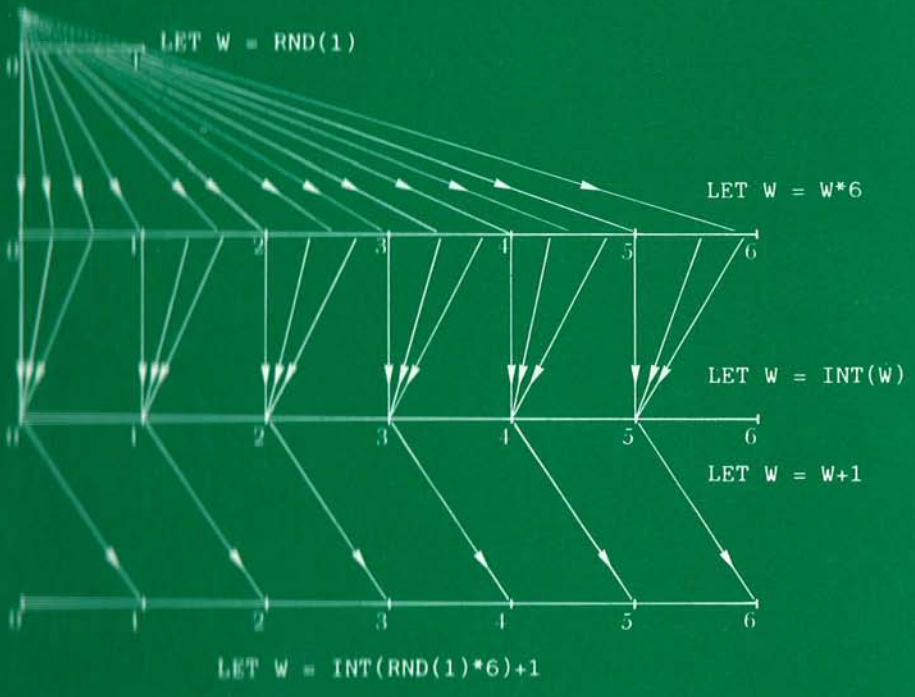
Jochen Ziegenbalg

Anwendungsbereiche für Kleincomputer

Unterrichtsbeispiele für die Sekundarstufe I mit 64 Programmen und 40 Aufgaben



Ferdinand Schöningh

Jochen Ziegenbalg

Anwendungsbereiche für Kleincomputer

Unterrichtsbeispiele für die Sekundarstufe I mit 64 Programmen und 40 Aufgaben

Ferdinand Schöningh

3. überarbeitete Auflage 1984

ISBN 3-506-37465-6

INHALTSVERZEICHNIS

VORWORT

Schon seit Jahren ist klar, daß Computern und Algorithmen im Mathematikunterricht der Zukunft eine außerordentlich wichtige Rolle zukommen wird. Die bisher vorgelegten Unterrichtsvorschläge bezogen sich vorwiegend auf den Oberstufenunterricht. Mit vielen Kollegen bin ich jedoch der Auffassung, daß gewisse Elemente des Algorithmierens und des Arbeitens mit Computern ihren eigentlichen Platz im Mittelstufenunterricht haben sollten. Die fachdidaktische Begründung dieser These werde ich u. a. in den Schlußbemerkungen ansprechen, wo auf konkrete Inhalte Bezug genommen werden kann.

Solange solche Überlegungen auf der theoretischen Ebene hochschulinterner Diskussionen durchgeführt werden, wirken sie nicht unbedingt überzeugend, so richtig sie auch sein mögen. Darüber hinaus steckt beim Arbeiten mit Computern der Teufel besonders intensiv im Detail.

Um konkrete Unterrichtserfahrungen im Mittelstufenunterricht zu erhalten, führe ich deshalb seit dem Sommer 1980 ein Unterrichtsprojekt zu diesem Problemkreis am Bildungszentrum Reutlingen-Nord (BZN) durch. Das BZN gliedert sich in einen Hauptschul-, einen Realschul- und einen Gymnasialzweig. Für den Oberstufenunterricht des Gymnasialzweiges waren einige Tischcomputer beschafft worden. Aufgrund der schulartintegrierenden Struktur des BZN, insbesondere im Bereiche der Arbeitsgemeinschaften, stehen diese Rechner im Prinzip aber auch den Mittelstufenschülern aller Schulzweige zur Verfügung. Diese organisatorischen Rahmenbedingungen und insbesondere das großzügige Entgegenkommen der Rektoren des BZN machten es mir möglich, das Unterrichtsprojekt

"Anwendungsbereiche für Kleincomputer"

mit Schülern der Mittelstufe durchzuführen. Das Projekt fand in Form mehrerer Einführungs- und eines Fortgeschrittenenkurses statt, an denen Schüler aller Schulzweige des BZN (hauptsächlich aus den Schuljahren 8 und 9) beteiligt waren. Dadurch war es insbesondere auch möglich, Erfahrungen zu der Frage zu machen, wie Haupt- und Realschüler auf das Arbeiten mit Computern reagieren. Das war für mich besonders deswegen sehr wertvoll, weil Computer im Hauptschul- und Realschulbereiche hierzulande normalerweise nicht zur Verfügung ste-

hen und solche Erfahrungen an den normalen Haupt- und Realschulen somit (noch) nicht gewonnen werden können.

Das Ziel des Unterrichtsprojekts bestand nicht vorrangig in der Einführung in eine bestimmte Programmiersprache oder einen bestimmten Rechnertyp, sondern in der inhaltlichen und methodischen Erschließung algorithmisch orientierter Themenkreise. Ich war insbesondere bestrebt, die in der Literatur häufig anzutreffende syntaktisch orientierte Methode zugunsten einer problemorientierten Einführung in das Arbeiten mit Computern zu überwinden; syntaktische Elemente wurden an geeigneten Themenkreisen entwickelt.

Kriterien für die Auswahl der Unterrichtsbeispiele waren vorrangig ihr Anwendungsbezug, aber auch eine gewisse Unterrichtsnähe. Im Bereiche der angesprochenen Fragestellungen war ich bemüht, zwar sehr elementare, aber dennoch von der Ausgangssituation her einigermaßen typische Probleme heranzuziehen. Obwohl sich die Schüler oft durch Computerspiele fasziniert zeigten, wurden letztere in der Unterrichtseinheit nicht berücksichtigt, da einerseits ihre Relevanz fraglich ist und da andererseits ihre Programmierung meist maschinensprachliche Elemente enthält, die im normalen Unterricht kaum sinnvoll zu vermitteln sind.
Die Auswahl der Inhalte ist auch in Verbindung mit dem in (R.Dürr/J.Ziegenbalg) beschriebenen Konzept der Ausgewogenheit im Mathematikunterricht zu sehen. Besonders auch in die Schlußbemerkungen ist viel von den dortigen Überlegungen eingeflossen. Allerdings wurde im Projekt weniger die begrifflich/strukturelle als vielmehr die konstruktiv/algorithmische Seite der Problematik betont, da es gerade in bezug auf das Algorithmieren und das Arbeiten mit Computern an Erfahrungen mangelt.

Da die zur Verfügung stehenden Rechner reine Tischcomputer ohne Peripherie waren, konnte auf Probleme der Dateienverarbeitung nicht eingegangen werden. Es stimmt mich bedenklich, daß unsere Gesellschaft es zwar für richtig hält, jedem Schüler Lernmittel zu schenken, die einen Minimalwert (im Unterrichtszeitraum: 1 DM) überschreiten, andererseits aber nicht in der Lage ist, einer Schule eine zukunftsweisende Investition von ca. 3.000,-- DM zu ermöglichen.

Am Unterrichtsprojekt nahmen zeitweise Lehrer des BZN zum Zwecke der persönlichen Weiterbildung teil. Ich habe die im

Projekt erarbeiteten Materialien insbesondere deswegen zur Veröffentlichung aufbereitet, weil ich die Hoffnung habe, daß diese Darstellung interessierten Lehrern, welche derartige Themenkreise in der Phase ihrer Ausbildung nicht kennengelernt haben (und das dürften die meisten sein), als Starthilfe dienen kann. Die Darstellung wendet sich demgemäß vorrangig an Lehrer und Lehrerstudenten, im weiteren Sinne aber auch an interessierte Schüler und Laien.
Sie ist nicht als Bedienungsanleitung für Computer-Neulinge gedacht und will eine solche auch nicht ersetzen.

Die Programme sind methodisch weitestgehend entsprechend dem Unterrichtsverlauf aufbereitet worden: Zunächst wurde in möglichst einfacher Form der Kern des Algorithmus herausgeschält und danach - oft in mehreren Stufen - weiterführende Aspekte hinzugenommen. Sie sind nicht mit dem Bestreben nach Perfektion geschrieben worden, sondern als Plattform zur weiteren Ausarbeitung und Verbesserung konzipiert.

Ein Teil des Unterrichtsprojekts wurde während eines vom Ministerium für Wissenschaft und Kunst dieses Landes bewilligten Forschungs- und Fortbildungssemesters durchgeführt.

Mein Dank gilt den Schülern und Lehrern des Bildungszentrums Reutlingen-Nord, die an dem Unterrichtsprojekt teilgenommen haben, für ihre Mitarbeit. Auch ich habe von ihnen gelernt. Den Rektoren des BZN, Herrn Dolderer, Herrn Greiner und Herrn Korger, bin ich für ihr großzügiges Entgegenkommen bei allen anfallenden schulorganisatorischen Problemen dankbar. Herrn Prof. M. Riedel von der University of Valparaiso, Indiana (U.S.A.), Gastprofessor an der Pädagogischen Hochschule Reutlingen, danke ich für die Hilfe bei der Übersetzung von Fachtermini (besonders aus dem wirtschaftlichen Bereiche) für eine Kurzbeschreibung des Projekts ins Englische bzw. Amerikanische. Herrn Rolf Dürr, Studienrat für Mathematik- und Physikunterricht am BZN, danke ich für seine uneingeschränkte Hilfsbereitschaft und Unterstützung.

Jochen Ziegenbalg

§1 EINFÜHRUNG IN DEN UMGANG MIT COMPUTERN

1.1 Grobüberblick über den Aufbau von Computern

Computer sind Maschinen. In den meisten Fällen arbeiten Maschinen so, daß sie zunächst mit Eingabe-Objekten gefüttert werden müssen. Die Eingabe wird dann intern in der Maschine verarbeitet, und das Ergebnis dieses Verarbeitungsvorganges verläßt die Maschine als Endprodukt ("Ausgabe"). Ein- und Ausgabeobjekte können materieller Natur sein, wie z. B. bei Holzverarbeitungsmaschinen. Beim Arbeiten mit Computern besteht die Ein- und Ausgabe meist aus Daten (Informationen). Diese sind nichtmaterieller Art, obwohl sie natürlich auf materiellen Datenträgern (z. B. Magnetbändern) gespeichert sind.

Das Grundschema der Arbeitsweise von Computern läßt sich also folgendermaßen darstellen:

Eingabe -> Verarbeitung -> Ausgabe

abgekürzt: E -> V -> A.

Entsprechend ihrer Arbeitsweise bestehen Computer aus verschiedenen, miteinander gekoppelten Funktionseinheiten:

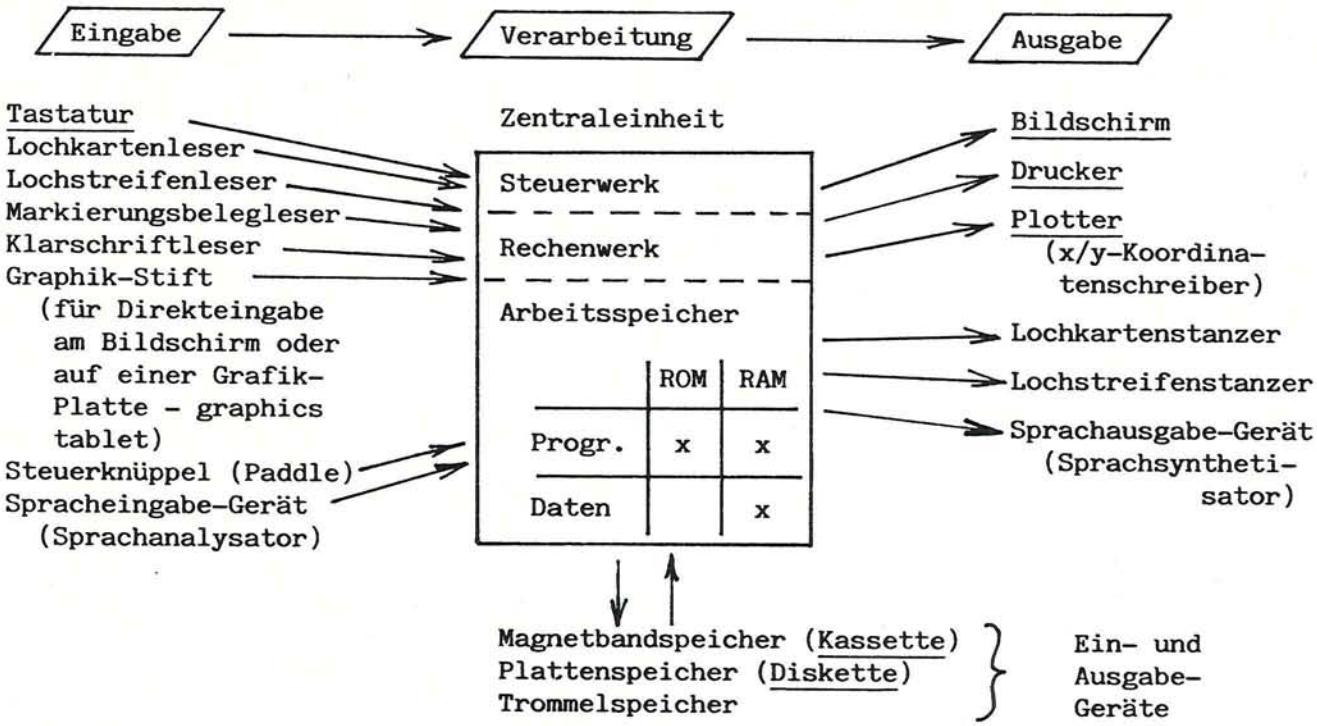
Eingabegeräte -> Zentraleinheit -> Ausgabegeräte.

Bei den im Unterricht vielfach verwendeten Tischcomputern ist die Standard-Eingabeeinheit die Tastatur und die Standard-Ausgabeeinheit der Bildschirm. Ein Überblick über weitere Ein- und Ausgabegeräte ist in Abb.1.1 gegeben. Die Zentraleinheit des Computers enthält das Steuerwerk, das Rechenwerk und den internen Speicher (Daten- und Programmspeicher). Über Datenleitungen stehen diese Funktionseinheiten miteinander in Verbindung. Den gesamten physikalisch-gerätetechnischen Bereich des Computers bezeichnet man auch als den Hardware-Bereich.

Ein Befehl oder eine Kette von Befehlen, durch die dem Computer in einer ihm verständlichen Sprache gesagt wird, was er tun soll, heißt ein Programm. Als physikalisches Gerät versteht der Computer zunächst einmal nur eine sehr unanschauliche und schwierig zu handhabende Maschinenspra-

Abb.1.1

Der Computer: ein grober Überblick



Bemerkungen: (1.) Die Abkürzungen bedeuten:
ROM - Read Only Memory: Nur-Lese-Speicher
RAM - Random Access Memory: Schreib-und-Lese-Speicher

(2.) Die unterstrichenen Geräte sind im Microcomputerbereich besonders häufig vertreten. Lochkarten- und Lochstreifengeräte waren in der Anfangszeit der Datenverarbeitung weit verbreitet; ihre Bedeutung geht zurück.

che. Um dem Menschen das Kommunizieren mit dem Computer zu erleichtern, wurden Übersetzungs- bzw. Interpretationsprogramme entwickelt, mit deren Hilfe man den Computer in einer "menschlicheren" Sprache ansprechen kann. Im Verlaufe der Zeit entstand so eine ganze Palette von immer komfortableren bzw. auf bestimmte Anwendungszwecke zugeschnittenen Programmier- und Systemsprachen.

Im Bereiche des Tischcomputersektors ist heute die von J. G. Kemeney und T. E. Kurtz vom Dartmouth College in New Hampshire (USA) entwickelte Programmiersprache BASIC weit verbreitet. Es ist eine leicht erlernbare und unkompliziert zu handhabende Sprache, in der sich kürzere und strukturell einfache Programme schnell und ohne tiefere Systemkenntnisse schreiben lassen. Aufgrund ihrer strukturellen Einfachheit läßt sie sich auch auf Tischrechnersystemen mit geringer Speicherkapazität "implementieren" (installieren?). Es treten aber i. a. Probleme auf, wenn man versucht, komplexe Aufgaben in BASIC zu lösen. Besonders für die Verwendung im Bildungsbereich wurde eine Reihe von Sprachen entwickelt, in denen einige der Schwachstellen von BASIC (an vorderster Stelle: die fehlenden Möglichkeiten zur logisch/sachlichen Gliederung von Programmen) überwunden wurden. Hierzu zählen Sprachen wie PASCAL, ELAN, LOGO und COMAL.

Im folgenden wird dennoch die Programmiersprache BASIC verwendet, weil:

(1.) sie zur Zeit noch die im Tischcomputerbereich am stärksten verbreitete Sprache ist,
(2.) die hier behandelten Themenkreise von ihrer logisch/sachlichen Seite her meist so einfach strukturiert sind, daß die fehlenden Strukturierungsmöglichkeiten von BASIC nicht allzu stark ins Gewicht fallen,
(3.) BASIC eine interaktiv angelegte Sprache mit "kurzen Wegen" zwischen Programmentwurf und Programmauswertung ist. (BASIC kommt mir in mancher Hinsicht wie eine Art Polaroid für Computer vor.)

Ergänzend zur Computer-Hardware kommen also noch die Übersetzer-, Interpreter- und sonstigen Betriebs- und Anwenderprogramme hinzu. Die Gesamtheit dieser Betriebs- und Dienstleistungsprogramme wird auch unter dem Begriff Computer-Software zusammengefaßt.

1.2 Grundtechniken im Umgang mit Computern (an einfachen Problemstellungen erläutert)

Ein sehr einfaches Problem, um das Grundschema

E → V → A

der Arbeitsweise von Computern zu veranschaulichen, ist die

Mehrwertsteuerberechnung

Problemstellung: Zu einem vorgegebenen Nettobetrag N sind die Mehrwertsteuer M und der Bruttobetrag B zu berechnen und auszugeben.
(Zusatzinformation: der Steuersatz beträgt 14 %.)

Das Programm MEHRWERTSTEUER-1 zeigt eine mögliche Problemlösung.

Programm: MEHRWERTSTEUER-1

```
10  INPUT N
20  LET M = N * 0.14
30  LET B = N + M
40  PRINT M
50  PRINT B
60  END
```

Der Nettobetrag N ist ein Eingabe-Objekt. Er wird durch den Befehl INPUT N eingelesen. Wenn der Computer beim Abarbeiten eines Programmes auf eine INPUT-Anweisung stößt, wartet er auf eine Eingabe des Benutzers über die Tastatur. Ist diese (durch carriage return - Wagenrücklauf) abgeschlossen, so wird der Eingabewert der Eingabevariablen (im obigen Falle: N) zugeordnet.
Der Verarbeitungsteil des Programmes besteht aus den Anweisungen LET M = N*0.14 und LET B = N+M. Sie stellen Wertzuweisungen dar. Der Wert 0.14*N (der Stern dient als Multiplikationszeichen) wird der Variablen M und der Wert N+M der Variablen B zugewiesen.

Durch die PRINT-Befehle werden die Werte von M und B ausgegeben.
Der Befehl END bedeutet: Ende der Programmausführung.
Jeder der Anweisungen ist eine Zeilennummer (hier im Zehnersprung) vorangestellt. Wenn ihm nichts anderes gesagt wird, arbeitet der Computer das Programm auf das Kommando RUN nach aufsteigenden Zeilennummern ab.

Die Eingabe des Programmes in den Computer erfolgt in der Reihenfolge: Zeilennummer, Anweisung (eventuell mit Parametern), RETURN-Taste, usw.
Das Gesamtprogramm wird mit Hilfe des Kommandos LIST aufgelistet. Es kann jederzeit durch Einfügen neuer oder Veränderung alter Zeilen modifiziert werden.
Das Kommando NEW löscht ein etwaiges im Speicher befindliches BASIC-Programm. Es ist empfehlenswert, dieses Kommando zu geben, wenn man mit dem Schreiben eines neuen Programmes beginnt.

(A.1.1) Ändern Sie Zeile 30 folgendermaßen ab:

```
30 LET B=N*1.14
```

und vergleichen Sie die Ergebnisse der alten und der neuen Version.

Das Programm MEHRWERTSTEUER-1 ist außerordentlich einfach und in manchen Punkten verbesserungsbedürftig. Aber auch für wesentlich augefeiltere Programme gilt der

"HAUPTSATZ" DER INFORMATIK:

Jedes Programm ist verbesserungsfähig.

Einer der wichtigsten Grundsätze, nach denen Programme geschrieben werden sollten, ist die Benutzerfreundlichkeit. Auch ein Benutzer, der ein bestimmtes Programm nicht geschrieben hat, sollte es laufen lassen können und während des Programmlaufes durch das Programm selbst mit Informationen versorgt werden, wo es nötig oder sinnvoll ist. So sollte z. B. vor Eingabe-Befehlen stets angegeben werden, was als nächstes einzugeben ist. Ebenso sollte die Ausgabe nicht nur aus "nackten" Zahlen bestehen.
Diesen Zielen wird im Programm MEHRWERTSTEUER-2 durch die Zeilen 10, 50 und 70 Rechnung getragen, in denen vor der jeweiligen Ein- bzw. Ausgabe von Variablen durch eine

in Anführungszeichen stehende Textkonstante die Bedeutung der Zahlenwerte vermittelt wird.

Programm: MEHRWERTSTEUER-2

```
10  PRINT "NETTOBETRAG: "
20  INPUT N
30  LET M = N * 0.14
40  LET B = N * 1.14
50  PRINT "MEHRWERTSTEUER: "
60  PRINT M
70  PRINT "BRUTTOBETRAG: "
80  PRINT B
90  END
```

Im Programm MEHRWERTSTEUER-3 wird durch das Semikolon jeweils hinter den Textkonstanten in Zeile 10, 50 und 70 bewirkt, daß die Zahlenwerte unmittelbar hinter die Erläuterungen geschrieben werden.

Programm: MEHRWERTSTEUER-3

```
10  PRINT "NETTOBETRAG: ";
20  INPUT N
30  LET M = N * 0.14
40  LET B = N * 1.14
50  PRINT "MEHRWERTSTEUER: ";
60  PRINT M
70  PRINT "BRUTTOBETRAG: ";
80  PRINT B
90  END
```

Zeile 10 in MEHRWERTSTEUER-4 ist eine abkürzende Schreibweise, durch welche die Befehle in Zeile 10 und 20 aus MEHRWERTSTEUER-3 in einer Anweisung vereinigt werden.

Programm: MEHRWERTSTEUER-4

```
10  INPUT "NETTOBETRAG: ";N
20  LET M = N * 0.14
30  LET B = N * 1.14
40  PRINT "MEHRWERTSTEUER: ";
50  PRINT M
60  PRINT "BRUTTOBETRAG: ";
70  PRINT B
80  END
```

Ebenso werden durch Zeile 40 in MEHRWERTSTEUER-5 die Zeilen 40 und 50 aus MEHRWERTSTEUER-4 zusammengefaßt.

Programm: MEHRWERTSTEUER-5

```
10  INPUT "NETTOBETRAG: ";N
20  LET M = N * 0.14
30  LET B = N * 1.14
40  PRINT "MEHRWERTSTEUER: ";M
50  PRINT "BRUTTOBETRAG: ";B
60  END
```

In MEHRWERTSTEUER-6 steht die Anweisung REM in den Zeilen 10 bis 50 für REMARK (Bemerkung). REM-Anweisungen dienen der Dokumentation des Programm-Textes; sie werden bei der Ausführung des Programmes übergangen.

Programm: MEHRWERTSTEUER-6

```
10  REM  *** DER COMPUTER ALS MEHRWERTSTEUER-MASCHINE ***
20  REM    BEZEICHNUNGEN:
30  REM      N: NETTOBETRAG
40  REM      M: MEHRWERTSTEUER
50  REM      B: BRUTTOBETRAG
60  INPUT "NETTOBETRAG: ";N
70  LET M = N * 0.14
80  LET B = N * 1.14
90  PRINT "MEHRWERTSTEUER: ";M
100  PRINT "BRUTTOBETRAG: ";B
110  END
```

(A.1.2) Ermitteln Sie unter ausschließlicher Verwendung des Programmes MEHRWERTSTEUER-6, wie hoch der Nettopreis eines Eises ist, das für 1 DM (Bruttopreis) verkauft wird.

(A.1.3) Ändern Sie das Programm MEHRWERTSTEUER-6 so ab, daß der zu einem Bruttopreis (Eingabe) gehörende Nettopreis auf direktem Weg ermittelt wird.

Prozentrechnung

Die Berechnung der Mehrwertsteuer ist ein Sonderfall des allgemeineren Problems: Berechnung des Prozentwertes aus Grundwert und Prozentsatz. In einem Programm zur Lösung dieses Problems muß neben dem Grundwert auch der Prozentsatz variabel eingebbar sein (siehe Zeile 80 in Programm PROZENT-WERT).

Programm: PROZENTWERT

```
10  REM  *** PROZENTWERT ***
20  HOME
30  PRINT "BERECHNUNG DES PROZENTWERTES NACH: "
40  PRINT
50  PRINT  TAB( 10)"PW = GW*PS"
60  PRINT
70  INPUT "GRUNDWERT:     GW = ";GW
80  INPUT "PROZENTSATZ:   PS = ";PS
90  PRINT
100  LET PW = GW * PS
110  PRINT "PROZENTWERT:   PW = ";PW
120  END
```

Durch den HOME-Befehl wird der Bildschirm gelöscht und der Cursor in die linke obere Bildschirmecke ("Home"-Position) plaziert. Der TAB-Befehl in Zeile 50 dient der Ausgabe--Formatierung durch Tabellierung.

(A.1.4) Der Name von Dezimalzahl-Variablen (REAL-Variablen) kann aus einem oder aus zwei Zeichen bestehen. Das erste Zeichen muß ein Buchstabe sein, das zweite Zeichen kann ein Buchstabe oder eine Zahl sein. Wie viele Namen stehen in BASIC für REAL-Variable zur Verfügung?

(A.1.5) Im Programm PROZENTWERT ist der Prozentsatz als reine Dezimalzahl einzugeben (z. B.: 4% muß als 0.04 eingegeben werden). Ändern Sie das Programm so ab, daß der Prozentsatz in % (d. h. in Hundertsteln) einzugeben ist. (Z. B. soll 25.75 % als 25.75 eingegeben werden.) Ändern Sie auch den Ein- und Ausgabedialog entsprechend ab.

(A.1.6) Modifizieren Sie das PROZENTWERT-Programm so, daß es zur Berechnung von Jahreszinsen und auch von unterjährigen Zinsen verwendet werden kann.

(A.1.7) (a) Ändern Sie das Prozentwertprogramm zu einem Prozentsatz- bzw. Grundwert-Programm ab. (D. h.: aus jeweils zwei Größen der Prozentrechnung ist die dritte Größe zu berechnen.)

(b) Vereinigen Sie alle drei Programme aus (a) in einem "Universalprogramm" zur Prozentrechnung.

