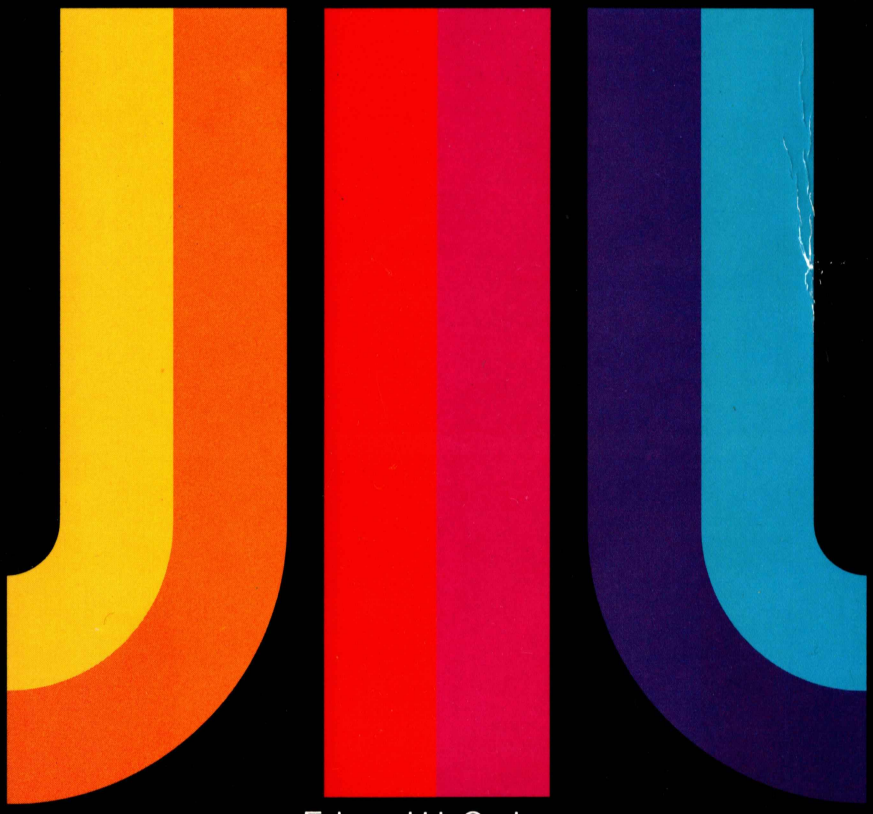
DATAMOST

HAPPY COMPUTER

Ein Markt & Technik Buch

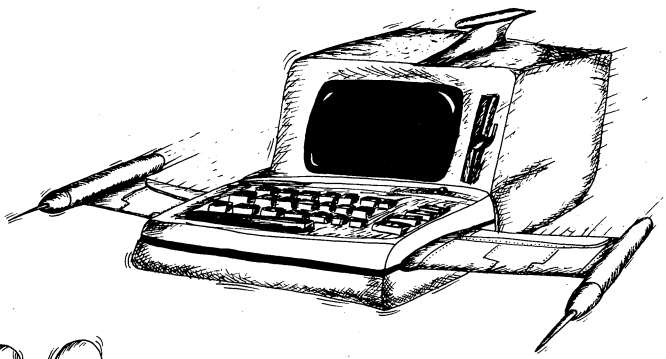
Lerne BASIC auf dem ATARI

Programmiertechnik für Groß und Klein



Edward H. Carlson

Lerne BASIC auf dem ATARI



Edward H. Carlson

Lerne BASIC auf dem ATARI

Programmiertechnik für Groß und Klein

Deutsche Übersetzung:
Irene Lüke

Markt & Technik Verlag

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Carlson, Edward H.:
Lerne BASIC auf dem ATARI / Edward H. Carlson.
Dt. Übers.: Irene Lüke. — Haar bei München : Markt-und-Technik-Verlag 1984.
(Happy Computer)
ISBN 3-89090-007-0
Einheitssacht.: Kids and the ATARI <dt.>

Die Informationen im vorliegenden Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Verlag, Herausgeber und Autoren können
für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine
Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Herausgeber dankbar.

