mitteilungen an das Betriebssystem des Rechners darstellen, während Programmanweisungen

nicht direkt ausgeführt, sondern zunächst "aufgesammelt“ werden, bis das Programm beendet ist und per RUN-Kommando gestartet wird.

Der Rechner erkennt BASIC-Programmanweisungen daran, daß sie mit Satznummern eingeleitet Werden, während Kommandos ohne Nummern eingegeben werden.

4. Unter ROM-Speichern versteht man diejenigen Bereiche des Arbeitsspeichers des Rechners, die gegen Überschreibungen durch den Benutzer geschützt sind. Sie enthalten Informationen, die dem Rechner immer zur Verfügung stehen müssen (zum Beispiel Teile des Betriebssystems), und die auch beim Ausschalten des Rechners nicht verloren gehen (ROM = Read Only Memory = Speicher, der nur gelesen werden kann).

   Der RAM-Speicher hingegen ist derjenige Teil des Arbeitsspeichers des Rechners, der vom Benutzer beschrieben oder auch gelesen werden kann. Hier sind also Überschreibungen möglich (und nützlich), d.h. mit diesem Speicherbereich kann der Benutzer konkret arbeiten (RAM = Random Access Memory = Speicher mit wahlfreiem Zugriff zum Lesen und Schreiben).

5. Ein Byte ist diejenige Speicherstelle im Arbeitsspeicher des Rechners, die ein Symbol aufnehmen kann. Ein Byte kann mit einer Folge von acht Nullen und/oder Einsen belegt werden, so daß insgesamt 28 = 256 verschiedene Symbole darstellbar sind.

6. Die Binärzahl 11Ø1Ø11Ø1 läßt sich ausführlich schreiben als :

$$1*2^8 + 1*2^7 + Ø*2^6 + 1*2^5 + Ø*2^4 + 1*2^3 + 1*2^2 + Ø*2^1 + 1*2^Ø$$

$$= 256 + 128 + Ø + 32 + Ø + 8 + 4 + Ø + 1$$

$$= 429_{(1Ø)}$$

Sie entspricht also der Dezimalzahl 429.

7. Die Dezimalzahl 16Ø läßt sich wie folgt in eine Binärzahl zerlegen :

$$16Ø_{(1Ø)} = 1*128 + Ø*64 + 1*32 + Ø*16 + Ø*8 + Ø*4 + Ø*2 + Ø*1$$

$$= 1*2^7 + Ø*2^6 + 1*2^5 + Ø*2^4 + Ø*2^3 + Ø*2^2 + Ø*2^1 + Ø*2^Ø$$

$$= 1Ø1ØØØØØ(2)$$

Sie entspricht also der Binärzahl 1Ø1ØØØØØ.

zu Kapitel 2

1. Die Fläche eines Rechtecks, das 7.4 cm lang und 2.Ø8 cm breit ist, kann mit dem ATARI im Direktmodus wie folgt berechnet werden :

```
PRINT 7.4 * 2.Ø8
```

Der ATARI gibt das folgende Ergebnis auf dem Bildschirm aus, wenn die obige Anweisung per RETURN-Taste abgeschickt wurde :

```
15.392
```

2. Die Fläche eines Kreises mit dem Radius 5 cm kann wie folgt berechnet werden :

```
PRINT 5*5*3.14
```

Es ergibt sich :

```
78.5
```

3. Die Mehrwertsteuer (14 %) bei einem Rechnungsbetrag (netto) von DM 45Ø.-- berechnet sich folgendermaßen :

```
PRINT 450 * 14/100
```

Es ergibt sich :

```
63
```

4. Lautet der Brutto-Rechnungsbetrag auf DM 45Ø.--, so erhalten wir die darin enthaltene Mehrwertsteuer mit der folgenden Anweisung :

```
PRINT (450/114)*14
```

Das Ergebnis lautet :

```
55.2631578
```

5. Die Fläche eines Trapezes berechnet sich, indem man die Höhe mit der Mittellinie multipliziert. Die Mittellinie wiederum ist der Mittelwert zwischen den beiden parallelen Trapezseiten. Somit gelangen wir zu der folgenden Anweisung :

```
PRINT 3.2 * (4.8 + 7.5)/2
```

Als Ergebnis erhalten wir bei dieser Flächenberechnung :

```
19.68
```

6. Der Rauminhalt eines Würfels mit der Kantenlänge 3.5 cm kann mit der folgenden Anweisung berechnet werden :

```
PRINT 3.5 ^ 3
```

Als Ergebnis erhalten wir :

```
42.8749886
```

zu Kapitel 3

1. Ein Flußdiagramm, wie das in dieser Übungsaufgabe vorgelegte, kann in der Weise überprüft werden, daß man sich Schritt für Schritt durch das Diagramm hindurcharbeitet und bei jedem Schritt notiert, wie (und ob) sich die einzelnen Felderbesetzungen verändern.

   Auf diese Weise kann man dann auch feststellen, was der Inhalt derjenigen Felder ist, die in PRINT-Anweisungen angesprochen werden, die also als Ergebnisse auf dem Bildschirm erscheinen.

   Übrigens empfiehlt sich diese Vorgehensweise insbesondere auch dann, wenn man feststellen will, warum ein Programm nicht das tut, was es eigentlich tun soll, wenn man also auf der Fehlersuche ist.

   Deshalb ist dieses Durcharbeiten von Flußdiagrammen (oder auch von fertigen Programmen) eine sehr wichtige und nützliche Übung. Man geht dabei so vor, daß man sich zunächst notiert, welche Felder überhaupt vorkommen :

   In dem vorliegenden Flußdiagramm sind das die folgenden Felder :

I und S

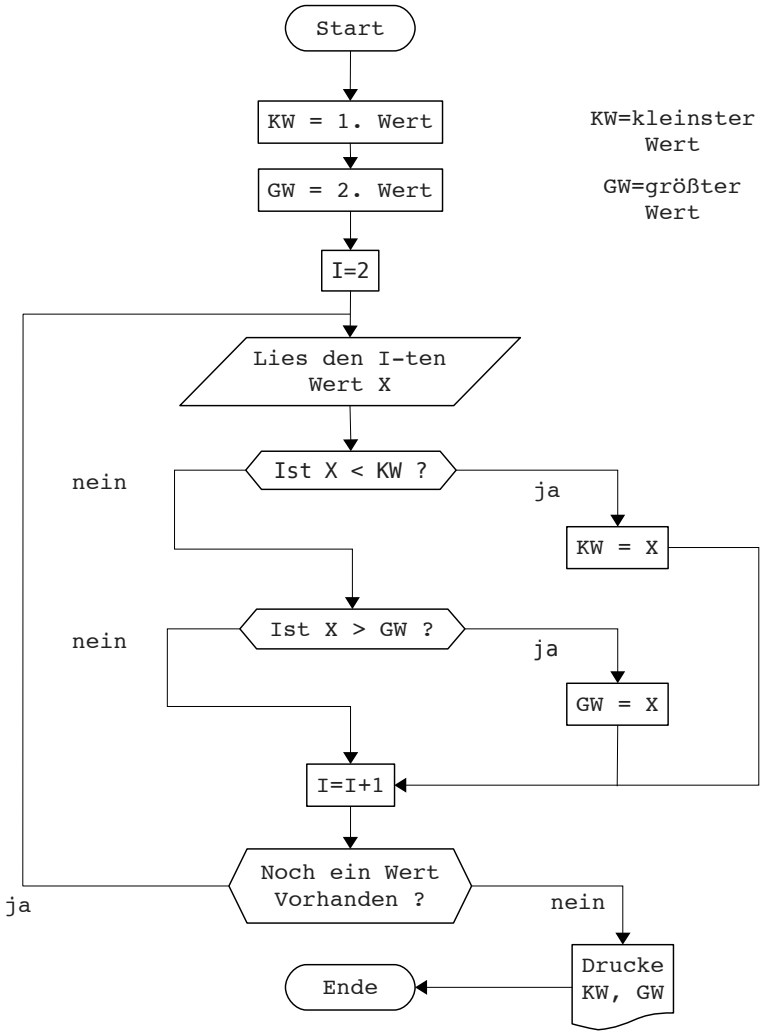
Die Besetzung dieser Felder im Ablauf wird aus der folgenden Tabelle deutlich :

I	S	
20	0	
	20	
17		I ist nicht kleiner 11
	37	
14		I ist nicht kleiner 11
	51	
11		I ist nicht kleiner 11
	62	
8		I ist kleiner 11, deshalb Ausgabe :
		I = 8 S = 62

Die obige Tabelle zeigt, wenn sie von oben nach unten zeilenweise gelesen wird, die Besetzung der Felder und ihre Veränderungen und führt somit zum gewünschten Ergebnis.

2. Soll aus einem Datenbestand der größte und der kleinste Welt herausgesucht werden, so benötigt man ein Flußdiaqramm, wie es die folgende Abbildung darstellt :

Flußdiagramm zur Aufgabe 2 :



Ein solches Flußdiagramm kann man in seiner Wirkungsweise sehr einfach dadurch überprüfen, daß man einen kleinen "Spieldatenbestand" vergibt und mit diesem den Programmablauf erprobt.

Beispielsweise seien die folgenden Daten geqeben :

15, 8, 27, 39, 9

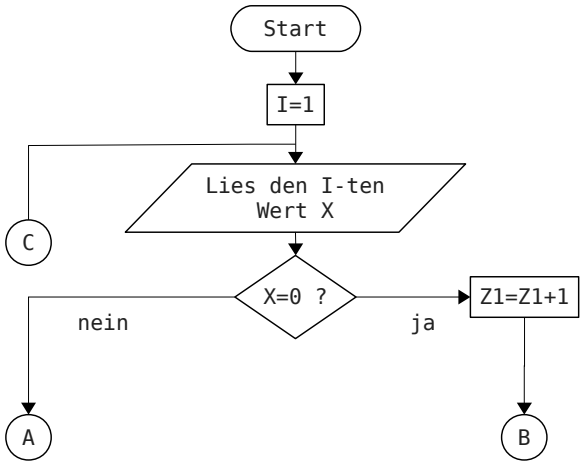
Die Felderbelegungen während des Programmablaufs sind nun die folgenden :

KW	GW	I	X	
15	15			
		2	8	Ist 8 kleiner 15 ?
8				Antwort : Ja, deshalb KW=8
				Ist 8 größer 15 ?
				Antwort : Nein
		3	27	Ist 27 kleiner 8 ?
				Antwort : Nein
				Ist 27 größer 15 ?
	27			Antwort : Ja, deshalb GW=27
	39			Ist 39 kleiner 8 ?
				Antwort : Nein
				Ist 39 größer 27 ?
				Antwort : Ja, deshalb GW=39
				Ist 9 kleiner 8 ?
				Antwort : Nein
				Ist 9 größer 39 ?
				Antwort : Nein
				Es gibt keinen 6.Wert mehr; deshalb Ausgabe der Inhalte der Felder KW und GW, d.h. der korrekten Werte 8 und 39.

4. Es handelt sich bei diesem Problem um die Aufstellung einer sog. Häufigkeitsverteilung. Wir gehen dabei sinnvollerweise so vor, daß jeder eingegebene Wert darauf geprüft wird, ob er Ø, 1, 2, 3 oder 9 ist. Ist der jeweilige Wert Ø, so wird ein erster Zähler um 1 erhöht, ist er 1, so wird ein zweiter Zähler um 1 erhöht usw.

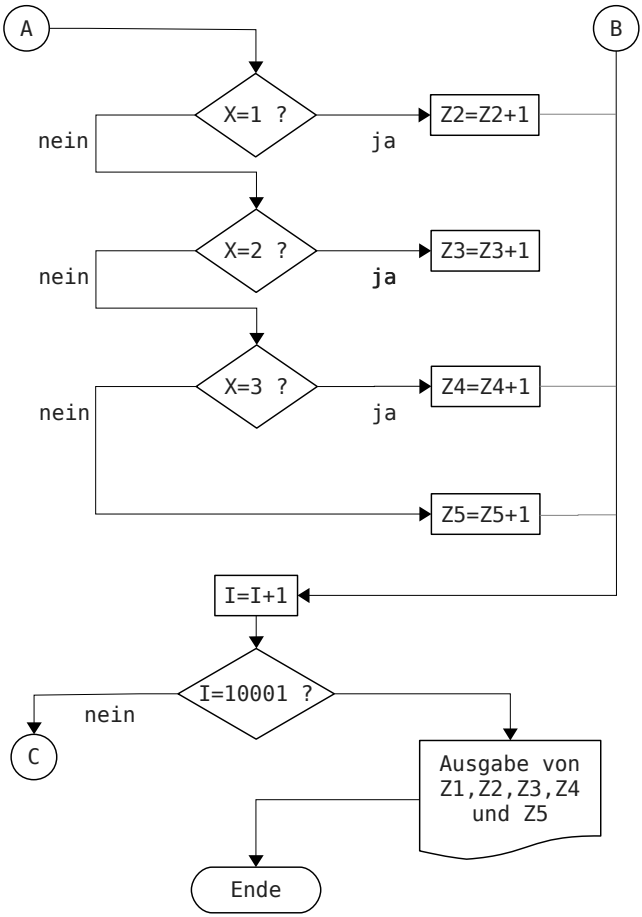
   Am Schluß dieses Klassifizierungsprozesses stehen in den Zählfeldern die gesuchten Häufigkeiten für die einzelnen Ausprägungen der statistischen Variablen "Familienstand".