Das Programm PROZENTRECHNUNG stellt eine mögliche Lösung dar. Beim Ausführen der Zeile

```
260 IF NR=1 THEN GOTO 310
```

wird die weitere Ausführung des Programmes von der Bedingung "NR = 1" abhängig gemacht. Ist sie erfüllt, so wird der Programmlauf in Zeile 310 fortgesetzt (durch den Sprungbefehl: GOTO 310); andernfalls wird die nächste Programmzeile (hier 270) ausgeführt. Zeile 300 zeigt, daß auch Kommandos der Betriebssprache (z. B. RUN) als Anweisungen in Programmen verwendet werden können.

(A.1.8) Wann wird der Befehl in Zeile 300 ausgeführt?

Im Gegensatz zu den bedingten Sprungbefehlen in den Zeilen 260, 270 und 280 stellt der GOTO-Befehl in Zeile 400 einen unbedingten Sprung dar. Zeile 600 zeigt, daß auch mehrere, durch : (Doppelpunkt) getrennte Anweisungen in einer Zeile stehen können. Die maximale Zeilenlänge ist rechnerabhängig. Meist ist die maximale Länge einer Programmzeile größer als eine Bildschirmzeile; die entsprechende Programmzeile wird dann einfach fortlaufend über den Bildschirmrand hinaus geschrieben. Da sehr komprimiert geschriebene Programme meist nicht gut lesbar sind, wird empfohlen, von der Möglichkeit, mehrere Befehle auf eine Zeile zu schreiben, keinen extensiven Gebrauch zu machen.
Die Handhabung des INPUT-Befehles ist stark rechnerabhängig. Deshalb führt Zeile 610 bei verschiedenen Rechnern zu unterschiedlichen Resultaten.

(A.1.9) Schreiben Sie ein Programm zur Berechnung des Wochenlohnes.
Eingabe: Arbeitszeit (Stunden), Stundenlohn, Lohnsteuersatz, Freibetrag;
Ausgabe: Bruttolohn,Lohnsteuer, Nettolohn.

Programm: PROZENTRECHNUNG

```
10  REM  *** PROZENTRECHNUNG ***
20  HOME
30  PRINT "PROZENTRECHNUNG"
40  PRINT : PRINT
50  PRINT "GRUNDGLEICHUNG DER PROZENTRECHNUNG:"
60  PRINT
70  PRINT "   PW = GW*PS"
80  PRINT
90  PRINT "BEZEICHNUNGEN:"
100  PRINT
110  PRINT "   GW : GRUNDWERT"
120  PRINT "   PW : PROZENTWERT"
130  PRINT "   PS : PROZENTSATZ"
140  PRINT "   P  : PROZENTSATZ IN PROZENT"
150  PRINT "   PS = P/100"
160  PRINT : PRINT
170  PRINT "MENU:"
180  PRINT
190  PRINT "1:  PROZENTWERT GESUCHT"
200  PRINT "2:  PROZENTSATZ GESUCHT"
210  PRINT "3:  GRUNDWERT GESUCHT"
220  PRINT "4:  ENDE"
230  PRINT
240  INPUT "BITTE WAEHLEN SIE:  NR.: ";NR
250  HOME
260  IF NR = 1 THEN  GOTO 310
270  IF NR = 2 THEN  GOTO 410
280  IF NR = 3 THEN  GOTO 510
290  IF NR = 4 THEN  END
300  RUN
310  PRINT "BERECHNUNG DES PROZENTWERTES NACH: "
320  PRINT
330  PRINT  TAB( 10)"PW = GW*P/100"
340  PRINT
350  INPUT "GRUNDWERT:    GW = ";GW
360  INPUT "PROZENTSATZ (IN %):  P = ";P
370  PRINT
380  LET PW = GW * P / 100
390  PRINT "PROZENTWERT:  PW = ";PW
400  GOTO 600
410  PRINT "BERECHNUNG DES PROZENTSATZES NACH: "
420  PRINT
430  PRINT  TAB( 10)"P = (PW/GW)*100"
440  PRINT
```

Programm: PROZENTRECHNUNG (Fortsetzung)

```
450  INPUT "PROZENTWERT:  PW = ";PW
460  INPUT "GRUNDWERT:    GW = ";GW
470  PRINT
480  LET P = (PW / GW) * 100
490  PRINT "PROZENTSATZ:  P = ";P;" %"
500  GOTO 600
510  PRINT "BERECHNUNG DES GRUNDWERTES NACH: "
520  PRINT
530  PRINT  TAB( 10)"GW = (PW/P)*100"
540  PRINT
550  INPUT "PROZENTWERT:  PW = ";PW
560  INPUT "PROZENTSATZ (IN %):  P = ";P
570  PRINT
580  LET GW = (PW / P) * 100
590  PRINT "GRUNDWERT:    GW = ";GW
600  PRINT : PRINT
610  INPUT "WEITER MIT DER (RETURN) - TASTE: ";RT$
620  RUN
```

Rechnungsschreibung

Eine typische Computer-Anwendung im Wirtschaftsleben besteht in der Erstellung von Rechnungen. Auch hierbei kommt das "E → V → A" - Prinzip deutlich zum Tragen. Die einfachste hier gegebene Version (Programm: RECHNUNGSSCHREIBUNG-1) ist zunächst nur auf zwei Rechnungspositionen ausgelegt. Sie dient im wesentlichen als ausbaufähiges Gerüst und insbesondere auch der Übung in der Ausgabe-Formatierung mit dem TAB-Befehl. Das an sich unverzichtbare stellengerechte Ausdrucken von Zahlenkolonnen ist in den meisten zur Zeit verfügbaren BASIC-Versionen ohne "Klimmzüge" nicht möglich.

(A.1.10) Ändern Sie das Programm RECHNUNGSSCHREIBUNG-1 so ab, daß die jeweiligen Zahlenkolonnen stellengerecht untereinander stehen, falls Ihr Rechner über entsprechende Formatierungsbefehle (z. B. PRINT USING) verfügt.

Programm: RECHNUNGSSCHREIBUNG-1

```
10  REM   *** RECHNUNG (2 ARTIKEL) ***
20  HOME
30  PRINT "RECHNUNGSSCHREIBUNG"
40  PRINT : PRINT
50  PRINT "1. ARTIKEL:  "
60  INPUT "   ARTIKELNUMMER:  ";A1
70  INPUT "   STUECKZAHL:  ";S1
80  INPUT "   EINZELPREIS:  ";P1
90  LET G1 = P1 * S1
100  PRINT
110  PRINT "2. ARTIKEL:  "
120  INPUT "   ARTIKELNUMMER:  ";A2
130  INPUT "   STUECKZAHL:  ";S2
140  INPUT "   EINZELPREIS:  ";P2
150  LET G2 = P2 * S2
160  LET N = G1 + G2
170  LET M = N * 0.14
180  LET B = N * 1.14
190  HOME
200  PRINT  TAB( 12)"R E C H N U N G"
210  PRINT
220  PRINT "ARTIKEL-" TAB( 11)"STUECK-" TAB( 20)"EINZEL-" TAB( 29)"GESAMT-"
230  PRINT  TAB( 3)"NR." TAB( 12)"ZAHL" TAB( 21)"PREIS" TAB( 30)"PREIS"
240  PRINT
250  PRINT A1 TAB( 13)S1 TAB( 21)P1 TAB( 30)G1
260  PRINT A2 TAB( 13)S2 TAB( 21)P2 TAB( 30)G2
270  PRINT
280  PRINT  TAB( 12)"NETTOBETRAG: " TAB( 30)N
290  PRINT  TAB( 12)"MEHRWERTSTEUER: " TAB( 30)M
300  PRINT
310  PRINT  TAB( 12)"BRUTTOBETRAG: " TAB( 30)B
320  END
```

Das obige Programm ist weniger wegen seiner Ausgereiftheit, sondern eher wegen seiner Defizite interessant; es kann als Diskussionsgrundlage für Verbesserungsmöglichkeiten dienen. Ein entsprechendes Gespräch mit Schülern führte zu der folgenden Vorschlagsliste:

(a) die Rechnungsartikel sollten durch ihren Namen und nicht nur durch eine Nummer beschreibbar sein;

(b) es sollten beliebig viele Rechnungspositionen eingebbar sei;

(c) der Einzelpreis sollte vom Rechner automatisch aus einer Waren-Preis-Tabelle entnommen werden;

(d) Kundenanschrift, Absender, Datum, Rechnungsnummer, Zahlungsbedingungen fehlen noch;

(e) die Rechnung sollte über einen Drucker ausgebbar sein.

Die Realisierbarkeit von Vorschlag (e) ist stark vom einzelnen Rechner abhängig. Beim APPLE II Computer genügt z.B. ein einziger Befehl: PR#n (n=Drucker-Slot-Nummer). Bei realen Anwendungen werden meist besondere Formulare verwendet, so daß dort noch spezielle Formatierungsfragen zu berücksichtigen sind.
Bei Vorschlag (d) sollten die Kundendaten von einer Diskette her einspeisbar sein.Da auch dies stark rechnerabhängig ist, wollen wir uns im folgenden nur mit den Verbesserungsvorschlägen (a) bis (c) befassen.

Bisher haben wir nur mit Dezimalzahl-Variablen gearbeitet. Zur Berücksichtigung von Vorschlag (a) benötigen wir eine neue Sorte von Variablen: nämlich Variable, in denen Texte bzw. beliebige Zeichenketten gespeichert werden können. Solche Variable heißen in BASIC STRING-Variable. Sie werden durch Anhängen eines $-Zeichens an den Variablen-Namen gekennzeichnet (z.B.: A$, C5$, MN$,...).

(A.1.11) Ändern Sie das Programm RECHNUNGSSCHREIBUNG-1 so ab, daß jeder Artikel durch seinen Namen und nicht durch eine anonyme Artikelnummer beschrieben wird.

Postleitzahlen

Vorschlag (c) führt geradewegs in den Problemkreis der Tabellenverarbeitung. Wir wollen ein mögliches Suchverfahren unabhängig vom Rechnungsschreibungskomplex anhand des Problems der Postleitzahlen behandeln. Die Tabelle der Städte-Namen und der zugehörigen Postleitzahlen wird im Programm POSTLEITZAHLEN-1 in den Feldvariablen A$(1),...,A$(12) bzw. B(1),...,B(12) abgespeichert (Zeile 40-270).

Programm: POSTLEITZAHLEN-1

```
10  REM     *** TABELLENSUCHEN ***
20  DIM A$(12)
30  DIM B(12)
40 A$(1) = "REUTLINGEN"
50 B(1) = 7410
60 A$(2) = "KARLSRUHE"
70 B(2) = 7500
80 A$(3) = "LINDAU"
90 B(3) = 8990
100 A$(4) = "TUEBINGEN"
110 B(4) = 7400
120 A$(5) = "ULM"
130 B(5) = 7900
140 A$(6) = "LANGENAU"
150 B(6) = 7907
160 A$(7) = "BALINGEN"
170 B(7) = 7460
180 A$(8) = "FREIBURG"
190 B(8) = 7800
200 A$(9) = "WUERZBURG"
210 B(9) = 8700
220 A$(10) = "SINDELFINGEN"
230 B(10) = 7031
240 A$(11) = "HAMBURG"
250 B(11) = 2000
260 A$(12) = "BERLIN"
270 B(12) = 1000
280  INPUT "NAME: ";N$
290  FOR I = 1 TO 12
300  IF A$(I) = N$ THEN  PRINT B(I)
310  NEXT I
320  END
```

Durch die "Dimensionierung" der A$-Tabelle und der B-Tabelle (in Zeile 20 und 30) wird der benötigte Speicherplatz reserviert. Die FOR-NEXT-Schleife (Zeile 290-310) ersetzt die für den Programmablauf gleichwertige, aber wesentlich umständlicher aufzuschreibende Befehlsfolge:

```
IF A$(1) = N$ THEN PRINT B(1)
IF A$(2) = N$ THEN PRINT B(2)
...
IF A$(12) = N$ THEN PRINT B(12).
```

(A.1.12) Warum ist das Programm POSTLEITZAHLEN-1 nicht durch Verwendung der STRING-Variablen A1$,A2$,...,A12$ bzw. der Zahl-Variablen B1,B2,...B12 lösbar?

(A.1.13) Erweitern Sie das Postleitzahlen-Programm auf mehr als 12 Postleitzahlen.

Das Programm POSTLEITZAHLEN-1 weist mehrere Schwächen auf:

(a) sollte eine "nicht gefunden"-Meldung kommen, wenn die gesuchte Postleitzahl nicht in der Liste vorkommt;

(b) kann der Suchvorgang (Zeile 290-310) noch erheblich optimiert werden.

Im Programm POSTLEITZAHLEN-2 ist eine mögliche Lösung des ersten Vorschlags gegeben. Der zweiten Anregung kann durch folgende Maßnahmen Rechnung getragen werden:

(b1) nicht weitersuchen, wenn Postleitzahl gefunden;

(b2) lexikographisches Anordnen der Städte-Namen und entsprechendes Abbrechen des Suchvorganges.

Programm: POSTLEITZAHLEN-2

```
10  REM     *** TABELLENSUCHEN ***
20  DIM A$(12)
30  DIM B(12)
40 A$(1) = "REUTLINGEN"
50 B(1) = 7410
60 A$(2) = "KARLSRUHE"
70 B(2) = 7500
80 A$(3) = "LINDAU"
90 B(3) = 8990
100 A$(4) = "TUEBINGEN"
110 B(4) = 7400
120 A$(5) = "ULM"
130 B(5) = 7900
140 A$(6) = "LANGENAU"
150 B(6) = 7907
160 A$(7) = "BALINGEN"
170 B(7) = 7460
```

```
(* Programm:  POSTLEITZAHLEN-2   ...   Fortsetzung *)

180 A$(8) = "FREIBURG"
190 B(8) = 7800
200 A$(9) = "WUERZBURG"
210 B(9) = 8700
220 A$(10) = "SINDELFINGEN"
230 B(10) = 7031
240 A$(11) = "HAMBURG"
250 B(11) = 2000
260 A$(12) = "BERLIN"
270 B(12) = 1000
280  INPUT "NAME: ";N$
290  LET D = 0
300  FOR I = 1 TO 12
310  IF A$(I) = N$ THEN  PRINT B(I): LET D = 1
320  NEXT I
330  IF D = 0 THEN  PRINT "NICHT VORHANDEN"
340  END
```

(A.1.14) Verändern Sie das Programm POSTLEITZAHLEN-2 so, daß die Vorschläge (b1) und (b2) Berücksichtigung finden.

Wenden wir uns nun wieder der Rechnungsschreibung zu. Das Programm RECHNUNGSSCHREIBUNG-2 trägt dem Vorschlag (b) ("beliebig" viele Positionen) Rechnung. Da das Ende der Artikel-Eingabe nicht von vornherein festliegt, muß es dem Rechner besonders signalisiert werden (hier durch Artikelname = ***). Im Verarbeitungsteil des Programmes wurde zur Ermittlung des Nettobetrages die Technik der saldierenden Summation angewandt (Zeile 280).

Programm: RECHNUNGSSCHREIBUNG-2

```
10  REM    *** RECHNUNGSSCHREIBUNG (MAXIMAL 50 POSITIONEN) ***
20  HOME
30  PRINT "RECHNUNGSSCHREIBUNG"
40  PRINT : PRINT
50  PRINT "    ENDE DER EINGABE WIRD DURCH:"
60  PRINT
70  PRINT "    ARTIKELEINGABE = ***  BEWIRKT."
```

```
(* Programm:  RECHNUNGSSCHREIBUNG-2   ...   Fortsetzung *)

80  PRINT : PRINT
90  DIM A$(50): DIM S(50): DIM P(50): DIM G(50)
100  LET I = I + 1
110  PRINT I;". ";
120  INPUT "ARTIKEL: ";A$(I)
130  IF A$(I) = "***" THEN  GOTO 190
140  INPUT "   STUECKZAHL:  ";S(I)
150  INPUT "   EINZELPREIS: ";P(I)
160  PRINT
170  LET G(I) = P(I) * S(I)
180  GOTO 100
190  PRINT : PRINT : PRINT : PRINT : PRINT
200  PRINT  TAB( 11)"R E C H N U N G"
210  PRINT
220  PRINT "ARTIKEL" TAB( 11)"STUECK-" TAB( 20)"EINZEL-" TAB( 29)"GESAMT-"
230  PRINT  TAB( 12)"ZAHL" TAB( 21)"PREIS" TAB( 30)"PREIS"
240  PRINT
250  LET N = 0
260  FOR J = 1 TO I - 1
270  PRINT A$(J) TAB( 13)S(J) TAB( 21)P(J) TAB( 30)G(J)
280  LET N = N + G(J)
290  NEXT J
300  PRINT
310  PRINT  TAB( 12)"NETTOBETRAG: " TAB( 30)N
320  LET M = 0.14* N
330  PRINT  TAB( 12)"MEHRWERTSTEUER: " TAB( 30)M
340  PRINT
350  LET B = N + M
360  PRINT  TAB( 12)"BRUTTOBETRAG: " TAB( 30)B
370  END
```

Im Programm RECHNUNGSSCHREIBUNG-3 sind schließlich noch die Waren-Preis-Tabelle und das entsprechende Suchprogramm (Zeile 170-220) aufgenommen worden. Die Zuweisung der Tabellenwerte erfolgt in einem "Unterprogramm" (Zeile 460-760). Der GOSUB-Befehl in Zeile 110 bewirkt die Verzweigung in das Unterprogramm; der RETURN-Befehl (Zeile 760) den Rücksprung ins Hauptprogramm.

Programm: RECHNUNGSSCHREIBUNG-3

```
10  REM    *** RECHNUNGSSCHREIBUNG (MAXIMAL 50 POSITIONEN) ***
20  HOME
30  PRINT "RECHNUNGSSCHREIBUNG"
40  PRINT : PRINT
50  PRINT "    ENDE DER EINGABE WIRD DURCH:"
60  PRINT
70  PRINT "    ARTIKELEINGABE = ***  BEWIRKT."
80  PRINT : PRINT
90  DIM A$(50): DIM S(50): DIM P(50): DIM G(50)
100  DIM AR$(15): DIM PR(15)
110  GOSUB 460
120  LET I = I + 1
130  PRINT I;". ";
140  INPUT "ARTIKEL: ";A$(I)
150  IF A$(I) = "***" THEN  GOTO 260
160  INPUT "   STUECKZAHL:  ";S(I)
170  LET D = 0
180  FOR J = 1 TO 15
190  IF A$(I) = AR$(J) THEN P(I) = PR(J): LET D = 1
200  NEXT J
210  IF D = 1 THEN  PRINT  TAB( 13)"EINZELPREIS: ";P(I): GOTO 230
220  INPUT "   EINZELPREIS: ";P(I)
230  LET G(I) = P(I) * S(I)
240  PRINT
250  GOTO 120
260  PRINT : PRINT : PRINT : PRINT : PRINT
270  PRINT  TAB( 11)"R E C H N U N G"
280  PRINT
290  PRINT "ARTIKEL" TAB( 11)"STUECK-" TAB( 20)"EINZEL-" TAB( 29)"GESAMT-"
300  PRINT  TAB( 12)"ZAHL" TAB( 21)"PREIS" TAB( 30)"PREIS"
310  PRINT
320  LET N = 0
330  FOR J = 1 TO I - 1
340  PRINT A$(J) TAB( 13)S(J) TAB( 21)P(J) TAB( 30)G(J)
350  LET N = N + G(J)
360  NEXT J
370  PRINT
380  PRINT  TAB( 8)"NETTOBETRAG: " TAB( 30)N
390  LET M = 0.14* N
400  PRINT  TAB( 8)"MEHRWERTSTEUER(14%): " TAB( 30)M
410  PRINT
420  LET B = N + M
430  PRINT  TAB( 8)"BRUTTOBETRAG: " TAB( 30)B
440  END
```

Programm: RECHNUNGSSCHREIBUNG-3 (Fortsetzung)

```
450  REM  UNTERPROGRAMM TABELLENZUWEISUNG
460 AR$(1) = "STUHL"
470 PR(1) = 87.50
480 AR$(2) = "TISCH"
490 PR(2) = 245.80
500 AR$(3) = "SESSEL"
510 PR(3) = 324.50
520 AR$(4) = "HOCKER"
530 PR(4) = 24.60
540 AR$(5) = "LAMPE"
550 PR(5) = 65.40
560 AR$(6) = "SOFA"
570 PR(6) = 568.20
580 AR$(7) = "KOMMODE"
590 PR(7) = 240.40
600 AR$(8) = "SPIEGEL"
610 PR(8) = 40.50
620 AR$(9) = "GARDEROBE"
630 PR(9) = 180.40
640 AR$(10) = "TEPPICH"
650 PR(10) = 280.30
660 AR$(11) = "BANK"
670 PR(11) = 148.50
680 AR$(12) = "REGAL"
690 PR(12) = 98.99
700 AR$(13) = "TRUHE"
710 PR(13) = 80.75
720 AR$(14) = "SCHREIBTISCH"
730 PR = 580.90
740 AR$ = "SCHRANK"
750 PR = 820.75
760  RETURN
```

(A.1.15) Bauen Sie das Programm RECHNUNGSSCHREIBUNG-3 unter Berücksichtigung der Vorschläge (d) und (e) zu einem kompletten Rechnungsschreibungsprogramm aus, falls Sie über geeignete Peripheriegeräte (Drucker und Disketten-Laufwerke) verfügen.

§2 PROBLEME AUS DEM WIRTSCHAFTSLEBEN

Kapitalverzinsung, Verdopplungszeit

Bei fast allen Problemen wirtschaftlicher Natur spielen Fragen der Kapitalverzinsung eine wichtige Rolle.
Erbringt ein Sparkonto einen Zinsertrag und läßt man diese Zinsen auf dem Konto stehen, so werden sie ihrerseits in der nächsten Zinsperiode (meist ein Jahr) weiter verzinst. Im Programm KAPITALVERZINSUNG wird der Prozeß dieser Zinseszins-Ermittlung durchgeführt.

Programm: KAPITALVERZINSUNG

```
10  HOME : PRINT "KAPITALVERZINSUNG"
20  PRINT
30  INPUT "ANFANGSKAPITAL:  K = ";K
40  INPUT "ZINSSATZ  (%) :   P = ";P
50  INPUT "LAUFZEIT (IN JAHREN): ";L
60  PRINT
70  FOR I = 1 TO L
80  LET K = K * (1 + P / 100)
90  PRINT I,K
100  NEXT I
110  END
```

Programm: KAPITALVERDOPPLUNG

```
10  HOME : PRINT "KAPITALVERDOPPLUNG"
20  PRINT
30  INPUT "ANFANGSKAPITAL:  KA = ";KA
40  INPUT "ZINSSATZ  (%) :   P = ";P
50  PRINT
60  LET K = KA
70  LET I = 0
80  LET K = K * (1 + P / 100)
90  LET I = I + 1
100  PRINT I,K
110  IF K < (2 * KA) THEN  GOTO 80
120  LET D = I
130  PRINT
140  PRINT "VERDOPPLUNGSZEIT:   D = ";D
150  END
```

Die besonders interessante Frage, nach welcher Laufzeit sich ein Anfangskapital durch Zinseszins-Wachstum verdoppelt, wird im Programm KAPITALVERDOPPLUNG behandelt.

(A.2.1) Begründen Sie, warum die Verdopplungszeit nur vom Zinssatz, nicht aber vom Anfangskapital abhängt.

(A.2.2) Führen Sie mehrere Programmläufe zur Ermittlung der Verdopplungszeiten für Zinssätze zwischen 0% und 10% durch. Versuchen Sie, eine Beziehung zwischen Zinssatz und Verdoppelungszeit herzustellen.

Im Programm TABELLE DER VERDOPPLUNGSZEITEN wird das Produkt P*D aus Zinssatz P und Verdopplungszeit D systematisch untersucht. Auf empirische Weise wird dadurch die folgende Gesetzmäßigkeit bestätigt:

p-mal-d-Regel:

Für Zinssätze zwischen 0% und 10% gilt näherungsweise:

$$p \cdot d = 70$$

(p = Zinssatz; d = Verdopplungszeit).

Programm: TABELLE DER VERDOPPLUNGSZEITEN

```
10  HOME : PRINT "TABELLE DER VERDOPPLUNGSZEITEN"
20  PRINT
30  INPUT "FUER ZINSSAETZE VON:  P1 = ";P1
40  INPUT "                 BIS:  P2 = ";P2
50  INPUT "SCHRITTWEITE:  S = ";S
60  PRINT
70  PRINT
80  PRINT "ZINSSATZ"; TAB( 12)"VERDOPP-"; TAB( 24)"PRODUKT"
90  PRINT  TAB( 12)"LUNGSZEIT"
100  PRINT "  P  "; TAB( 12)"  D  "; TAB( 24)" P*D"
110  PRINT
120  FOR P = P1 TO P2 STEP S
130  LET K = 1
140  LET I = 0
150  LET K = K * (1 + P / 100)
160  LET I = I + 1
170  IF K < 2 THEN  GOTO 150
180  LET D = I
190  PRINT P; TAB( 12)D; TAB( 24)P * D
200  NEXT P
210  END
```

Unterjährige Verzinsung

Unterjährige Zinsen (also Zinsen, die innerhalb des Zeitraumes von einem Kalenderjahr erwirtschaftet werden) ermittelt man häufig nach der folgenden Vereinbarung:

Sparkassenkonvention:

Der unterjährige Zinsbetrag wird proportional zur verstrichenen Zeit berechnet.

(A.2.3) (a) Berechnen Sie den Zinsertrag eines Kapitals von 1.000,-- DM, wenn es zweimal halbjährlich nach der Sparkassenkonvention verzinst wird und die Zinsen des ersten Halbjahres im zweiten Halbjahr ebenfalls verzinst werden.
(b) Verfeinern Sie die Jahreseinteilung aus (a).
(c) Reflektieren Sie die Berechtigung der Sparkassenkonvention.

Im Programm UNTERJAEHRIGE VERZINSUNG-1 können Sie den theoretischen Zinsverlauf innerhalb eines Jahres bei beliebiger Unterteilung des Jahres in K Abschnitte verfolgen. Das Programm UNTERJAEHRIGE VERZINSUNG-2 zeigt nur noch die Jahresergebnisse für K = 1 bis K = 1000, und im Programm UNTERJAEHRIGE VERZINSUNG-3 können schließlich Untergrenze, Obergrenze und Schrittweite beliebig festgelegt werden.
(Der ursprüngliche Jahreszinssatz betrage jeweils 100 %; d. h. Verdoppelung pro Jahr.)

Programm: UNTERJAEHRIGE VERZINSUNG-1

```
10  HOME : PRINT "UNTERJAEHRIGE VERZINSUNG"
20  PRINT
30  INPUT "UNTERTEILUNG:  K = ";K
40  LET A = 1
50  FOR I = 1 TO K
60  LET A = A * (1 + 1 / K)
70  PRINT I,A
80  NEXT I
90  END
```

Programm: UNTERJAEHRIGE VERZINSUNG-2

```
10  HOME : PRINT "UNTERJAEHRIGE VERZINSUNG (TABELLE)"
20  FOR K = 1 TO 1000
30  LET A = 1
40  FOR I = 1 TO K
50  LET A = A * (1 + 1 / K)
60  NEXT I
70  PRINT K,A
80  NEXT K
90  END
```

Programm: UNTERJAEHRIGE VERZINSUNG-3

```
10  HOME : PRINT "UNTERJAEHRIGE VERZINSUNG (TABELLE)"
20  PRINT : PRINT
30  PRINT "BEREICH DER LAUFVARIABLEN K:"
40  PRINT
50  INPUT "START BEI:   KA = ";KA
60  IF KA = 0 THEN  GOTO 50
70  INPUT "STOP BEI:    KE = ";KE
80  INPUT "SCHRITTWEITE:  KS = ";KS
90  PRINT
100  FOR K = KA TO KE STEP KS
110  LET A = 1
120  FOR I = 1 TO K
130  LET A = A * (1 + 1 / K)
140  NEXT I
150  PRINT K,A
160  NEXT K
170  END
```

Ratensparen, Renten, Barwert einer Rente

Beim RATENSPAREN kommt zusätzlich zur Verzinsung durch Zinseszinsen noch eine periodische Einzahlung in gleichbleibender Höhe hinzu.