Titel der amerikanischen Originalausgabe
»KIDS AND THE ATARI«
by Edward H. Carlson
ISBN 0-88190-055-9

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

88 87 86 85 84

ISBN 3-89090-007-0

Einbandgestaltung: Grafikdesign Heinz Rauner
Zeichnungen: René Nestler
Druck: Eimannsberger, München
Printed in Germany

ÜBER DIESES BUCH

Das Buch ist aus 33 Kapiteln zusammengesetzt. Jedes enthält Übungsaufgaben und Hinweise für Eltern und Lehrer. Die Kapitel sind auch für nicht mit der Materie vertraute Erwachsene geeignet. Übungen und Hinweise sind unterschiedlich gestaltet. Die Abschnitte mit den Lehrinhalten sind sprachlich so gehalten, daß sie auch von Kindern und Jugendlichen ohne Schwierigkeiten bearbeitet werden können. Ein Begriffslexikon, das nötige Hintergrundinformationen liefert, und die Hinweise zu den Übungen erklären detailliert die verwendeten BASIC-Befehle.

Das Buch führt zunächst in die grundlegenden Denkweisen der Programmierung ein und gelangt dann schnell zu dem Punkt, an dem interessante Programme selbst geschrieben werden können. Im mittleren Teil des Buchs werden etwas kompliziertere Befehle behandelt. Dies setzt sich im letzten Abschnitt fort, wobei hier auch auf die wichtigen Programmierhilfen wie Fehlersuche und Editieren eingegangen wird.

Die Übungsaufgaben verlangen von den Schülern das Schreiben einfacher Programme. Am Ende des Buches befinden sich Lösungsbeispiele. Diese sollten jedoch nicht als einzige Möglichkeit begriffen werden.

Die Übungen 14: "Das Speichern auf Band" und 18: "Der Editier- und der RUN-Modus" können auch gleich nach dem ersten Kapitel durchgearbeitet werden.

INHALTSVERZEICHNIS

WIDMUNG

Mein aufrichtiger Dank gilt Paul Sheldon Foote, der mich zu diesem Buch angeregt hat.

Ich habe in den letzten zwei Jahren im Computer Camp in Michigan mitgeholfen und gelehrt. Ich danke meinen Kollegen und den Mitgliedern des Sommer Camps: Mark Lardie, Mary Winter und John Forsyth. Alle haben mich an ihren Erfahrungen teilhaben lassen und vernünftige Wege gefunden, den Lehrstoff zu vermitteln.

Mark Lardie hat in großzügiger Weise seine großen Erfahrungen im Unterricht mit Kindern weitergegeben und das Manuskript überlesen.

Das große Vergnügen der Schüler im Sommer-Camp hat mich dazu ermutigt, dieses Buch zu schreiben.

Einige Familien besitzen bereits dieses Buch und haben auch schon Verbesserungsvorschläge gemacht. Besonderen Dank sagen möchte ich Steve Peter und den Mädchen Karen und Kristy, George Campbell und seinen Kindern Andrew und Sarah, Beth O'Malia und Scott, John und Matt, Chris Clark und Chris Jr. Tryn, Daniel und Vicky, und Paul Foote und David.

John Decarli half mir, einen kritischen Punkt beim Erstellen dieses Manuskripts zu überstehen, als er mir einen Drucker lieh.

Meine Familie hat lange Rücksicht auf mich genommen und sich ruhig verhalten. Deshalb gilt mein besonderer Dank meiner Frau Louise und unseren Kindern Karen, Brian und Minda.

AN EUCH JUGENDLICHE

In diesem Buch lernt Ihr, wie man für den ATARI-Computer Programme schreibt.

Ihr erfahrt, wie man Aktion-, Brett- und Wortspiele machen kann. Damit könnt Ihr langweilige Parties in Schwung bringen und Euch zusammen mit Euren Freunden amüsieren.

Vielleicht wollt Ihr auch Eure Plattensammlung archivieren oder Euer Taschengeld verwalten. All das könnt Ihr mit dem Computer erledigen.

Ihr könnt auch Euren kleineren Geschwistern helfen, mit ihnen das Rechnen oder Buchstabieren zu üben. Auch das Hausaufgabenmachen könnt Ihr Euch erleichtern, wenn Ihr zum Beispiel ein Programm über Geschichte oder eine Fremdsprache schreibt.

So benutzt man dieses Buch: Führe alle Beispiele aus. Weißt Du nicht mehr weiter, lese das Kapitel noch einmal von Anfang an. Vielleicht hast Du einen wichtigen Hinweis übersehen. Findest Du Dich aber gar nicht mehr zurecht, solltest Du Deine Eltern oder einen Lehrer um Hilfe bitten.