   Ein Flußdiagramm, das diesen Auszählprozeß beschreibt, könnte folgendermaßen aussehen :



Anhänge

Anhang 1 : Übungsaufgaben



zu Kapitel 4

1. Ein BASIC-Programm, welches die Fläche beliebiger Dreiecke ausrechnen soll, muß von der folgenden generellen Flächenberechnungsformel ausgehen :

   F = (c * h)/2

   Die Dreiecksfläche F ergibt sich aus dem Produkt aus Grundseite mal Höhe, geteilt durch zwei.

   Das folgende Programm setzt diese Formel in die BASIC-Schreibweise um :

```
10 REM U1-DREIECKSBERECHNUNGEN
20 PRINT CHR$(125)
30 PRINT "                DREIECK"
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET DIE FLAECHE"
80 PRINT "         BELIEBIGER DREIECKE."
110 PRINT :PRINT :PRINT
120 PRINT "DAS PROGRAMM BENOETIGT DIE FOLGENDEN"
130 PRINT "INPUT-INFORMATIONEN :":PRINT
140 PRINT "     GRUNDSEITE :  ";:INPUT C
150 PRINT "     HOEHE      :  ";:INPUT H
160 F=(C*H)/2
170 PRINT :PRINT :PRINT
180 PRINT "DIE DREIECKSFLAECHE BETRAEGT ";F
240 PRINT :PRINT :PRINT
250 PRINT "ENDE DER BERECHNUNGEN"
260 PRINT "NEUSTART NUR MIT  RUN"
270 END
```

Variablenliste :

C = Grundseite des Dreiecks
F = Fläche des Dreiecks
H = Höhe des Dreiecks

Programmbeschreibung :

Satz 1Ø-13Ø	:	Überschrift und Erläuterungen
Satz 14Ø-15Ø	:	Anforderung der Input-Informationen
Satz 16Ø	:	Berechnung der Dreiecksfläche
Satz 17Ø-24Ø	:	Ausgabe des Berechnungsergebnisses
Satz 25Ø-27Ø	:	Beendigung des Programms

2. Das folgende Programm rechnet aus, wie groß die Bevölkerung eines Landes nach einer beliebigen Anzahl von Jahren wird, wenn die Bevölkerungswachstumsrate und der Ausgangsbestand beliebig vorgegeben werden können.

   Dies bedeutet, daß das Programm mit drei, vom Benutzer vorgebbaren Parametern arbeitet :

   - Ausgangsbestand (BØ)
   - Jährliche Wachstumsrate (P)
   - Dauer des Wachstumsprozesses in Jahren (T)

   Die Bevölkerungszahl nach T Jahren, die sich bei einer jährlichen Wachstumsrate von P % und einem Ausgangsbestand von BØ ergibt, berechnet sich nach der folgenden Formel :

   BT = BØ * (1 + p/1ØØ)T

   Das Programm, das sich dieser Berechnungsformel bedient, sieht folgendermaßen aus :

```
10 REM U2-BEVOELKERUNGSENTWICKLUNG
20 PRINT CHR$(125)
30 PRINT "     BEVOELKERUNGSENTWICKLUNG"
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET DIE ENT-"
80 PRINT "WICKLUNG EINER VORGEGEBENEN BEVOELKE-"
90 PRINT "                RUNG."
110 PRINT :PRINT :PRINT
120 PRINT "DAS PROGRAMM BENOETIGT DIE FOLGENDEN"
130 PRINT "INPUT-INFORMATIONEN :":PRINT
140 PRINT "      URSPRUENGLICHE BEV.ZAHL : ";
145 INPUT B0
150 PRINT "      JAEHRL.ZUWACHS IN %     : ";
155 INPUT P
160 PRINT "      ANZAHL DER JAHRE        : ";
165 INPUT T
168 BT=B0*(1+P/100)^T
170 PRINT :PRINT :PRINT
180 PRINT "NACH ";T;" JAHREN BETRAEGT DIE "
190 PRINT "BEVOELKERUNGSZAHL : ";BT
240 PRINT :PRINT :PRINT
250 PRINT "ENDE DER BERECHNUNGEN"
260 PRINT "NEUSTART NUR MIT  RUN"
270 END
```

Variablenliste :

BØ = Ausgangsbestand
BT = Bevölkerungszahl zum Zeitpunkt T (nach T Jahren)
P = Bevölkerungswachstumsrate in Prozent
T = Anzahl der Jahre

Programmbeschreibung :

Satz 1Ø-13Ø : Überschrift und Erläuterungen
Satz 14Ø-165 : Anforderung der benötigten Input-Informationen
Satz 168 : Berechnung der Bevölkerungszahl zum Zeitpunkt T
Satz 17Ø-24Ø : Präsentation der Ergebnisse
Satz 25Ø-27Ø : Beendigung des Programms

Es ist leicht einsichtig, daß ein derartiges Programm auch bei Berechnungen zu anderen Wachstumsprozessen verwendet werden kann.

3. Das folgende Programm dient dazu, auszurechnen, wieviel Prozent ein bestimmter absoluter Wert, gemessen an einem anderen Absolutwert ausmacht. Es beantwortet also beispielsweise die folgende Frage :

   Wieviel Prozent sind 4Ø DM verglichen mit 2ØØ DM ?

   Antwort : 2Ø %

   Der mathematische Hintergrund einer derartigen Berechnung ist die Dreisatzrechnung, deren Grundkonzept wir hier sicherlich nicht vorzuführen brauchen.

   Das Programm soll allgemein gehalten werden, d.h. sowohl der Basis-(Vergleichs-)Wert, wie auch derjenige Wert, dessen Prozentanteil bestimmt werden soll, müssen vom Programmbenutzer frei eingegeben werden können.

   Das entsprechende Programm sieht nun folgendermaßen aus :

```
10 REM U3-PROZENTRECHNEN
20 PRINT CHR$(125)
30 PRINT "          PROZENTRECHNEN."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIES IST EIN PROZENTBERECHNUNGSPRO-"
80 PRINT "              GRAMM."
110 PRINT :PRINT :PRINT
120 PRINT "DAS PROGRAMM BENOETIGT DIE FOLGENDEN"
130 PRINT "INPUT-INFORMATIONEN :":PRINT
140 PRINT "     BASISWERT           : ";
145 INPUT X
146 PRINT
150 PRINT "     WERT, DESSEN PROZENT-"
152 PRINT "     ANTEIL BESTIMMT WER-"
154 PRINT "     DEN SOLL            : ";
156 INPUT Y
160 P=(Y/X)*100
170 PRINT :PRINT :PRINT
180 PRINT "DER PROZENTUALE ANTEIL VON ";Y
190 PRINT "AN ";X;" BETRAEGT ";P;" %"
240 PRINT :PRINT :PRINT
250 PRINT "ENDE DER BERECHNUNGEN"
260 PRINT "NEUSTART NUR MIT  RUN"
270 END
```

Variablenliste :

P = Prozentwert
X = Basiswert
Y = Zu vergleichender Wert

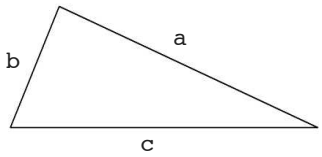
Programmbeschreibung :

Satz 1Ø-13Ø	:	Überschrift und Erläuterungen
Satz 14Ø-156	:	Anforderung der benötigten Input-Informationen
Satz 16Ø	:	Berechnung des Prozentwertes
Satz 17Ø-24Ø	:	Ergebnisausgabe
Satz 25Ø-27Ø	:	Beendigung des Programms

zu Kapitel 5

1. Das folgende Programm berechnet die Hypothenuse eines beliebigen rechtwinkligen Dreiecks.

   Zugrunde liegt der Lehrsatz des Pythagoras, der an der nebenstehenden Skizze erläutert werden kann und zu der folgenden Berechnungsformel für die Hypothenuse c führt :



$$c = \sqrt{a^2 + b^2}$$

   Damit das Programm generell einsetzbar ist, muß dem Benutzer Gelegenheit geboten werden, die beiden Dreiecksseiten a und b beliebig eingeben zu können.

   Dieser Forderung entspricht das folgende BASIC-Programm :

```
10 REM U4-PYTHAGORAS
20 PRINT CHR$(125)
30 PRINT "              PYTHAGORAS."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET DIE HYPO-"
80 PRINT "THENUSE EINES BELIEBIGEN RECHTWINKLI-"
90 PRINT "            GEN DREIECKS."
100 FOR I=1 TO 5:PRINT :NEXT I
110 PRINT "KATHETE A :  ";:INPUT A
120 PRINT "KATHETE B :  ";:INPUT B
140 C=SQR(A*A+B*B)
150 PRINT :PRINT
160 PRINT "HYPOTHENUSE :  ";C
170 PRINT :PRINT :PRINT
250 PRINT "ENDE":END
```

Variablenliste :

A = Erste Kathete
B = Zweite Kathete
C = Zu berechnende Hypothenuse

Programmbeschreibung :

Satz 1Ø-9Ø	:	Überschrift und Erläuterungen
Satz 1ØØ	:	Eingabe der benötigten Input-Informationen
Satz 11Ø-12Ø	:	Berechnung der Hypothenuse gemäß des Lehrsatzes des Pythagoras
Satz 15Ø-17Ø	:	Ausgabe des Ergebnisses
Satz 18Ø	:	Beendigung des Programms

2. Das folgende Programm berechnet die Summe aller ganzen Zahlen von 1 bis 1ØØ (Gauß–Problem).

Um diese Aufgabe zu bewältigen, wird lediglich eine geeignete Programmschleife benötigt, wie aus dem folgenden Programm deutlich hervorgeht :

```
10 REM U5-GAUSS
20 PRINT CHR$(125)
30 PRINT "                GAUSS."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET DIE SUMME"
80 PRINT "ALLER GANZEN ZAHLEN VON 1 BIS 100."
100 FOR I=1 TO 5:PRINT :NEXT I
110 FOR I=1 TO 100
120 S=S+I
130 NEXT I
140 PRINT :PRINT :PRINT
150 PRINT "SUMME  =  ";S
160 PRINT :PRINT "ENDE":END
```

Variablenliste :

I = Laufindex (und Summand)
S = Summe

Programmbeschreibung :

Satz 1Ø-8Ø	:	Überschrift und Erläuterungen
Satz 1ØØ	:	Fünf Leerzeilen
Satz 11Ø	:	Beginn der Schleife (1ØØ Runden)
Satz 12Ø	:	Summenbildung
Satz 13Ø	:	Schleifenende (nächster Summand)
Satz 14Ø-15Ø	:	Ausgabe des Ergebnisses
Satz 16Ø	:	Beendigung des Programms

3. Das folgende Programm berechnet Fakultäten.

Unter der Fakultät einer ganzen positiven Zahl n versteht man die folgende Produktkette :

n! = n*(n-1)*(n-2)*(n-3)* ... * 3 * 2 * 1

Demnach ist zum Beispiel 5! (sprich : Fünf-Fakultät) gleich :

5! = 5 * 4 * 3 * 2 * 1 = 12Ø

Es ist in diesem Zusammenhang zu beachten, daß mit wachsenden Werten von n die dazugehörigen Fakultäten sehr rasch anwachsen. Beispielsweise ist 34! schon so groß, daß die entsprechende Berechnung durch den ATARI nicht mehr durchgeführt werden kann.