Eine Folge von konstanten Zahlungen auf ein durch Zinseszinsen verzinstes Konto wird in wirtschaftswissenschaftlicher Terminologie auch als eine Rente bezeichnet. Im Programm RENTE kann man zwischen vorschüssiger (d. h. jeweils zu Beginn einer Verzinsungsperiode erfolgender) und nach-

schüssiger (d. h. jeweils am Ende einer Verzinsungsperiode erfolgender) Rentenzahlung wählen. In Zeile 100 dieses Programmes wurde von der Möglichkeit mehrerer Anweisungen pro Zeile Gebrauch gemacht, um dadurch ein gehäuftes Springen im Programm zu vermeiden.

Programm: RATENSPAREN

```
10  HOME : PRINT "RATENSPAREN"
20  PRINT
30  INPUT "ZINSSATZ:  P = ";P
40  INPUT "ANFANGSKAPITAL:   KO = ";KO
50  INPUT "EINZAHLUNGSBETRAG:  E = ";E
60  INPUT "LAUFZEIT:  L = ";L
70  LET K = KO
80  LET I = 0
90  LET K = (1 + P / 100) * K + E
100  LET I = I + 1
110  PRINT I,K
120  IF I < L THEN  GOTO 90
130  END
```

Programm: RENTE

```
10  HOME : PRINT "RENTENRECHNUNG"
20  PRINT
30  INPUT "RENTENBETRAG:  R = ";R
40  INPUT "ZINSSATZ (IN %):  P = ";P
50  LET Q = (1 + P / 100)
60  INPUT "LAUFZEIT:  L = ";L
70  PRINT
80  INPUT "VORSCHUESSIG/NACHSCHUESSIG? (V/N): ";VN$
90  PRINT
100  IF VN$ = "V" THEN  LET K = K + R: LET K = K * Q: GOTO 120
110  LET K = K * Q + R
120  LET I = I + 1
130  PRINT I,K
140  IF I < L THEN  GOTO 100
150  LET KE = K
160  PRINT
170  PRINT "RENTEN-ENDWERT: ";KE
180  END
```

Im Zusammenhang mit der Bewertung von Renten tritt gelegentlich die Frage nach dem Wert einer vorzeitig auszuzahlenden Rente auf:

Barwert-Problem: Herr Müller hat sich durch laufende Zahlungen an eine Lebensversicherung in der Vergangenheit das Anrecht auf eine Rente zur Aufbesserung seines Einkommens im Alter erworben. Jetzt steht er kurz vor seinem 60. Geburtstag, dem Termin, zu dem die Rentenzahlungen von 5.000,-- DM pro Jahr beginnen sollen. Die Laufzeit der Rente ist auf 15 Jahre festgelegt.
Aufgrund unvorhersehbarer Entwicklungen wollen die Kinder von Herrn Müller ein Geschäft eröffnen und benötigen dazu Kapital. Herr Müller überlegt: Vielleicht könnte er sich an Stelle der (über 15 Jahre) laufenden Rentenzahlungen die gesamte Rente "auf einen Schlag" auszahlen lassen, um seine Kinder mit diesem Geld zu unterstützen. Was aber ist der "gerechte" Gegenwert für den Verzicht auf die Rente?

Barwert einer Rente: Durch eine Rente mit jährlicher Zahlung von R DM werde nach n Jahren durch ständige Verzinsung zum Zinssatz von p% ein Kapital K(n) erwirtschaftet. Steigt ein einmalig eingezahlter Betrag B durch Zinseszinsen (Zinssatz: ebenfalls p%) nach n Jahren auch auf die Höhe K(n) an, so heißt B der Barwert der Rente.

Im Diagramm:

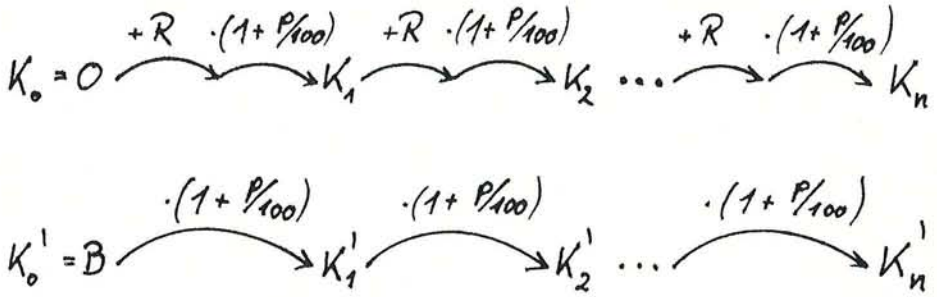


Abb.2.1

(Ist K'(n) = K(n), so ist B (=K'(0)) der Barwert der Rente.)

Im Beispiel: R = 5.000,-- DM (vorschüssig);
Zinssatz: p% = 5%;
Laufzeit: 15 Jahre;
ergibt näherungsweise:
Rentenendwert: K(15) = 113.287,-- DM;
Barwert: 54.493,-- DM.

Die Verzinsung des Barwerts einer Rente durch Zinseszinsen ist eine Beispiel für exponentielles Wachstum, von dem D. Meadows in dem Buch "Die Grenzen des Wachstums" schreibt:

> Obwohl exponentielles Wachstum so alltäglich ist, bereitet es immer wieder Überraschungen, schon seit Jahrtausenden.

Es ist deshalb lehrreich, den Barwert einer Rente zunächst einmal schätzen zu lassen, wie dies im Programm BARWERT-SCHAETZUNG geschieht. Dagegen wird er im Programm RENTEN-BARWERT auf direkte Weise (durch fortgesetztes Dividieren)

Programm: RENTEN-BARWERT

```
10  HOME : PRINT "BARWERT EINER RENTE"
20  PRINT
30  INPUT "RENTENBETRAG:  R = ";R
40  INPUT "ZINSSATZ (IN %):  P = ";P
50  LET Q = (1 + P / 100)
60  INPUT "LAUFZEIT:  L = ";L
70  PRINT
80  INPUT "VORSCHUESSIG/NACHSCHUESSIG? (V/N): ";VN$
90  PRINT
100  IF VN$ = "V" THEN  LET K = K + R: LET K = K * Q: GOTO 120
110  LET K = K * Q + R
120  LET I = I + 1
130  PRINT I,K
140  IF I < L THEN  GOTO 100
150  PRINT
160  PRINT "RENTEN-ENDWERT: ";K
170  PRINT
180  FOR I = 1 TO L
190  LET K = K / Q
200  NEXT I
210  PRINT "BARWERT DER RENTE: ";K
220  END
```

ermittelt. (Das fortgesetzte Dividieren hat den Vorteil, daß man sowohl im Hinblick auf die Lernvoraussetzungen der Schüler als auch auf die Ausstattung des Rechners auf Potenzfunktionen bzw. Exponentialfunktionen verzichten kann. In unterrichtlicher Behandlung ergibt sich die Begründung zudem in sehr einfacher Weise durch umgekehrtes Durchlaufen der unteren Pfeil-Kette im obigen Diagramm.)

Programm: BARWERTSCHAETZUNG

```
10  HOME : PRINT "BARWERT EINER RENTE (SCHAETZUNG)"
20  PRINT
30  INPUT "RENTENBETRAG:  R = ";R
40  INPUT "ZINSSATZ (IN %):  P = ";P
50  LET Q = (1 + P / 100)
60  INPUT "LAUFZEIT:  L = ";L
70  PRINT
80  INPUT "VORSCHUESSIG/NACHSCHUESSIG? (V/N): ";VN$
90  PRINT
100  IF VN$ = "V" THEN  LET K = K + R: LET K = K * Q: GOTO 120
110  LET K = K * Q + R
120  LET I = I + 1
130  PRINT I,K
140  IF I < L THEN  GOTO 100
150  LET KE = K
160  PRINT
170  PRINT "RENTEN-ENDWERT: ";KE
180  PRINT
190  INPUT "SCHAETZUNG DES BARWERTES:  B = ";B
200  LET I = 0
210  LET K = B
220  LET K = Q * K
230  LET I = I + 1
240  IF I < L THEN  GOTO 220
250  PRINT "DIE SCHAETZUNG ERGIBT DEB ENDWERT:"
260  PRINT K
270  PRINT
280  INPUT "NOCH EINMAL SCHAETZEN?  (J/N):";JN$
290  IF  NOT (JN$ = "N") THEN  GOTO 180
300  END
```

Tilgung von Darlehen

In bezug auf seine algorithmische Struktur ist das Programm TILGUNG fast identisch mit dem Programm RATENSPAREN. Abgesehen davon, daß die Eingabegrößen anders zu interpretieren sind, besteht nur ein Unterschied in der Abbruchbedingung: eine Tilgung ist dann beendet, wenn die Restschuld gleich Null ist.

Programm: TILGUNG

```
10  HOME : PRINT "TILGUNG"
20  PRINT
30  INPUT "ZINSSATZ:  P = ";P
40  INPUT "ANFANGSKREDIT:    KO = ";KO
50  INPUT "EINZAHLUNGSBETRAG:  E = ";E
60  LET K = KO
70  LET I = 0
80  LET K = (1 + P / 100) * K - E
90  LET I = I + 1
100  PRINT I,K
110  IF K > 0 THEN  GOTO 80
120  END
```

Fast allen Programmen dieses Paragraphen liegt als zentrale Rekursionsbedingung eine Zuweisung vom Typ

```
LET Y = A*Y + B
```

zugrunde. In mathematischer Notation entspricht diese Zuweisung der Rekursionsgleichung:

$$y_{k+1} = a \cdot y_k + b$$

Ihr formaler Typ ist: lineare Differenzengleichung erster Ordnung mit konstanten Koeffizienten und konstanter Inhomogenität. Wir wollen sie jedoch ihrer prägnanten Anwendung wegen kurz als Tilgungsgleichung bezeichnen.

Man kann sie sich durch "Verschmelzung" der Rekursionsgleichungen zur arithmetischen und geometrischen Folge entstanden denken:

$$y_{k+1} = y_k + b$$

$$y_{k+1} = a \cdot y_k$$

(A.2.4) Stellen Sie die Größe y_k aus der Tilgungsgleichung in Abhängigkeit vom Anfangswert y_0 in expliziter Form dar.

(A.2.5) Jeder Tilgungsprozeß hängt von den Parametern: Anfangsdarlehen, Annuität, Zinssatz und Laufzeit ab. Durch Vorgabe von je drei dieser Größen ist (im Falle der Lösbarkeit) auch die vierte Größe festgelegt. Analysieren Sie die Möglichkeiten zur Ermittlung dieser vierten Größe einerseits mit Hilfe eines Tilgungsprogramms und andererseits durch analytisch/geschlossene Methoden (d. h.: Darstellung der gesuchten Größe durch eine explizite Formel).

Trotz ihres elementaren Charakters besitzt die Tilgungsgleichung viele interessante Anwendungen:

(1.) Ratensparen

(2.) Rente

(3.) Tilgung

(4.) Finanzierung von Stiftungen aus Zinserträgen

(5.) Erntevorgänge: exponentielles Wachstum, das durch periodische Abschöpfungen in konstanter Höhe eingeschränkt wird (z. B. Karpfenzucht mit Abfischvorgang, Holzeinschlag in der Forstwirtschaft)

(6.) Produktion radioaktiver Substanzen (z. B. für medizinische Zwecke); Rekursionsgleichung:

$$N_{k+1} = (1-\lambda)\cdot N_k + P$$

N: Anzahl der radioaktiven Kerne einer bestimmten Substanz

λ: Zerfallswahrscheinlichkeit pro Zeitintervall

P: (konstante) pro Zeitintervall produzierte Menge

(7.) Bei Spielen: z. B. Anzahl der Züge beim "Turm-von-Hanoi-Spiel" bei der rekursiven Standardstrategie:

$$U_{k+1} = 2 \cdot U_k + 1$$

(8.) Dynamische Wirtschaftsmodelle: z. B. Angebot-Nachfrage-Zyklen ("Schweinezyklus"); Literatur: Dürr/Ziegenbalg oder Goldberg.

Ratenkauf, Zinsvergleiche

Ein weiterer wichtiger Problemkreis im Wirtschaftsleben sind die Zinsvergleiche. Durch Ratenzahlung soll der Kauf einer Ware auch solchen Interessenten schmackhaft gemacht werden, die knapp bei Kasse sind. Im Programm RATENKAUF wird der Zinsbetrag eines Geschäftes auf Ratenzahlungsbasis ermittelt und in Jahreszinsen umgerechnet.

Programm: RATENKAUF

```
10  HOME : PRINT "EFFEKTIV-ZINSEN BEI RATENKAEUFEN"
20  PRINT
30  INPUT "WARENWERT: ";W
40  INPUT "MONATLICHE RATE: ";R
50  INPUT "LAUFZEIT (IN MONATEN): ";L
60  PRINT
70  LET G = L * R
80  LET Z = G - W
90  LET P = (Z / W) * (12 / L) * 100
100  PRINT "GESAMTZAHLUNG: ";G
110  PRINT "ZINSBETRAG: ";Z
120  PRINT "ENTSPRECHENDER JAHRES-ZINSSATZ: ";P;" %"
130  END
```

Das Programm ZINSVERGLEICH ermöglicht den Vergleich der folgenden beiden Tilgungsformen:

- die Zinsen werden als Jahreszinsen angegeben und nur aus der Restschuld berechnet;
- die Zinsen werden als Monatszinsen angegeben und stets aus dem Anfangsbetrag des Kredits berechnet.

Programm: ZINSVERGLEICH

```
10  HOME : PRINT "ZINSVERGLEICH"
20  PRINT
30  INPUT "KREDITBETRAG: ";KB
40  INPUT "LAUFZEIT: ";LZ
50  PRINT
60  PRINT "1. ANGEBOT: SPARKASSE"
70  PRINT
80  PRINT "   DIE ZINSEN WERDEN AUS DER RESTSCHULD"
90  PRINT "   BERECHNET UND JEWEILS MONATLICH "
100  PRINT "   ZUZUEGLICH ZUM TILGUNGSBETRAG BEZAHLT"
110  PRINT
120  LET TS = KB / LZ
130  PRINT "   MONATL. TILGUNGSBETRAG: ";TS
140  INPUT "   JAHRES-ZINSSATZ: ";PS
150  LET K = KB
160  FOR L = 1 TO LZ
170  LET Z = K * (PS / 1200)
180  LET ZS = ZS + Z
190  LET K = K - TS
200  NEXT L
210  LET ZA = ZS / KB
220  PRINT
230  PRINT "GESAMTZINSEN: ";ZS
240  PRINT "ZINSANTEIL: ";ZA * 100;" %"
250  PRINT : PRINT
260  PRINT "2. KREDIT-AGENTUR"
270  PRINT
280  PRINT "   DIE (MONATS-) ZINSEN WERDEN VOM"
290  PRINT "   URSPRUENGLICHEN KREDITBETRAG "
300  PRINT "   BERECHNET."
310  PRINT
320  INPUT "   MONATS-ZINSSATZ (IN %): ";PM
330  LET ZK = KB * (PM / 100) * LZ
340  LET ZA = ZK / KB
350  PRINT
360  PRINT "GESAMTZINSEN: ";ZK
370  PRINT "ZINSANTEIL: ";ZA * 100;" %"
380  END
```

§3 STOCHASTISCHE SIMULATIONEN

Die Realität wird von uns Menschen mit Hilfe von Bildern und Modellen erfaßt, die wir uns von ihr machen. Bei der Komplexität realer Vorgänge ist es unvermeidlich, daß ein Beschreibungsmodell nicht in jedem Detail mit der Realität übereinstimmt. Modelle sind also praktisch nie perfekt; es kommt eher darauf an, ob sie als tauglich akzeptiert werden können oder nicht.

Die Beschreibungsmittel für die Modellbildungen in §2 und insbesondere die rekursiven Beschreibungen durch Differenzengleichungen waren deterministischer Natur. Das heißt, daß sich der Zustand des jeweiligen Systems in einer bestimmten Periode eindeutig aus dem Zustand des Systems in den Vorperioden ergibt. Demgegenüber sind viele Prozesse in der Realität für uns nicht (bzw. zumindest nicht erkennbar) determiniert; der "Zufall" spielt bei ihnen eine Rolle. Man nennt solche Prozesse dann nicht-determiniert oder stochastisch.

Computer verkörpern in vieler Hinsicht ein Höchstmaß an Determiniertheit. Der Versuch, den Zufall mit Hilfe von Computern zu simulieren, kann deshalb zunächst als paradox erscheinen. Es kommt aber im folgenden nicht darauf an, wirklich Zufälliges, wie z. B. zufällige Zahlenfolgen zu erzeugen, sondern nur solche, die mit echten Zufallsfolgen (falls es so etwas überhaupt gibt) wesentliche Merkmale gemeinsam besitzen.

Die meisten BASIC-Dialekte besitzen den Befehl RND (für RANDOM). In der Form RND(1) erzeugt er eine Zufallszahl zwischen 0 und 1; die Zahl 1 selbst wird nicht erzeugt. Alle erzeugten Zufallszahlen sollen gleich wahrscheinlich sein. Wenn dieser Generator für Zufallszahlen als brauchbar akzeptiert werden soll, dann müssen zumindest die folgenden Anforderungen erfüllt sein: die Zufallszahlen müssen im Intervall (0,1) gleich verteilt sein, und sie müssen unabhängig voneinander sein. Es gibt eine Reihe von Tests, mit denen die Güte von Zufallszahlen-Generatoren überprüft werden kann. Wir wollen diesen fortgeschrittenen Themenkreis hier nicht explizit behandeln und gehen im folgenden im Sinne einer Arbeitshypothese davon aus, daß durch den Befehl RND(1) gleichverteilte und unabhängige Zufallszahlen zwischen 0 und 1 erzeugt werden.

Der Würfel

Ein aus dem Spieltrieb des Menschen heraus häufig in Gang gesetzter Zufallsprozeß ist das Würfeln. Beim gewöhnlichen Würfel sollte jede der sechs Zahlen 1, 2, 3, 4, 5 und 6 unabhängig von den anderen auftreten, und alle sollten bei einer großen Anzahl von Versuchen etwa gleich oft vorkommen. Im Programm WUERFEL-1 wird durch den Befehl

```
LET W = RND(1)
```

eine Zufallszahl zwischen 0 und 1 erzeugt. Durch die Anweisung

```
LET W = W*6
```

wird dieses Intervall auf das Intervall (0,6) "gestreckt". Durch den Befehl

```
LET W = INT(W)
```

wird die größte ganze Zahl, die kleiner oder gleich W ist, ermittelt. So ist z. B. INT(3.14)=3. Da durch RND(1) nur Zahlen echt unterhalb von 1 erzeugt werden, bestehen die möglichen Ergbnisse des Befehls LET W = INT(W) in Zeile 30 aus den Zahlen 0, 1, 2, 3, 4 und 5. Durch den Befehl

```
LET W = W+1
```

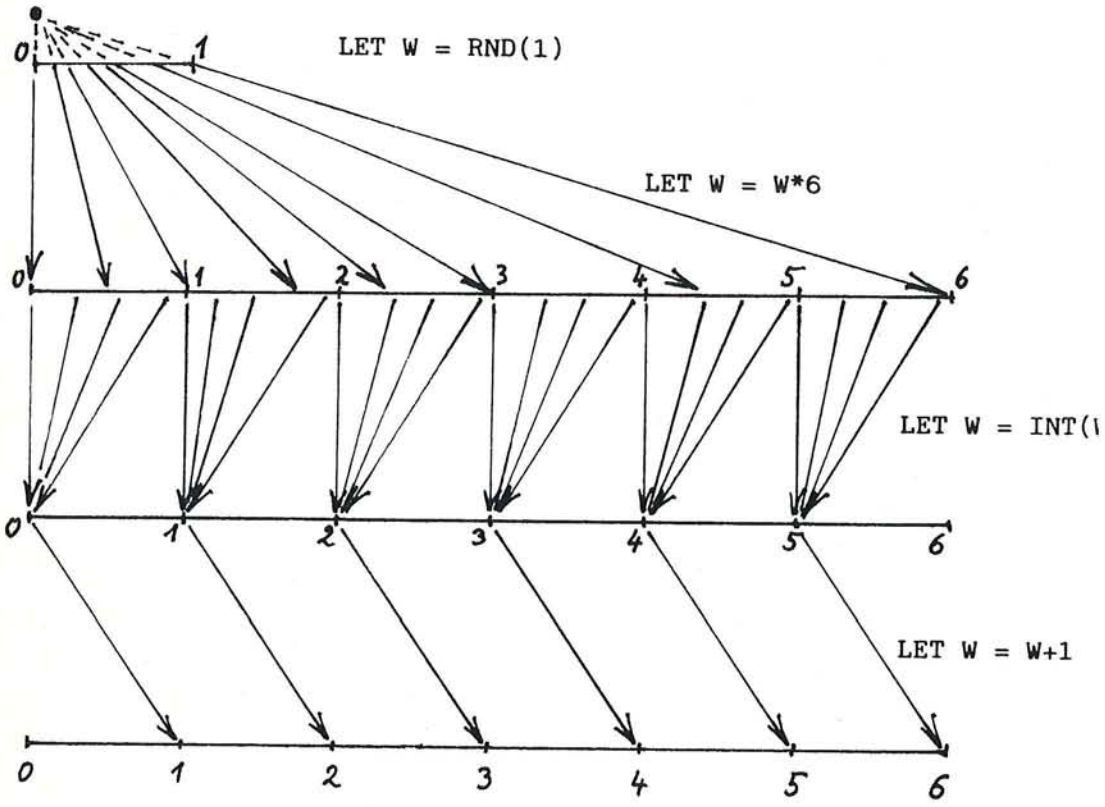
wird der Wert von W um 1 erhöht, und wir erhalten schließlich eine Würfelzahl zwischen 1 und 6. Der hier geschilderte Transformationsprozeß ist in Abb.3.1 graphisch dargestellt. Die gesamte Befehlskette von Zeile 20 bis Zeile 50 im Programm WUERFEL-1 kann mit Hilfe der Technik der Funktionsverkettung in einem einzigen Befehl formuliert werden:

```
LET W = INT(RND(1)*6)+1.
```

Beim Lauf von WUERFEL-1 drängt sich die Frage auf, wie viele Einsen, Zweien, ... ,Sechsen insgesamt gewürfelt wurden. Diese Summenhäufigkeiten werden im Programm WUERFEL-2 ermittelt. In Zeile 30 wird nach dem Erwürfeln der Zahl W der Inhalt des "Schubfaches" S(W) um 1 erhöht. Die Tabelle S(1), S(2), ... ,S(6) braucht nicht explizit dimensioniert zu werden, da sie nicht mehr als 10 Feldelemente besitzt.

Simulation des Würfelns

mit Hilfe des RND-Zufallszahlen-Generators



Komposition: **LET W = INT(RND(1)*6)+1**

Abb.3.1

Programm: WUERFEL-1

```
10  HOME : PRINT "WUERFEL-SIMULATION"
20  LET W =  RND (1)
30  LET W = W * 6
40  LET W =  INT (W)
50  LET W = W + 1
60  PRINT W;" ";
70  GOTO 20
```

Programm: WUERFEL-2

```
10  HOME : PRINT "WUERFEL-SIMULATION"
20  LET W =  INT ( RND (1) * 6) + 1
30  LET S(W) = S(W) + 1
40  FOR I = 1 TO 6
50  PRINT S(I);"   ";
60  NEXT I
70  PRINT
80  GOTO 20
```

Die Programmläufe von WUERFEL-2 werden erfahrungsgemäß praktisch von allen Beobachtern als ein erstes grobes Indiz zur Beurteilung des Gleichverteilungs-Kriteriums benutzt.

(A.3.1) Ändern Sie das Programm WUERFEL-2 so ab, daß ein "Glücksrad" mit einer vorgebbaren Anzahl von gleich großen Kreissektoren simuliert wird.

Das Glücksrad mit zwei Feldern

Viele stochastischen Simulationen beruhen auf dem wiederholten Drehen eines Glücksrades mit zwei Feldern A und B.

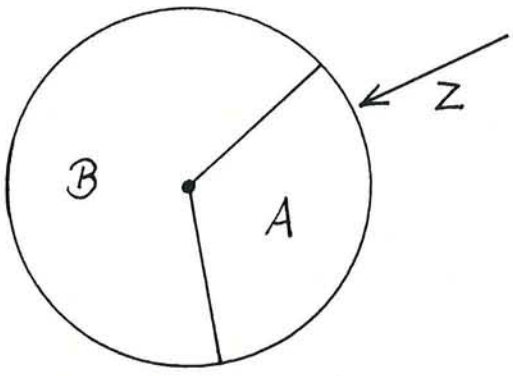


Abb.3.2

Zeigt der fest montierte Zeiger Z nach dem Drehen auf das Feld A, so hat der Spieler A gewonnen, andernfalls verloren. Im Programm GLUECKSRAD-1 wird dieser Zufallsprozeß beliebig oft wiederholt. Die Gewinnwahrscheinlichkeit PA, also der Anteil des zu Spieler A gehörenden Kreissektors, ist dabei beliebig vorgebbar.

Programm: GLUECKSRAD-1

```
10  HOME : PRINT "GLUECKSRAD MIT ZWEI FELDERN"
20  PRINT
30  INPUT "WAHRSCHEINLICHKEIT FUER FELD A: ";PA
40  LET R =  RND (1)
50  IF (R < = PA) THEN  PRINT "GEWONNEN"
60  IF (R > PA) THEN  PRINT  TAB( 10)"VERLOREN"
70  GOTO 40
```

Im Programm GLUECKSRAD-2 ist eine feste Anzahl von Läufen vorgesehen, nach welcher die relative Häufigkeit der Gewinne von A berechnet wird. Zum Vergleich wird die Gewinnwahrscheinlichkeit PA des Spielers A ebenfalls ausgegeben.

Programm: GLUECKSRAD-2

```
10  HOME : PRINT "GLUECKSRAD MIT ZWEI FELDERN"
20  PRINT
30  INPUT "WAHRSCHEINLICHKEIT FUER FELD A: ";PA
35  INPUT "ANZAHL DER LAUEFE:  L = ";L
36  PRINT
38  FOR I = 1 TO L
40  LET R =  RND (1)
50  IF (R < = PA) THEN  PRINT "GEWONNEN": LET GA = GA + 1
60  IF (R > PA) THEN  PRINT "   VERLOREN": LET GB = GB + 1
70  NEXT I
75  PRINT
80  PRINT "GEWINNE VON A: ";GA
90  PRINT "GEWINNE VON B: ";GB
100  PRINT
110  PRINT "RELATIVE HAEUFIGKEIT DER GEWINNE VON A:"
120  PRINT  TAB( 10)GA / L
130  PRINT "PA = ";PA
140  END
```

(A.3.2) Ändern Sie das Programm GLUECKSRAD-2 so ab, daß die relativen Häufigkeiten der Gewinne von A fortlaufend berechnet und ausgegeben werden.

(A.3.3) Schreiben Sie ein Programm GLUECKSRAD-3, durch das ein Glücksrad mit beliebig vielen Feldern und beliebig vorgebbaren Gewinnwahrscheinlichkeiten simuliert wird.