Jede Übung endet mit Aufgabenstellungen. Du solltest sie alle beantworten können, ehe Du zum nächsten Kapitel weitergehst.

AN DIE ELTERN

Dieses Buch soll Jugendliche im Alter von 10 bis 14 mit dem ATARI-Computer in die Rechnersprache BASIC einführen. Es beinhaltet Erklärungen, Übungen, Wiederholungen und Fragespiele. Bei einigen Übungen können die Schüler die Antworten in vorgegebene Zeilen schreiben. Am Schluß des Buchs findet man mögliche Lösungen für die gestellten Aufgaben.

Wahrscheinlich wird Ihr Kind am Anfang Hilfe brauchen und es sollte ermutigt werden, wenn es Schwierigkeiten hat.

Das Programmieren ist nicht leicht zu erlernen, denn es verlangt Genauigkeit und die Beachtung von Einzelheiten. Aber Ihr Kind wird reich belohnt, wenn es sich mit dem Buch lang genug beschäftigt hat und mit den erlernten Befehlen tolle Programme erstellen kann.

So benutzt man dieses Buch: Das Buch ist in 33 Kapitel unterteilt. Am Anfang jeder Übung steht ein Hinweisteil, den Sie lesen sollten. Er erklärt kurz den Inhalt der Übung, gibt wertvolle Hinweise und schlägt Fragen vor, die Sie stellen können, damit Sie sehen, ob der Stoff des Kapitels verstanden worden ist.

Diese Hinweise sind für Erwachsene geschrieben worden, aber sie können auch älteren Schülern von Nutzen sein. Schüler der unteren Klassen werden sie wahrscheinlich nicht lesen. Sie werden sich mit dem Übungsteil begnügen. Bei kleineren Kindern ist es ratsam, die Übungen mit ihnen zusammen laut zu lesen und durchzusprechen, bevor sie an den Computer dürfen.

AN DEN LEHRER

Dieses Buch wurde für Schüler zwischen 10 und 14 Jahren geschrieben. Es lehrt ATARI-BASIC auf einem Kassettensystem.

In den Kapiteln sind Erklärungen (und Bilder), Beispiele, Übungen und Wiederholungsfragen enthalten. Ein Anmerkungsteil zu Beginn jeder Übung stellt die verwendeten Befehle vor, gibt Hinweise und klärt über die Fragestellungen auf.

Dieses Buch eignet sich zum Selbststudium, kann aber auch als Lehrbuch in einem Kurs verwendet werden.

Ich habe das Buch so angelegt, das BASIC nicht nur gelernt, sondern daß die BASIC-Sprache beherrscht wird. Seymour Papert hat einmal gesagt, daß durch das Programmieren der Horizont erweitert werden kann. Dazu gehört auch, daß man erkennt, daß Programme eigenständige Gebilde sind. Man kann ihnen einen Namen geben, sie in Einzelteile zerlegen und Fehler suchen. Sie können in "mundgerechte" Häppchen zerlegt und hierarchisch aufgebaut werden. Schleifen können durchlaufen und Bedingungen (die IF...THEN-Anweisung) können abgefragt werden.

All dies wird dem Schüler beim Durcharbeiten der Übungen vermittelt.

ÜBER DAS PROGRAMMIEREN

Es besteht die allgemeine Vorstellung, daß das Programmieren eines Computers etwas mit Algebra zu tun habe. Das ist nicht unbedingt so. Am ehesten ist das Arbeiten mit dem Computer noch mit dem Bauen mit Bauklötzen und dem Schreiben eines Aufsatzes vergleichbar.

So, wie ein Baukasten aus nur wenigen verschiedenen Grundformen besteht, hat auch BASIC nur eine kleine Anzahl von Grundelementen (Anweisungen). Genauso, wie man aus diesen Bauklötzen ein phantasievolles Schloß bauen kann, ist es möglich, mit BASIC interessante Programme zu schreiben.

Ähnlich wie bei der Beschäftigung mit Bauklötzen, beginnt man mit einfachen Strukturen und erarbeitet sich langsam das gesamte zur Verfügung stehende Instrumentarium.