Das folgende Programm berechnet derartige Fakultäten für beliebige Werte von n, sofern der vom Benutzer eingegebene Wert für n kleiner als 34 ist :

Anhänge

Anhang 1 : Übungsaufgaben

```
10 REM U6-FAKULTAETEN
20 PRINT CHR$(125)
30 PRINT "            FAKULTAETEN."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BERECHNET DIE FAKUL-"
80 PRINT "TAET  N! = N*(N-1)*(N-2)* ... 3*2*1"
90 DIM A$(1)
100 FOR I=1 TO 5:PRINT :NEXT I
110 PRINT "BITTE DIE GANZE POSITIVE"
120 PRINT "ZAHL EINGEBEN, DEREN FA-"
130 PRINT "KULTAET GEWUENSCHT WIRD: ";
140 INPUT N
150 IF N>33 THEN 230
160 P=N
170 FOR I=N-1 TO 1 STEP -1
180 P=P*I
190 NEXT I
200 PRINT :PRINT :PRINT
210 PRINT "DIE FAKULTAET VON ",N
220 PRINT "BETRAEGT : ";P:GOTO 260
230 PRINT :PRINT :PRINT
240 PRINT "DIE EINGEGEBENE ZAHL IST FUER DIESEN"
250 PRINT "RECHNER ZU GROSS."
260 PRINT :PRINT :PRINT
270 PRINT "NOCH EINE RECHNUNG (J/N) ";
280 INPUT A$
290 IF A$="J" THEN PRINT CHR$(125):GOTO 110
300 PRINT :PRINT "ENDE":END
```

Variablenliste :

I = Laufindex (bzw. Faktor in der Produktkette)
N = Zahl, deren Fakultät gewünscht wird
P = Produkt (Fakultät)

Programmbeschreibung :

Satz 1Ø-1ØØ	:	Überschrift und Erläuterungen
Satz 11Ø-14Ø	:	Anforderung der Zahl, deren Fakultät gewünscht wird
Satz 15Ø	:	Wenn die eingegebene Zahl N grösser als 33 ist, kann die Fakultät nicht berechnet werden; deshalb Sprung zum Satz 23Ø
Satz 16Ø-19Ø	:	Berechnung der Fakultät in einer Programmschleife
Satz 2ØØ-22Ø	:	Ausgabe des Ergebnisses und Sprung zum Satz 26Ø
Satz 23Ø-25Ø	:	Ausgabe einer Meldung, falls N zu groß war
Satz 26Ø-28Ø	:	Abfrage, ob noch eine weitere Berechnung gewünscht wird
Satz 29Ø	:	Wenn ja, Löschen des Bildschirms und zurück zum Satz 11Ø
Satz 3ØØ	:	Wenn nein, Beendigung des Programms

4. Das folgende Programm simuliert 1ØØ Würfelwürfe und gibt die Häufigkeitsverteilung der gewürfelten Augenzahlen aus.

Das Simulieren von Würfelwürfen bedeutet, daß Zufallszahlen zwischen 1 und 6 erzeugt werden müssen. Die Zufallsfunktion RND erzeugt Zufallszahlen zwischen Ø bis unter 1.

Wenn man eine solche Zufallszahl mit 6 multipliziert, dann erhält man eine Zufallszahl zwischen Ø.ØØØ und 5.999... . Addiert man jetzt noch eine Eins hinzu, so erhält man eine Zufallszahl zwischen 1.ØØØ und 6.999... . Wendet man schließlich auf diese Zahl die INT-Funktion an, so erhält man eine ganze Zufallszahl zwischen 1 und 6.

Die Erzeugung einer solchen Zufallszahl in BASIC sieht folgendermaßen aus :

```
W = INT(RND(1) * 6 + 1)
```

Der Leser möge sich überlegen, wie nach diesem Muster Zufallszahlen zwischen 1 und 49 erzeugt werden können, mit deren Hilfe vielleicht Lotto-Ergebnisse simuliert werden können.

Die Erstellung der gewünschten Häufigkeitsverteilung der Würfel-Augenzahlen bereitet keine neuen Schwierigkeiten : Es ist im Prinzip genauso vorzugehen, wie in einem früher geübten Flußdiagramm, in dem

eine Häufigkeitsverteilung über die statistische Variable "Familienstand" erzeugt werden sollte :

Wir vergleichen jedes "Würfelergebnis" mit den Zahlen 1 bis 6; wenn bei einem dieser Vergleiche Identität festgestellt wird, ist einer von sechs Zählern um 1 zu erhöhen und das nächste Würfelergebnis wird betrachtet.

Das folgende Programm führt diese Arbeitsschritte durch :

```
10 REM U7-WUERFEL
20 PRINT CHR$(125)
30 PRINT "              WUERFEL."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60  PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM SIMULIERT 100 WUERFE"
80 PRINT "MIT EINEM WUERFEL, BESTIMMT DIE SICH"
90 PRINT "ERGEBENDE HAEUFIGKEITSVERTEILUNG UND"
100 PRINT "DIE DURCHSCHNITTLICHE AUGENZAHL."
110 PRINT :PRINT :PRINT
115 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN"
116 STOP
117 PRINT CHR$(125)
120 FOR I=1 TO 100
130 W=INT(RND(1)*6+1)
140 PRINT W;" ";
150 S=S+W
```

```
160 IF W=1 THEN H1=H1+1:GOTO 220
170 IF W=2 THEN H2=H2+1:GOTO 220
180 IF W=3 THEN H3=H3+1:GOTO 220
190 IF W=4 THEN H4=H4+1:GOTO 220
200 IF W=5 THEN H5=H5+1:GOTO 220
210 H6=H6+1
220 NEXT I
230 PRINT :PRINT
240 PRINT "DURCHSCHNITT = ";S/100
250 PRINT :PRINT
260 PRINT "AUGENZAHL     HAEUFIGKEIT"
270 FOR I=1 TO 25:PRINT "-";:NEXT I
280 PRINT :PRINT
290 PRINT 1,,H1
300 PRINT 2,,H2
310 PRINT 3,,H3
320 PRINT 4,,H4
330 PRINT 5,,H5
340 PRINT 6,,H6
350 PRINT :PRINT "ENDE":END
```

Variablenliste :

H1 bis H6	=	Zählfelder für die Häufigkeiten der "gewürfelten" Augenzahlen
I	=	Laufindex
S	=	Summenfeld
W	=	Augenzahl bei einem "Würfelwurf"

Programmbeschreibung :

Satz 1Ø-11Ø	:	Überschrift und Erläuterungen
Satz 115-116	:	Unterbrechung des Programms
Satz 117	:	Löschen des Bildschirms
Satz 12Ø	:	Beginn der "Würfelschleife" (1ØØ mal)
Satz 13Ø	:	Simulieren eines Würfelwurfs
Satz 14Ø	:	Ausgabe des Würfelergebnisses auf dem Bildschirm
Satz 15Ø	:	Erhöhung einer Summe S um die "gewürfelte" Augenzahl zur Vorbereitung einer späteren Durchschnittsberechnung
Satz 16Ø-21Ø	:	Einsortieren des jeweiligen Würfelwurfs in eine der sechs Häufigkeitsklassen
Satz 22Ø	:	Beendigung der Schleife, d.h. nächster Würfelwurf
Satz 23Ø-25Ø	:	Berechnung und Ausgabe der durchschnittlichen Augenzahl (bei einem völlig korrekten Würfel, d.h. bei einem guten Zufallszahlengenerator muß dieser Wert 3.5 sein)
Satz 26Ø	:	Ausgabe einer Tabellenüberschrift
Satz 27Ø	:	Unterstreichung der Tabellenüberschrift
Satz 28Ø-34Ø	:	Ausgabe der erzeugten Häufigkeitstabelle
Satz 35Ø	:	Beendigung des Programms

zu Kapitel 6

1. Das folgende Programm bestimmt aus einem beliebigen Datenbestand, der vom Benutzer frei eingegeben werden kann, den <u>größten</u> und den <u>kleinsten Zahlenwert</u> (für diese Problemstellung hatten wir an anderer Stelle schon ein Flußdiagramm entwickelt, das sich der Leser noch einmal anschauen sollte).

   Dieser Aufgabenstellung entspricht es, daß das Programm aus zwei Teilen besteht : Einem Dateneingabeteil, den man so oder in ähnlicher Form im Prinzip immer wieder verwenden kann, und aus dem Teil, der den kleinsten und den größten Wert suchen soll.

   Um die Dateneingabe zu optimieren, lassen wir den Benutzer zunächst eingeben, wieviele Daten er in das Programm "einspeisen" möchte, um daraufhin mit dieser Information eine Eingabeschleife zu konstruieren, die die entsprechende Zahl von INPUT-State-ments enthält.

   Bei dem Aufsuchen des größten und des kleinsten Werts gehen wir etwas anders vor als in dem oben erwähnten Flußdiagramm : Wir belegen zunächst das Feld XG (größter Wert) mit einem unrealistisch kleinen Wert und das Feld XK (kleinster Wert) mit einem unrealistisch

großen Wert. Danach vergleichen wir jeden "echten" Wert mit dem bisher erreichten kleinsten bzw. mit dem bisher erreichten größten Wert. Ist der echte Wert kleiner als der bisher kleinste Wert, so ersetzt er diesen im Feld XK, ist er größer als der bisher größte Wert, so ersetzt er diesen im Feld XG.

Auf diese Weise wird erreicht, daß am Ende des Vergleichsprozesses im Feld XK der tatsächlich kleinste und im Feld XG der tatsächlich größte Wert des Datenbestands zu finden ist.

Der Leser wird aufgefordert, diese Überlegungen anhand der folgenden Programmliste zu überprüfen und nachzuvollziehen :

```
10 REM U8-MINIMAX
20 PRINT CHR$(125)
30 PRINT "   GROESSTER UND KLEINSTER WERT."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60  PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BESTIMMT DEN GROESS-"
80 PRINT "TEN UND DEN KLEINSTEN WERT AUS EINEM"
85 PRINT "BELIEBIGEN EINZUGEBENDEN DATENBESTAND."
90 PRINT :PRINT :PRINT
100 PRINT "WIEVIELE DATEN :  ";:INPUT N
110 DIM X(N)
120 PRINT CHR$(125)
130 PRINT "DATENEINGABE":PRINT
140 FOR I=1 TO N
150 PRINT I;".WERT :  ";:INPUT X
160 X(I)=X:NEXT I
170 XG=-1000:XK=1000
180 REM SUCHEN
190 FOR I=1 TO N
200 IF X(I)<XK THEN XK=X(I)
210 IF X(I)>XG THEN XG=X(I)
220 NEXT I
230 REM AUSGABE
240 PRINT :PRINT :PRINT
250 PRINT "GROESSTER WERT = ";XG
260 PRINT
270 PRINT "KLEINSTER WERT = ";XK
280 PRINT :PRINT "ENDE":END
```

Variablenliste :

I	=	Laufindex
N	=	Anzahl der Werte im Datenbestand
X	=	Wert
XG	=	Größter Wert
XK	=	Kleinster Wert

Programmbeschreibung :

Satz 1Ø-9Ø	:	Überschrift und Erläuterungen
Satz 1ØØ	:	Anforderung der Anzahl der einzugebenden Daten
Satz 11Ø	:	Dimensionieren des Daten-Arrays
Satz 12Ø	:	Löschen des Bildschirms
Satz 13Ø	:	Ausgabe der Überschrift "Dateneingabe"
Satz 14Ø-16Ø	:	Eingabeschleife (Anforderung jedes einzelnen Wertes)
Satz 17Ø	:	Belegung von XK und XG mit unrealistischen Werten
Satz 18Ø	:	Kommentar (Zwischenüberschrift im Programm-Listing
Satz 19Ø-22Ø	:	Schleife zum Suchen des kleinsten und des größten Wertes durch Ersetzung des jeweils kleinsten Werts, wenn ein noch kleinerer gefunden wurde, bzw. Ersetzung des jeweils

größten Wertes, wenn ein noch grösserer gefunden wurde

Satz 23Ø-27Ø : Ausgabe der Ergebnisse

Satz 28Ø : Beendigung des Programms

Der Leser überlege sich in Ergänzung zu dieser Übungsaufgabe, wie das obige Programm verändert werden müßte, wenn man auf die Dimensionierung (Satz 11Ø) verzichten wollte, ob dies überhaupt möglich ist und welche Vor- oder Nachteile gegebenenfalls damit verbunden wären.

2. Das folgende Programm simuliert wieder 1ØØ Würfelwürfe, berechnet den Durchschnitt aus den 1ØØ Augenzahlen und berechnet zusätzlich die Streuung der Häufigkeitsverteilung der Augenzahlen durch Bestimmung der sog. Standardabweichung.