Das empirische Gesetz der großen Zahlen

Wiederholte Programmläufe mit GLUECKSRAD-2 vermitteln den Eindruck, daß sich die relative Häufigkeit GA/L der Siege von A bei einer großen Anzahl von Wiederholungen der Wahrscheinlichkeit PA annähert.

Hierin kommt das folgende allgemeinere empirische Gesetz der großen Zahlen zum Ausdruck:

> Bei einer Folge voneinander unabhängiger Wiederholungen eines Zufallsexperiments stabilisieren sich im Laufe der Zeit die relativen Häufigkeiten für einen bestimmten Ausgang des Experiments. Ist die Wahrscheinlichkeit des betrachteten Ausgangs nicht von vornherein bekannt, so dient der Wert, dem sich die relativen Häufigkeiten annähern, häufig als Schätzwert für die unbekannte Wahrscheinlichkeit.

Besonders anschaulich wird dieser Sachverhalt, den man natürlich nicht mit rein innermathematischen Methoden beweisen

Programm: GESETZ DER GROSSEN ZAHLEN

```
10  REM     *** EMPIRISCHES GESETZ DER GROSSEN ZAHLEN ***
20  HOME
30  HGR
40  HPLOT 0,0 TO 0,159
50  HPLOT 0,159 TO 279,159
60  VTAB 22
70  INPUT "TESTWAHRSCHEINLICHKEIT: ";PT
80  HPLOT 0,159 TO 0,159
90  LET S = 0
100  FOR I = 0 TO 279
110  LET R =  RND (1)
120  IF R < PT THEN S = S + 1
130  LET H = S / (I + 1)
140  LET Y =  - 159 * H + 159
150  HPLOT  TO I,Y
160  NEXT I
170  INPUT "NOCH EINMAL? (J/N): ";JN$
180  IF JN$ = "J" THEN  GOTO 60
190  TEXT
200  END
```

kann, da er sich auf Beobachtungen stützt, durch die graphische Darstellung, wie sie z. B. im Programm GESETZ DER GROSSEN ZAHLEN gegeben wird. In Abhängigkeit von der Anzahl der Versuche (dargestellt auf der x-Achse) wird die relative Häufigkeit auf der y-Achse wiedergegeben. Dieses Programm arbeitet mit der hochauflösenden Graphik des APPLE-II Computers. Erläuterungen zum graphischen Arbeitsmodus und zu den notwendigen Umrechnungen (Transformationen) der Koordinatenwerte ist bei der Beschreibung des Programmes FUNKTIONSSCHAUBILD in §4 zu finden.

Irrfahrten

Eine weitere graphische Simulation ist im Programm IRRFAHRT-APPLE gegeben. Es kann z. B. als Simulation der Brownschen Bewegung interpretiert werden. Ein Teilchen habe die Koordinaten (X,Y). Durch den Zufallsprozeß "Würfeln mit einem Viererwürfel" wird festgelegt, ob es sich um eine Einheit nach rechts, unten, links oder oben bewegt.

Im Programm IRRFAHRT-CBM ist ein entsprechendes Programm für den Text-Bildschirm eines CBM-Computers (8-K-Version) gegeben. Der Bildschirmspeicher liegt zwischen den Adressen 32768 und 33767. Durch den Befehl POKE I,J wird das Zeichen mit dem ASCII-Wert J in der Speicherzelle Nr.I abgelegt. Der Befehl ASC("c") gibt den ASCII-WERT des Zeichens c wieder.

Programm: IRRFAHRT-APPLE

```
10  HOME : VTAB 21: PRINT "IRRFAHRT";
20  LET X = 140
30  LET Y = 80
40  HGR
50  HPLOT X,Y
60  LET R =  INT ( RND (1) * 4)
70  IF R = 0 THEN X = X + 1
80  IF R = 1 THEN Y = Y + 1
90  IF R = 2 THEN X = X - 1
100  IF R = 3 THEN Y = Y - 1
110  IF (X < 0) OR (X > 279) OR (Y < 0) OR (Y > 159) THEN  END
120  HPLOT X,Y
130  GOTO 60
```

Programm: IRRFAHRT-CBM

```
10  REM     ***   IRRFAHRT   ***
20  LET A = 32768
30  LET E = 33767
40  REM (BEI CBM 40-ZEILEN-BILDSCHIRM)
50  LET M = (A + E) / 2
60  LET P = M
70  LET R =  INT ( RND (1) * 4)
80  IF R = 0 THEN P = P + 1
90  IF R = 1 THEN P = P + 40
100  IF R = 2 THEN P = P - 1
110  IF R = 3 THEN P = P - 40
120  IF (P < A) OR (P > E) THEN  END
130  POKE P, ASC ("*")
140  GOTO 70
```

(A.3.4) Schreiben Sie ein Irrfahrt-Programm, bei dem auch die Länge der zurückgelegten Wegstrecke durch einen Zufallsprozeß bestimmt wird.

Verlängerung eines Tischtennisspiels

Wenn es gegen Ende eines Tischtennisspiels 20:20 steht, so siegt derjenige Spieler, der als erster einen Vorsprung von zwei Punkten erspielen kann. Dieses Spielende wird im Programm TISCHTENNIS-1 simuliert. Die Gewinnwahrscheinlichkeit pro Ballwechsel von Spieler A ist frei vorgebbar (in der Variablen PA). Damit ist die Gewinnwahrscheinlichkeit von B als 1-PA festgelegt. Fällt der Wert der Zufallszahl R in das Intervall (0,PA), so wird dies wie im Programm GLUECKSRAD als Punktgewinn für A, andernfalls als Punktgewinn für B interpretiert.

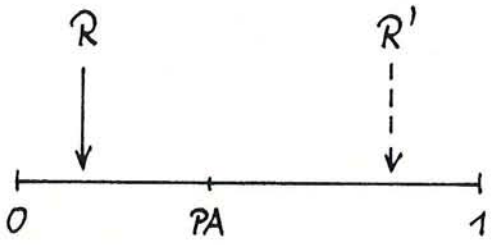


Abb.3.3

Die Funktion ABS ermittelt den Absolutbetrag einer Zahl. Das Tischtennisspiel ist zu Ende, wenn ABS(A-B)=2 ist.

Die Ergebnisse einzelner Zufallsexperimente sind nicht sehr aussagekräftig. Erst wenn dasselbe Experiment sehr oft unter gleichwertigen Bedingungen wiederholt wird, kann man statistische Schlüsse daraus ziehen. Im Programm TISCHTENNIS-2 wird das Endspiel deshalb in einer Wiederholungsschleife L-mal wiederholt. Vor jedem Spielende müssen jetzt die betreffenden Zählvariablen auf Null gesetzt werden (Zeile 110-130). Im Programm TISCHTENNIS-1 geschah dies automatisch beim Aufruf des RUN-Kommandos. Nach dem Durchlaufen aller L Simulationen werden die durchschnittliche Spieldauer und die Anzahl der Siege von A und B ausgegeben.

Programm: TISCHTENNIS-1

```
10  HOME : PRINT "SIMULATION DES ENDSPIELS"
20  PRINT "BEIM TISCHTENNIS"
30  PRINT
40  PRINT "GEWINNWAHRSCHEINLICHKEIT VON"
50  INPUT "          SPIELER A:  PA = ";PA
60  LET R =  RND (1)
70  LET N = N + 1
80  IF (R < = PA) THEN A = A + 1
90  IF (R > PA) THEN B = B + 1
100  PRINT "A = ";A;"   B = ";B
110  IF  ABS (A - B) < 2 THEN  GOTO 60
120  PRINT "SPIELDAUER: N = ";N
130  END
```

(A.3.5) Ändern Sie das TISCHTENNIS-Programm so ab, daß
- (a) eine Tabelle erstellt wird, welche die mittlere Spieldauer in Abhängigkeit von der Spielstärke PA des Spieler A wiedergibt.
- (b) eine Häufigkeitstabelle der verschiedenen Spieldauern ausgegeben wird.

(A.3.6) Die Spielstärke pro Ballwechsel des Spielers A sei: PA = 0,3. Schätzen Sie, wie viele von 100000 Endspielen der Spieler A voraussichtlich gewinnen wird. Führen Sie eine entsprechende Computer-Simulation durch und analysieren Sie das Ergebnis.

Schüler schätzen die Anzahl der Gesamtgewinne von A praktisch regelmäßig auf etwa 30000 ein. Das Beispiel ist eine gute Illustration der Tatsache, daß man in ungewohnten Situationen versucht ist, zunächst einmal Proportionalitätsschlüsse anzuwenden. Gerade im Bereiche der stochastischen Simulationen gibt es eine Fülle von Beispielen, die sehr gut geeignet sind, die Grenzen des Proportionalitätsdenkens aufzuzeigen.

Programm: TISCHTENNIS-2

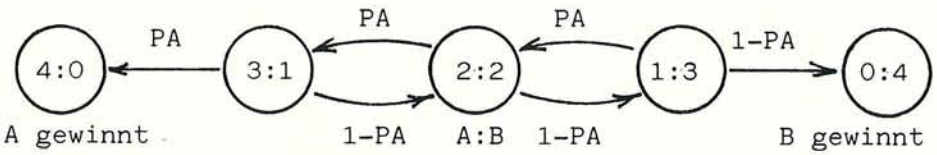
```
10  HOME : PRINT "SIMULATION DES ENDSPIELS"
20  PRINT "BEIM TISCHTENNIS"
30  PRINT
40  INPUT "ANZAHL DER LAEUFE:  L = ";L
50  PRINT
60  PRINT "GEWINNWAHRSCHEINLICHKEIT VON"
70  INPUT "          SPIELER A:  PA = ";PA
80  FOR I = 1 TO L
90  PRINT
100  PRINT "SPIEL NR.: ";I
110  LET A = 0
120  LET B = 0
130  LET R =  RND (1)
140  LET N = N + 1
150  IF (R < = PA) THEN A = A + 1
160  IF (R > PA) THEN B = B + 1
170  PRINT "A = ";A;"   B = ";B
180  IF  ABS (A - B) < 2 THEN  GOTO 130
190  IF (A > B) THEN GA = GA + 1
200  IF (B > A) THEN GB = GB + 1
210  NEXT I
220  PRINT
230  PRINT "DURCHSCHNITTLICHE SPIELDAUER:"
240  PRINT N / L
250  PRINT
260  PRINT "SIEGE VON A: ";GA
270  PRINT "SIEGE VON B: ";GB
280  END
```

Spielerruin

Strukturell sehr eng mit dem Tischtennis-Spielende verwandt ist das Problem des Spielerruins: zwei Spieler A und B spielen gegeneinander. Zu Beginn erhält jeder der Spieler V DM; (V steht für Vorrat). Die Gewinnwahrscheinlichkeit PA für Spieler A ist beliebig vorgebbar; die von Spieler B berechnet sich dann zu 1-PA. Pro Einzelspiel setzt jeder Spieler 1 DM. Der Gewinner erhält den Gesamteinsatz; der Verlierer geht leer aus. Wer zuerst bankrott ist, hat verloren.

Das Tischtennis-Spielende kann als Spezialfall dieses "Ruinproblems" angesehen werden: jeder der Spieler erhält 2 DM. Die folgende graphische Darstellung macht dies besonders deutlich:

Spielerruin mit V = 2:



Tischtennis-Spielende:

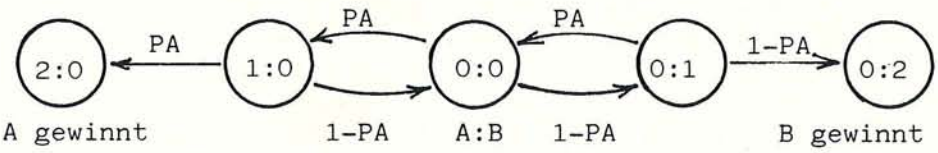


Abb.3.4

Die Pfeile stellen Übergänge zwischen den jeweiligen Spielzuständen dar. Die Übergangswahrscheinlichkeiten sind jeweils bei den Pfeilen angegeben.

Im Programm SPIELERRUIN-1 wird wieder zunächst das Einzelspiel, in SPIELERRUIN-2 eine ganze Spielserie simuliert. Im zweiten Programm werden fortlaufend die durchschnittlichen Wartezeiten ermittelt.

Programm: SPIELERRUIN-1

```
10  HOME : PRINT "SPIELERRUIN"
20  PRINT
30  PRINT "SPIELSTAERKE DES SPIELERS A "
40  INPUT "BEZOGEN AUF EIN EINZELSPIEL: P = ";P
50  PRINT
60  PRINT "SPIELSTAERKE B: ";1 - P
70  PRINT
80  INPUT "GELDVORRAT JEDES SPIELERS:  V = ";V
90  PRINT
100  PRINT "SPIEL NR.","KONTO A","KONTO B"
110  LET VA = V
120  LET VB = V
130  LET I = 0
140  LET R =  RND (1)
150  IF R < P THEN VA = VA + 1:VB = VB - 1
160  IF R > = P THEN VB = VB + 1:VA = VA - 1
170  LET I = I + 1
180  PRINT I,VA,VB
190  IF (VA > 0) AND (VB > 0) THEN  GOTO 140
200  END
```

Programm: SPIELERRUIN-2

```
10  HOME : PRINT "SPIELERRUIN"
20  PRINT
30  PRINT "SPIELSTAERKE DES SPIELERS A "
40  INPUT "BEZOGEN AUF EIN EINZELSPIEL: P = ";P
50  PRINT
60  PRINT "SPIELSTAERKE B: ";1 - P
70  PRINT
80  INPUT "GELDVORRAT JEDES SPIELERS:  V = ";V
90  PRINT
100  INPUT "ANZAHL DER LAEUFE:  L = ";L
110  PRINT
120  PRINT "RUNDE" TAB( 10)"WARTEZEIT" TAB( 20)"DURCHSCHNITT"
130  FOR J = 1 TO L
140  LET VA = V: LET VB = V
150  LET I = 0
160  LET R =  RND (1)
170  IF R < P THEN VA = VA + 1:VB = VB - 1
180  IF R > = P THEN VB = VB + 1:VA = VA - 1
190  LET I = I + 1
```

```
(* Programm:  SPIELERRUIN-2    ...   Fortsetzung *)

200  IF (VA > 0) AND (VB > 0) THEN  GOTO 160
210  LET SI = SI + I
220  LET DI = SI / J
230  PRINT J; TAB( 10)I; TAB( 20)DI
240  NEXT J
250  END
```

(A.3.7) Schreiben Sie ein Programm, das in Tabellenform ausgibt:

(a) die relative Häufigkeit der Gesamtsiege von A in Abhängigkeit von seiner Gewinnwahrscheinlichkeit PA pro Einzelspiel;

(b) die mittlere Spieldauer in Abhängigkeit vom Einsatz V.

Stellen Sie die Ergebnisse graphisch dar und versuchen Sie, etwaige Gesetzmäßigkeiten durch Formeln auszudrükken.

Warteschlangen

Vor Bank- oder Postschaltern, Tankstellen oder Supermarktkassen bilden sich erfahrungsgemäß zu gewissen Zeiten längere oder kürzere Schlangen. Der Inhaber eines Dienstleistungsunternehmens muß, insbesondere wenn es ein nach privatwirtschaftlichen Rentabilitätsgrundsätzen geführtes Unternehmen ist, daran interessiert sein, daß

- einerseits die Schlangen nicht zu groß werden (weil sonst die Kunden abgeschreckt werden);

- andererseits an der Kasse keine allzu großen Leerlaufzeiten durch Überkapazitäten entstehen.

Grundlegend für das Verständnis der Warteschlangen-Problematik ist das Konzept der Ankunftswahrscheinlichkeit bzw. Bedienungswahrscheinlichkeit pro Zeitintervall. Eine Ankunftswahrscheinlichkeit von 0.2 soll bedeuten, daß in 100 Zeiteinheiten etwa mit 20 Zugängen zu rechnen ist. Entsprechendes gelte für die Bedienungswahrscheinlichkeit. Indem man notfalls die Dauer des Zeitintervalls hinreichend klein

wählt, kann man durchaus von Wahrscheinlichkeiten an Stelle von beliebigen Raten ausgehen.

Im Programm WARTESCHLANGEN-1 wird der Prozeß der Schlangenbildung bei frei festlegbarer Ankunfts- und Bedienungswahrscheinlichkeit anhand mehrer Durchläufe simuliert.

Bei realen Warteschlangen wird ein potentieller Kunde meist durch zu große Schlangenlängen abgeschreckt. Dieser Tatsache wird im Programm WARTESCHLANGEN-2 durch "Einbau" einer oberen Schranke für die Schlangenlänge Rechnung getragen. Darüber hinaus wird in diesem Programm die durchschnittliche Schlangenlänge ermittelt.

(A.3.8) Ändern Sie das Programm WARTESCHLANGEN-2 so ab, daß
- (a) die durchschnittliche Schlangenlänge fortlaufend ermittelt wird;
- (b) eine Häufigkeitsverteilung (-tabelle) der verschiedenen Schlangenlängen ermittelt und ausgegeben wird.

Programm: WARTESCHLANGEN-1

```
10  REM  WARTESCHLANGEN
20  PRINT "WARTESCHLANGEN"
30  PRINT "BEDIENUNGSWAHRSCHEINLICHKEIT"
40  PRINT "      PRO ZEITEINHEIT:  BW = ";
50  INPUT BW
60  PRINT "ANKUNFTSWAHRSCHEINLICHKEIT"
70  PRINT "      PRO ZEITEINHEIT:  AW = ";
80  INPUT AW
90  PRINT "ANZAHL DER LAEUFE:  L= ";
100  INPUT L
110  LET SL = 0
120  PRINT "ZEITEINH.  ABGANG  ZUGANG  SCHLANGE"
130  FOR I = 1 TO L
140  LET AB = 0
150  LET ZU = 0
160  LET PB =  RND (1)
170  IF (PB < = BW) AND (SL > 0) THEN  LET AB = 1
180  LET SL = SL - AB
190  LET PA =  RND (1)
200  IF PA < = AW THEN ZU = 1
210  LET SL = SL + ZU
220  PRINT  TAB( 5)I; TAB( 15)AB; TAB( 23)ZU; TAB( 32)SL
230  NEXT I
240  END
```

Programm: WARTESCHLANGEN-2

```
10  HOME : PRINT "WARTESCHLANGEN"
20  PRINT
30  PRINT "BEDIENUNGSWAHRSCHEINLICHKEIT"
40  PRINT "      PRO ZEITEINHEIT:  BW = ";
50  INPUT BW
60  PRINT
70  PRINT "ANKUNFTSWAHRSCHEINLICHKEIT"
80  PRINT "      PRO ZEITEINHEIT:  AW = ";
90  INPUT AW
100  PRINT
110  PRINT "MAXIMALE SCHLANGENLAENGE: MS = ";
120  INPUT MS
130  PRINT
140  PRINT "ANZAHL DER LAEUFE:  L= ";
150  INPUT L
160  PRINT
170  LET SL = 0
180  LET SS = 0
190  PRINT "ZEITEINH.  ABGANG  ZUGANG  SCHLANGE"
200  FOR I = 1 TO L
210  LET AB = 0
220  LET ZU = 0
230  LET PB =  RND (1)
240  IF (PB <  = BW) AND (SL > 0) THEN  LET AB = 1
250  LET SL = SL - AB
260  LET PA =  RND (1)
270  IF (PA <  = AW) AND (SL < MS) THEN ZU = 1
280  LET SL = SL + ZU
290  LET SS = SS + SL
300  PRINT  TAB( 5)I; TAB( 15)AB; TAB( 23)ZU; TAB( 32)SL
310  NEXT I
320  PRINT
330  PRINT "DURCHSCHNITTLICHE SCHLANGENLAENGE:"
340  PRINT  TAB( 19)SS / L
350  END
```

Maschinenausfälle

Eine Firma habe einen Maschinenpark von M Maschinen desselben Typs. Der Maschinentyp habe eine bestimmte Ausfallwahrscheinlichkeit PA pro Zeiteinheit (im folgenden: pro Tag).

Programm: MASCHINENAUSFAELLE-1

```
10  HOME : PRINT "MASCHINENAUSFAELLE"
20  PRINT
30  INPUT "ANZAHL DER MASCHINEN:  M = ";M
40  INPUT "AUSFALLWAHRSCHEINLICHKEIT: PA = ";PA
50  INPUT "ANZAHL DER LAEUFE:  L = ";L
60  PRINT
70  FOR I = 1 TO L
80  LET AF = 0
90  FOR J = 1 TO M
100  LET R =  RND (1)
110  IF (R < PA) THEN AF = AF + 1
120  NEXT J
130  PRINT "TAG NR.: ";I; TAB( 15)" AUSFAELLE: ";AF
140  NEXT I
150  END
```

Programm: MASCHINENAUSFAELLE-2

```
10  HOME : PRINT "MASCHINENAUSFAELLE"
20  PRINT
30  INPUT "ANZAHL DER MASCHINEN:  M = ";M
40  DIM MA(M)
50  INPUT "AUSFALLWAHRSCHEINLICHKEIT: PA = ";PA
60  INPUT "ANZAHL DER LAEUFE:  L = ";L
70  PRINT
80  FOR I = 1 TO L
90  LET AF = 0
100  FOR J = 1 TO M
110  LET R =  RND (1)
120  IF (R < PA) THEN AF = AF + 1
130  NEXT J
140  LET MA(AF) = MA(AF) + 1
150  PRINT "TAG NR.: ";I; TAB( 15)" AUSFAELLE: ";AF
160  NEXT I
170  PRINT
180  FOR I = 0 TO M
190  PRINT "AUSFAELLE: ";I; TAB( 20)" AN ";MA(I);"  TAGEN"
200  NEXT I
210  END
```

Im Programm MASCHINENAUSFAELLE-1 wird in der inneren Schleife (Zeile 90-120) durch eine Simulation die Anzahl der an einem bestimmten Tage ausgefallenen Maschinen ermittelt. Diese Simulation wird durch die Kontrollstruktur der äußeren Schleife (Zeile 70-140) L-mal wiederholt.

Im Programm MASCHINENAUSFAELLE-2 wird darüber hinaus eine Häufigkeitstabelle der ausgefallenen Maschinen ermittelt und ausgegeben.

(A.3.9) Überlegen Sie, welche weiteren statistischen Kenndaten (wie z. B. mittlere Ausfallzahl pro Tag) wünschenswert wären, und ermitteln Sie diese Parameter in einem entsprechenden Programm.

Das Sammlerproblem

In Cornflake- oder Haferflocken-Packungen sind gelegentlich Sammelmarken ("Pennies") enthalten. Auf jedem Penny steht eine Ziffer zwischen 1 und 6. Einen vollständigen Satz, also eine Sammlung der sechs verschiedenen Pennies können die Kinder beim Kaufmann gegen ein kleines Geschenk eintauschen. Wieviele Pakete muß man durchschnittlich kaufen, um einen vollständigen Satz von Pennies zu bekommen? Das hängt natürlich davon ab, mit welcher Häufigkeit die einzelnen Penny-Marken auftreten. Wir wollen im folgenden davon ausgehen, daß sie alle gleich häufig vorkommen.(Bei einem ähnlichen "Spiel" für Autofahrer waren einmal Benzinmarken des Typs A bis F zu sammeln. Gelegentlich konnte man in der Zeitung Annoncen der Art "Tausche B gegen D" lesen. B kam jedoch sehr häufig, D dagegen sehr selten vor.)

Im Programm SAMMLER-1 wird die Entnahme eines Pennies aus der Packung durch "Erwürfeln" einer Zufallsziffer simuliert und die Wartezeit ermittelt. Nach jedem Würfeln (Zeile 50 und 60) findet eine Überprüfung auf Vollständigkeit statt (Zeile 70-100). Dazu wird zunächst der "Schalter" KP (für komplett) auf 1 gesetzt. Ist eines der Felder A(1),...,A(6), in denen die Häufigkeiten der gewürfelten Zahlen gespeichert sind, gleich Null, so ist der Satz noch nicht komplett. Der Schalter KP wird dann auf 0 gesetzt, und es muß weitergewürfelt werden. (Ein Schüler machte einmal den folgenden Vorschlag: als Vollständigkeitstest solle man die Speicherinhalte von A(1) bis A(6) alle miteinander multiplizieren.

Programm: SAMMLER-1

```
10  HOME : PRINT "SAMMLERPROBLEM"
20  PRINT
30  LET R =  INT ( RND (1) * 6) + 1
40  PRINT R;" ";
50  LET A(R) = A(R) + 1
60  LET WZ = WZ + 1
70  LET KP = 1
80  FOR I = 1 TO 6
90  IF A(I) = 0 THEN KP = 0
100  NEXT I
110  IF KP = 0 THEN  GOTO 30
120  PRINT
130  PRINT
140  PRINT "WARTEZEIT   ";WZ
150  END
```

Programm: SAMMLER-2

```
10  HOME : PRINT "SAMMLERPROBLEM"
20  PRINT
30  INPUT "ANZAHL DER LAEUFE:  L = ";L
40  PRINT
50  FOR J = 1 TO L
60  LET WZ = 0
70  FOR I = 1 TO 6
80  LET A(I) = 0
90  NEXT I
100  LET R =  INT ( RND (1) * 6) + 1
110  LET A(R) = A(R) + 1
120  LET WZ = WZ + 1
130  LET KP = 1
140  FOR I = 1 TO 6
150  IF A(I) = 0 THEN KP = 0
160  NEXT I
170  IF KP = 0 THEN  GOTO 100
180  LET WS = WS + WZ
190  PRINT "J = ";J; TAB( 10)"WARTEZEIT: ";WZ; TAB( 26)"D = ";WS / J
200  NEXT J
210  PRINT
220  PRINT "DURCHSCHNITTLICHE WARTEZEIT:  ";
230  PRINT  TAB( 20)WS / L
240  END
```

Programm: SAMMLER-3

```
10  HOME : PRINT "SAMMLERPROBLEM"
20  PRINT
30  INPUT "ANZAHL DER LAEUFE:  L = ";L
40  DIM U(L): REM  U...URLISTE
50  PRINT
60  FOR J = 1 TO L
70  LET WZ = 0
80  FOR I = 1 TO 6
90  LET A(I) = 0
100  NEXT I
110  LET R =  INT ( RND (1) * 6) + 1
120  LET A(R) = A(R) + 1
130  LET WZ = WZ + 1
140  LET KP = 1
150  FOR I = 1 TO 6
160  IF A(I) = 0 THEN KP = 0
170  NEXT I
180  IF KP = 0 THEN  GOTO 110
190  LET WS = WS + WZ
200  PRINT "J = ";J; TAB( 10)"WARTEZEIT: ";WZ; TAB( 26)"D = ";WS / J
210  LET U(J) = WZ
220  IF MU < WZ THEN  LET MU = WZ
230  NEXT J
240  PRINT
250  PRINT "DURCHSCHNITTLICHE WARTEZEIT:  ";
260  PRINT  TAB( 20)WS / L
270  DIM V(MU): REM    V...VERTEILUNGSTABELLE
280  FOR J = 1 TO L
290  LET V(U(J)) = V(U(J)) + 1
300  NEXT J
310  PRINT : PRINT
320  PRINT "VERTEILUNG DER WARTEZEITEN:"
330  PRINT
340  PRINT "WARTEZEIT" TAB( 12)"ABSOLUT" TAB( 20)"RELATIV"
350  PRINT
360  FOR J = 1 TO MU
370  IF  NOT (V(J) = 0) THEN  PRINT  TAB( 3)J TAB( 14)V(J) TAB( 22)V(J) / L
380  IF MV < V(J) THEN  LET MV = V(J)
390  NEXT J
400  END
```

Ist das Ergebnis von Null verschieden, so ist der Satz vollständig, andernfalls nicht.)