Wie eine Kurzgeschichte ist auch ein Programm eine in sich selbst geschlossene Einheit, die einem bestimmten Zweck dient. Das Schreiben des berühmten Aufsatzes "Wie ich meine Sommerferien verbrachte" kann als Beispiel für die verschiedenen Arbeitsmethoden herangezogen werden. Der Erstklässler sitzt etwas ratlos vor seinem Blatt, kaut am Bleistift und überlegt, was er schreiben könnte. Einige Klassen später wird nicht mehr lange überlegt, sondern einfach drauflosgeschrieben. Die Bedeutung der inhaltlichen Gliederung findet keine Beachtung. Erst am Ende der Schulzeit hat sich ein Gefühl für die Gestaltung von Form und Inhalt gebildet.

Einige Ähnlichkeiten mit Arithmetik hat das Programmieren aber doch. Es gibt viele Regeln zu lernen. Bereits der kleinste Fehler kann zu einem schlechten Gesamtergebnis führen. Anders als im Bereich der Arithmetik steckt im Programmieren ein schier unübersehbares Kreativpotential.

Das Programmieren wirkt sich allgemein positiv auf die Ausbildung aus. Plastisch auf einem Bildschirm angebotener Lehrstoff erlaubt die Konzentration auf das Wesentliche. Dazu kommt das Erlernen sowohl analytischer als auch konstruktiver Denkweisen.

Diese beim Programmieren erworbenen Denkweisen lassen sich auch im Alltag vorteilhaft einsetzen. Die analytische Betrachtungsweise führt zu mehr Überblick über vorhandene Situationen und führt damit auch zu besseren Entscheidungen.

1

Einführung

1. **ANMERKUNGEN FÜR DEN LEHRER** NEW, PRINT, REM und RUN

Dieses Kapitel ist eine Einführung in die Welt des Computers. Auf den nächsten Seiten erfahren Sie, wie man den Computer einschaltet.

Inhaltsübersicht:

1. Einschalten des Computers.
2. Eingeben von Befehlen oder Zeilen. Die RETURN-Taste.
3. Der Computer versteht nur eine begrenzte Anzahl von Befehlen.
4. Mit REM kann man im Programm Bemerkungen ablegen.
5. Was ist ein Programm? Numerierte Zeilen.
6. Löschen des Bildschirms.
7. Der Speicher kann mit NEW gelöscht werden.
8. Die Ausgabe auf dem Bildschirm und der Inhalt im Speicher unterscheiden sich. Dies mag für den Schüler am Anfang schwer zu verstehen sein.
9. RUN veranlaßt, daß der Computer in seinem Speicher nach den Befehlen in den Zeilen (der Reihe nach) sucht und sie ausführt.
10. Warum beim Durchnumerieren der Zeilen größere Zahlenabstände gemacht werden sollen.

Übung 1: Die Befehle NEW, PRINT, REM und RUN

Fragen:

1. Schreibe ein Programm, das Deinen Namen ausgibt.

3. Laß es laufen.

3. Laß das Programm vom Bildschirm verschwinden, im Speicher soll es noch vorhanden sein.

4. Laß es laufen.

5. Lösche das Programm aus dem Speicher.

ÜBUNG 1: NEW, PRINT, REM UND RUN

SO FÄNGT MAN AN

Stecke das Modul mit der Bezeichnung BASIC in den ATARI-Computer. Das Etikett muß auf Dich zeigen. Wenn Du einen ATARI 800 besitzt, mußt Du das Modul in den linken Einschub stecken.

Schalte den Computer ein. Du siehst eine Meldung auf dem Bildschirm:

READY

Unter READY erscheint ein Quadrat. Dieses Quadrat nennt man "Cursor". Der Cursor signalisiert Dir, daß der Computer bereit ist, Eingaben anzunehmen.

Cursor heißt soviel wie "Läufer", denn das kleine Quadrat saust auf dem Bildschirm herum und zeigt Dir, wo das Zeichen, das Du eintippst, erscheint.

Was Du eingibst, siehst Du am Bildschirm.

DEN BILDSCHIRM LÖSCHEN

Mit zwei Tasten löschst Du den Bildschirm.

Halte eine der SHIFT-Tasten fest und drücke dann die CLEAR-Taste. Der Bildschirm ist gelöscht.

Das englische Wort clear bedeutet: löschen.

Übung 1: Die Befehle NEW, PRINT, REM und RUN

DEM COMPUTER BEFEHLE ERTEILEN

Versuch das mal. Gib ein: GIB MIR BONBONS

und drücke die RETURN-Taste.