Nur in dieser letzten Berechnung unterscheidet sich dieses Programm von einem zuvor schon vorgestellten Programm :

Die Standardabweichung aus einer gegebenen Anzahl von Werten berechnet sich nach der folgenden Formel :

$$S = \sqrt{\frac{1}{N} \sum (x_i - \bar{x})^2}$$

Dabei ist : x_i = Merkmalswert Nr i (i-ter Würfelwurf in diesem Beispiel)
$\bar{x}$ = Durchschnitt aller x_i (arithmetisches Mittel)
s = Standardabweichung
N = Anzahl der Merkmalswerte

Anhänge

Anhang 1 : Übungsaufgaben

```
10 REM U9-WUERFEL2
20 PRINT CHR$(125)
30 PRINT "             WUERFEL2"
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM SIMULIERT 100 WUERFE"
80 PRINT "MIT EINEM WUERFEL UND BERECHNET DEN"
85 PRINT "DURCHSCHNIT UND DIE STREUUNG (STAN-"
87 PRINT "  DARDABWEICHUNG) DER AUGENZAHLEN."
90 PRINT :PRINT :PRINT
100 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN"
110 STOP
115 PRINT CHR$(125)
117 PRINT "WUERFELERGEBNISSE ":PRINT
120 DIM W(100)
130 FOR I=1 TO 100
140 W=INT(RND(1)*6+1)
150 W(I)=W:PRINT W;" ";
160 S=S+W(I):NEXT I
170 AM=S/100
180 PRINT :PRINT
190 PRINT "DURCHSCHNITT        = ";AM
200 REM STANDARDABWEICHUNG
205 S=0
210 FOR I=1 TO 100
220 D=W(I)-AM:D=D*D
230 S=S+D:NEXT I
240 S=SQR(S/100)
250 S=INT(S*100+0.5)/100
260 PRINT :PRINT
270 PRINT "STANDARDABWEICHUNG  = ";S
280 PRINT :PRINT "ENDE":END
```

Variablenliste :

AM = Arithmetisches Mitte1
D = Differenz zwischen Wert und arithmetischem Mittel
I = Laufvariable
S = Summenfeld
W = Würfelergebnis (Wert)

Programmbeschreibung :

Satz 1Ø-9Ø : Überschrift und Erläuterungen
Satz 1ØØ-11Ø : Programmunterbrechung
Satz 115 : Löschen des Bildschirms
Satz 117 : Ausgabe einer Überschrift
Satz 12Ø : Dimensionierung
Satz 13Ø-16Ø : Erzeugung von 1ØØ Würfelwürfen, Ausgabe auf dem Bildschirm und Erzeuder Summe aller Augenzahlen
Satz 17Ø : Berechnung des arithmetischen Mittels
Satz 18Ø-19Ø : Ausgabe des Mittelwerts
Satz 2ØØ : Kommentar im Programm-Listing
Satz 2Ø5 : Wiederbelegung des Summenfeldes S mit Ø
Satz 21Ø : Beginn der Summenschleife zur Berechnung der Standardabweichung
Satz 22Ø : Bestimmung der Differenz zwischen Merkmalswert und arithmetischem Mit-

		tel und Quadrierung dieser Differenz (vergleiche Berechnungsformel für die Standardabweichung 5)
Satz 23Ø	:	Aufsummieren der quadrierten Differenzen
Satz 24Ø	:	Berechnung der Standardabweichung
Satz 25Ø	:	Runden der Standardabweichung auf zwei Dezimalstellen
Satz 26Ø-27Ø	:	Ausgabe der Standardabweichung
Satz 28Ø	:	Beendigung des Programms

3. Das folgende Programm erzeugt eine Häufigkeitsverteilung über beliebig viele einzugebende Familienstandsangaben.

Dabei gilt :

Ø = ledig
1 = verheiratet
2 = geschieden
3 = verwitwet

In diesem Programm Werden zwei Elemente miteinander gekoppelt, die wir getrennt voneinander schon in anderen Aufgabenstellungen kennengelernt haben :

Zum einen benutzen wir wieder ein allgemeines Dateneingabeprogramm, hier zur Bereitstellung der Familienstandsangaben; zum anderen benötigen wir ein Klassifizierungsprogramm, wie wir es schon in einem Flußdiagramm zur gleichen Problemstellung und in einem Auszählprogramm für Würfel-Augenzahlen benutzt haben.

Insoweit bietet das folgende Programm also keine neuen Elemente, sondern greift auf schon erprobte Bausteine zurück :

Anhänge

Anhang 1 : Übungsaufgaben

```
10 REM U10-FAMILIENSTAND
20 PRINT CHR$(125)
30 PRINT "     FAMILIENSTANDSVERTEILUNG."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BESTIMMT EINE HAEU-"
80 PRINT "FIGKEITSVERTEILUNG UEBER EINZUGEBENDE"
85 PRINT "     FAMILIENSTANDSANGABEN."
90 PRINT :PRINT :PRINT
100 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN"
110 STOP
120 PRINT "ES GILT : ":PRINT
130 PRINT "     0  =  LEDIG"
140 PRINT "     1  =  VERHEIRATET"
150 PRINT "     2  =  GESCHIEDEN"
160 PRINT "     3  =  VERWITWET"
170 PRINT :PRINT
180 PRINT "WIEVIELE WERTE : ";:INPUT N
190 DIM X(N),H(3)
200 PRINT :PRINT "EINGABE":PRINT
210 FOR I=1 TO N
220 PRINT I;".WERT : ";:INPUT X
230 X(I)=X:NEXT I
240 REM VERTEILUNG
250 PRINT CHR$(125)
260 FOR K=0 TO 3:H(K)=0:NEXT K
270 FOR I=1 TO N
280 FOR K=0 TO 3
290 IF X(I)=K THEN H(K)=H(K)+1:GOTO 310
300 NEXT K
310 NEXT I
320 REM AUSGABE
330 PRINT CHR$(125)
340 PRINT "VERTEILUNG":PRINT :PRINT
350 PRINT "FAM.STAND","ANZAHL":PRINT
360 FOR I=1 TO 26:PRINT "-";:NEXT I:PRINT
370 PRINT "LEDIG      ",H(0)
390 PRINT "GESCHIEDEN ",H(2)
380 PRINT "VERHEIRATET",H(1)
400 PRINT "VERWITWET  ",H(3)
410 PRINT :PRINT "ENDE":END
```

Variablenliste :

H	=	Häufigkeit
I	=	Laufindex (über die Werte)
K	=	Laufindex (über die Häufigkeitsklasse)
N	=	Anzahl der Werte
X	=	Wert (Familienstand)

Programmbeschreibung :

Satz 1Ø-9Ø	:	Überschrift und Erläuterungen
Satz 1ØØ-11Ø	:	Programmunterbrechung
Satz 115-17Ø	:	Löschen des Bildschirms und Ausgabe weiterer Erläuterungen
Satz 18Ø	:	Anforderung der Anzahl der einzugebenden Werte
Satz 19Ø	:	Dimensionierungen (Werte-Array und Häufigkeiten-Array)
Satz 2ØØ	:	Ausgabe einer Überschrift
Satz 21Ø-23Ø	:	Eingabeschleife
Satz 24Ø	:	Kommentar im Programm-Listing
Satz 25Ø	:	Löschen des Bildschirms
Satz 26Ø	:	Belegen aller Zählfelder (Häufigkeiten) mit Ø
Satz 27Ø	:	Beginn der Schleife über alle Werte
Satz 28Ø	:	Beginn der Schleife über alle Häufigkeitsklassen (Schleife zum Einsortieren)

Satz 29Ø	:	Erhöhung des betreffenden Zählers um 1, wenn der jeweilige Wert in die entsprechende Klasse hineingehört und Sprung zum Satz 31Ø (nächster Wert)
Satz 3ØØ	:	Nächste Klasse
Satz 31Ø	:	Nächster Wert
Satz 32Ø	:	Kommentar
Satz 33Ø	:	Löschen des Bildschirms
Satz 34Ø-36Ø	:	Tabellenüberschrift und Unterstreichung
Satz 37Ø-4ØØ	:	Ausgabe der Häufigkeitstabelle
Satz 41Ø	:	Beendigung des Programms

zu Kapitel 7

1. Das folgende Programm nimmt ein beliebiges Wort auf und gibt dieses in einer etwas seltsamen Form auf dem Bildschirm wieder aus.

   Gehen wir beispielsweise das Wort ATARI ein, so erhalten wir :

   A
   AT
   ATA
   ATAR
   ATARI

```
10 REM U11-WORTSPIEL
20 PRINT CHR$(125)
30 PRINT "             WORTSPIEL."
40 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM DRUCKT EIN BELIEBIG"
80 PRINT "EINZUGEBENDES WORT IN EINER ETWAS"
90 PRINT "    SELTSAMEN SCHREIBWEISE."
100 PRINT :PRINT :PRINT
110 DIM A$(1),S$(25)
120 PRINT "BITTE EIN WORT EINGEBEN :"
130 PRINT "     ";:INPUT S$
140 FOR I=1 TO LEN(S$)
150 PRINT S$(1,I)
160 NEXT I
170 PRINT :PRINT :PRINT
180 PRINT "NOCHMAL (J/N) ";:INPUT A$
190 IF A$="J" THEN PRINT CHR$(125):GOTO 120
200 PRINT :PRINT "ENDE":END
```

Variablenliste :

A$ = Stringvariable zur Aufnahme eines Symbols (J für Ja oder N für Nein)
I = Laufindex
S$ = Stringvariable zur Aufnahme des zu verarbeitenden Wortes

Programmbeschreibung :

Satz 1Ø-1ØØ : Überschrift und Erläuterungen
Satz 11Ø : Programmunterbrechung
Satz 12Ø-13Ø : Anforderung des zu bearbeitenden Wortes
Satz 14Ø-16Ø : Schrittweise Zerlegung des eingegebenen Wortes und Ausgabe der jeweiligen "Bruchstücke"; in der ersten Schleifenrunde nur der erste Buchstabe, in der zweiten Runde die ersten beiden Buchstaben usw..
Es werden so viele Runden durchlaufen wie das zu zerlegende Wort Buchstaben hat
Satz 17Ø-18Ø : Abfrage, ob ein erneuter Programmlauf gewünscht wird
Satz 19Ø : Wenn ja, Löschen des Bildschirms und Sprung zum Satz 12Ø
Satz 2ØØ : Wenn nein, Beendigung des Programms

2. Das folgende Programm gibt aus einer Reihe vorgegebener Namen alle diejenigen auf dem Bildschirm aus, die mit einem bestimmten Anfangsbuchstaben anfangen.

   Damit dieses Programm generell nutzbar ist: kann der jeweilige Such-Anfangsbuchstabe vom Benutzer frei gewählt werden.

   Die Namen, mit denen dieses Programm arbeitet, werden über DATA-Statements vorgegeben (genauso gut wäre aber auch eine Eingabe mit Hilfe von INPUT-Anweisungen denkbar). Wenn der Programmbenutzer deshalb mit eigenen Namen arbeiten möchte, sollte er über INPUT eingeben, oder er muß die DATA-Statements (Satz 5ØØ ff.) ändern.

   Damit der Rechner bei der Programmabarbeitung das Ende der Namensliste erkennen kann, haben wir als letzten Namen einen unrealistischen Namen eingegeben, nämlich den Namen "999". Durch Überprüfung eines jeden eingelesenen Namens auf Identität mit "999" verfügt das Programm also über ein funktionierendes Abbruchkriterium.

```
10 REM U12-NAMENSSUCHE
20 PRINT CHR$(125)
30 PRINT "          NAMENSSUCHE."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM GIBT AUS EINER REIHE"
80 PRINT "VORZUGEBENDER NAMEN ALLE AUS, DIE MIT"
90 PRINT "EINEM BESTIMMTEN BUCHSTABEN ANFANGEN."
100 PRINT :PRINT :PRINT
110 DIM N$(20),A$(1)
120 PRINT "GESUCHTER ANFANGSBUCHSTABE : ";
130 INPUT A$
140 FOR I=1 TO 1000
150 READ N$
160 IF N$="999" THEN 200
170 IF N$(1,1)=A$ THEN PRINT N$
180 NEXT I
200 PRINT :PRINT :PRINT
210 PRINT "ENDE DER AUSGABE":END
500 DATA OTTO,BERTA,ADAM,EMIL,FRITZ
510 DATA ERNST,KARL,EVA,SUSI,EGON
520 DATA XAVER,ALFONS
555 DATA 999
```

Variablenliste :

A$ = Stringvariable zur Aufnahme eines Symbols (gewünschter Anfangsbuchstabe)
I = Laufindex
N$ = Stringvariable zur Aufnahme eines Namens

Programmbeschreibung :

Satz 10-90 : Überschrift und Erläuterungen
Satz 100 : Drei Leerzeilen
Satz 110 : Dimensionierungen (es wird unterstellt, daß die einzugebenden Namen maximal 20 Symbole umfassen)
Satz 120-130 : Anforderung des gewünschten Such-Anfangsbuchstabens
Satz 135 : Löschen des Bildschirms
Satz 140 : Beginn der Leseschleife (sie muß mehr Runden aufweisen als tatsächlich Namen vorhanden sind, es sei denn man kennt die Anzahl der Namen ganz genau; dann kann das Schleifenende präzis angegeben werden und auf Satz 160 kann dann verzichtet werden)
Satz 150 : Lesen eines Namens
Satz 160 : Prüfung, ob der (unrealistische) Namen "999" gelesen wurde

		Wenn ja, Sprung zum Satz 2ØØ
Satz 17Ø	:	Überprüfung, ob der gelesene ("echte") Namen mit dem gewünschten Anfangsbuchstaben anfängt; wenn ja, wird der Namen gedruckt
Satz 18Ø	:	Betrachtung des nächsten Namens
Satz 2ØØ-21Ø	:	Beendigung des Programms
Satz 5ØØ-52Ø	:	Daten (hier : Namen)
Satz 555	:	Unrealistischer Name als Abbruchkriterium

3. Das folgende Programm dient dazu, einen beliebigen, in Großbuchstaben eingegebenen String wieder auszugeben, dann aber in Kleinschreibweise.

   Dies ist nicht so schwierig zu programmieren, wie es zunächst vielleicht erscheinen mag : Man muß sich nur daran erinnern, daß die ASCII-Codezahlen für Kleinbuchstaben gerade immer um 32 größer sind als die für Großbuchstaben.