Im Programm SAMMLER-2 wird diese Simulation L-mal wiederholt, wobei nur noch die Wartezeiten und ihre laufenden Durchschnitte ausgegeben werden. Die Frage, wie oft die einzelnen Wartezeiten vorkommen, wird in SAMMLER-3 untersucht. Dazu werden die einzelnen Wartezeiten zunächst in einer Urliste U gespeichert. Das Maximum MU dieser Urlisten-Werte wird zur Dimensionierung der Tabelle V für die Häufigkeiten der Wartezeiten herangezogen.

(A.3.10) Die graphische Darstellung von Verteilungstabellen in Form von Stäbchendiagrammen ("Histogrammen") ist besonders anschaulich und einprägsam.
Erweitern Sie das Programm SAMMLER-3 durch eine Histogrammdarstellung der Verteilungstabelle V.

Histogramme

Das Programm HISTOGRAMM-DEMO zeigt die Umsetzung einer Zufallsziffern-Tabelle in ein Histogramm. Im Hinblick auf die Arbeitsweise mit der hochauflösenden Graphik des APPLE-II Computers wird auf das Programm FUNKTIONSSCHAUBILD in §4 verwiesen.

(A.3.11) Schreiben Sie eigene Simulationsprogramme für Spiele oder Abläufe aus dem Berufs- und Alltagsleben (z. B. Roulette, Fremdwörter-Abfrage-Programm, Verkehrsregelung).

(A.3.12) Schreiben Sie im Rahmen der Möglichkeiten Ihres Computers einen möglichst universellen "Graphik-Modul", d. h. ein Programm, das sich gut in bestehende Programme zur graphischen Darstellung von zweidimensionalen Tabellen eingliedern läßt.
Manche Computer verfügen auch über Aufrufmechanismen für Graphik-Moduln, die (sei es vom Hersteller, sei es vom Benutzer selbst) in eine "Programm-Bibliothek" eingegliedert worden sind.

Programm: HISTOGRAMM-DEMO

```
10  TEXT : HOME : PRINT "ZUFALLSZAHLEN-TABELLE"
20  PRINT
30  INPUT "UMFANG:  N = ";N
40  DIM T(N)
50  PRINT
60  PRINT "SCHRANKEN FUER DIE ZUFALLSZIFFERN:"
70  INPUT "MINIMUM: MI = ";MI
80  INPUT "MAXIMUM: MX = ";MX
90  FOR I = 1 TO N
100  LET T(I) =  RND (1) * (MX - MI) + MI
110  NEXT I
120  LET X1 =  - 1:X2 = N + 1
130  LET Y1 =  - 1:Y2 = MX
140  IF MI < - 1 THEN  LET Y1 = MI
150  REM    *** HISTOGRAMM ***
160  DEF  FN F(X) = (X - X1) * 279 / (X2 - X1)
170  DEF  FN G(Y) = (Y - Y1) * ( - 159) / (Y2 - Y1) + 159
180  LET XO =  FN F(O)
190  LET YO =  FN G(O)
200  HGR
210  HCOLOR= 3
220  HOME : VTAB 21
230  PRINT "X-MIN = ";X1; TAB( 20)"X-MAX = ";X2
240  PRINT "Y-MIN = ";Y1; TAB( 20)"Y-MAX = ";Y2
250  HPLOT 0,YO TO 279,YO
260  HPLOT XO,0 TO XO,159
270  FOR I = 1 TO N
280  LET X = I - 0.5
290  LET U =  FN F(X)
300  LET V =  FN G(T(I))
310  HPLOT U,YO
320  HPLOT  TO U,V
330  LET X = X + 1
340  LET U =  FN F(X)
350  HPLOT  TO U,V
360  HPLOT  TO U,YO
370  NEXT I
380  END
```

§4 EINZELPROBLEME

Primfaktorzerlegung

Im Programm PRIMFAKTORZERLEGUNG werden die Primfaktoren der Zahl N der Größe nach ausgedruckt. Zur Untersuchung von Teilbarkeitsfragen ist die INTEGER-Funktion nützlich, denn für natürliche Zahlen A und B gilt:

B teilt A genau dann, wenn A/B = INT(A/B).

Durch den Befehl LET N = N/T (Zeile 50) werden die ermittelten Primfaktoren "herausdividiert".

(A.4.1) Schreiben Sie ein Programm zur Ermittlung aller Teiler einer Zahl N.

Programm: PRIMFAKTORZERLEGUNG

```
10  HOME : PRINT "PRIMFAKTORZERLEGUNG": PRINT
20  INPUT "ZAHL: N = ";N
30  PRINT : PRINT "PRIMFAKTOREN:": PRINT
40  LET T = 2
50  IF N / T =  INT (N / T) THEN  PRINT T: LET N = N / T: GOTO 50
60  IF T > N THEN  END
70  IF T = 2 THEN  LET T = 3: GOTO 50
80  LET T = T + 2: GOTO 50
```

Wertetafeln

Bei der Funktions- und Termauswertung im mathematisch/naturwissenschaftlichen Unterricht leistet der Computer wertvolle Dienste. Im Programm WERTETAFEL wird die Auswertung der Funktion S(X)=SIN(X) bei vorgegebener Unterschranke, Oberschranke und Schrittweite gezeigt. Die Sinus-Funktion kann natürlich durch jede beliebige, in BASIC definierbare Funktion ersetzt werden. Zeile 30 ist ein weiteres Beispiel dafür, daß Kommandos der Betriebssprache auch vom Programm aus aufgerufen werden können.

Programm: WERTETAFEL

```
10  HOME : PRINT "WERTETAFEL"
20  DEF  FN S(X) =  SIN (X)
30  LIST 20
40  INPUT "X-MIN = ";X1
50  INPUT "X-MAX = ";X2
60  INPUT "SCHRITTWEITE:  H = ";H
70  FOR X = X1 TO X2 STEP H
80  PRINT X TAB( 20) FN S(X)
90  NEXT X
100  END
```

Termauswertung

In diesem Programm wird der Term R1*R2/(R1+R2) ausgewertet. Er tritt z. B. bei der Berechnung des Gesamtwiderstandes in einem Schaltnetz auf, das aus den beiden parallelgeschalteten Widerständen R1 und R2 besteht. Bereits in diesem einfachen Programm zeigen sich leider schon die Nachteile der stark eingeschränkten Bezeichnungsmöglichkeiten für Variable in den meisten BASIC-Dialekten. Für eine gute Programmdokumentation wäre es sinnvoll, die untere und obere Grenze des Definitionsbereiches im Programm WERTETAFEL z. B. als XMIN und XMAX zu bezeichnen; doch diese Namen können die

Programm: TERMAUSWERTUNG

```
10  HOME : PRINT "TERMAUSWERTUNG"
20  PRINT : LIST 120: PRINT
30  INPUT "R1-MIN = ";G1
40  INPUT "R1-MAX = ";G2
50  INPUT "R2-MIN = ";G3
60  INPUT "R2-MAX = ";G4
70  INPUT "SCHRITTWEITE FUER R1:  H1 = ";H1
80  INPUT "SCHRITTWEITE FUER R2:  H2 = ";H2
90  FOR R1 = G1 TO G2 STEP H1
100  FOR R2 = G3 TO G4 STEP H2
110  IF R1 + R2 = 0 THEN  GOTO 140
120  LET R = R1 * R2 / (R1 + R2)
130  PRINT R1 TAB( 12)R2 TAB( 24)R
140  NEXT R2
150  NEXT R1
160  END
```

meisten BASIC-Systeme nicht auseinanderhalten. Deshalb wurde für XMIN der Name X1 und für XMAX der Name X2 verwendet. Im Sinne dieser Bezeichnungssystematik wäre es nun konsequent, die entsprechenden Grenzen für die Variable R1 im Programm TERMAUSWERTUNG als R11 und R12 zu bezeichnen. Aber auch diese Namen werden nicht unterschieden. Bei komplexen Programmen können sich diese Einschränkungen in der Bezeichnungstechnik wegen der damit verbundenen Unsystematik als sehr störend erweisen.

Funktionsschaubild

Eine der Grundtechniken bei der Diskussion mathematischer Funktionen ist ihre Darstellung im kartesischen Koordinatensystem. Um diese Aufgabe in sinnvoller Weise mit dem Computer anzugehen, muß dieser über die folgenden Ausstattungsmerkmale verfügen:

(1.) eine (einigermaßen) hochauflösende Graphik;

(2.) graphik-orientierte Grundbefehle (wie z. B. PLOT-Befehle u.ä.).

Die folgende Diskussion der Funktionsdarstellung im Koordinatensystem bezieht sich zwar auf die technische Ausstattung des APPLE-II Computers; sie ist aber leicht auf andere Computer mit den obengenannten graphischen Möglichkeiten übertragbar.
Der Bildschirm des APPLE-II ist (im Graphik-Modus 1, auf den wir uns hier beschränken wollen) in ein Raster von 160 Zeilen und 280 Spalten eingeteilt:

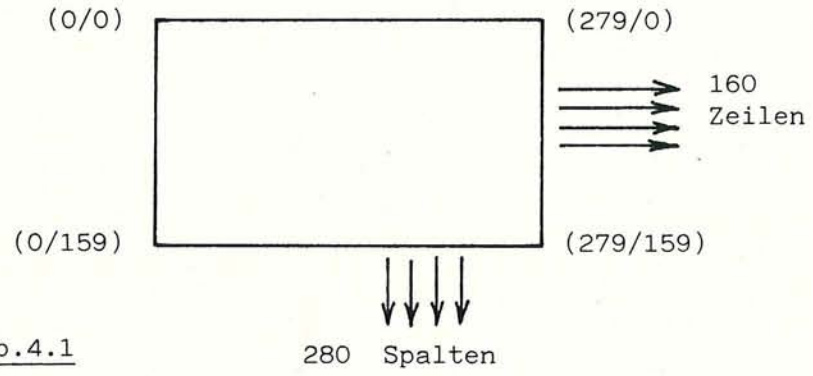


Abb.4.1

Der Punkt mit den Koordinaten (U,V) befindet sich in der Spalte Nr. U und der Zeile Nr. V. (Die Spaltenwerte entsprechen also den Werten auf der x-Achse und die Zeilenwerte denjenigen auf der y-Achse). Unterhalb dieses Graphikfensters stehen im Graphik-Modus 1 noch vier Textzeilen zur Wiedergabe bestimmter Informationen oder für Dialogzwecke zur Verfügung.

Durch den Befehl HGR (für: High resolution GRaphics) wird der Rechner in den graphischen Arbeitsmodus versetzt. Bei der Verwendung von Farb-Bildschirmen sind durch HCOLOR=I (für I = 0,1,...,7) verschiedene Farben einstellbar; für Schwarzweißbildschirme erweist sich die Farbe I = 3 als relativ günstig. Durch den Befehl HPLOT A,B wird der Punkt mit den Bildschirmkoordinaten (A,B) gezeichnet. Der Befehl HPLOT A,B TO C,D verbindet (so gut es geht) die Punkte (A,B) und (C,D) durch eine gerade Linie und HPLOT TO A,B bewirkt, daß eine gerade Linie vom augenblicklichen Standort zum Punkt (A,B) gezeichnet wird.

Der darzustellende Ausschnitt der Objektebene (realen Ebene) sei auf der x-Achse durch die Werte X1 und X2, auf der y-Achse durch Y1 und Y2 begrenzt:

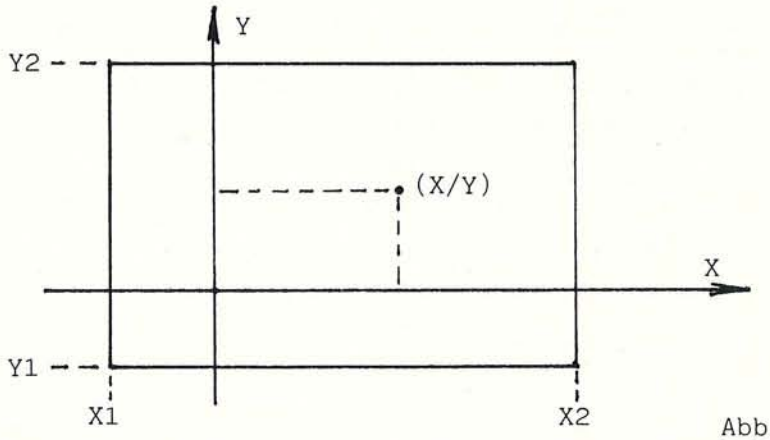


Abb.4.2

Wir stehen nun vor dem Problem, das Rechteck (X1,X2)x(Y1,Y2) auf das Rechteck (0,279)x(159,0) abzubilden. Die natürlichste Art ist, dies komponentenweise zu tun (also die Intervalle auf der x-Achse und auf der y-Achse getrennt abzubilden). Dies reduziert das Problem auf die Abbildung des Intervalls (X1,X2) auf (0,279) und des Intervalls (Y1,Y2) auf (159,0).

LINEARE TRANSFORMATION VON INTERVALLEN

Wir wollen im folgenden allgemein die Aufgabe lösen, ein Intervall (a,b) auf ein Intervall (c,d) abzubilden. Da wir hinsichtlich des Abbildungstyps nicht eingeschränkt sind, machen wir es uns nicht unnötig schwer und wählen eine lineare Funktion y=f(x), deren Graph im kartesischen Koordinatensystem eine Gerade ist:

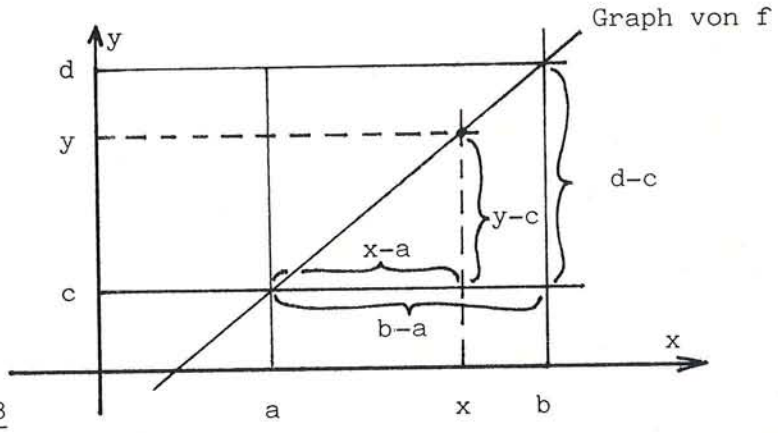


Abb.4.3

Mit Hilfe des zweiten Strahlensatzes bzw. der Zwei-Punkte-Form der Geradengleichung erhalten wir:

$$\frac{y-c}{d-c} = \frac{x-a}{b-a} \qquad \text{bzw.} \qquad y = (x-a).\frac{d-c}{b-a} + c.$$

Im Hinblick auf das Problem der Transformation eines rechteckigen Ausschnitts der Ebene auf den Computer-Bildschirm erhalten wir hieraus die Transformationsgleichungen:

Für die x-Achse (a=X1, b=X2, c=0, d=279):

```
U = F(X) = (X-X1)*279/(X2-X1).
```

Für die y-Achse (a=Y1, b=Y2, c=159, d=0):

```
V = G(Y) = (Y-Y1)*(-159)/(Y2-Y1) + 159.
```

Im Programm FUNKTIONSSCHAUBILD sind diese Transformationsfunktionen in Zeile 140 und 150 definiert. Die Befehle

Programm: FUNKTIONSSCHAUBILD

```
10  TEXT : HOME : PRINT "FUNKTIONSSCHAUBILD"
20  PRINT
30  DEF  FN S(X) =  SIN (X)
40  LIST 30
50  VTAB 21
60  INPUT "X-MIN = ";X1
70  VTAB 21: HTAB 20
80  INPUT "X-MAX = ";X2
90  INPUT "Y-MIN = ";Y1
100  VTAB 22: HTAB 20
110  INPUT "Y-MAX = ";Y2
120  INPUT "SCHRITTWEITE:  H = ";H
130  VTAB 23
140  DEF  FN F(X) = (X - X1) * 279 / (X2 - X1)
150  DEF  FN G(Y) = (Y - Y1) * ( - 159) / (Y2 - Y1) + 159
160  LET XO =  FN F(0)
170  LET YO =  FN G(0)
180  HGR : HCOLOR= 3
190  IF (0 <= YO) AND (YO <= 159) THEN  HPLOT 0,YO TO 279,YO
200  IF (0 <= XO) AND (XO <= 279) THEN  HPLOT XO,0 TO XO,159
210  LET F1 = 0
220  FOR X = X1 TO X2 STEP H
230  LET Y =  FN S(X)
240  LET U =  FN F(X)
250  LET V =  FN G(Y)
260  IF (V < 0) OR (V > 159) THEN  LET F1 = 0: GOTO 300
270  IF F1 = 0 THEN  HPLOT U,V
280  IF F1 = 1 THEN  HPLOT  TO U,V
290  LET F1 = 1
300  NEXT X
310  END
```

VTAB 21, VTAB 22 und VTAB 23 bewirken, daß der Cursor (bezüglich des Text-Fensters) auf die entsprechenden Zeilen am unteren Bildrand gesetzt wird. Die so am unteren Rand erzeugten Informationen bleiben dann auch im Graphik-Modus 1 lesbar. Die Zeilen 190 und 200 dienen dem Zeichnen der Koordinatenachsen. Die Variable F1 (F für "flag"), stellt eine Art Schalter dar, der anzeigt, ob der zuletzt ermittelte Wert (U,V) außerhalb (F1=0) oder innerhalb (F1=1) des Bildschirmbereiches gelegen hat. Je nachdem wird nur ein Punkt gesetzt oder eine Verbindungslinie zum neuen Wert gezogen (falls der neu ermittelte Punkt im Bildschirmbereich liegt).

(A.4.2) Verbessern Sie das Programm FUNKTIONSSCHAUBILD so, daß die Koordinatenachsen skaliert werden.

(A.4.3) Im Graphik-Modus 2 des APPLE-II Computers steht ein Bildschirmfenster von 192 Zeilen zur Verfügung. Die Spaltenzahl (280) bleibt unverändert; der untere Textrand (vier Zeilen) entfällt.
Ändern Sie die Graphik-Programme so ab, daß sie wahlweise im Modus 1 oder im Modus 2 laufen.

Wurzelbestimmung nach Heron

Die Quadratwurzel der Zahl a kann als Seitenlänge eines Quadrats vom Flächeninhalt a interpretiert werden. Beim Heron-Verfahren wird das gesuchte Quadrat über eine Folge von Rechtecken angenähert, die alle den Flächeninhalt a besitzen. Das Ausgangsrechteck ist im Prinzip beliebig vorgebbar; wir wählen die Seitenlängen x = a und y = 1.
Solange die gewünschte Genauigkeit noch nicht erreicht ist, wird der folgende Ausgleichsprozeß durchlaufen:

(1.) die Seite x(neu) wird durch das arithmetische Mittel der Seiten x(alt) und y(alt) ersetzt;

(2.) die Seite y(neu) wird so festgelegt, daß der Flächeninhalt des neuen Rechtecks gleich a ist: y=a/x.

Dieser Ausgleichsprozeß ist in Abb.4.4 graphisch dargestellt. Er wird in der Literatur häufig in abgekürzter Form durch die rekursive Folge

$$x_{k+1} = \frac{1}{2} \cdot (x_k + \frac{a}{x_k})$$

beschrieben. Im Programm HERON ist dieser Algorithmus in ein BASIC-Programm umgesetzt.

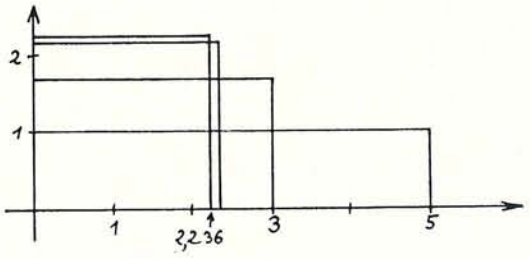


Abb.4.4

Programm: HERON

```
10  HOME : PRINT "HERON-VERFAHREN": PRINT
20  PRINT "ZUR ERMITTLUNG DER QUADRATWURZEL"
30  PRINT "    EINER ZAHL A": PRINT
40  INPUT "A = ";A
50  INPUT "GENAUIGKEIT:  E = ";E
60  PRINT
70  LET X = A: LET Y = 1
80  LET X = (X + Y) / 2
90  LET Y = A / X
100  PRINT X,Y
110  IF  ABS (X * X - A) > E THEN  GOTO 80
120  PRINT
130  PRINT "WURZEL(";A;") = ";X
140  END
```

(A.4.4) Schreiben Sie ein Programm zur Berechnung beliebiger n-ter Wurzeln unter Zugrundelegung des Ausgleichsverfahrens von Heron.
(Grundidee: Die n-te Wurzel von a ist die Kantenlänge eines n-dimensionalen Würfels vom Inhalt a).

(A.4.5) Stellen Sie die Rechtecks-Approximation des Heron-Verfahrens graphisch dar.

Quadratische Gleichungen

Ein weiteres Standardproblem des mathematisch/naturwissenschaftlichen Unterrichts ist die Lösung quadratischer Gleichungen. Wir gehen von der folgenden Gleichung aus:

$$A*X*X + B*X + C = 0$$

In Abhängigkeit von den eingegebenen Koeffizienten A, B und C ergeben sich die Fallunterscheidungen in Abb.4.5.

Das Programm QUADRATISCHE GLEICHUNG stellt im wesentlichen eine Umsetzung dieses Lösungsbaumes in einen Computer-Algorithmus dar. Durch die Verwendung eines von verschiedenen

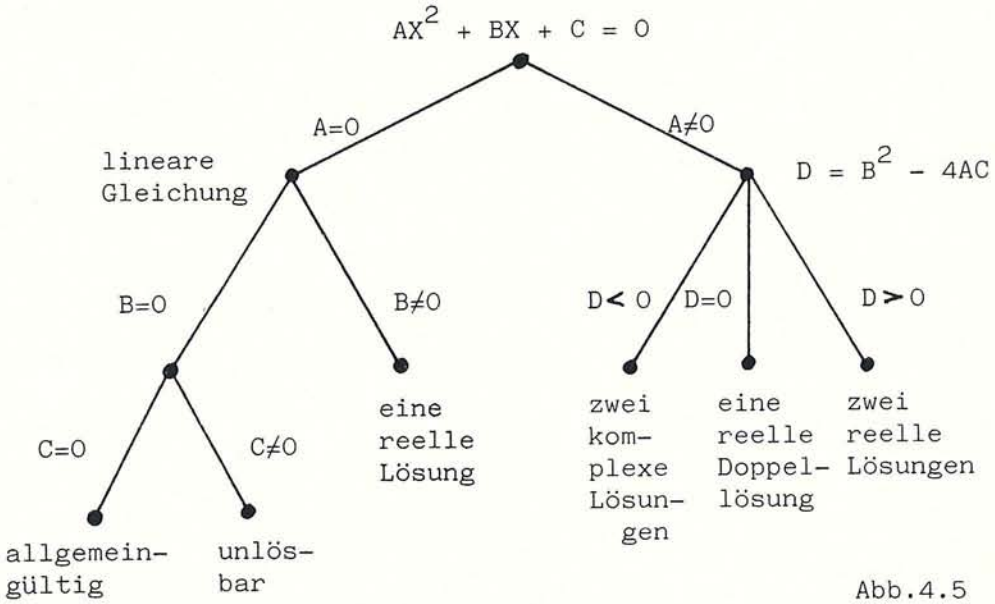


Abb.4.5

Stellen aus aufgerufenen Unterprogramms (Zeile 440-530) wird verhindert, daß die Ausgabe Dokumentation für jeden Lösungszweig einzeln aufgeschrieben werden muß.

Programm: QUADRATISCHE GLEICHUNG

```
10  HOME : PRINT "QUADRATISCHE GLEICHUNG": PRINT
20  PRINT "KOEFFIZIENTENEINGABE": PRINT
30  INPUT "KOEFFIZIENT VON X*X:  A= ";A
40  INPUT "KOEFFIZIENT VON X  :  B= ";B
50  INPUT "ABSOLUTGLIED          C= ";C
60  REM  LOESUNGSTEIL
70  IF  NOT A = 0 THEN  GOTO 220
80  IF  NOT B = 0 THEN  GOTO 170
90  IF B = 0 AND C = 0 GOTO 140
100  REM    ES BLEIBT DER FALL: B=0 UND C UNGLEICH 0
110  GOSUB 470
120  PRINT "DIE GLEICHUNG IST UNLOESBAR"
130  END
140  GOSUB 470
150  PRINT "DIE GLEICHUNG IST ALLGEMEINGUELTIG"
160  END
170  GOSUB 470
180  PRINT "X= ";( - C) / B
190  PRINT
```

Programm: QUADRATISCHE GLEICHUNG (Fortsetzung)

```
200  PRINT "DIE GLEICHUNG WAR IN WIRKLICHKEIT LINEAR"
210  END
220  LET D = B * B - 4 * A * C
230  IF D < 0 GOTO 330
240  IF D = 0 GOTO 400
250  REM   ES BLEIBT D>0
260  LET X1 = ( - B +  SQR (D)) / (2 * A)
270  LET X2 = ( - B -  SQR (D)) / (2 * A)
280  GOSUB 470
290  PRINT "X1= ";X1
300  PRINT "X2= ";X2
310  PRINT : PRINT "DIE LOESUNGEN SIND REELL"
320  END
330  LET X = ( - B) / (2 * A)
340  LET Z =  SQR ( - D) / (2 * A)
350  GOSUB 470
360  PRINT "X1= ";X" + "Z" * I"
370  PRINT "X2= ";X" - "Z" * I"
380  PRINT : PRINT "DIE LOESUNGEN SIND KOMPLEX"
390  END
400  LET X1 = ( - B) / (2 * A)
410  GOSUB 470
420  PRINT "X1= ";X1
430  PRINT "X2= ";X1
440  PRINT : PRINT "DIE LOESUNGEN SIND REELL"
450  PRINT "UND FALLEN ZUSAMMEN"
460  END
470  REM  AUSGABETEIL
480  PRINT : PRINT
490  PRINT "DIE QUADRATISCHE GLEICHUNG"
500  PRINT "    MIT DEN KOEFFIZIENTEN:": PRINT
510  PRINT "A =  ";A
520  PRINT "B =  ";B
530  PRINT "C =  ";C
540  PRINT : PRINT
550  PRINT "HAT DIE LOESUNG(EN):": PRINT
560  RETURN
```

(A.4.6) Ändern Sie das Programm QUADRATISCHE GLEICHUNG im folgenden Sinne zu einer "Schülerversion" ab:
Falls $D < 0$ ist (komplexe Lösungen), soll an Stelle der bisherigen Lösungsausgabe der Ausdruck
"DIE GLEICHUNG HAT KEINE LÖSUNGEN"
erfolgen.

Katzenvermehrung

Die in §2 behandelten Wachstumstypen und insbesondere das einfache Modell des exponentiellen Wachstums lassen außer acht, daß die neugeborenen Individuen einer Population meist nicht sofort geschlechtsreif sind. Im Programm KATZENVERMEHRUNG ist ein einfaches Modell für generationsbezogenes Wachstum gegeben. Der Wert der Variablen K gibt die Anzahl aller Katzen und der von G die Anzahl der geschlechtsreifen Tiere an. Der Nachwuchs N macht pro Periode das 1,4-fache von G aus. Er ist nicht sofort, sondern erst in der übernächsten Periode geschlechtsreif. Dies wird im Modell durch die verzögerte Summierung des Nachwuchses berücksichtigt, dessen Anzahl zunächst in dem "Schieberegister" N1 zwischengespeichert wird.