Der Computer gibt aus:

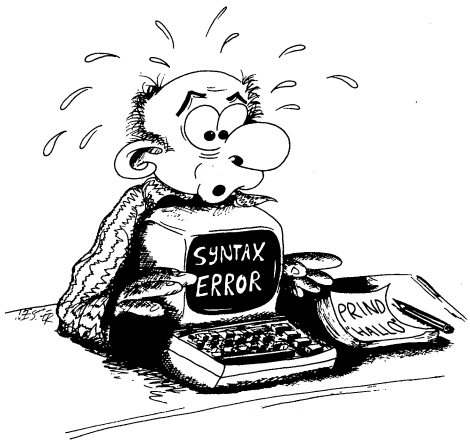
ERROR- GIB MIR BONBONS

Wenn der Computer ERROR- druckt, dann hat er Deine Eingabe nicht verstanden. ERROR ist englisch und bedeutet hier Eingabefehler.

Der Computer versteht nur etwa 80 Wörter. Du mußt sie lernen.

Anschließend findest Du die ersten vier, die Du Dir merken mußt:

NEW, PRINT, REM und RUN



DER NEW-BEFEHL

Gib ein: NEW

und drücke RETURN.

Mit NEW wird der Speicher des Computers gelöscht, und Du kannst nun Dein Programm eingeben.

SO GIBT MAN EINE ZEILE EIN

Wenn es heißt: gib ein oder tippe, dann bedeutet das, daß Du zwei Sachen erledigen mußt.

1. Eine Zeile eingeben.
2. Dann RETURN drücken.

Drücke gleichzeitig die SHIFT- und die CLEAR-Taste und gib die nächste Zeile ein:

```
10 PRINT "HALLO"
```

(Die Anführungszeichen (") schreibst Du, indem Du die Tasten SHIFT und 2 gleichzeitig drückst.)

Hast Du auch daran gedacht, RETURN am Ende der Zeile einzugeben? Jetzt befindet sich die Zeile 10 im Speicher des Computers. Sie bleibt dort so lange, bis Du NEW tippst oder den Computer ausschaltest. Zeile 10 ist ein sehr kurzes Programm.

DIE ZAHL 0 UND DER BUCHSTABE "O"

Der Computer schreibt die Null folgendermaßen:

Null 0

und den Buchstaben O:

Buchstabe O O

Das nächste Beispiel solltest Du nicht nachmachen:

```
10 PRINT "HALLO"    falsch

10 PRINT "HALLO"    richtig
```

WAS IST EIN PROGAMM?

Ein Programm besteht aus einer Reihe von Befehlen, die der Computer ausführen soll. Sie stehen in Zeilen. Das Programm, das Du vorher eingegeben hast, hat nur eine Zeile.

SO LÄUFT EIN PROGRAMM

Du hast folgendes Programm im Speicher stehen:

```
10 PRINT "HALLO"
```

Gib jetzt ein: RUN (Denk daran, daß Du RETURN drücken mußt.)

Die RUN-Anweisung bringt den Computer dazu, daß er in seinem Speicher nach einem Programm sucht und dann die Befehle, die in den Zeilen stehen, ausführt. RUN heißt losrennen oder laufen.

Hat der Computer das PRINT-Kommando ausgeführt? Mit PRINT wird alles, was zwischen den Anführungszeichen steht, auf dem Bildschirm ausgedruckt. (PRINT bedeutet ausdrucken.) Der Computer druckt:

HALLO

EIN LÄNGERES PROGRAMM

Lösch den Speicher mit NEW.

(Denkst Du auch daran, daß Du wieder RETURN drücken mußt?)

Gib das Programm ein:

```
1 REM PROGRAMM
2 PRINT "HALLO"
3 PRINT "MEIN FREUND"
```

Dieses Programm besteht aus drei Zeilen. Jede davon beginnt mit einem Befehl.

Übung 1: Die Befehle NEW, PRINT, REM und RUN

Gib ein: RUN

Zeile 1: Der Computer überspringt diese Zeile, weil sie einen REM-Befehl enthält.

Zeile 2: Der Computer druckt "HALLO".

Zeile 3: Der Computer gibt dann "MEIN FREUND" aus. Mit dem REM-Befehl kannst Du für Dich selbst kleine Programmanmerkungen machen.

In unserem Fall haben wir die REM-Anweisung in Ziele 1 dazu verwendet, dem Programm einen Namen zu geben. Der Name lautet: "PROGRAMM". Der Computer führt die Befehle in den Zeilen aus. Er beginnt mit der niedrigsten Zeilennummer und arbeitet das Programm dann der Reihe nach ab.