   Ist beispielsweise die ASCII-Codezahl für A gleich 65, so ist die Codezahl für a gleich 65 + 32 = 97; entsprechend gilt für B 66 und für b 93 usw.

   Der eingegebene String muß also in seine einzelnen Bestandteile "zerhackt"werden, deren Codezahlen dann mit der ASC-Funktion festgestellt werden können. Addiert man zu diesen Codezahlen den Wert 32 und wendet auf die sich ergebende neue Zahl die CHR$-Funktion an, so produziert man die entsprechenden Kleinbuchstaben.

   Das folgende Programm leistet diese Umwandlungen für beliebig eingebbare Strings :

```
10 REM U13-GROSS-KLEINSCHREIBUNG
20 PRINT CHR$(125)
30 PRINT "      GROSS-KLEINSCHREIBUNG."
40 PRINT :PRINT :PRINT
50 PRINT "        PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM WANDELT EINEN BELIE-"
80 PRINT " BIGEN STRING VON GROSS- IN KLEIN-"
90 PRINT "        SCHREIBUNG UM."
100 PRINT :PRINT :PRINT
110 DIM G$(20),A$(1),C$(1),B$(1)
120 PRINT "UMZUWANDELNDER STRING : "
130 PRINT :INPUT G$
140 PRINT :PRINT :PRINT
150 FOR I=1 TO LEN(G$)
160 B$=G$(I,I)
170 Z=ASC(B$):Z=Z+32:C$=CHR$(Z)
180 PRINT C$;
190 NEXT I
200 PRINT :PRINT :PRINT
210 PRINT "NOCHMAL (J/N) ";:INPUT A$
220 IF A$="J" THEN PRINT CHR$(125):GOTO 120
230 PRINT :PRINT "ENDE":END
```

Variablenliste :

A$ = Stringvariable zur Eingabe für J (für Ja) oder N (für Nein)
B$ = Einzelner Buchstabe des zu behandelnden Strings
C$ = Umgewandelter Buchstabe des zu behandelnden Strings
G$ = Zu behandelnder String
I = Laufvariable
Z = Code

Programmbeschreibung :

Satz 1Ø-1ØØ : Überschrift und Erläuterungen
Satz 11Ø : Dimensionierungen (es wird dabei unterstellt, daß der zu behandelnde String maximal 2Ø Symbole umfaßt)
Satz 12Ø-13Ø : Anforderung des zu behandelnden Strings
Satz 14Ø : Drei Leerzeilen
Satz 15Ø : Beginn der Umwandlungsschleife
Satz 16Ø : Isolierung eines einzelnen Buchstabens
Wenn ja, Sprung zum Satz 2ØØ
Satz 17Ø : Bestimmung der Codezahl des Buchstabens, Erhöhung dieser Zahl um 32 und Aufsuchen des entsprechenden Symbols mit der CHR$-Funktion

Satz 18Ø	:	Ausgabe des gefundenen Zeichens
Satz 19Ø	:	Beendigung der Umwandlungsschleife, d.h. Übergang zum nächsten Symbol des zu behandelnden Strings
Satz 2ØØ-21Ø	:	Drei Leerzeilen und Frage; ob noch eine erneute Umwandlung gewünscht wird
Satz 22Ø	:	Wenn ja, dann Löschen des Bildschirms und Rücksprung zum Satz 12Ø
Satz 23Ø	:	Wenn nein, Beendigung des Programms

zu Kapitel 8

1. Das folgende Programm zeichnet drei Rechtecke zufälliger Größe an zufällig ausgewählte Stellen des Bildschirms.

   Bei einer solchen Aufgabenstellung muß nur darauf geachtet werden, daß bei der Vorgabe der Startkoordinaten zum Zeichnen eines der Rechtecke der zulässige Bildschirmbereich nicht verlassen wird. Das gleiche gilt für die übrigen Ecken des Rechtecks, die sich ergeben, wenn wir zu den Startkoordinaten zufällig bestimmte Längen und Breiten hinzuaddieren.

   Deshalb müssen in dem Programm entsprechende Abfragen vorgesehen werden und es ist Sorge dafür zu tragen, daß Längen und Breiten der Rechtecke nicht beliebig groß werden können.

   In dem folgenden Programm wird dies dadurch erreicht, daß die Startspalte nur zwischen Ø und 26, die Startzeile nur zwischen Ø und 14, die Länge nur zwischen Ø und 14 und die Breite nur zwischen Ø und 7 liegen kann (dies sind natürlich willkürliche Setzungen, die der Programmbenutzer auch bei Bedarf verändern kann).

   Zudem sorgen die oben erwähnten Abfragen für das "Abfangen" von "Ausreißern".

Das Zeichnen der Rechtecke selbst übernimmt die Anweisung PRINT CHR$(16Ø), die ein einzelnes kleines Rechteckchen zeichnet, das - mehrfach aneinandergesetzt - die erwünschten Rechtecksflächen erzeugen kann :

```
10 REM U14-RECHTECKE
20 PRINT CHR$(125)
30 PRINT "                RECHTECKE."
40 PRINT :PRINT :PRINT
50 PRINT "          PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM ZEICHNET DREI ZUFAEL-"
80 PRINT "LIG ANGEORDNETE UND ZUFAELLIG GROSSE"
90 PRINT "                RECHTECKE."
100 PRINT :PRINT :PRINT
110 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !"
120 STOP
130 PRINT CHR$(125)
140 FOR I=1 TO 3
150 S=INT(RND(1)*27)+8:Z=INT(RND(1)*15)
160 L=INT(RND(1)*15):B=INT(RND(1)*8)
170 S1=S+L:IF S1>38 THEN S1=38
180 Z1=Z+B:IF Z1>18 THEN Z1=18
190 FOR J=S TO S1
200 FOR K=Z TO Z1
210 POSITION J,K:PRINT CHR$(160)
220 NEXT K
230 NEXT J
240 NEXT I
250 POSITION 1,20
260 PRINT "ENDE":END
```

Variablenliste :

B = Breite des Rechtecks
I = Laufindex (über alle Rechtecke)
J = Laufindex (über alle Spalten, die ein Rechteck überdecken soll)
K = Laufindex (über alle Zeilen, die ein Rechteck überdecken soll)
L = Länge des Rechtecks
S = Startspalte eines Rechtecks
S1 = Startspalte + Länge
Z = Startzeile eines Rechtecks
Z1 = Startzeile + Breite

Programmbeschreibung :

Satz 1Ø-1ØØ : Überschrift und Erläuterungen
Satz 11Ø-12Ø : Warten (Programmunterbrechung)
Satz 13Ø : Löschen des Bildschirms
Satz 14Ø : Beginn einer Dreierschleife (drei Rechtecke sollen gezeichnet werden)
Satz 15Ø : Zufällige Bestimmung der Startspalte und der Startzeile
Satz 16Ø : Zufällige Bestimmung von Länge und Breite

Satz 17Ø	:	Bestimmung der Endspalte des Rechtecks und Festlegung auf Spalte 38, falls diese überschritten werden sollte
Satz 18Ø	:	Bestimmung der Endzeile und Festlegung auf 18, falls dieser Wert überschritten sein sollte
Satz 19Ø	:	Zeichnen in allen Spalten von S bis S1
Satz 2ØØ	:	Zeichnen in allen Zellen von Z bis Z1
Satz 21Ø	:	Zeichnen eines kleinen Quadrats
Satz 22Ø	:	Nächste Zeile
Satz 23Ø	:	Nächste Spalte
Satz 24Ø	:	Nächstes Rechteck
Satz 25Ø	:	Beendigung des Programms

2. Eine gegebene Häufigkeitsverteilung soll in Form eines Balkendiagramms graphisch dargestellt werden (die Ausgangsdaten finden sich in der nebenstehenden Tabelle).

Klasse	Anzahl
1	15
2	33
3	17
4	28
5	7

Zum Zeichnen dieser Balken benutzen wir wieder die Anweisung PRINT CHR$(16Ø), wobei nun darauf zu achten ist, daß die Fläche der zu zeichnenden Balken die darzustellenden Häufigkeiten (Anzahlen) repräsentiert.

Es werden deshalb mit dieser Anweisung so viele Quadrate aneinandergehängt, daß Balken der gewünschten Länge entstehen.

Da auch dabei der erlaubte Bildschirmbereich nicht verlassen werden darf, normieren wir die gegebenen Anzahlen derart, daß der größte Wert (33) den Bildschirm in seiner Höhe gut ausnutzt. Dies haben wir so festgelegt, daß der entsprechende Maximalbalken sich über 13 Bildschirmzeilen erstreckt.

Eine derartige Normierung macht es erforderlich, daß in einem ersten Programmschritt zunächst der Maximalwert gesucht wird, und daß dann alle anderen Häufigkeiten darauf bezogen werden.

Anhänge

Anhang 1 : Übungsaufgaben

```
10 REM U15-BALKENDIAGRAMM
20 PRINT CHR$(125)
30 PRINT "PROGRAMM ZUR GRAPHISCHEN DARSTELLUNG"
40 PRINT "    EINER HAEUFIGKEITSVERTEILUNG."
50 PRINT :PRINT :PRINT :PRINT
60 PRINT "         PROF.DR.W.VOSS, 1984"
70 PRINT :PRINT :PRINT
80 DIM F(5),G(5)
90 DATA 15,33,17,28,7
100 FOR I=1 TO 5:READ F
110 F(I)=F:NEXT I
120 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !"
130 STOP
150 PRINT CHR$(125)
160 FM=0
180 FOR I=1 TO 5:IF F(I)>FM THEN FM=F(I)
190 NEXT I
200 FOR I=1 TO 5:G(I)=(13/FM)*F(I)
210 G(I)=INT(G(I)+0.5):NEXT I
230 FOR J=1 TO 5
240 B=17:A=17-G(J)
250 FOR I=A TO B
260 POSITION J*6,I:PRINT CHR$(160)
270 POSITION J*6+1,I:PRINT CHR$(160)
280 NEXT I
290 NEXT J
300 FOR I=1 TO 35:PRINT CHR$(18);:NEXT I
310 PRINT :PRINT :PRINT "ENDE":END
```

Variablenliste :

A = Anfang des zu zeichnenden Balkens
B = Ende des zu zeichnenden Balkens
F = Darzustellende Häufigkeit
FM = Maximale Häufigkeit
G = Normierte Häufigkeit
I = Laufindex
J = Laufindex

Programmbeschreibung :

Satz 10-70 : Überschrift und Erläuterungen
Satz 80 : Dimensionierungen
Satz 90 : Ausgangsdaten
Satz 100-110 : Einlesen der Ausgangsdaten
Satz 120-130 : Warten (Programmunterbrechung)
Satz 150 : Löschen des Bildschirms
Satz 160 : Vorgabe eines unrealistisch kleinen Maximalwertes
Satz 180-190 : Aufsuchen der maximalen Häufigkeit
Satz 200 : Beginn der Normierung der einzelnen Häufigkeiten
Satz 210 : Zeichnen in allen Zellen von Z bis Z1
Satz 210 : Runden der normierten Häufigkeiten auf ganze Zahlen und Beendigung der Normierungsschleife
Satz 230 : Beginn der Zeichenschleife

Satz 24Ø	:	Beginn und Ende des zu zeichnenden Balkens werden bestimmt
Satz 25Ø	:	Beginn der Schleife, die je einen Balken zeichnet
Satz 26Ø	:	Zeichnen eines einspaltigen Balkans
Satz 27Ø	:	Daneben wird der gleiche einspaltige Balken noch einmal gesetzt, so daß insgesamt ein zwei Spalten breiter Balken entsteht
Satz 28Ø	:	Beendigung der Balkenschleife
Satz 29Ø	:	Nächster Balken
Satz 3ØØ	:	Zeichnen eines waagrechten Strichs unter die Balken mit Hilfe von PRINT CHR$(18)
Satz 31Ø	:	Beendigung des Programms

3. Das folgende Programm dient dazu, die Farbmöglichkeiten des ATARI dadurch zu demonstrieren, daß Bildschirmrahmen und Bildschirmhintergrund gemeinsam verschiedene Farben durchlaufen.

Wir erinnern uns daran, daß dies mit der Anweisung SETCOLOR möglich ist. Damit die verschiedenen Farben auch schön ins Auge fallen, muß dafür gesorgt werden, daB der Farbwechsel nicht zu rasch erfolgt, weshalb wir an passender Stelle eine Warteschleife vorgesehen haben :

```
10 REM U16-FARBENSPIEL
20 PRINT CHR$(125)
30 PRINT "PROGRAMM ZUM FARBWECHSEL DES GESAMTEN"
40 PRINT "              BILDSCHIRMS."
50 PRINT :PRINT :PRINT :PRINT
60 PRINT "         PROF.DR.W.VOSS, 1984"
70 PRINT :PRINT :PRINT
80 FOR F=0 TO 15
90 SETCOLOR 2,F,6
100 SETCOLOR 4,F,6
110 FOR I=1 TO 400:NEXT I
120 NEXT F
130 PRINT :PRINT "ENDE":END
```

Variablenliste :

F = Codezahl für die auszuwählende Farbe
I = Laufindex

Programmbeschreibung :

Satz 1Ø-7Ø : Überschrift und Erläuterungen
Satz 8Ø : Beginn der Farbenschleife mit den Farben F = Ø bis F = 15 (vergleiche Farbtabelle)
Satz 9Ø : Färbung des Bildschirmhintergrunds
Satz 1ØØ : Färbung des Bildschirmrahmens
Satz 11Ø : Warteschleife
Satz 12Ø : Beendigung der Farbenschleife (nächste Farbe)
Satz 13Ø : Beendigung des Programms

zu Kapitel 9

1. Das folgende Programm soll im Blockgraphik-Modus ein Gitter auf dem Bildschirm zeichnen, wobei Zeilen- und Spaltenabstand vom Benutzer frei gewählt werden können.