(A.4.7) Schreiben Sie ein Programm, das ein Wachstumsmodell simuliert, bei dem der Nachwuchs erst nach einer beliebig vorgebbaren Anzahl von Perioden geschlechtsreif wird.

Programm: KATZENVERMEHRUNG

```
10  HOME : PRINT "KATZEN-VERMEHRUNG (MIT VERZOEGERUNG)"
20  PRINT
30  INPUT "LAUFZEIT (PERIODEN):  L = ";L
40  PRINT
50  LET K = 2
60  LET G = 2
70  FOR I = 1 TO L
80  LET N1 = N
90  LET G = G + N1
100  LET N = G * 1.4
110  LET K = K + N
120  PRINT "PERIODE:  ";I
130  PRINT  TAB( 3)"NACHWUCHS (PERIODE ALT):" TAB( 30)N1
140  PRINT  TAB( 3)"GESCHLECHTSREIF: " TAB( 30)G
150  PRINT  TAB( 3)"NACHWUCHS (NEUGEBOREN): " TAB( 30)N
160  PRINT  TAB( 3)"GESAMTZAHL DER KATZEN: " TAB( 30)K
170  PRINT
180  NEXT I
190  END
```

Zehngang-Fahrrad

Zehngang-Fahrräder verkaufen sich gut, denn es scheint klar zu sein, daß zehn Gänge besser sind als drei. Dennoch bleiben einige Fragen zu klären:

(1.) Welche Übersetzungsverhältnisse kommen überhaupt vor?

(2.) Wie groß ist die Spannweite zwischen dem kleinsten und dem größten Übersetzungsverhältnis?

(3.) In welcher Reihenfolge muß man schalten, um die verschiedenen Übersetzungsverhältnisse in monoton auf- oder absteigender Reihe zu durchlaufen?

(4.) Wie groß sind die absoluten und relativen Unterschiede zwischen aufeinanderfolgenden Übersetzungsverhältnissen?

(5.) Wie groß ist der mittlere absolute bzw. relative (prozentuale) Zuwachs aufeinanderfolgender Übersetzungsverhältnisse?

(6.) Welche Zuwächse sind physiologisch optimal?

Im Programm FAHRRAD-1 werden die Übersetzungsverhältnisse in Abhängigkeit von den eingegebenen Zahnkranzkombinationen unsortiert ausgedruckt. Darüber hinaus werden die Übersetzungsverhältnisse V(I)/H(J) im Programm FAHRRAD-2 aufsteigend sortiert ausgegeben. Dazu wird das zweidimensionale Feld, bestehend aus den Werten V(I)/H(J) (mit I = 1, 2 und J = 1,...,5) zunächst in ein eindimensionales Feld S(K) (mit K = 1,...,10) übertragen. Die zum Wert S(K) gehörenden Begleitdaten werden im Feld KB$(K) als Strings aufbereitet. Dabei dient der Befehl STR$(I) zur Überführung der Dezimalzahlvariablen I in eine Textvariable desselben Inhalts. Die "Addition" von Texten (mit Hilfe des "+"-Zeichens) geschieht in der Form der Konkatenation (Verkettung). So wird z. B. aus der Stringvariablen A$="BAHN", der Stringkonstanten "STEIG " und der numerischen Variablen C=3 durch die Operationen

```
LET D$ = A$ + "STEIG " + STR$(C)
```

der zusammengesetzte String

```
D$ = "BAHNSTEIG 3"
```

hergestellt.

(A.4.8) (a) Bauen Sie das Programm FAHRRAD-2 so aus, daß alle der eingangs angeführten Fragen (1.) bis (5.) beantwortet werden.
(b) Sammeln Sie Informationen zu Frage (6.).

Programm: FAHRRAD-1

```
10  HOME : PRINT "UEBERSETZUNGSVERHAELTNISSE BEIM"
20  PRINT "     ZEHNGANG-FAHRRAD"
30  PRINT
40  PRINT "VORDERRAD   -   BITTE GEBEN SIE"
50  PRINT "DIE ENTSPRECHENDEN ZAHLEN EIN !"
60  PRINT
70  FOR I = 1 TO 2
80  PRINT "ZAHNKRANZ NR.: ";I;" ";
90  INPUT V(I)
100  NEXT I
110  PRINT
120  PRINT "HINTERRAD   -   BITTE GEBEN SIE"
130  PRINT "DIE ENTSPRECHENDEN ZAHLEN EIN !"
140  PRINT
150  FOR J = 1 TO 5
160  PRINT "ZAHNKRANZ NR.: ";J;" ";
170  INPUT H(J)
180  NEXT J
190  PRINT
200  PRINT "I  J   V   H  V/H"; TAB( 28)"H/V"
210  PRINT
220  FOR I = 1 TO 2
230  FOR J = 1 TO 5
240  PRINT I TAB( 4)J TAB( 7)V(I) TAB( 11)H(J);
250  PRINT  TAB( 15)V(I) / H(J) TAB( 27)H(J) / V(I)
260  NEXT J
270  NEXT I
280  END
```

Programm: FAHRRAD-2

```
10  HOME : PRINT "UEBERSETZUNGSVERHAELTNISSE BEIM"
20  PRINT "     ZEHNGANG-FAHRRAD"
30  PRINT
40  PRINT "VORDERRAD   -   BITTE GEBEN SIE"
50  PRINT "DIE ENTSPRECHENDEN ZAHLEN EIN !"
60  PRINT
70  FOR I = 1 TO 2
80  PRINT "ZAHNKRANZ NR.: ";I;" ";
90  INPUT V(I)
100  NEXT I
110  PRINT
120  PRINT "HINTERRAD   -   BITTE GEBEN SIE"
130  PRINT "DIE ENTSPRECHENDEN ZAHLEN EIN !"
140  PRINT
150  FOR J = 1 TO 5
160  PRINT "ZAHNKRANZ NR.: ";J;" ";
170  INPUT H(J)
180  NEXT J
190  PRINT
200  PRINT "I  J   V   H  V/H"; TAB( 28)"H/V"
210  LET K = 0
220  PRINT
230  FOR I = 1 TO 2
240  FOR J = 1 TO 5
250  PRINT I TAB( 4)J TAB( 7)V(I) TAB( 11)H(J);
260  PRINT  TAB( 15)V(I) / H(J) TAB( 27)H(J) / V(I)
270  LET K = K + 1
280  LET KB$(K) =  STR$ (I) + "  " +  STR$ (J) + "  "
290  LET KB$(K) = KB$(K) +  STR$ (V(I)) + "  " +  STR$ (H(J))
300  LET S(K) = V(I) / H(J)
310  NEXT J
320  NEXT I
330  PRINT : PRINT "V/H: AUFSTEIGEND SORTIERT:"
340  FOR I = 1 TO 10
350  FOR J = 1 TO 10 - I
360  IF S(J) <= S(J + 1) THEN  GOTO 390
370  LET H = S(J): LET S(J) = S(J + 1): LET S(J + 1) = H
380  LET H$ = KB$(J): LET KB$(J) = KB$(J + 1): LET KB$(J + 1) = H$
390  NEXT J
400  NEXT I
410  FOR K = 1 TO 10
420  PRINT KB$(K);"  ";S(K)
430  NEXT K
440  END
```

Bubblesort

Das im Programm FAHRRAD-2 verwendete Sortierverfahren wird als Bubblesort bezeichnet. Es läßt sich zwar schnell und einfach aufschreiben; in der Ausführung ist es aber nicht sehr effektiv. Man sollte es daher nur bei kleinen Datenbeständen verwenden. Im Programm BUBBLESORT wird zunächst eine Liste von Zufallszahlen generiert. Das eigentliche Sortierverfahren nimmt nur die Zeilen 150-200 in Anspruch.

Die Grundidee des Bubblesort besteht darin, daß im noch nicht sortierten Teil der Liste benachbarte Feldelemente in mehreren Durchläufen miteinander verglichen und - falls sie noch nicht in der richtigen Reihenfolge stehen - vertauscht werden (Zeile 180).

Auch bei der Abschätzung der Sortierdauer wandten die Schüler zunächst Proportionalschlüsse an (doppelte Datenmenge - doppelte Sortierzeit). Ein Testlauf macht die Unzulässigkeit dieser Schlußweise sofort deutlich. Die folgende Überlegung ist auf viele kombinatorische Situationen übertragbar: Zerlegt man die verdoppelte Datenmenge in zwei Hälften, so wird für das getrennte Sortieren jeder der Hälften die gleiche Sortierzeit benötigt wie für die Ausgangsmenge. Also benötigt die doppelte Datenmenge mehr als die doppelte Sortierzeit.

Programm: BUBBLESORT

```
10  HOME : PRINT "BUBBLE-SORT"
20  INPUT "N = ";N
30  PRINT "TABELLE AUS ZUFALLSZIFFERN:"
40  DIM A(N)
50  FOR I = 1 TO N
60  LET A(I) =  INT ( RND (1) * 32000) + 1
70  PRINT I,A(I)
80  NEXT I
90  GOSUB 150
100  PRINT "SORTIERTE TABELLE:"
110  FOR I = 1 TO N
120  PRINT I,A(I)
130  NEXT I
140  END
150  FOR I = 1 TO N - 1
160  FOR J = 1 TO N - I
170  IF A(J) <  = A(J + 1) THEN  GOTO 190
180  LET H = A(J): LET A(J) = A(J + 1): LET A(J + 1) = H
190  NEXT J
200  NEXT I
210  RETURN
```

Quicksort

Der Quicksort ist (insbesondere bei großen Datenmengen) erheblich schneller als der Bubblesort. Das eigentliche Sortierverfahren im Programm QUICKSORT steht in Zeile 180-370.

Die Grundidee des Quicksort entspricht dem Motto:

Die Guten ins Töpfchen, die Schlechten ins Kröpfchen.

Dazu wird die zu sortierende Liste durch das Trennelement TE in zwei Teile zerlegt. Als Trennelement wird das erste (bzw. am weitesten "links" stehende) Feldelement der Liste verwendet (Zeile 290). In den Zeilen 300-350 werden die Feldelemente, die kleiner als TE sind, nach "links" und diejenigen, welche größer als TE sind, nach "rechts" sortiert. Dies geschieht mit Hilfe des linken Zeigers LZ und des rechten Zeigers RZ. Diese Zeiger werden zu Beginn des Sortiervorgangs auf die Grenzen LA und RA des zu sortierenden Feldes gesetzt. Während des Austauschvorganges nach der Methode von Aschenputtel werden die Zeiger so mitgeführt, daß sie stets die Position des geprüften und eventuell ausgetauschten Feldelements angeben. Dadurch rücken sie immer näher aufeinander zu. Der Austauschvorgang mit Trennelement TE wird so lange fortgesetzt, bis der linke Zeiger LZ und der rechte Zeiger RZ zusammenfallen (Zeile 300). Zum Schluß wird das Trennelement TE wieder an der richtigen Stelle in die Liste eingefügt (Zeile 370) und seine Position durch den Trennzeiger TZ festgehalten (Zeile 360).
Nun werden die beiden Teillisten links und rechts vom Trennelement TE nach demselben Verfahren weitersortiert. Die "Buchhaltung" über die zu sortierenden Teilhaufen wird mit Hilfe des Zählers SZ und des zweidimensionalen Feldes S(I,J) geführt. SZ gibt jeweils die Nummer der zu sortierenden Teilliste an, während im Feld S(SZ,1) seine linke Grenze und im Feld S(SZ,2) seine rechte Grenze notiert ist. (Die Festlegung der linken und rechten Grenze erfolgt natürlich mit Hilfe des Trennzeigers TZ - siehe Zeile 260 und 270). Die Bearbeitung der Teillisten geht nach dem Prinzip "last in - first out" vonstatten; das heißt, die Teilliste mit der höchsten Nummer wird zuerst sortiert.

Programm: QUICKSORT

```
10  HOME : PRINT "QUICKSORT"
20  INPUT "N = ";N
30  DIM A(N)
40  DIM S(N,2): REM  GEHT SPARSAMER
50  PRINT "TABELLE VON ZUFALLSZIFFERN:"
60  FOR I = 1 TO N
70  LET A(I) =  INT ( RND (1) * 32000) + 1
80  PRINT I,A(I)
90  NEXT I
100  LET L = 1
110  LET R = N
120  GOSUB 180
130  PRINT "SORTIERTE TABELLE:"
140  FOR I = 1 TO N
150  PRINT I,A(I)
160  NEXT I
170  END
180  LET SZ = 1
190  LET S(1,1) = L
200  LET S(1,2) = R
210  IF SZ < 1 THEN  RETURN
220  LET LA = S(SZ,1)
230  LET RA = S(SZ,2)
240  GOSUB 290
250  LET SZ = SZ - 1
260  IF LA < TZ - 1 THEN  LET SZ = SZ + 1: LET S(SZ,1) = LA: LET S(SZ,2) = TZ - 1
270  IF (TZ + 1 < RA) THEN  LET SZ = SZ + 1: LET S(SZ,1) = TZ + 1: LET S(SZ,2) = R
280  GOTO 210
290  LET LZ = LA: LET RZ = RA: LET TE = A(LZ)
300  IF LZ = RZ THEN  GOTO 360
310  IF (A(RZ) > = TE) AND (LZ < RZ) THEN  LET RZ = RZ - 1: GOTO 310
320  LET A(LZ) = A(RZ)
330  IF (A(LZ) < = TE) AND (LZ < RZ) THEN  LET LZ = LZ + 1: GOTO 330
340  LET A(RZ) = A(LZ)
350  GOTO 300
360  LET TZ = LZ
370  LET A(TZ) = TE
380  RETURN
```

§5 SCHLUSSBEMERKUNGEN

5.1 Erfahrungen

Der Arbeitsstil im Unterrichtsprojekt ging von der Voraussetzung aus, daß der Unterrichtende im Normalfall zwar die Einführung in das Gesamtprojekt und in die Problematik der jeweiligen Einzelprobleme gab, daß die Problemlösungen aber nach Möglichkeit von den Schülern selbst oder in Gruppenarbeit erarbeitet werden sollten. An besonders neuralgischen Punkten wurde im Lehrer-Schüler-Gespräch ein grober Rahmen für die Problemlösung erarbeitet, der dann von den einzelnen Gruppen selbständig zu einer kompletten Lösung ausgearbeitet wurde. Somit sahen die Programme der einzelnen Schülergruppen jeweils durchaus verschieden aus; die in dieser Darstellung wiedergegebenen Programme stellen stets nur eine von mehreren denkbaren Lösungen dar.

Wie im Vorwort erwähnt, orientierte sich das Projekt vorrangig an algorithmischen Problemstellungen und erst in zweiter Linie am syntaktischen Rahmen der verwendeten Programmiersprache. Natürlich mußten dabei Kompromisse geschlossen werden. Die Probleme wurden so ausgewählt, daß die syntaktischen Kenntnisse im Rahmen der Problemlösungen erarbeitet werden konnten. Dabei wurden die syntaktischen Kenntnisse beständig in kleinen Schritten erweitert. Bei jedem neuen Programm oder jeder neuen Version kamen im allgemeinen etwa zwei neue syntaktische Varianten vor. Da die Schüler volle Freiheit bei der Umsetzung des gemeinsam erarbeiteten algorithmischen Rahmens in ein konkretes Programm hatten, führte dies gelegentlich zu kleinen Abweichungen von der Projektplanung, die aber nicht weiter störten, da der Projektrahmen sowieso auf Flexibilität ausgelegt war. Ich ermunterte die Schüler auch, auftretende Fragen problem- oder rechnerspezifischer Natur durch (systematisches) eigenständiges Ausprobieren verschiedener Möglichkeiten selbst zu klären. Gerade diesem Arbeitsstil kam die Interaktivität des benutzten BASIC-Systems zugute.

Im Hinblick auf die Erfahrungen mit den einzelnen Unterrichtsthemen möchte ich mich auf die Beispiele des ersten Paragraphen beschränken. Denn die Besonderheiten des Arbeitens mit Computern kommen gerade in der Einführungsphase besonders intensiv zum Tragen. In Anbetracht der zunächst sehr bescheiden anmutenden Problemstellungen in §1 war ich immer wieder überrascht, zu welch hochinteressanten Unter-

richtssituationen sie Anlaß gaben. Weitere Beobachtungen zu den Themen der Paragraphen drei bis fünf sind an Ort und Stelle beschrieben.

Der in §1 gegebene Grobüberblick über den Aufbau von Computern sollte nur ein Minimum an Vorinformation über die Komponenten eines Computers und ihre Interaktionsmöglichkeiten vermitteln. Diese Unterrichtsphase wurde deshalb so knapp wie möglich gehalten. Sie diente auch der Anknüpfung an das Vorwissen der Schüler: zu meiner Freude reagierten die Schüler spontan auf das Grundschema

Eingabe → Verarbeitung → Ausgabe

und komplettierten es selbständig, nachdem ich nur das Wort "Eingabe" erwähnt hatte. Dies sei doch genau wie bei den Operatoren, sagten sie.

Die Lösung des Mehrwertsteuer-Problems diente der Festigung dieses Vorverständnisses. Ihre algorithmische Formulierung wurde gemeinsam in einer umgangssprachlichen Normsprache erarbeitet:

1. Gib den Nettobetrag N ein.
2. Berechne N*0,14 und nenne diesen Wert M.
3. Berechne N+M und nenne diesen Wert B.
4. Drucke M.
5. Drucke B.
6. Ende.

Die vom Unterrichtenden gegebene Übersetzung bzw. Transkription dieses Algorithmus ins Englische führte zum Programm MEHRWERTSTEUER-1. Das erste komplette Programm war (zusätzlich zur Einführung in den Aufbau von Computern) in der ersten Sitzung von zwei Unterrichtsstunden erarbeitet worden; ein erster Grundwortschatz der Programmiersprache BASIC und der Systemsprache (LIST, RUN) stand zur Verfügung. Der sukzessive Ausbau bis zum Programm MEHRWERTSTEUER-6 erfolgte ohne große Probleme und gab zugleich eine gute Editier-Übung ab. Beim Laufenlassen dieser Programme beobachtete ich immer wieder, daß die Schüler von Anfang an möglichst "wilde", also z. B. möglichst große Zahlen eingaben. Dies vermittelte ihnen zwar ein erhebendes Gefühl für die Arbeitserleichterung durch den Computer, es machte ihnen andererseits aber die Überprüfung der Ergebnisse unmöglich. Das Vertrauen in die fehlerfreie Arbeitsweise des Computers und des Programms war offenbar unerschütterlich.

Meine Frage, welcher Nettopreis einem Eis zum Verkaufspreis von 1 DM zugrunde liege, führte zu unterschiedlichen Reaktionen: Real- und Gymnasialschüler reagierten eher verduzt mit den Worten "Was? - Da muß ich ja probieren!" Den Hauptschülern dagegen schien diese Lösungsmethode weniger anrüchig vorzukommen. Dennoch machte es allen Schülern großen Spaß, und sie ließen nicht locker, bis sie den gesuchten Wert möglichst genau ermittelt hatten. Vom unsystematischen Probieren ausgehend, entwickelten sie bald effektive Einschachtelungs-Strategien. Einige der Schüler waren aber mit dem Probierverfahren immer noch unzufrieden und versuchten, das Mehrwertsteuer-Programm so abzuwandeln, daß der gesuchte Wert durch Division auf direktem Wege ermittelt wurde.

Bereits bei der Behandlung des ersten Programms zur Mehrwertsteuerberechnung fragte ein Schüler: "Woher weiß der Computer, welchen Variablen die Eingabewerte entsprechen, falls einmal mehr als eine Variable durch INPUT zu bestücken sind?" Im Prozentwert-Programm konnte diese Frage aufgegriffen werden. Vielen Schülern wurde erst im Laufe dieser Arbeit klar, daß die Mehrwertsteuerberechnung ein Sonderfall der Prozentwert-Berechnung ist. (Allerdings fragte einer der Schüler dann auch, welcher Größe der Bruttobetrag beim Prozentwert-Problem entspreche).

Obwohl die folgenden Formen der Grundgleichung

```
PW = GW * PS

PW = GW * P / 100        (P = 100 * PS)
```

beide behandelt worden waren, kam es beim Programmlauf häufig vor, daß die Schüler bei der Eingabe von z. B. 14 % einfach 14 an Stelle von 0.14 eingaben.

Durch algebraische Umformung der obigen Grundgleichung, die erfreulicherweise auch mit Hauptschülern gut gelang (selbst Lehrer der Schule hatten dies nicht erwartet), kamen wir zu den anderen Grundaufgaben der Prozentrechnung, deren Kombination dann zu einem "Superprogramm" PROZENTRECHNUNG führte. Das Kommando RUN in Zeile 300 dieses Programms hatte ich zunächst ohne viel Aufhebens eingebracht. Die Beantwortung meiner Frage, wann dieses Kommando ausgeführt würde, bereitete den Schülern erhebliche Schwierigkeiten.
Sie hatten angenommen, daß der Computer Eingabefehler selbständig erkenne und schon "richtig" reagieren werde. Deshalb waren sie erstaunt darüber, daß Maßnahmen zur Behebung

von Eingabefehlern im Algorithmus selbst berücksichtigt werden müssen. Nachdem sie das erkannt hatten, brachten sie dann aber saftige Fehlermeldungen an Stelle des kargen RUN-Kommandos aus.

Das Prozentrechnungs-Programm stellte auch eine gute Gelegenheit dar, die Schüler mit der Standard-Technik der Menu-Tafeln bekannt zu machen.

Bei der Bearbeitung des Rechnungsschreibungs-Problems fiel den Schülern die Formulierung von Anforderungen an ein verbessertes Programm (vgl. Seite 20) relativ schwer. Die Berücksichtigung des Wunsches (a) gab Anlaß für eine etwas systematischere Diskussion der Variablentypen und -namen. Einer der Schüler stellte spontan die Frage nach der Anzahl aller möglichen Variablennamen.

Das Programm POSTLEITZAHLEN-1 hatten wir (allerdings mit weniger als 12 Städten) tatsächlich in der umständlicheren Form von Seite 22 unten formuliert:

```
IF A$(1) = N$ THEN PRINT B(1)
IF A$(2) = N$ THEN PRINT B(2)
IF A$(3) = N$ THEN PRINT B(3)      (*)
IF A$(4) = N$ THEN PRINT B(4)
IF A$(5) = N$ THEN PRINT B(5)
```

Bei allen Schülergruppen fing es dabei fast immer irgendwie zu "knistern" an. Ein Schüler meinte z.B.: "Hier müßte man doch so etwas wie ein X verwenden können". In einer anderen Gruppe schlug ein Schüler vor, daß man die Frage nur einmal stellen und dann "irgendwie" immer wieder zu ihr zurückspringen sollte. Als ich dann die auf Seite 22 gegebene Lösung vorstellte, reagierte er spontan: "Ja! - Genau so habe ich es gemeint; nur konnte ich es nicht in der Computersprache ausdrücken." Besonders erfreut war ich darüber, daß ich auch in einer Gruppe, die vorwiegend aus Hauptschülern bestand, auf die Frage, wie die "allgemeine" Zeile im Schema (*) lauten müßte, spontan die Antwort

```
IF A$X = N$ THEN PRINT BX
```

erhielt. Es waren nur noch Klammern um die Laufvariable X zu setzen (was wegen der bekannten Funktionsschreibweise als ganz natürlich akzeptiert wurde), und der Kern der Lösung war erarbeitet. Am Programm POSTLEITZAHLEN-1 wird deutlich, daß es immer wieder schwerfällt, das optionale LET bei Wertzuweisungen in BASIC tatsächlich aufzuschreiben. Was syntak-

tisch und funktional nicht zwingend erforderlich ist, gerät erfahrungsgemäß in Gefahr, unter den Tisch zu fallen.

Eine BASIC-spezifische Lösung mit Hilfe der DATA-Anweisung, wie sie nicht selten in der Literatur anzutreffen ist, habe ich wegen der fehlenden Übertragbarkeit auf andere Programmiersprachen bewußt vermieden.

Bei dem Test auf "Vorhandensein des Suchbegriffs" im Programm POSTLEITZAHLEN-2 waren die Schüler immer wieder versucht, die Abfrage auf "D=0" in die Schleife zu verlegen. Hier hilft z. B. ein Papier-und-Bleistift-Lauf oder konkretes Ausprobieren, um den Irrtum zu erkennen.

Im Programm RECHNUNGSSCHREIBUNG-3 zeigten sich langsam die fehlenden Gliederungsmöglichkeiten der Programmiersprache BASIC auch für die Schüler. Die gemeinsame Erstellung eines Lösungsrahmens führte zur Formulierung der folgenden übergeordneten Ziele:

1. Eingaberoutine mit Suchroutine für den Einzelpreis

2. Rechnungskopf

3. Ausgabe der Rechnungspositionen

4. Ermittlung der Rechnungssumme (Nettobetrag)

5. Ermittlung und Ausgabe der Rechnungsendbeträge

6. Zuweisung der Artikel-Preis-Tabelle

Hierbei zeigte sich, daß besonders die Hauptschüler größere Probleme hatten, solche übergeordneten Teilaufgaben zu formulieren. In der endgültigen Fassung des Programms RECHNUGSSCHREIBUNG-3 wurden diese Teilaufgaben dann auf der ausgedruckten Programmliste eingerahmt und bezeichnet:

```
Teilaufgabe 1.: Zeile 120-250
                (dabei: Tabellensuchen: Zeile 170-210)
            2.: Zeile 260-310
            3.  und 4.: Zeile 320-360
            5.: Zeile 370-440
            6.: Zeile 450-760.
```

Dies kann jedoch nur als Notbehelf gewertet werden, da alle Dokumentationen, die nicht in der Programmliste selbst stehen und dort nach Möglichkeit eine Funktion im Programmab-

lauf erfüllen, erfahrungsgemäß in kürzester Zeit überholt und veraltet sind. Da es sich bei der Arbeit an den verschiedensten Computersystemen erwiesen hat, daß parallele Dokumentationen praktisch nie konsequent auf den neuesten Stand gebracht werden, muß man m. E. davon ausgehen, daß die einzig gültige Dokumentation eines Programmes der Programmtext selbst ist. Wenn man diese Einsicht akzeptiert, so führt dies allerdings zu der Forderung nach besserer logisch-sachlicher, syntaktischer und auch optischer Gliederbarkeit von Programmen, als es bei den meisten BASIC-Versionen möglich ist.

Im Hinblick auf die Zugehörigkeit der Schüler zu den einzelnen Schulzweigen sei zunächst einmal erwähnt, daß es keine reinen schulzweigspezifischen Gruppen gab. Dennoch bildeten sich bestimmte Schwerpunkte heraus, so daß es im Laufe der Zeit zu je einer Gruppe kam, in der schwerpunktmäßig die Schüler je eines Schulzweiges vertreten waren.

Im Laufe des Projekts konnte ich beobachten, daß die Schüler des Gymnasialzweiges am meisten auf die Behandlung möglichst raffinierter Spiele drängten, während die Haupt- und Realschüler die Behandlung wirtschaftlich orientierter Probleme begrüßten. Hier kam wohl der unterschiedliche Abstand der Schüler vom bevorstehenden Berufsleben zum Ausdruck. Die Haupt- und Realschüler erkannten durchaus einen in der Arbeitsgemeinschaft liegenden berufspraktischen Effekt; die Schüler des Gymnasialzweiges reizte eher die spielerisch/sportliche Komponente. Sie waren auch sehr rührig im Auffinden von allen möglichen Tricks (wie z. B. die Ersetzbarkeit des Kommandos RUN durch R shift U), die sie dann als eine Art "Geheimsprache für Eingeweihte" benutzten. Genau derartige Belanglosigkeiten wurden dann von den anderen Schülern am schnellsten und sichersten aufgenommen und verarbeitet.