SO NUMERIERT MAN DIE ZEILEN IN EINEM PROGRAMM

Normalerweise macht man die Abstände zwischen den Zeilennummern größer. Zum Beispiel so:

```
10 REM PROGRAMM
15 PRINT "HALLO"
20 PRINT "MEIN FREUND"
```

Das Programm ist gleich, aber es hat unterschiedliche Zeilennummern. Sie stehen der Reihe nach, aber mit Abstand. Das ist deshalb wichtig, weil Du später in diese Lücken neue Zeilen einfügen kannst, wenn Dein Programm größer wird.

AUFGABE 1

1. Zeige, wie man den Bildschirm löscht.

2. Verwende den NEW-Befehl. Erkläre seine Bedeutung.

3. Schreibe ein Programm, in dem REM einmal und PRINT zweimal vorkommt. Starte mit dem RUN-Befehl das Programm.

2. **ANMERKUNGEN FÜR DEN LEHRER** Der Piepser, Inversdarstellung Zeichenkettenkonstanten

Diese Übung beginnt mit der CONTROL-2-Sequenz, die einen Ton erzeugt. Wir wollen eine ganze Menge "Geräusche" hervorbringen, und damit unsere Programme bereichern.

Die in Übung 1 verwendete "Zeichenkettenkonstante" wird erklärt. Zahlen, die in einer Zeichenkette vorkommen, zum Beispiel "19", können nicht unmittelbar arithmetisch umgesetzt werden.

Der INVERS-Befehl bringt ein wenig Abwechslung auf den Bildschirm.

Obwohl der ATARI auch mit Kleinbuchstaben schreiben kann, werden wir sie in diesem Buch nicht verwenden. Dafür gibt es mehrere Gründe:

Wir werden mit den Kleinbuchstaben in PRINT-Anweisungen besondere Befehle verdeutlichen, wie "piepser, invers, normal, clear".

Sie erschweren das Programmerstellen, weil die Befehle nicht in Kleinbuchstaben geschrieben werden dürfen.

Fragen:

1. Wie führst Du die folgenden Dinge aus: Der ATARI soll "ein Geräusch machen". Der Bildschirm soll gelöscht werden. Der Speicher soll gelöscht werden.

2. Was ist eine "Zeichenkette"?

3. Welche bestimmte Taste mußt Du drücken, damit eine Zeile eingegeben werden Kann?

4. Was ist ein Befehl? Gib einige Beispiele.

5. Wie kannst Du das Wort "FEUER" in inversen Buchstaben darstellen und dabei den Computer piepsen lassen?

ÜBUNG 2: DER PIEPSER, INVERSDARSTELLUNG, ZEICHENKETTENKONSTANTEN

Gib ein: NEW (Denk daran: RETURN-Taste drücken!)

Lösche den Bildschirm. (Denk daran: SHIFT- und CLEAR-Taste!)

Nun kannst Du mit der Übung anfangen.

DER PIEPSER MACHT GERÄUSCHE

Der ATARI hat einen Piepser. Mit zwei Tasten kannst Du ihn dazu bringen, daß er Geräusche von sich gibt.

Halte die CONTROL-Taste fest. Drücke dann die Taste mit der Aufschrift "2".

Du hörst ein lautes Geräusch.

CONTROL bedeutet: steuern. Die Taste hilft uns dabei, den Computer zu steuern.

WIE MAN EINE LEERZEILE DRUCKT

Laß das Programm laufen:

```
10 REM LEERZEILEN
20 PRINT "DAS IST EINE ZEILE"
30 PRINT
40 PRINT "EINE ZEILE WURDE UEBERSPRUNGEN"
```

Mit Zeile 30 wurde eine Leerzeile ausgedruckt.

ZEICHENKETTENKONSTANTEN

Sieh Dir diese PRINT-Anweisungen genau an:

```
10 PRINT "THOMAS"
10 PRINT "#D47*%"
10 PRINT "19"
10 PRINT "3.1416"
10 PRINT "ICH BIN 14"
```

Buchstaben, Zahlen und Satzzeichen werden "Zeichen" genannt.

Auch ein Leerraum ist ein Zeichen. Zum Beispiel:

```
10 PRINT " "
```

(Später werden wir auch noch die Grafikzeichen und Sonderzeichen wie "piepser" und "clear" kennenlernen.) Zeichen in einer Zeile werden Zeichenkette genannt.