   Zum Zeichnen benutzen wir jetzt nicht mehr etwa die Anweisung PRINT CHR$(16Ø) oder eine ähnliche Version dieser Anweisung, sondern wir bedienen uns der Graphik-Anweisungen PLOT und DRAWTO.

   Die frei wählbaren Zeilen- und Spaltenabstände geben wir über INPUT ein. Darüber hinaus ergeben sich keine prinzipiellen Schwierigkeiten bei der Erstellung dieses Programms :

```
10 REM U17-GITTER
20 PRINT CHR$(125)
30 PRINT :PRINT :PRINT
40 PRINT "                GITTER."
50 PRINT :PRINT :PRINT
60 PRINT "         PROF.DR.W.VOSS, 1984"
70 PRINT :PRINT :PRINT
75 PRINT "PROGRAMM ZUM ZEICHNEN EINES GITTERS."
77 PRINT :PRINT :PRINT
80 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !"
90 STOP
95 PRINT CHR$(125)
100 PRINT "ZEILENABSTAND (MAX.19) : ";
110 INPUT Z
120 PRINT "SPALTENABSTAND (MAX.39): ";
130 INPUT S
140 GRAPHICS 3
145 COLOR 1
150 FOR I=0 TO 39 STEP S
160 PLOT I,0:DRAWTO I,19
170 NEXT I
180 FOR I=0 TO 19 STEP Z
190 PLOT 0,I:DRAWTO 39,I
200 NEXT I
205 PRINT "BITTE WARTEN"
210 FOR I=1 TO 3000:NEXT I
220 GRAPHICS 0
230 PRINT "ENDE":END
```

Variablenliste :

I = Laufindex
S = Spaltenabstand
Z = Zeilenabstand

Programmbeschreibung :

Satz 1Ø-77	:	Überschrift und Erläuterungen
Satz 8Ø-9Ø	:	Warten (Programmunterbrechung)
Satz 95	:	Löschen des Bildschirms
Satz 1ØØ-13Ø	:	Anforderung von Zeilen- und Spaltenabstand des Gitters
Satz 14Ø	:	Umschalten in den Blockgraphik-Modus
Satz 145	:	Vorgabe einer Zeichenfarbe
Satz 15Ø-17Ø	:	Zeichnen der senkrechten Strichs durch Setzen eines Punktes in den Spalten Ø bis 39 mit der Schrittweite S (Zeile Ø) und Ziehen des Strichs bis Zeile 19
Satz 18Ø-2ØØ	:	Zeichnen der waagrechten Striche nach dem gleichen Prinzip wie oben
Satz 2Ø5-21Ø	:	Wartemeldung und Warteschleife
Satz 22Ø-23Ø	:	Zurückschalten in den Textmodus und Beendigung des Programms

2. Nach dem gleichen Muster wie das vorangegangene Programm ein Gitter erzeugte, soll das nächste Programm im Blockgraphik-Modus eine Diagonale von links oben nach rechts unten auf dem Bildschirm erzeugen.

Hier ergeben sich keine prinzipiellen neuen Schwierigkeiten. so daß wir sofort das Programm vorstellen können :

```
10 REM U18-DIAGONALE
20 PRINT CHR$(125)
30 PRINT :PRINT :PRINT
40 PRINT "              DIAGONALE."
50 PRINT :PRINT :PRINT
60 PRINT "        PROF.DR.W.VOSS, 1984"
70 PRINT :PRINT :PRINT
75 PRINT "PROGRAMM ZUM ZEICHNEN EINER DIAGONALEN"
77 PRINT :PRINT :PRINT
80 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !"
90 STOP
95 PRINT CHR$(125)
140 GRAPHICS 3
145 COLOR 1
150 FOR S=0 TO 39
160 Z=INT(S/2+0.5)
170 PLOT S,Z
180 NEXT S
200 PRINT "BITTE WARTEN"
210 FOR I=1 TO 3000:NEXT I
220 GRAPHICS 0
230 PRINT "ENDE":END
```

Variablenliste :

I = Laufindex
S = Spalte, in der gezeichnet wird
Z = Zeile, in der gezeichnet wird

Programmbeschreibung :

Satz 1Ø-77	:	Überschrift und Erläuterungen
Satz 8Ø-9Ø	:	Programmunterbrechung (Warten)
Satz 14Ø	:	Umschalten in den Blockgraphik-Modus
Satz 145	:	Auswahl einer Zeichenfarbe
Satz 15Ø	:	Beginn der Zeichenschleife
Satz 16Ø	:	Bestimmung der Zeile, in der gezeichnet werden soll (die Spalte wird durch den Laufindex S der Zeichenschleife gegeben)
Satz 17Ø	:	Zeichnen eines Punktes
Satz 18Ø	:	Ende der Zeichenschleife (nächste Spalte)
Satz 2ØØ-21Ø	:	Wartemeldung und Warteschleife
Satz 22Ø-23Ø	:	Zurückschalten in den Textmodus und Beendigung des Programms

zu Kapitel 1Ø

1. Das folgende Programm dient dazu, in hochauflösender Graphik eine <u>Ellipse</u> zu zeichnen.

   Die Lage der Ellipse (genauer : die Lage des Ellipsenmittelpunkts) und die beiden Halbachsen sollen vom Benutzer frei vorgegeben werden können.

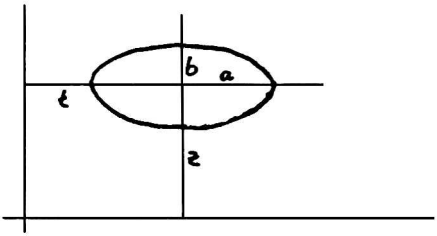
   Zur Erstellung einer solchen Graphik muß man von der allgemeinen Ellipsengleichung ausgehen, welche - nach y aufgelöst - folgendermaßen lautet :

$$y_{1/2} = z + \frac{b}{a} * \sqrt{a^2 - (x-t)^2}$$

   dabei bedeutet :

   x = X-Koordinate (Spalte)
   y = Y-Koordinate (Zeile; zu jedem X-Wert gibt es zwei Y-Werte : Obere und untere Halbellipse)
   a = horizontale Halbachse
   b = vertikale Halbachse
   z = Bildschirmzeile des Mittelpunkts
   t = Bildschirmspalte des Mittelpunkts

Die folgende Skizze verdeutlicht die Aussage der einzelnen Parameter :



Wenn nun die Ellipse gezeichnet werden soll, brauchen wir nur im erlaubten Bereich x–t bis x+t zu allen X-Werten die Werte Y1 und Y2 zu berechnen und können dann an den Bildschirmpunkten X,Y1 bzw. X,Y2 Punkte zeichnen, die in ihrer Verbindung das graphische Bild der Ellipse auf dem Bildschirm ergeben.

Allerdings muß dabei jeweils überprüft werden, ob der zulässige Bildschirmbereich nicht verlassen wird. Sollte dies für bestimmte Koordinatenwerte geschehen, so muß der Zeichenbefehl ignoriert werden.

Hilfreich ist es, wenn man zusätzlich ein Achsenkreuz zeichnet und als Hilfslinien Parallelen zu den Achsen durch den Ellipsenmittelpunkt.

Anhänge

Anhang 1 : Übungsaufgaben

```
10 REM U19-ELLIPSE
20 PRINT CHR$(125)
30 PRINT "PROGRAMM ZUM ZEICHNEN EINER BELIEBI-"
40 PRINT "              GEN ELLIPSE."
45 PRINT :PRINT :PRINT
50 PRINT "         PROF.DR.W.VOSS, 1984"
60 PRINT :PRINT :PRINT :PRINT
70 PRINT "DIESES PROGRAMM BENOETIGT ALS INPUT-"
80 PRINT "INFORMATIONEN :":PRINT
90 PRINT "KOORDINATEN DES MITTELPUNKTS :":PRINT
100 PRINT "     - SPALTE :  ";:INPUT T
110 PRINT "     - ZEILE  :  ";:INPUT Z
115 PRINT :PRINT
120 PRINT "1. HALBACHSE  :  ";:INPUT A
125 PRINT "2. HALBACHSE  :  ";:INPUT B
130 GRAPHICS 8:COLOR 1
140 PLOT 0,159:DRAWTO 319,159
150 PLOT 0,159:DRAWTO 0,0
160 PLOT 0,Z:DRAWTO 319,Z
170 PLOT T,159:DRAWTO T,0
390 REM ELLIPSE
400 C=T-A:D=T+A
410 IF C<0 THEN C=0
420 IF D>319 THEN D=319
430 FOR X=C+1 TO D-1
440 DS=(B/A)*SQR(A*A-(X-T)^2)
450 Y=Z+DS
460 IF Y<0 OR Y>159 THEN 480
470 PLOT X,Y
480 NEXT X
490 FOR X=D-1 TO C+1 STEP -1
500 DS=(B/A)*SQR(A*A-(X-T)^2)
510 Y=Z-DS
520 IF Y<0 OR Y>159 THEN 540
530 PLOT X,Y
540 NEXT X
550 FOR I=1 TO 2000:NEXT I
560 GRAPHICS 0
570 PRINT "ENDE":END
```

Variablenliste :

A = Horizontale Halbachse
B = Vertikale Halbachse
C = Ellipsenmittelpunkt - A
D = Ellipsenmittelpunkt + A
DS = Hilfsgröße in der Ellipsenberechnung
I = Laufindex
T = Spalte des Ellipsenmittelpunkts
X = X-Koordinate
Y = Y-Koordinate gemäß der Ellipsengleichung
Z = Zeile des Ellipsenmittelpunkts

Programmbeschreibung :

Satz 1Ø-6Ø : Überschrift und Erläuterungen
Satz 7Ø-125 : Anforderung der Input-Informationen, die das Programm benötigt :
- Koordinaten des Ellipsenmittelpunkts
- Halbachsen der Ellipse

Satz 13Ø : Umschalten in den hochauflösenden Graphikmodus und Auswahl einer Zeichenfarbe
Satz 14Ø-15Ø : Zeichnen eines Achsenkreuzes
Satz 16Ø-17Ø : Zeichnen von Parallelen zu den Achsen durch den Ellipsenmittelpunkt

Satz 39Ø	:	Kommentar im Programm-Listing
Satz 4ØØ	:	Bestimmung des Wertebereichs für die X-Variable
Satz 41Ø-42Ø	:	Beschränkung des Wertebereichs auf den zulässigen Bildschirmbereich, falls dies erforderlich ist
Satz 43Ø	:	Beginn der Zeichenschleife für die untere Ellipsenhälfte
Satz 44Ø	:	Bestimmung einer Hilfsgröße in der Ellipsengleichung
Satz 45Ø	:	Bestimmung der Y—Koordinate
Satz 46Ø	:	Überprüfung, ob die Y-Koordinate den zulässigen Bildschirmbereich verläßt; wenn ja, weiter bei Satz 48Ø
Satz 47Ø	:	Zeichnen eines Punktes mit den Koordinaten X und Y
Satz 48Ø	:	Beendigung der Zeichenschleife, d.h. Übergang zum nächsten X-Wert und Zeichnen des nächsten Punktes
Satz 49Ø-54Ø	:	Gleiche Prozedur wie in den Sätzen 43Ø bis 48Ø nun für die obere Ellipsenhälfte
Satz 55Ø	:	Warteschleife
Satz 56Ø	:	Zurückschalten in den Textmodus
Satz 57Ø	:	Beendigung des Programms

2. Das folgende Programm erzeugt eine Graphik: die schematisch etwa so aussieht, wie es auf Seite 263 vorgestellt wurde.

   Um ein derartiges Bild mit einem BASIC-Programm erzeugen zu können, benötigt man einfach eine Programmschleife, die so viele Quadrate, wie gewünscht werden, zeichnen kann.

   In dem folgenden Programm werden 12 Quadrate gezeichnet, d.h. wir arbeiten mit einer Schleife, deren Laufindex von I=1 bis I=12 läuft. In jeder Runde werden Startspalte und Startzeile für die Quadratzeichnung in Abhängigkeit.von I bestimmt, d.h. S wächst mit wachsendem I und auch Z wächst mit anwachsendem I (S und Z seien die Startkoordinaten für das jeweilige Quadrat).