In der hauptschulzentrierten Gruppe waren verhältnismäßig viele ausländische Schüler vertreten, von denen sehr viele gute Anregungen kamen. Es war offensichtlich, daß sie die bei der Arbeit mit dem Computer verringerten sprachlichen Benachteiligungen zu besonders intensiver Mitarbeit beflügelten.

Formale "objektive" Tests habe ich nicht durchgeführt. Erstens wurde die dafür notwendige Zeit durch die Behandlung weiterer Probleme besser genutzt, zweitens sollte der erreichte Freiraum nicht sogleich wieder zunichte gemacht werden, drittens wären in einem derartigen Test (wie in vielen

solchen Tests) nur die belangloseren Aspekte des Projekts testbar, und viertens bezweifle ich, daß die Mehrzahl der "objektiven" Tests objektivere Einsichten in relevante Sachverhalte vermittelt, als es die kontext- und situationsgebundene Beobachtung des Unterrichts mit offenen Augen ermöglicht.

Abschließend sei erwähnt, daß im Gegensatz zu häufig auch im Kreise der Lehrerschaft geäußerten Vermutungen die englischsprachigen Schlüsselwörter praktisch kein Problem darstellten.

Den insgesamt nachhaltigsten Eindruck auf mich machten die außerordentlich große Einsatzbereitschaft und der große Arbeitswille der Schüler. Ich bin der Auffassung, daß ein sehr hoher Stimulationsgrad auf die unmittelbare Rückkopplung zwischen der geleisteten Arbeit (Programmentwurf, Programmierung) und dem daraus entstandenen Produkt (Programmlauf, Auswertung) zurückzuführen ist. Dies spricht m. E. für die Verwendung interaktiver Systeme im Mittelstufenunterricht.

Durch diese unmittelbare Rückkopplung zwischen ausgeübter Tätigkeit und entstandenem Produkt gewinnt das Arbeiten mit Computern einen geradezu handwerklichen Charakter. Ähnlich wie an einem Stück Holz kann man auch an einem Programm "schnitzen". Gerade dieser praxisorientierte, handwerkliche Arbeitsstil kommt der intellektuellen Disposition besonders von Haupt- und Realschülern mehr entgegen, als es einem an abstrakten Begrifflichkeiten orientierten Unterricht möglich ist.

Ich bin der Auffassung, daß das aktive Arbeiten mit Computern nicht etwa nur eine Domäne der gymnasialen Oberstufe sein sollte, sondern daß es schon in den Mittelstufenunterricht der allgemeinbildenden Schulen gehört.

5.2 Bemerkungen zum unterschiedlichen Stellenwert der algorithmischen Methode

Die atemberaubende Entwicklung im Bereiche der Computerindustrie, deren Geräte innerhalb kürzester Zeit um ganze Größenordnungen billiger, leistungsfähiger und bedienungsfreundlicher wurden, ermöglichte es, einem jahrtausendealten Anliegen der Mathematik, nämlich der algorithmischen Denk-

und Arbeitsweise, im Mathematikunterricht wieder verstärkt Aufmerksamkeit zu schenken. Einer der ersten mathematischen Algorithmen, der zugleich von paradigmatischer Qualität für den Algorithmusbegriff überhaupt ist, war das von Euklid vorgestellte Verfahren zur Bestimmung des größten gemeinsamen Teilers zweier natürlicher Zahlen, der "Euklidische Algorithmus". (Diese Bemerkung zeigt übrigens auch, daß Algorithmen eine uralte Domäne der Mathematik sind und nicht erst - wie von einigen Informatikern gern unterstellt - von der Informatik "erfunden" wurden.)

Algorithmen spielten im traditionellen Mathematikunterricht eine große, jedoch gelegentlich etwas zwielichtige Rolle. Die Behandlung von Algorithmen umfaßt nämlich die beiden folgenden Aspekte:

- die Konstruktion, den Entwurf, die "Architektur" von (neuen) algorithmischen Problemlösungen;

- das Ausführen von (vorgegebenen) Algorithmen.

Algorithmen werden von "Prozessoren" ausgeführt. Computer sind außerordentlich schnelle und zuverlässige Prozessoren. Da man im traditionellen Mathematikunterricht (abgesehen von einigen Geräten mit sehr eingeschränkten Möglichkeiten) über keine Universalprozessoren (Computer) verfügte, war man gezwungen, die Algorithmen zum großen Teil manuell auszuführen. Einen der wichtigsten algorithmischen Grundbausteine stellt die Wiederholung (Schleifenbildung) dar. Die manuelle Aufführung von Algorithmen - insbesondere solcher mit Wiederholungselementen - ist im allgemeinen äußerst zeitaufwendig und wenig inspirierend. Der Gefahr, daß man sich im Rahmen eines zeitlich beschränkten Mathematikunterrichts zu sehr auf die "Ausführung von Algorithmen" und zu wenig auf die "Konstruktion von Algorithmen" konzentrierte, wurde häufig nicht energisch genug entgegengewirkt.

Diese "kalkülhafte" Komponente erdrückte somit häufig den konstruktiven bzw. problemorientierten Aspekt von Algorithmen. Sie führte bei den Leuten, die den Algorithmusbegriff ausschließlich durch seinen kalkülhaften Aspekt mißdeuteten, zu einem gewissen Verruf und einer Geringschätzung von Algorithmen.

Natürlich stellt die Verfolgung des Unterrichtszieles "Entwurf von Algorithmen" erheblich höhere Anforderungen an den Unterrichtenden als die Einübung des Abspulens von Algorithmen.

Der Computer befreite seine Benutzer davon, sich zeitlich über Gebühr mit der Ausführung von Algorithmen beschäftigen zu müssen, und schuf dadurch insbesondere auch im Mathematikunterricht die Möglichkeit, der Konstruktion von Algorithmen den ihr gebührenden Stellenwert einzuräumen. Diese Ziele verfolgt insbesondere der computerorientierte Unterricht - nicht zu verwechseln mit Entwicklungen, in denen versucht wurde, Elemente der Lehrtätigkeit des Lehrers auf den Computer zu übertragen und die unter den Bezeichnungen "programmierter Unterricht" bzw. "computerunterstützter Unterricht" (CUU) liefen.

Es bleibt festzuhalten, daß durch die verstärkte Berücksichtigung der (im vollen und nicht etwa im mißdeuteten Sinne verstandenen) algorithmischen Komponente als Ausgleich zu begrifflich/strukturellen Verfahren wieder ein größeres Maß an Ausgewogenheit im Mathematikunterricht erreicht wird, als dies in den letzten Jahren der Fall war. Dies sei im folgenden in bezug auf inhaltliche und methodologische Aspekte etwas eingehender diskutiert. Obwohl es nicht immer möglich ist, inhaltliche und methodologische Aspekte sauber zu trennen, wird versucht, die Diskussion durch diese Grobeinteilung etwas zu strukturieren.

(A) Inhaltliche Aspekte der Algorithmik im Unterricht

Inhaltliche Schwerpunkte der Algorithmik sehe ich vorrangig in den folgenden Bereichen:

- Mathematisierung
- das Arbeiten mit Modellen
- Simulationen
- Anwendungsoriertheit / Berufs- und Praxisnähe
- Integration

Ein vorrangiges Ziel des Mathematikunterrichts auf ziemlich hoher Stufe jeder Lernzielhierarchie ist das der Mathematisierung. Ihr liegt implizit oder explizit praktisch immer ein Denken in und Arbeiten mit Modellen zugrunde. In den klassischen Anwendungsfeldern des traditionellen Mathematik-

unterrichts der Mittelstufe, etwa im Sachrechnen oder in der Physik, waren die zugrundeliegenden mathematischen Modelle meist kanonischer Natur, so daß dem Prozeß der Modellbildung vergleichsweise wenig Aufmerksamkeit geschenkt wurde. Erst durch die Einbringung nichtklassischer Gebiete wie z.B. Stochastik und Algorithmik in den Unterricht, sowie durch eine verbreiterte Anwendungsorientierung trat der Modellentwurf stärker in den Vordergrund.

Sehr viele mathematischen Modelle in den Natur-, Wirtschafts- und Sozialwissenschaften sind rekursiver Natur. In den meisten Fällen sind die Modellansätze nicht in geschlossen/analytischer Form, sondern nur durch konkrete, iterative "Hochrechnung" von Anfangswerten auswertbar. Ohne Computer und entsprechende Algorithmen ist diese Arbeit außerordentlich mühselig und im unterrichtlichen Rahmen praktisch nicht durchführbar. Durch die Verwendung von Computern wird dieser hochinteressante und anwendungsträchtige Themenkreis dem Mittelstufenunterricht mit elementarsten mathematischen Methoden (siehe Stichwort: Vereinfachung) zugänglich gemacht.

Dies gilt einerseits für deterministische Modelle, wie z. B. Modelle für modifiziertes exponentielles Wachstum, und andererseits in ganz besonderem Maße für stochastische Modelle, deren Verhalten praktisch nur noch in Form von gut variierten Computer-Simulationen analysierbar ist. Genau dies sind zugleich die Verfahren, die in den realen Anwendungen der Mathematik verstärkt vorkommen. Es erscheint mir unbestreitbar, daß dieses durch Computereinsatz ermöglichte Arbeiten mit Modellen den Anwendungs- und Praxisbezug des Mathematikunterrichts außerordentlich erhöht. (Die Tatsache, daß die im Mittelstufenunterricht behandelbaren Grundmodelle meist sehr einfach und für reale Anwendungen im Detail verbesserungsbedürftig sind, beeinträchtigt die obige Aussage nicht im geringsten).

Sehr viele Simulationsmodelle sind aus naturwissenschaftlichen Fragestellungen abgeleitet oder dem wirtschafts- und sozialwissenschaftlichen Bereiche entlehnt. Hier besteht die Chance, einer möglichen Isolierung des Matematikunterrichts entgegenzuwirken und ihn wieder stärker mit anderen Schulfächern (Biologie, Physik, Chemie, Wirtschaftslehre) zu verzahnen. Algorithmen und Computer bergen also beträchtliche Chancen zur Integration von Unterrichtsfächern und zur Einbeziehung unkonventioneller, aber dennoch bedeutsamer Probleme in sich.

In bezug auf die inhaltliche Diskussion sei abschließend noch auf die Möglichkeit der Eingliederung algorithmischer Entwicklungsstränge in den Mathematikunterricht im Sinne eines spiralförmigen Aufbaus hingewiesen. Besonders gut algorithmisch erschließbare Entwicklungslinien stellen im Mittelstufenunterricht z. B. die folgenden (auch miteinander verflechtbaren) Bereiche dar:

- Teilbarkeitslehre / Bruchrechnen / Gleichungslehre
- Sachrechnen / Funktionenlehre / Wachstumsprozesse
- Stochastik / Simulationen

(B) Methodologische Aspekte des Arbeitens mit Computern

Aus methodologischer Sicht stellen die folgenden Aspekte wesentliche, durch das Algorithmieren und das Arbeiten mit Computern geförderte Zielsetzungen dar:

- empirische Fundierung des Lernprozesses
- Elementarisierung / Vereinfachung
- operatives, "parametrisches" Lernen und Arbeiten
- modulares Arbeiten als heuristische Grundtechnik
- konstruktives Arbeiten
- Vertiefung des Variablenverständnisses.

Diese Stichworte seien im folgenden näher erläutert.

(a) Empirische Fundierung des Lernprozesses:

Ein ziemlich unumstrittener Grundsatz des Mathematikunterrichts besagt, daß mathematische Begriffe und Verfahren an sorgfältig ausgesuchten Beispielen vorbereitet, entwickelt und erläutert werden sollten. Es kommt jedoch nicht nur auf die Anzahl der behandelten Beispiele, sondern auch auf ihre Typizität, ihren paradigmatischen Charakter an. Ein schlagendes Beispiel ist oft geeignet, den Sinn einer schwer ein-

gängigen abstrakten Definition oder eines schwer zu fassenden mathematischen Sachverhalts zu erhellen.

Da das Arbeiten mit Computern von stupider Rechenarbeit entlastet, kann mehr Gewicht auf die Festigung der kognitiven Basis durch gut variiertes Beispielmaterial gelegt werden. Auf die Vermeidung krummer Zahlen oder nichtlinearer Modelle braucht aus rechentechnischen Gründen keine Rücksicht mehr genommen werden. Besonders nichtlineare Funktionen und Wachstumsprozesse sind durch aussagekräftige Wertetafeln und Diagramme besser illustrierbar.

Das Verhalten mathematischer Modelle ist anhand sehr unterschiedlicher Anfangs- und Rahmenbedingungen studierbar. Dieses beispielgebundene Arbeiten kommt den Schülern aller Erfahrung nach grundsätzlich, ganz besonders aber in Einführungsphasen zugute. Sicherheit im jeweiligen Kontext wird meist erst durch die Verarbeitung eines hinreichend großen und gut variierten Beispielmaterials bewirkt.

Erste empirische Erfahrungen sind meist auch der Ausgangspunkt im - häufig zyklisch bzw. spiralförmig verlaufenden - Prozeß des Problemlösens. In Abb.5.1 wurde versucht, diesen "heuristischen Zyklus" in graphischer Form darzustellen.

(b) Elementarisierung / Vereinfachung

Durch die Technik des iterativen Hochrechnens mathematischer Modelle und Prozesse lassen sich Fragestellungen, die ohne Computereinsatz nur mit relativ hohen theoretischen Vorkenntnissen lösbar sind, derart vereinfachen, daß ihre Bearbeitung praktisch nur noch die Grundrechenarten, gelegentlich etwas Prozentrechnung und elementarste (in ca. 1 Stunde zu vermittelnde) Kentnisse über das Arbeiten mit Computern voraussetzt. Dies sollte aus den Beispielen von §2,3 und 4 deutlich geworden sein. Darüber hinaus werden Probleme elementar lösbar, die sich einer geschlossenen Lösung (sei es auf Schulniveau, sei es grundsätzlich) entziehen. Als Beispiel hierfür betrachte man etwa die Frage, welcher Zinssatz einem in N Jahren durch eine Annuität A getilgten Kredit K zugrunde liegt.

(c) Operatives, parametrisches Lernen und Arbeiten

Die Grundfrage der operativen Vorgehensweise lautet:

"Was geschieht, wenn ... "

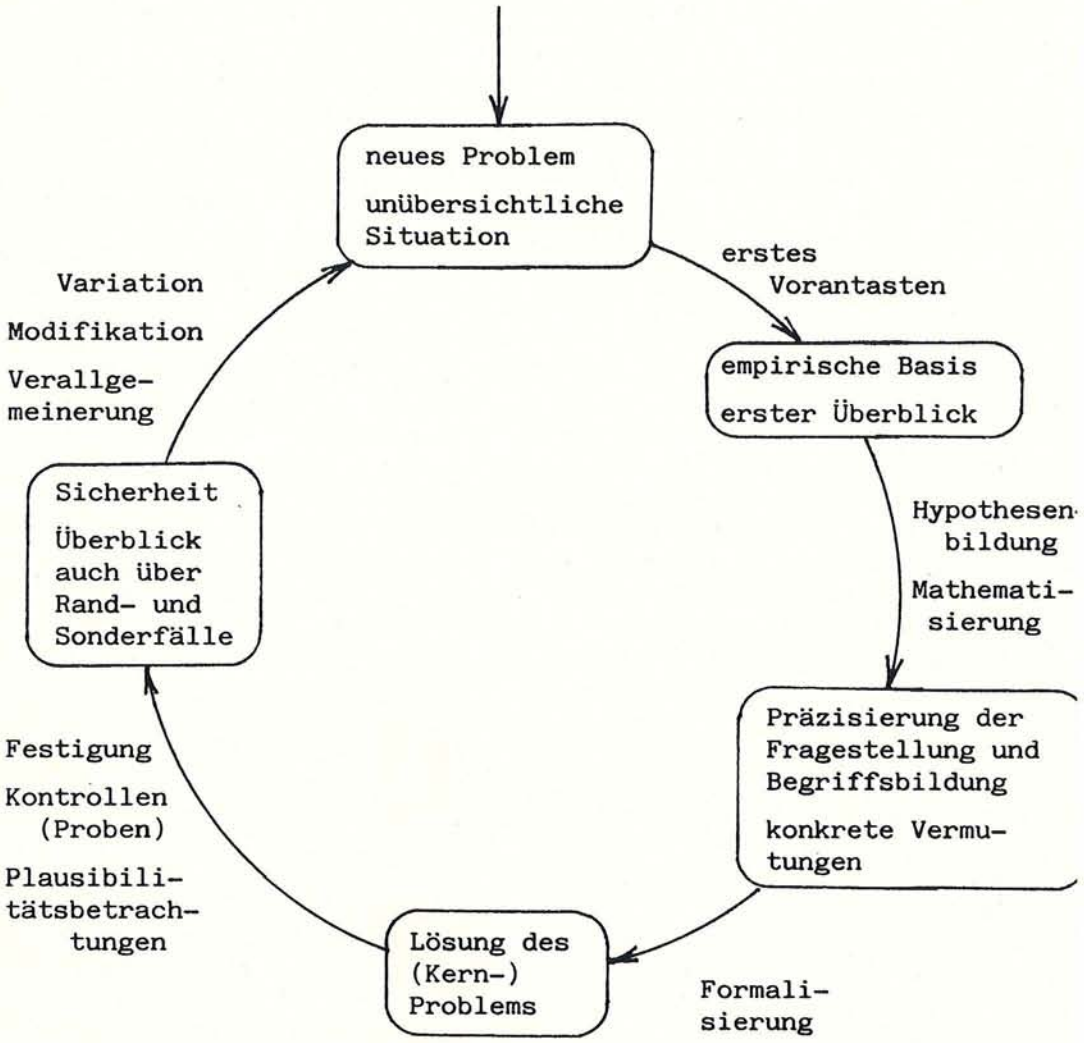


Abb.5.1: "Heuristischer Zyklus"

Gerade im Bereiche der Computersimulationen wird diese Frage in vielfacher Weise gestellt: was geschieht, wenn man die Anfangswerte eines Simulationsproblems, wenn man Rahmenbedingungen und Parameter verändert, ja wenn man das gesamte Modell modifiziert?

Besonders im Zusammenhang mit der empirisch fundierten Lernmethode läßt sich das Variieren der Parameter über flexible Eingabemöglichkeiten am Computer oder über gezielte Parameterdurchlaufschleifen gut realisieren.

Graphische Ausgabemöglichkeiten, mit deren Hilfe die jeweiligen Ergebnisse in bildhaft anschaulicher Form darstellbar sind, erweisen sich bei diesem "parametrischen" Lernen als außerordentlich nützlich.

Bei der Modifikation der Modellbedingungen muß der Benutzer die verwendeten Programme selbständig abändern können. Die klassische Arbeitsweise des CUU, wo der Benutzer nur vorgefertigte Programme laufen lassen kann, scheint mir den Zielen des operativen Lernens nicht gerecht zu werden.

Notwendig erscheint mir die Feststellung, daß die Variation der Parameter stets in kontrollierter Form vorgenommen werden sollte. Durch ein wildes gleichzeitiges "Drehen an allen Knöpfen" wird selten eine Erkenntnis über die Wirkungsweise der verschiedenen Einflußfaktoren zu gewinnen sein. So wird der Schüler z.B. nur dann einen Einblick in die Linearitätsverhältnisse der Prozentrechnung gewinnen, wenn von den drei Größen Grundwert, Prozentwert und Prozentsatz stets nur eine in systematischer Weise variiert und diese Variation mit Hilfe von Schaubildern graphisch umgesetzt wird.

(d) Modulares Arbeiten als heuristische Technik

Die Gliederung eines komplexen Problems in überschaubare, relativ abgeschlossene Teilprobleme, das Lösen der einzelnen Teilprobleme und das Zusammenfügen dieser Einzellösungen zu einer Gesamtlösung des Ausgangsproblems stellt eine heuristische Grundtechnik dar, die durch das Problemlösen mit Hilfe von Computern eine besondere Aktualität gewonnen hat. Hand in Hand mit der Untergliederung eines Problems in Teilprobleme geht natürlich auch die konkrete Benennung der Teilprobleme und ihrer Parameter, also der Aufbau problemadäquater begrifflicher Hierarchien.

Besondere Sorgfalt ist auf die saubere Beschreibung der Schnittstellen zwischen den einzelnen Teilproblemen zu legen. Erst wenn diese einwandfrei zusammenpassen, ist der Aufbau einer Gesamtlösung aus den einzelnen Teillösungen gewährleistet.

In der Terminologie der Informatik werden derartige Lösungen von Teil- bzw. Einzelproblemen als Prozeduren bezeichnet. Zur Verfolgung der hier genannten Ziele wäre natürlich eine Programmiersprache mit Prozedurkonzept und Parameterübergabemechanismen höchst wünschenswert. Ihr Fehlen stellt eine der größten Schwachstellen von BASIC dar.

(e) Konstruktives Arbeiten

Jeder Algorithmus stellt eine konstruktive Lösung einer Aufgabe dar; die algorithmische Problemlösung kann geradezu als Paradigma für Konstruktivität angesehen werden.

Diese konstruktive Vorgehensweise kommt in Verbindung mit der oben erwähnten empirischen Fundierung und mit operativen Arbeitstechniken meiner Erfahrung nach der intellektuellen Disposition der meisten Schulkinder weit mehr entgegen als axiomatisch/deduktive Arbeitsweisen. Dies gilt in besonderem Maße für Haupt- und Realschüler.

Im traditionellen Mathematikunterricht wurde diese konstruktive Vorgehensweise besonders bei den m. E. zu Unrecht in den Hintergrund gedrängten Konstruktionstexten und Planskizzen im Geometrieunterricht praktiziert. Die klare, gut gegliederte sprachliche Beschreibung von Lösungsverfahren wird von den Schülern zwar als lästig empfunden; sie gehört m. E. aber zu den vorrangigen allgemeinen Lernzielen des Mathematikunterrichts.

In Verbindung mit einer gut strukturierbaren, umgangssprachennahen Programmiersprache eröffnet das Arbeiten mit Computern also auch Chancen, die sprachlich-kommunikative Ebene des Mathematikunterrichts wieder stärker zu pflegen.

(f) Vertiefung des Variablenverständnisses

Beim Algorithmieren kommt sowohl der syntaktische als auch der semantische Aspekt des Variablenverständnisses zum Tragen. Der syntaktische Aspekt (Variable als Leerstelle, Platzhalter) ist durch die physische Realisierung von Variablen als Speicherzellen im Computer und mit der Wertzuweisung als Vorgang des Überschreibens gegeben.

Der semantische Aspekt (Variable als Kurzbezeichnung bzw. Bedeutungsträger) spielt in der Phase der Algorithmierung und Programmierung eine große Rolle durch die Festlegung von Variablennamen als Identifikatoren.

Der syntaktische Aspekt der Verwendung von Variablen spielt sich also beim Arbeiten mit dem Computer eher auf der hardware- und maschinenorientierten Ebene, der semantische Aspekt dagegen eher auf der Software-Ebene ab.

Nachdem in der letzten Zeit im Rahmen von Vorschlägen zur größeren begrifflichen Strenge der syntaktische Aspekt des Variablenbegriffs überbetont wurde, kommt der verstärkt semantischen Orientierung beim Vorgang des Algorithmierens und Programmierens eine positive Bedeutung zu.

In bezug auf die Wertzuweisung bleibt zu hoffen, daß sich hier endlich eine Notationsform durchsetzt, die keine Konflikte mit der Gleichungslehre hervorruft. Positivbeispiele in anderen Programmiersprachen sind etwa der "MAKE"-Befehl in LOGO oder das wegen der Kürze der Schreibweise chancenreiche definitorische Gleichheitszeichen ":=" von PASCAL und anderen Sprachen.

ANHANG: PASCAL-PROGRAMME ZU AUSGEWÄHLTEN PROBLEMEN DES UNTERRICHTSPROJEKTS

Im folgenden sind die Pascal-Versionen zu einigen ausgewählten Problemen wiedergegeben. Diese Programme wurden auf einem den Schülern nicht zur Verfügung stehenden Rechner außerhalb der Schule geschrieben. Sie dienten im Unterricht nur zur Demonstration der sachlich/logischen und optischen Gliederungsmöglichkeiten von Pascal. Dabei wurde auf die Diskussion der modularen Vorgehensweise in Verbindung mit dem Prozedur-Konzept besonderer Wert gelegt.

Es war aber nicht mein Ziel und lag auch nicht im Rahmen der Möglichkeiten, eine formale Einführung in Pascal zu geben.

Für ähnliche Verwendungsmöglichkeiten sind diese Pascal-Programme hier aufgenommen worden. Außerdem setzen sich ja höhere Sprachen im Laufe der Zeit auch auf der Ebene der Micro-Computer durch.

Die Pascal-Programme entsprechen den jeweiligen BASIC-Programmen mehr oder weniger direkt. Die stärkste Übereinstimmung liegt bei den Programmen RECHNUNGSSCHREIBUNG, RENTEN-BARWERT-SCHAETZUNG, BUBBLESORT und QUICKSORT vor; die Programme zu den stochastischen Simulationen sind etwas freier übertragen worden.