   Gleichzeitig soll, um einen perspektivischen Effekt zu erzeugen, die Quadratseite der zu zeichnenden

Quadrate immer kleiner werden, je weiter unten ein Quadrat auf dem Bildschirm ist, d.h. je höher der Wert der Laufvariablen I ist. Dies erreichen wir einfach, indem wir die Quadratseite L mit wachsendem I verkleinern nach der Berechnungsformel :

L = 5Ø – 2*I

Des weiteren bietet diese Aufgabenstellung keine Schwierigkeiten :

```
 10 REM U20-PERSPEKTIVE
 20 PRINT CHR$(125)
 30 PRINT "PROGRAMM ZUM PERSPEKTIVISCHEN ZEICH-"
 40 PRINT "    NEN (BEISPIEL : QUADRATE)"
 45 PRINT :PRINT :PRINT
 50 PRINT "        PROF.DR.W.VOSS, 1984"
 60 PRINT :PRINT :PRINT :PRINT
 70 PRINT "ZUR FORTSETZUNG BITTE CONT EINGEBEN !"
 80 STOP
 90 GRAPHICS 8
 100 FOR I=1 TO 12
 110 S=I*10:Z=I*7
 120 PLOT S,Z
 130 L=50-I*2
 140 DRAWTO S,Z+L
 150 DRAWTO S+L,Z+L
 160 DRAWTO S+L,Z
 170 DRAWTO S,Z
 180 NEXT I
 190 FOR I=1 TO 3000:NEXT I
 200 GRAPHICS 0
 210 PRINT "ENDE":END
```

Variablenliste :

I = Laufindex
L = Quadratseite
S = Spalte des Startpunkts der Quadratzeichnung
Z = Zeile des Startpunkts der Quadratzeichnung

Programmbeschreibung :

Satz 1Ø-6Ø	:	Überschrift und Erläuterungen
Satz 7Ø-8Ø	:	Programmunterbreohung
Satz 9Ø	:	Umschalten in den hochauflösenden Graphikmodus
Satz 1ØØ	:	Beginn der Zwölferschleife zum Zeichnen von 12 Quadraten
Satz 11Ø	:	Bestimmung der Startkoordinaten
Satz 12Ø	:	Zeichnen des Startpunktes
Satz 13Ø	:	Berechnung der Quadratseite
Satz 14Ø-17Ø	:	Zeichnen der Quadratseiten
Satz 18Ø	:	Nächstes Quadrat
Satz 19Ø	:	Warteschleife
Satz 2ØØ	:	Zurückschalten in den Textmodus
Satz 21Ø	:	Beendigung des Programms

Anhang 2 : Fehlermeldungen

Bei der Eingabe der hier vorgeführten Programme oder auch bei der Entwicklung eigener Programme wird der Leser - nicht nur wenn er noch Anfänger ist - häufig feststellen, daß nach dem Kommando RUN das Programm vom Rechner mit einer Fehlermeldung abgebrochen wird.

Die am häufigsten auftretenden Fehlermeldungen sollen hier kurz besprochen werden, damit der Leser erkennen kann, wie derartige Fehler behoben werden können, so daß dann das eingegebene Programm korrekt abgearbeitet werden kann.

Die ATARI-Fehlermeldungen werden als Nummern ausgegeben, deren Bedeutung nun kurz, zusammen mit einigen Hinweisen zur Fehlerbehebung, vorgestellt werden sollen :

Anhang 2 : Fehlermeldungen

ERROR
(Fehler)
Nr.

2 Der zur Verfügung stehende Speicherplatz reicht nicht aus.

Korrektur : Es muß mit kleineren Datenbeständen, mit Teil-Datenbeständen oder mit kleineren Dimensionierungen gearbeitet werden.

3 Falscher Wert : Ein negativer Wert wurde benutzt, wo ein positiver Wert verlangt ist oder es wurde ein Wert benutzt, der nicht im erlaubten Wertebereich liegt.

Korrektur : Sie muß anhand inhaltlicher Überlegungen erfolgen.

6 Keine Daten mehr : Es wurde versucht mit einer READ-Anweisung Daten zu lesen, obwohl die DATA-Anweisungen keine Daten mehr beinhalten.

Korrektur : Verlängerung des Inhalts der DATA-Anweisungen oder ggf. Gebrauch des Statements RESTORE.

8 Typenfehler : Es wurde versucht, eine Stringgröße in ein numerisches Feld einzugeben.

Korrektur : Änderung des betreffenden Variablennamens

ERROR
(Fehler)
Nr.

9 Dimensionierungsfehler : Der Index einer indizierten Variablen hat einen Wert angenommen, der durch das zuständige DIM-Statement nicht zugelassen wird.

Korrektur : Änderung der betreffenden DIM-Anweisung.

11 Speicher-Overflow oder Speicher-Underflow : Es wurde versucht, durch Null zu dividieren oder generell eine zu kleine oder eine zu große Zahl zu speichern.

Korrektur : Sie wird aufgrund inhaltlicher Überlegungen möglich.

12 Nicht existierender Satz : Es wurde versucht, einen Satz anzuspringen (GOTO, GOSUB oder IF ... THEN), der im Programm nicht existiert.

Korrektur : Satznummern überprüfen und ggf. an der entsprechenden Stelle ändern bzw. fehlenden Satz einfügen.

13 NEXT ohne FOR : Es wurde ein NEXT-Statement erreicht, ohne daß ein dazugehöriges FOR ... TO-Statement vorgefunden wurde.

Korrektur : Fehlendes Statement einfügen.

ERROR
(Fehler)
Nr.

16 RETURN ohne GOSUB : Es wurde ein RETURN-Statement erreicht, ohne daß das dazugehörige GOSUB-Statement berührt wurde.

Korrektur : Einfügung des fehlenden Statements

Da dieser Fehler häufig dann auftritt, wenn der Benutzer vergessen hat, sein Programm mit einem END-Statement zu beschliessen, sollte er zunächst überprüfen, ob sein Hauptprogramm auf END endet.

141 Cursor ist außerhalb des zulässigen Bereichs : Es wurde versucht, in einem der Graphik-Modi den Cursor an einen Punkt zu schicken, der außerhalb des zulässigen Bildschirmbereichs liegt.

Korrektur : Überprüfung der Spalten- und Zeilenparameter der Graphik-Anweisungen.

17Ø Datei nicht gefunden : Es wurde versucht, von einer Diskette eine Datei einzulesen, die auf dieser Diskette nicht vorhanden ist.

Korrektur : Überprüfung, ob die richtige Diskette eingelegt wurde; wenn ja, Überprüfung, ob der korrekte Dateiname im LOAD-Kommando verwendet wurde.

Anhang 2 : Fehlermeldungen

Die vorangegangene Auswahl von Fehlermeldungen soll es dem Benutzer ermöglichen, seine Programme im Bedarfsfall zu verbessern. Diese Fehlermeldungsliste ist nicht vollständig. Weitere Hinweise findet der Leser in der ATARI-Spezialliteratur.

In dieser Liste taucht nicht der bekannte (und sehr häufig auftretende) Syntax-Fehler auf, der reine Sprachfehler also, der beispielsweise dann zu verzeichnen ist, wenn der Benutzer statt PRINT vielleicht PRUNT eintippt. Derartige Syntaxfehler meldet der ATARI sofort, nachdem der fehlerhafte Satz abgeschickt wurde, so daß dieser dann sofort auch korrigiert werden kann und nicht erst dann, wenn per RUN-Kommando versucht wurde, das Programm zu starten.

Die Korrektur selbst ist beim ATARI sehr einfach :

Meldet der Rechner einen Fehler, so empfiehlt es sich, das Programm zu listen und dann per CONTROL-Taste, zusammen mit den Pfeiltasten (Cursor-Steuertasten) den Cursor an die fehlerhafte Stelle zu schicken.

Dort kann dann neu geschrieben werden und der korrekte Satz kann dann per RETURN-Taste abgeschickt werden.

Ist ein Zeichen einzufügen, weil es vergessen wurde, so muß zunächst die Taste CONTROL zusammen mit der Taste INSERT gedrückt werden. Ist hingegen ein Zeichen löschen, so muß die Taste CONTROL zusammen mit DELETE gedrückt werden.

Der Leser wird es nach einigem Üben sehr rasch lernen, geschickt mit diesen Tasten und Tastenkombinationen umzugehen, um dann sehr schnell in diesem Korrekturmodus arbeiten zu können.

Anhang 3 : Wichtige EDV-Stichworte Englisch-Deutsch

Access	Zugriff
Add	Addieren
Alphabetic	Alphabetisch
AND	Und
Array	Datenspalte
Binary	
Bit	Binäre Einheit
Blank	Leerzeichen
Byte	Speicherstelle (8 Bits)
Case	
Central Unit	Zentraleinheit
Character	Symbol (Zeichen)
CLEAR	Löschen
CLOSE	Schließen
Code	Code
COLOR	Farbe
Compute	Rechnen
Computer	Rechner
Constant	Konstante
Continue	Fortsetzen
Control	Kontrolle
Cursor	Blinker
DATA	Datum (Daten)
Data Processing	Datenverarbeitung
Decimal	Dezimal
Delete	Löschen
Digit	Ziffer
Dimension	Bereich
Direct access	Direkter Zugriff
Direct mode	Direktmodus
Disc	Platte
Display	Bildschirm
Divide	Dividieren
End	Ende
Enter	Eingeben
Equal	Gleich
Error	Fehler

Anhang 3 : Wichtige EDV-Stichworte Englisch-Deutsch

Field	Feld
File	Datei
Floppy Disc	Diskette
FOR ... TO	Für ... bis
Format	Formatieren (Format)
Free	Frei
Function	Funktion
GOTO	Gehen ... nach
GRAPHICS	Graphiken
Hardware	Geräte
High resolution	Hohe Auflösung
Home	Cursor-Rückkehr
If	Wenn
IF ... THEN	Wenn ... dann
INPUT	Eingabe
Instruction	Anweisung
Integer	Ganzzahliger Wert
Interchange	Austausch
Interface	Schnittstelle
Keyboard	Tastatur
Language	Sprache
Left	Links
Length	Länge
LET	Lassen
LIST	Auflisten
LOAD	Laden
Loop	Schleife
Memory	Speicher
Middle	Mitte
Minus	Minus
Multiply	Multiplizieren
NEW	Neu
NEXT	Nächste(r)
NOT	Nicht
Number	Anzahl
Numeric	Numerisch
Off-Line	Nicht angeschlossen
Online	Angeschlossen

Anhang 3 : Wichtige EDV-Stichworte Englisch-Deutsch

OPEN	Öffnen
Operate	Operieren (arbeiten)
Operating System	Betriebs-system
OR	Oder
Output	Ausgabe
Play	Spielen
PLOT	Zeichnen
Plotter	Zeichengerät
Plus	Plus
PRINT	Drucken
Printer	Drucker
Procedure	Prozedur
Process	Prozeß (Verarbeitung)
Program	Programm
Random	Zufall
READ	Lesen
Real	reell
Remark	Anmerken
RESTORE	Restaurieren
Right	Rechts
RUN	Laufen
SAVE	Sichern (Speichern)
Screen	Bildschirm
Set	Setzen (Datensatz)
Shift	Umschalten
Software	Programme
Sort	Sortieren
Space	Leerer Raum
Square root	Quadratwurzel
Statement	Anweisung
STEP	Schrittweite
STOP	Halt (anhal-ten)
String	Zeichenkette
Subroutine	Unterprogramm
Subscript	Index
Symbol	Symbol
Syntax Error	Sprachfehler
System	System
Tape	Band
Terminal	Konsole
Time	Zeit
Unequal	Ungleich

Unit	Einheit
Use	Benutzen
User	Benutzer
Value	Wert
Write	Schreiben
Zero	Null

Anhang 4 : Im Buch erwähnte Kommandos

** C **

CONT	89

** L **

LIST	66
LOAD	267, 362

** N **

NEW	66

** R **

RUN	61

** S **

SAVE	267

Anhang 5 : Im Buch erwähnte BASIC-Statements

** C **

COLOR 219

** D **

DATA 124, 36Ø
DIM 128, 361
DRAWTO 217

** E

END 6Ø, 362

** F **

FOR...TO 92, 361

** G **

GOSUB 131, 261, 362
GOTO 88, 361
GRAPHICS 218

** I **

IF...THEN 9Ø, 361
INPUT 64

** L **

LET 61
LPRINT 27Ø

** N **

NEXT 92, 361

** O **

ON...GOSUB 27Ø
ON...GOTO 269

** P **

PEEK 271
PLOT 216
POKE 27Ø
POSITION 192
PRINT 28, 38, 57, 215

** R **

READ	124, 36Ø
REM	94
RESTORE	126, 36Ø
RETURN	131, 362

** S **

SETCOLOR	21Ø
SOUND	271
STOP	93

Anhang 6 : Stichwortverzeichnis

** A **

ABKUERZUNG 268
ABS 86
ADDITION 4Ø
AND 96
ANFANGSBUCHSTABE 328
ANFUEHRUNGSZEICHEN 42
ANWEISUNG 16
ARBEITSSPEICHER 18, 281
ARGUMENT 86
ARITHMETISCHE, FUNKTION 86
ARITHMETISCHER, AUSDRUCK 58
ARITHMETISCHER, OPERATOR 4Ø
ARITHMETISCHES, MITTEL 116
ARRAY 1Ø, 127, 279
ASC 29, 163
ASCII-CODE 28, 194
AUFLOESUNG 242
AUFRUNDEN 99
AUSDRUCK, ARITHMETISCHER 58
AUSDRUCK, LOGISCHER 9Ø
AUSGABE 12, 57, 61
AUSGABEGERAET 18
AUSPRAEGUNG 9