Persönlich habe ich die Erfahrung gemacht, daß es bei einigen Problemen (so z. B. beim Quicksort) sehr hilfreich war, wenn ich das Programm zunächst in Pascal schrieb und dann in BASIC übertrug. Diese Vorgehensweise war wesentlich weniger fehleranfällig und somit per Saldo auch zeitsparender, als wenn ich diese Programme von vornherein in BASIC geschrieben hätte.

```
PROGRAM RECHNUNGSSCHREIBUNG;   (* MAXIMAL 50 POSITIONEN *)

CONST POSMAXIMUM=50;
      TABELLENMAXIMUM=20;

VAR ARTIKEL: STRING;
    STUECKZAHL,POSZAHL: INTEGER;
    EINZELPREIS,GESAMTPREIS,NETTOBETRAG,
    MEHRWERTSTEUER,BRUTTOBETRAG: REAL;

    ARTIKELSPEICHER: ARRAY[1..POSMAXIMUM] OF STRING;
    STUECKZSPEICHER: ARRAY[1..POSMAXIMUM] OF INTEGER;
    EINZPREISSPEICHER: ARRAY[1..POSMAXIMUM] OF REAL;

    ARTIKELTABELLE: ARRAY[1..TABELLENMAXIMUM] OF STRING;
    PREISTABELLE: ARRAY[1..TABELLENMAXIMUM] OF REAL;

PROCEDURE UEBERSCHRIFT;
  BEGIN
    PAGE(OUTPUT);  (* CLEAR SCREEN UND HOME *)
    WRITELN('RECHNUNGSSCHREIBUNG');  WRITELN;  WRITELN;
  END;

PROCEDURE TABELLEEINLESEN;
  BEGIN
    ARTIKELTABELLE[1]:='STUHL';       PREISTABELLE[1]:=87.50;
    ARTIKELTABELLE[2]:='TISCH';       PREISTABELLE[2]:=245.80;
    ARTIKELTABELLE[3]:='SESSEL';      PREISTABELLE[3]:=324.50;
    ARTIKELTABELLE[4]:='HOCKER';      PREISTABELLE[4]:=24.60;
    ARTIKELTABELLE[5]:='LAMPE';       PREISTABELLE[5]:=65.40;
    ARTIKELTABELLE[6]:='SOFA';        PREISTABELLE[6]:=568.20;
    ARTIKELTABELLE[7]:='KOMMODE';     PREISTABELLE[7]:=240.40;
    ARTIKELTABELLE[8]:='SPIEGEL';     PREISTABELLE[8]:=40.50;
    ARTIKELTABELLE[9]:='GARDEROBE';   PREISTABELLE[9]:=180.40;
    ARTIKELTABELLE[10]:='TEPPICH';    PREISTABELLE[10]:=280.30;
  END;
```

```
(* Programm:  RECHNUNGSSCHREIBUNG   ...   Fortsetzung *)

PROCEDURE PREISERMITTELN;
  VAR J: INTEGER;
      GEFUNDEN: BOOLEAN;
  BEGIN
    GEFUNDEN:=FALSE;
    FOR J:=1 TO 10 DO IF ARTIKEL=ARTIKELTABELLE[J] THEN
      BEGIN
        EINZELPREIS:=PREISTABELLE[J];
        GEFUNDEN:=TRUE;
        WRITELN('   EINZELPREIS: ',EINZELPREIS:10:2);
      END;
    IF NOT GEFUNDEN THEN
      BEGIN
        WRITE('   BITTE EINZELPREIS EINGEBEN: ');
        READLN(EINZELPREIS);
      END;
  END;

PROCEDURE ARTIKELEINGABE;
  BEGIN
    POSZAHL:=0;
    REPEAT
      WRITE('ARTIKEL: '); READLN(ARTIKEL);
      IF NOT (ARTIKEL = '***') THEN
        BEGIN
          POSZAHL:=POSZAHL+1;
          PREISERMITTELN;
          WRITE('   STUECKZAHL: '); READLN(STUECKZAHL);
          ARTIKELSPEICHER[POSZAHL]:=ARTIKEL;
          STUECKZSPEICHER[POSZAHL]:=STUECKZAHL;
          EINZPREISSPEICHER[POSZAHL]:=EINZELPREIS;
        END;
    UNTIL ARTIKEL='***';
  END;

PROCEDURE KOPFSCHREIBEN;
  BEGIN
    WRITELN; WRITELN; WRITELN; WRITELN; WRITELN;
    WRITELN('          R E C H N U N G');  WRITELN;
    WRITELN('ARTIKEL','STUECK-':11,'EINZEL-':11,'GESAMT-':11)
    WRITELN('ZAHL':16,'PREIS':12,'PREIS':11);
    WRITELN; WRITELN;
  END;
```

```
(* Programm:  RECHNUNGSSCHREIBUNG   ...   Fortsetzung *)

PROCEDURE POSITIONENAUSGABE;
  VAR I,L: INTEGER;
  BEGIN
    NETTOBETRAG:=0;
    FOR I:=1 TO POSZAHL DO
      BEGIN
        GESAMTPREIS:=EINZPREISSPEICHER[I]*STUECKZSPEICHER[I];
        NETTOBETRAG:=NETTOBETRAG+GESAMTPREIS;
        ARTIKEL:=ARTIKELSPEICHER[I];
        L:=LENGTH(ARTIKEL);
        WRITELN(ARTIKEL, STUECKZSPEICHER[I]:15-L,
                EINZPREISSPEICHER[I]:13:2,GESAMTPREIS:11:2);
        (* RECHTSBUENDIG FORMATIERTE AUSGABE *)
      END;
  END;

PROCEDURE SUMMENAUSGABE;
  BEGIN
    WRITELN;
    WRITELN('NETTOBETRAG: ':27,NETTOBETRAG:12:2);
    MEHRWERTSTEUER:=NETTOBETRAG*0.13;
    WRITELN('MEHRWERTSTEUER: ':27,MEHRWERTSTEUER:12:2);
    BRUTTOBETRAG:=NETTOBETRAG+MEHRWERTSTEUER;
    WRITELN('BRUTTOBETRAG: ':27,BRUTTOBETRAG:12:2);
  END;

BEGIN   (* HAUPTPROGRAMM *)
  UEBERSCHRIFT;
  TABELLEEINLESEN;
  ARTIKELEINGABE;
  KOPFSCHREIBEN;
  POSITIONENAUSGABE;
  SUMMENAUSGABE
END.
```

```
PROGRAM RENTEN_BARWERT_SCHAETZUNG;
VAR R,P,Q,K,KE: REAL;
    L,I: INTEGER;
    genau_genug: BOOLEAN;
PROCEDURE Parameter_einlesen;
  BEGIN
    PAGE(OUTPUT);
    WRITELN('Rentenrechnung mit Barwert');
    WRITELN('   *** nachschuessig ***'); WRITELN;
    WRITE('Rentenbetrag:  R = '); READLN(R);
    WRITE('Zinssatz (in %):   P = '); READLN(P);
    WRITE('Laufzeit:  L = '); READLN(L);
  END;
PROCEDURE Rentenendbetrag_ermitteln;
  BEGIN
    K:=0;
    Q:=(1+P/100);
    FOR I:=1 TO L DO
        BEGIN
          K:=K*Q+R;
          WRITELN(I:4, K:10:2);
        END;
    KE:=K;
    WRITELN('Renten-Endwert: ',KE:10:2);
  END;
PROCEDURE Barwert_schaetzen;
  VAR B: REAL;
  BEGIN
    WRITE('Ihre Barwert-Schaetzung:  B = '); READLN(B);
    K:=B;
    FOR I:=1 TO L DO
        BEGIN
          K:=Q*K;
          WRITELN(I:4, K:10:2);
        END;
    IF ABS(KE-K) < (0.01 * KE) THEN genau_genug:=TRUE
                               ELSE genau_genug:=FALSE;
  END;
BEGIN   (* Hauptprogramm *)
  Parameter_einlesen;
  Rentenendbetrag_ermitteln;
  REPEAT
    Barwert_schaetzen;
    WRITELN('Renten-Endwert: ',KE:10:2);
  UNTIL genau_genug;
END.
```

```
PROGRAM WARTESCHLANGEN;

USES APPLESTUFF;

VAR AW,BW,P,Q,DS: REAL;
    L,I,MAXLAENGE,SCHLALAENGE,
    SCHLANGENSUMME,ZUGANG,ABGANG: INTEGER;

BEGIN
  PAGE(OUTPUT);
  WRITELN('WARTESCHLANGEN');
  WRITELN;
  WRITELN('BEDIENUNGSWAHRSCHEINLICHKEIT');
  WRITE('   PRO ZEITEINHEIT:  BW = '); READLN(BW);
  WRITELN('ANKUNFTSWAHRSCHEINLICHKEIT');
  WRITE('   PRO ZEITEINHEIT:  AW = '); READLN(AW);
  WRITE('MAXIMALE SCHLANGENLAENGE: '); READLN(MAXLAENGE);
  WRITE('ANZAHL DER LAEUFE:  L = '); READLN(L);
  WRITELN;
  SCHLANGENSUMME:=0;
  SCHLALAENGE:=0;
  RANDOMIZE;
  WRITELN('ZEITEINH. ABGANG  ZUGANG  SCHLANGE');
  FOR I:=1 TO L DO
    BEGIN
      ABGANG:=0;
      ZUGANG:=0;
      Q:=RANDOM/32767;
      IF (Q <= BW) AND (SCHLALAENGE > 0)  THEN ABGANG:=1;
      SCHLALAENGE:=SCHLALAENGE-ABGANG;
      P:=RANDOM/32767;
      IF (P <= AW) AND (SCHLALAENGE < MAXLAENGE) THEN ZUGANG:=1;
      SCHLALAENGE := SCHLALAENGE + ZUGANG;
      SCHLANGENSUMME := SCHLANGENSUMME + SCHLALAENGE;
      WRITELN(I:5,ABGANG:10,ZUGANG:7,SCHLALAENGE:10);
    END;
  DS:=SCHLANGENSUMME/L;
  WRITELN;
  WRITELN('DURCHSCHNITTLICHE SCHLANGENLAENGE: ',DS:8:4);
END.
```

```
PROGRAM MASCHINENAUSFAELLE;

USES APPLESTUFF;

CONST MASCHZAHL=4;

VAR AUSFALL: ARRAY[1..MASCHZAHL] OF INTEGER;
    AUSFALLTAGE: ARRAY[0..MASCHZAHL] OF INTEGER;
    AW,P: ARRAY[1..MASCHZAHL] OF REAL;
    L,TAG,GESAMTAUSFALL,I: INTEGER;

PROCEDURE ZEILENAUSDRUCK;
  BEGIN
    IF TAG MOD 18 = 1 THEN
      BEGIN
        WRITELN;
        WRITELN('*************************************');
        WRITELN('*TAG * M1 * M2 * M3 * M4 * GES.AUSF.*');
        WRITELN('*************************************');
        WRITELN;
      END;
    WRITE(TAG:4);
    FOR I:=1 TO MASCHZAHL DO  WRITE(AUSFALL[I]:5);
    WRITELN(GESAMTAUSFALL:9);
  END;

BEGIN   (*HAUPTPROGRAMM *)
  PAGE(OUTPUT);
  WRITELN('AUSFALL VON MASCHINEN  -  SIMULATION');  WRITELN;
  WRITELN('AUSFALLWAHRSCHEINLICHKEITEN:');
  FOR I:=1 TO MASCHZAHL DO
    BEGIN
      WRITE('   MASCHINE NR. ',I:2,'   AW[',I:1,'] = ');
      READLN(AW[I]);
    END;
  WRITELN;
  WRITE('LAUFZEIT:   L = ');  READLN(L);
  FOR I:=0 TO MASCHZAHL DO AUSFALLTAGE[I]:=0;
  RANDOMIZE;
```

```
(* Programm:  MASCHINENAUSFAELLE    ...   Fortsetzung *)

  FOR TAG:=1 TO L DO
    BEGIN
      GESAMTAUSFALL:=0;
      FOR I:=1 TO MASCHZAHL DO
        BEGIN
          P[I]:=RANDOM/32767;
          IF P[I] <= AW[I] THEN AUSFALL[I]:=1
            ELSE AUSFALL[I]:=0;
          GESAMTAUSFALL:=GESAMTAUSFALL + AUSFALL[I];
        END;
      AUSFALLTAGE[GESAMTAUSFALL]:=AUSFALLTAGE[GESAMTAUSFALL] + 1;
      ZEILENAUSDRUCK;
    END;
  WRITELN;
  WRITELN('ZUSAMMENSTELLUNG DER AUSFALLTAGE: ');  WRITELN;
  FOR I:=0 TO MASCHZAHL DO
    WRITELN(I:1,' MASCHINE(N) AUSGEFALLEN AN ',
                  AUSFALLTAGE[I]:4,'  TAGEN');
END.
```

```
PROGRAM SAMMLERPROBLEM;

(* VERSION:    WUERFEL MIT 6 SEITEN
               MEHRERE LAEUFE
               MIT VERTEILUNG DER WARTEZEITEN *)

USES APPLESTUFF;

VAR A: ARRAY[1..6] OF INTEGER;
    B: ARRAY[6..100] OF INTEGER;
    L,R,I,J,WARTEZEIT,SW,
    WMIN,WMAX,IHAEUF,WHAEUF: INTEGER;
    MW: REAL;
    KOMPLETT: BOOLEAN;

PROCEDURE VOLLSTAENDIGKEITSTEST;
  BEGIN
    KOMPLETT := TRUE;
    FOR I:=1 TO 6 DO
      IF A[I]=0 THEN KOMPLETT := FALSE;
  END;

BEGIN
  PAGE(OUTPUT);
  WRITELN('SAMMLERPROBLEM');  WRITELN;
  WRITE('ANZAHL DER LAEUFE:  L = ');  READLN(L);
  WRITELN;
  SW:=0;
  FOR I:=6 TO 100 DO B[I]:=0;
  WMAX:=0;
  WMIN:=999;
  IHAEUF:=0;
  WHAEUF:=0;
  RANDOMIZE;
```

```
(* Programm:  SAMMLERPROBLEM    ...    Fortsetzung *)

  FOR J:=1 TO L DO
    BEGIN
      KOMPLETT := FALSE;
      FOR I:=1 TO 6 DO A[I]:=0;
      WARTEZEIT:=0;
      WHILE NOT KOMPLETT DO
        BEGIN
          R:=(RANDOM MOD 6) + 1;
          A[R]:=A[R] + 1;
          WARTEZEIT:=WARTEZEIT + 1;
          VOLLSTAENDIGKEITSTEST;
        END;
      WRITELN('LAUF NR.: ',J:4,'  WARTEZEIT: ',WARTEZEIT:8);
      SW := SW + WARTEZEIT;
      IF WARTEZEIT < WMIN THEN WMIN := WARTEZEIT;
      IF WARTEZEIT > WMAX THEN WMAX := WARTEZEIT;
      B[WARTEZEIT] := B[WARTEZEIT] + 1;
    END;
  MW:=SW/L;
  WRITELN;  WRITELN('VERTEILUNG DER WARTEZEITEN');  WRITELN;
  FOR I:=6 TO 100 DO
    BEGIN
      IF NOT (B[I] = 0) THEN
         WRITELN('WARTEZEIT: ',I:4,'  ',B[I]:6,'-MAL');
      IF B[I] > WHAEUF THEN
        BEGIN
           IHAEUF := I;
           WHAEUF := B[I];
        END;
    END;
  WRITELN;
  WRITELN('MINIMALE WARTEZEIT: ',WMIN:4);
  WRITELN('MAXIMALE WARTEZEIT: ',WMAX:4);  WRITELN;
  WRITELN('MITTLERE WARTEZEIT: ',MW:8:4,' ',MW);
  WRITELN('HAEUFIGSTE WARTEZEIT: ',IHAEUF:4);
  WRITELN('HAEUFIGKEIT IHRES AUFTRETENS: ',WHAEUF:4);
END.
```

```
PROGRAM QUADRATISCHE_GLEICHUNG;

USES TRANSCEND;

VAR A,B,C,D,X1R,X2R,X1I,X2I: REAL;
    LINFALL, UNLOESB, ALLGEMG, DOPPELFALL, ZWEIREFALL,
    KOMPFALL: BOOLEAN;

PROCEDURE LINEAR(B,C: REAL);
  BEGIN
    LINFALL := TRUE;
    IF NOT (B=0) THEN  X1R := -C/B
       ELSE IF NOT (C=0) THEN UNLOESB := TRUE
                ELSE ALLGEMG := TRUE;
  END;

PROCEDURE DOPPELLOES;
  BEGIN
    DOPPELFALL := TRUE;
    X1R := -B/(2*A);
    X2R := -B/(2*A);
  END;

PROCEDURE ZWEIREELL;
  BEGIN
    ZWEIREFALL := TRUE;
    X1R := (-B + SQRT(D))/(2*A);
    X2R := (-B - SQRT(D))/(2*A);
  END;

PROCEDURE KOMPLEX;
  BEGIN
    KOMPFALL := TRUE;
    X1R := -B/(2*A);
    X2R := -B/(2*A);
    X1I := SQRT(-D)/(2*A);
    X2I := -SQRT(-D)/(2*A);
  END;
```

```
(* Programm:  QUADRATISCHE_GLEICHUNG   ...   Fortsetzung *)

PROCEDURE QUADLOES(A,B,C: REAL);
  BEGIN
    LINFALL := FALSE;
    UNLOESB := FALSE;
    ALLGEMG := FALSE;
    DOPPELFALL := FALSE;
    ZWEIREFALL := FALSE;
    KOMPFALL := FALSE;
    IF A=0 THEN LINEAR(B,C)
    ELSE
       BEGIN
         D := B*B - 4*A*C;
         IF D=0 THEN DOPPELLOES
            ELSE IF  D > 0  THEN ZWEIREELL
               ELSE KOMPLEX;
       END;
  END;

PROCEDURE QUADEING;
  BEGIN
    PAGE (OUTPUT);
    WRITELN('QUADRATISCHE GLEICHUNG'); WRITELN;
    WRITELN('KOEFFIZIENTENEINGABE'); WRITELN;
    WRITE('KOEFFIZIENT VON X*X:   A = '); READLN(A);
    WRITE('KOEFFIZIENT VON X:     B = '); READLN(B);
    WRITE('ABSOLUTGLIED:          C = '); READLN(C);
  END;
```

```
(* Programm:  QUADRATISCHE_GLEICHUNG   ...   Fortsetzung *)

PROCEDURE QUADAUSG;
  BEGIN
    WRITELN;  WRITELN;
    WRITELN('DIE QUADRATISCHE GLEICHUNG');
    WRITELN('     MIT DEN KOEFFIZIENTEN:');  WRITELN;
    WRITELN('     A = ',A:5:2);
    WRITELN('     B = ',B:5:2);
    WRITELN('     C = ',C:5:2);  WRITELN;
    WRITELN('HAT DIE LOESUNG(EN):');  WRITELN;
    IF LINFALL THEN
       IF UNLOESB THEN WRITELN('DIE GLEICHUNG IST UNLOESBAR.'
       ELSE IF ALLGEMG THEN
            WRITELN('DIE GLEICHUNG IST ALLGEMEINGUELTIG.')
       ELSE
         BEGIN
           WRITELN('X = ',X1R);
           WRITELN;
           WRITELN('DIE GLEICHUNG WAR IN WIRKLICHKEIT LINEAR.'
         END;
    IF DOPPELFALL OR ZWEIREFALL THEN
       BEGIN
         WRITELN;
         WRITELN('  X1 = ',X1R);
         WRITELN('  X2 = ',X2R);
         WRITELN;
         WRITELN('DIE LOESUNGEN SIND REELL');
         IF DOPPELFALL THEN
            WRITELN('UND FALLEN ZUSAMMEN.');
       END;
    IF KOMPFALL THEN
       BEGIN
         WRITELN;
         WRITELN('  X1 =',X1R:5:2,' + ', X1I:5:2,' * I');
         WRITELN('  X2 =',X2R:5:2,' - ',-X2I:5:2,' * I');
         WRITELN;
         WRITELN('DIE LOESUNGEN SIND KOMPLEX.');
       END;
  END;

BEGIN   (* HAUPTPROGRAMM *)
  QUADEING;
  QUADLOES(A,B,C);
  QUADAUSG;
END.
```

```
PROGRAM DEMO_BUBBLESORT;
USES APPLESTUFF;
VAR A: ARRAY[1..10000] OF INTEGER;
    N,I,J: INTEGER;

PROCEDURE BUBBLESORT(L,R: INTEGER);

  PROCEDURE TAUSCH(I,J: INTEGER);
    VAR B: INTEGER;
    BEGIN   (* Tausch *)
      B:=A[I];
      A[I]:=A[J];
      A[J]:=B;
    END;   (* Tausch *)

  BEGIN   (* BUBBLESORT *)
    FOR J:=L TO R-1 DO
      FOR I:=L TO R-J DO
        IF A[I] > A[I+1] THEN TAUSCH(I,I+1);
  END;   (* BUBBLESORT *)

BEGIN   (* HAUPTPROGRAMM *)
  PAGE(OUTPUT);
  WRITELN('BUBBLESORT - DEMO');  WRITELN;
  WRITE('N = '); READLN(N);  WRITELN;
  WRITELN('JETZT WERDEN DIE DATEN GENERIERT');  WRITELN;
  RANDOMIZE;
  FOR I:=1 TO N DO
    BEGIN
      A[I]:=RANDOM;
      WRITELN(I:6,A[I]:8);
    END;
  WRITELN;
  WRITELN('JETZT WERDEN DIE DATEN SORTIERT');  WRITELN;
  NOTE(30,10);
  BUBBLESORT(1,N);
  NOTE(30,10);
  WRITELN('FERTIG MIT SORTIEREN');  WRITELN;
  WRITELN('SORTIERTE DATEN: ');  WRITELN;
  FOR I:=1 TO N DO WRITELN(I:6,A[I]:8);
  WRITELN;  WRITELN('ENDE BUBBLESORT');
END.
```

```
PROGRAM DEMO_QUICKSORT;
USES APPLESTUFF;
TYPE FELD = ARRAY[1..10000] OF INTEGER;
VAR N,L,R,I: INTEGER;
    A: FELD;

PROCEDURE QUICKSORT(L,R: INTEGER);
VAR LZ,RZ,G: INTEGER;
BEGIN   (* QUICKSORT *)
  IF L < R THEN
  BEGIN
  LZ:=L; RZ:=R; G:=A[LZ];
  REPEAT
    WHILE (A[RZ] >= G) AND (LZ < RZ) DO RZ:=RZ-1;
    A[LZ]:=A[RZ];
    WHILE (A[LZ] <= G) AND (LZ < RZ) DO LZ:=LZ+1;
    A[RZ]:=A[LZ];
  UNTIL LZ=RZ;
  A[LZ]:=G;
  IF L < LZ-1 THEN QUICKSORT(L,LZ-1);
  IF LZ+1 < R THEN QUICKSORT(LZ+1,R);
  END;
END;   (* QUICKSORT *)

BEGIN   (* HAUPTPROGRAMM *)
  PAGE(OUTPUT);
  WRITELN('QUICKSORT - DEMO');  WRITELN;
  WRITE('N = ');  READLN(N);
  WRITELN;  WRITELN('JETZT WERDEN DIE DATEN GENERIERT');WRITELI
  RANDOMIZE;
  FOR I:=1 TO N DO
    BEGIN
      A[I]:=RANDOM;
      WRITELN(I:6,A[I]:8);
    END;
  WRITELN;
  WRITELN('JETZT WERDEN DIE DATEN SORTIERT');  WRITELN;
  NOTE(30,10);
  QUICKSORT(1,N);
  NOTE(30,10);
  WRITELN('FERTIG MIT SORTIEREN');  WRITELN;
  WRITELN('SORTIERTE DATEN: ');  WRITELN;
  FOR I:=1 TO N DO WRITELN(I:6,A[I]:8);
  WRITELN; WRITELN('ENDE QUICKSORT');
END.
```

LITERATURVERZEICHNIS

1 Abelson, H. / A. di Sessa: Turtle Geometry
The MIT Press; Cambridge, Massachusetts 1981

2 Balzert, H.: Informatik 1 / 2
Hueber-Holzmann Verlag; München 1976 / 1978

3 Baumann, R.: Programmieren mit PASCAL
Vogel-Verlag; Würzburg 1980

4 --- Informatik mit Pascal
Ernst Klett Verlag; Stuttgart 1981

5 --- Strukturiertes Programmieren mit BASIC
Ernst Klett Verlag; Stuttgart 1983

6 Bowles, K.: Problem Solving Using PASCAL
Springer-Verlag; New York 1977

7 Dürr, R. / J. Ziegenbalg: Dynamische Prozesse und ihre Mathematisierung durch Differenzengleichungen
Ferdinand Schöningh Verlag; Paderborn 1984

8 Engel, A.: Elementarmathematik vom algorithmischen Standpunkt
Ernst Klett Verlag; Stuttgart 1977

9 Goldberg, S.: Introduction to Difference Equations with Illustrative Examples from Economics, Psychology and Sociology
John Wiley & Sons, Inc.; New York 1958

10 Graf, K.-D.: Informatik
Verlag Herder; Freiburg im Breisgau 1981

11 Jensen, K. / N. Wirth: PASCAL Manual and User Report
Springer-Verlag; New York 1978

12 Kemeny, J.G. / Th.E. Kurtz: BASIC Programming
John Wiley & Sons, Inc.; New York 1971

13 Klingen, L.H.: Elementare Algorithmen
Verlag Herder; Freiburg im Breisgau 1981

14 Löthe, H. / W. Quehl: Systematisches Arbeiten mit BASIC
Teubner Verlag; Stuttgart1982

15 Meadows, D. et al.: Die Grenzen des Wachstums
Rowohlt Taschenbuch Verlag;
Reinbek bei Hamburg 1973

16 Menzel, K.: Elemente der Informatik
Teubner Verlag; Stuttgart 1978

17 --- BASIC in 100 Beispielen
Teubner Verlag; Stuttgart 1981

18 Ocker, S. / L. Schöttle / W. Simon: Informatik
R.Oldenbourg Verlag; München 1979

19 Papert, S.: Mindstorms: Children, Computers, and Powerful Ideas
Basic Books; New York, 1980

20 Schupp, W.: Schüler programmieren in BASIC
Ferdinand Schöningh Verlag; Paderborn 1980

21 --- Schüler programmieren in PASCAL
Ferdinand Schöningh Verlag; Paderborn 1984

22 Wirth, N.: Systematisches Programmieren
Teubner Verlag; Stuttgart 1972

Bücher für den Lehrer

Jochen Ziegenbalg
Anwendungsbereiche für Kleincomputer
Unterrichtsbeispiele für die Sekundarstufe I mit 64 Programmen und 40 Aufgaben.
122 Seiten, Best.-Nr. 37465

Rolf Dürr, Jochen Ziegenbalg
Dynamische Prozesse
und ihre Mathematisierung durch Differenzengleichungen.
317 Seiten, Best.-Nr. 37462

Norbert Christmann u. a.
Anwendungsorientierter Mathematikunterricht
unter besonderer Berücksichtigung der Möglichkeiten von Rechnern.
288 Seiten, Best.-Nr. 37464

Annemarie Hauf, Leonhard Sturm (Hrsg.)
Grundkurs S II Informatik
unter Mitarbeit von Peter Dresch, Lisa Franzen, Gunter Frobel, Hans-Jürgen Koschorreck und Josef Teufel.

Band 1
Grundkursfolge – Aufgabensammlung
200 Seiten, Best.-Nr. 37457

Band 2
Einführung in die Informatik – Algorithmik I/II – Datenstrukturen
360 Seiten, Best.-Nr. 37458

Band 3
Struktur und Arbeitsweise einer DV-Anlage – Probleme aus der praktischen Anwendung
293 Seiten, Best.-Nr. 37459

Georg Smolek, Martin Weissenböck
Einführung in die EDV von A – Z
168 Seiten, 80 Abb., 12 Tab.
= Uni-Taschenbücher, UTB 610

Hermann O. Huppertz
FORTRAN IV
131 Seiten, Best.-Nr. 37471

Ferdinand Schöningh, Paderborn

Informatik

Wilfried Schupp

Schüler programmieren in BASIC

Lehr- und Übungsbuch mit 150 Programmbeispielen und 260 Übungsaufgaben.
3. verbesserte Auflage
160 Seiten, Best.-Nr. 37449
Lehrerbuch: 80 Seiten, Best.-Nr. 37450
Programmdiskette (Apple II): Best.-Nr. 62020

Donald Alcock

Das kleine BASIC-Handbuch

Übersetzt von Wilfried Schupp
136 Seiten, Best.-Nr. 37479

Wilfried Schupp

Schüler programmieren in PASCAL

Einführung in UCSD-PASCAL mit 140 Programmbeispielen und 260 Aufgaben.
160 Seiten, Best.-Nr. 37469
Lehrerbuch: Best.-Nr. 37470
Programmdiskette (Apple II): Best.-Nr. 62021

Wilfried Schupp

PLUS-Informatik

für die Sekundarstufe I.
80 Seiten, Best.-Nr. 32210
Lehrerbuch: Best.-Nr. 32211

Peter Dresch, Gunter Frobel, Hans-Jürgen Koschorreck

Informatik für die Sekundarstufe II

Ein Unterrichtswerk für die Sekundarstufe II.
Band 1: **Elementare Algorithmen**
192 Seiten, Best.-Nr. 37041
Begleit- und Lösungsheft: 39 Seiten, Best.-Nr. 37046

Band 2: **Algorithmen und Datenstrukturen**
168 Seiten, Best.-Nr. 37042
Begleit- und Lösungsheft: 72 Seiten, Best.-Nr. 37047

Band 3: **Aufbau und Arbeitsweise von Datenverarbeitungsanlagen und ihre Anwendung in der Praxis**
ca. 160 Seiten, Best.-Nr. 37043
Begleit- und Lösungsheft: ca. 60 Seiten, Best.-Nr. 37048

Ferdinand Schöningh, Paderborn