** B **

BACKSPACE 35, 36, 8Ø
BALKENDIAGRAMM 34Ø
BALLSPIEL 2Ø4
BASIC 3, 51
BASIC-DIALEKT 51
BASIC-FUNKTION 85
BASIC-STATEMENT 85
BASIS 25, 26
BEDINGTER, SPRUNG 88, 9Ø
BEDINGUNG, LOGISCHE 9Ø
BESCHAEFTIGTENSTATISTIK 233

BETRIEBSSYSTEM 14, 2Ø, 29, 266, 28Ø
BEVOELKERUNG 83, 295
BILDSCHIRM 18
BILDSCHIRMHINTERGRUND 21Ø, 211, 212, 213
BILDSCHIRMRAHMEN 21Ø, 211, 212, 213
BINAERSYSTEM 24
BINAERZAHL 24
BINOMIALKOEFFIZIENT 154
BIT 7, 26
BLOCKGRAPHIK 215
BLOCKGRAPHIK-MODUS 217, 346
BREAK 35
BREAK-TASTE 89
BUCHSTABE 6, 8, 174
BYTE 8, 26, 28Ø, 281

** C **

C. 268
CAPS 36, 37
CASE 1Ø
CHARACTER 28Ø
CHR$ 28, 59, 162, 193. 215
CLOG 86
CODE 24, 27
COLOR 219, 288
COMPUTER 6
COMPUTERTECHNOLOGIE 5
CONT 89, 94
CONTROL 35, 66, 81
COS 87
COSINUS 87
CURSOR 23, 35, 81

** D **

D. 268
DATA 124, 125, 268
DATEI 11, 28Ø
DATEN 6, 15, 16, 124
DATENMATRIX 3Ø
DATENSATZ 1Ø
DATENVERARBEITUNG 5, 7, 123
DELETE 35, 36, 8Ø
DEZIMALSYSTEM 24
DEZIMALZAHL 24, 4Ø
DIAGONALE 24Ø, 349
DIALOG 64
DIM 113, 128, 129, 13Ø
DIMENSIONIERUNG 164
DIREKTMODUS 31, 38, 57, 283
DISKETTE 21
DISKETTEN-LAUFWERK 266
DIVISION 4Ø
DOPPELPUNKT 59
DOS 266
DR. 268
DRAWTO 217, 268
DREIECK 83, 12Ø, 292
DREIECK, RECHTWINKLIGES 3Ø1
DREISATZRECHNUNG 298
DRUCKER 18, 27Ø
DUALSYSTEM 24
DURCHSCHNITT 47, 116, 159

** E **

E 87
EINGABE 12, 57
EINGABEGERAET 18
EINMALEINS 1Ø1
ELLIPSE 263, 351
END 6Ø
ENDLOSPROGRAMM 89
ERROR 81, 82, 36Ø
EVA-PRINZIP 12, 68, 28Ø
EXP 87
EXTERNER, SPEICHER 18, 21

** F **

FAKULTAET 120, 154, 306
FAMILIENSTAND 159
FARB-BRILLIANZ 211
FARBAUSWAHL 211
FARBE 191, 210, 211, 219, 220
FARBGEBUNG 210
FARBMOEGLICHKEIT 344
FARBTABELLE 211
FEHLER 360
FEHLERMELDUNG 82, 359
FELD 8, 279, 286
FILE 11, 280
FILES 266
FLUSSDIAGRAMM 47
FOR...TO 92
FUNKTION 86
FUNKTION, ARITHMETISCHE 86
FUNKTION, TRIGONOMETRISCHE 87
FUNKTIONSNAME 86
FUNKTIONSTASTE 33
FUNKTIONSWEISE 12

** G **

G. 268
GAUSS-PROBLEM 120, 304
GERADE 245
GERAETEKONFIGURATION 18
GEWINN 73
GITTER 240, 346
GOS. 268
GOSUB 131, 132, 268
GOTO 88, 132, 268, 269
GRAPHICS 218, 243
GRAPHICS, BUSINESS 233
GRAPHIK, HOCHAUFLOESENDE 241
GRAPHIK-CURSOR 217
GRAPHIK-SYMBOL 191, 193
GROESSTER, WERT 159, 287, 313
GROSSBUCHSTABE 190, 332

** H **

HAEUFIGKEITSVERTEILUNG 146, 16Ø, 213, 29Ø, 322, 34Ø
HELP 34
HINTERGRUNDFARBE 22Ø
HOCHAUFLOESENDE, GRAPHIK 241
HYPOTHEKENTILGUNG 1Ø5
HYPOTHENUSE 3Ø1

** I **

I. 268
IF...THEN 9Ø, 96, 97, 269
INDEX 127
INDIZIERTE, VARIABLE 126, 127
INFORMATION 6, 57
INFORMATIONSAUFNAHME 12
INFORMATIONSAUSGABE 18
INFORMATIONSEINGABE 18, 61
INFORMATIONSSPEICHERUNG 18
INFORMATIONSVERARBEITUNG 12, 18
INPUT 64, 65, 124, 268
INSERT 35, 36
INT 87, 97 99

** K **

KILOBYTE 19
KLAMMER 4Ø
KLEINBUCHSTABE 19Ø
KLEINSCHREIBWEISE 332
KLEINSTER, WERT 159, 287, 313
KOMMA 41, 59
KOMMANDO 15, 16, 51, 28Ø, 281
KOMMANDOMODUS 31
KOMMUNIKATION 15
KORRIGIEREN 8Ø
KREIS 44, 68, 251, 283
KREISFLAECHE 69
KREISPARAMETER 199
KREISUMFANG 69

** L **

L. 268
LAUTSTAERKE 273
LEERZEILE 58
LEN 163
LESEN 124
LET 61, 62, 124
LIST 66, 81, 268
LOAD 267
LOG 87
LOGARITHMUS, NATUERLICHER 87
LOGIK 22
LOGISCHE, BEDINGUNG 90
LOGISCHER, AUSDRUCK 90
LOTTO 154
LPRINT 270

** M **

MAGNETBANDKASSETTE 21
MASTER-DISKETTE 266
MEHRWERTSTEUER 44, 284
MITTEL, ARITHMETISCHES 116
MITTELWERT 116, 141
MULTIPLIKATION 40

** N **

N. 268
NAME 9, 62
NATUERLICHER, LOGARITHMUS 87
NEW 66
NEXT 92, 268
NOT 96
NOTBREMSE 89
NUMERISCHE, VARIABLE 63, 279
NUMERISCHER, WERT 280

** O **

ON...GOSUB 27Ø
ON...GOTO 269
OPERATOR, ARITHMETISCHER 4Ø
OPTION 34
OR 96

** P **

PEEK 271
PERIPHERER, SPEICHER 21
PL. 268
PLOT 216, 268
POKE 27Ø
POS. 268
POSITION 192, 215, 268
POTENZIERUNG 4Ø
PR. 268
PRIMZAHL 111
PRINT 8, 28, 38, 41 57, 58, 13Ø, 268
PROBLEMANALYSE 46, 47
PROGRAMM 32
PROGRAMMMODUS 31, 32
PROGRAMMABLAUFPLAN 47
PROGRAMMANWEISUNG 15, 16, 28Ø, 281
PROGRAMMSPRUNG 88
PROGRAMMVERZWEIGUNG 88
PROZENT 298
PROZENTBERECHNUNG 83
PUNKT 4Ø
PYTHAGORAS 3Ø1

** Q **

QUADRAT 95, 263, 356
QUADRATWURZEL 87
QUADRATZAHL 77

** R **

RADIUS 69
RAM 2Ø
RAM-SPEICHER 19, 281
READ 124, 125, 126
READY 23, 39
RECHENWERK 18
RECHNERSYSTEM 12
RECHTECK 44, 196, 213, 283, 336
RECHTWINKLIGES, DREIECK 3Ø1
RECORD 1Ø
REM 94
RESET 34
RESTORE 126
RET. 268
RETURN 37, 131, 132, 268
RETURN-TASTE 28, 39, 58, 8Ø, 266
RND 87
ROM 19
ROM-SPEICHER 19, 281
RUN 16, 61, 8Ø
RUNDUNG 1ØØ

** S **

S. 268
SATZNUMMER 57, 281
SAVE 267, 268
SCHLEIFE 86, 92
SE. 268
SELECT 34
SEMIKOLON 41
SETCOLOR 21Ø, 22Ø, 268
SHIFT 36, 37
SIN 87

SINUS 87
SINUSLINIE 198, 258
SONDERTASTE 33
SONDERZEICHEN 6, 8
SORTIEREN 133
SOUND 271
SPALTENINDEX 128
SPEICHER, EXTERNER 18, 21
SPEICHER, PERIPHERER 21
SPEICHERERWEITERUNG 18, 19
SPEICHERFELD 58
SPEICHERKAPAZITAET 19, 21
SPEICHERSTELLE 9, 27Ø
SPRUNG, BEDINGTER 88, 9Ø
SPRUNG, UNBEDINGTER 88
SPRUNGANWEISUNG 88
SQR 87
STABDIAGRAMM 213
STANDARDABWEICHUNG 159, 318
START 34
STEP 92, 93
STEUERWERK 18
STOP 93, 95, 97
STR$ 163
STREUUNG 318
STREUUNGSMASS 159
STRICHPUNKT 41, 59
STRING 1Ø, 42, 58, 162, 279
STRING-FUNKTION 162
STRING-VARIABLE 63
STRINGVARIABLE 129, 164
SUBROUTINE 131
SUBTRAKTION 4Ø
SUCHEN 174
SYMBOL 6, 8, 28Ø
SYMBOLTASTE 33

** T **

TASTATUR 18, 33
TEXT 6
TEXTMODUS 217
TEXTVERARBEITUNG 161
TONHOEHE 272
TONKANAL 272
TRAPEZ 44, 284
TRIGONOMETRICCHE, FUNKTION 87

** U **

UEBERSETZUNG 166
UNBEDINGTER, SPRUNG 88
UNTERPROGRAMM 124, 131, 132

** V **

VAL 164
VARIABLE 9, 1Ø, 42
VARIABLE, INDIZIERTE 126, 127
VARIABLE, NUMERISCHE 63, 279
VARIABLENNAME 58, 63, 86
VERARBEITUNG 12
VERKNUEPFUNGSOPERATOR 96
VERZERRUNG 273
VERZWEIGUNG 86
VOKABEL 166
VOKABELTEST 18Ø

** W **

WARTESCHLEIFE 223
WERT 6, 9, 279
WERT, GROESSTER 159, 287, 313
WERT, KLEINSTER 159, 287, 313
WERT, NUMERISCHER 28Ø
WORT 6
WUERFEL 44, 285
WUERFELWURF 121, 159, 3Ø9
WURZEL 95

** Z **

ZAHL 6
ZAHL, EULERSCHE 87
ZAHLENSYSTEM 24
ZEHNERSYSTEM 25
ZEICHENKETTE 1Ø, 42
ZEILENINDEX 128
ZENTRALEINHEIT 18
ZIFFER 6, 8
ZUFALLSLINIE 228
ZUFALLSMUSTER 222
ZUFALLSZAHL 87, 3Ø9
ZWEIERPOTENZ 26
ZWEIERSYSTEM 26

DAS STEHT DRIN:

Das BASIC-TRAININGSBUCH zu ATARI 600XL/ 800XL ist eine ausführliche, didaktisch gut geschriebene Einführung in das ATARI BASSIC. Von den BASIC-Befehlen über die Problemanalyse bis zum fertigen Algorithmus lernt man schnell und sicher das Programmieren.

Aus dem Inhalt:

- Grundlagen des Programmierens
- Zahlensysteme und Codes
- BASIC-Befehle
- Funktionen, Verzweigungen, Schleifen
- Komplexere BASIC-Programme
- Nutzung von Unterprogrammen
- Blockgrafik
- Hochauflösende Grafik
- Grundelemente der Textverarbeitung
  und viele Beispielprogramme.

UND GESCHRIEBEN HAT DIESES BUCH:

Werner Voß ist Professor für Statistik an der Universität Bochum. Zahlreiche Veröffentlichungen im Bereich Statistik und Datenverarbeitung.

ISBN 3-89011-057-6

Voß / Das BASIC-Trainingsbuch zu ATARI ® 600XL / 800XL

ATARI