Die Buchstaben sind wie auf einer Perlenkette aufgereiht.

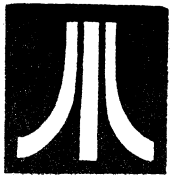
Eine Zeichenkette, die in Anführungszeichen steht, wird Zeichenkettenkonstante genannt.

Sie ist eine Zeichenkette, weil sie aus Buchstaben, Zahlen und Satzzeichen besteht.

Sie ist eine Konstante, weil sie immer gleich bleibt. Sie ändert sich nicht, wenn das Programm läuft.

"VON INNEN NACH AUSSEN DRUCKEN"

Auf dem ATARI-Computer gibt es eine besondere Taste. Sie sieht folgendermaßen aus:



Man nennt sie "die ATARI-Taste".

Drücke sie. Gib was ein.

Du siehst, daß alle Zeichen "invers" dargestellt werden. Das heißt, der Hintergrund und die Zeichen sind genau vertauscht.

Drücke die ATARI-Taste nochmal. Gib wieder etwas ein.

Nun kannst Du die Buchstaben wieder normal schreiben.

Verwende die Inversdarstellung, damit Deine Ausgaben am Bildschirm interessanter aussehen.

INVERS UND NORMAL IN PRINT-BEFEHLEN

Wir werden Dir in diesem Buch auf besondere Art zeigen, wenn Du die ATARI-Taste drücken sollst.

Wir sagen nämlich "invers" oder "normal". Das geht folgendermaßen:

```
10 REM SCHWARZ UND WEISS
20 PRINT "invers WEISS normal SCHWARZ"
```

Das bedeutet folgendes:

Gib ein: 20 PRINT "

Drücke die ATARI-Taste.

Gib ein: WEISS

Drücke nochmals die ATARI-Taste.

Gib ein: SCHWARZ"

Drücke die RETURN-Taste.

Laß das Programm laufen.

GROSSE UND KLEINE BUCHSTABEN

Der ATARI kann auch Kleinbuchstaben schreiben.

In diesem Buch werden wir allerdings keine kleinen Buchstaben verwenden.

WICHTIG!
Immer, wenn Du innerhalb von Anführungszeichen bei einem PRINT-Befehl Kleinbuchstaben siehst, haben diese eine besondere Bedeutung.

```
20 PRINT invers    heißt nicht      20 PRINT "INVERS"
20 PRINT normal    hieß nicht       20 PRINT "NORMAL"
```

Wir haben gesehen, daß die ATARI-Taste eine besondere Stellung einnimmt. Es gibt noch eine Sondertaste:

DIE ESC-TASTE

Such die ESC-Taste. Sie befindet sich in der oberen linken Ecke der Tastatur. ESC ist die Abkürzung für ESCAPE.

Mit dieser Taste können wir dem Computer besondere Anweisungen, wie "piepser" und "clear" geben.

Immer, wenn Du in einem PRINT-Befehl "clear" siehst, mußt Du folgendes ausführen:

Drücke einmal die ECS-Taste.

Halte dann die SHIFT-Taste fest und drücke die CLEAR-Taste. Verstanden?

Gib ein:

```
10 REM SONDERDRUCK
20 PRINT "clear"
30 PRINT "ALLES GELOESCHT"
```

Natürlich kannst Du den Ausdruck "clear" in Zeile 20 nicht sehen. Statt dessen siehst Du einen lustig gebogenen Pfeil.

Immer, wenn Du in einem PRINT-Befehl "piepser" siehst, mußt Du folgendes ausführen:

Drücke einmal die ESC-Taste.

Halte die CONTROL-Taste fest und drücke gleichzeitig die Taste mit der Ziffer 2.

Schreibe noch diese Zeile dazu:

```
40 PRINT "piepser"
```

Natürlich siehst Du auch hier nicht den Ausdruck "piepser". Statt dessen siehst Du einen inversen, lustig gebogenen Pfeil.

Laß das Programm laufen. Es sollte den Bildschirm löschen, ALLES GELOESCHT ausgeben und ein Geräusch machen.

AUFGABE 2

1. Schreibe ein Programm, daß Deinen Vor-, Deinen Spitz- und Deinen Nachnamen schreibt. Die ersten beiden Namen sollen in normalen Buchstaben, der Nachname soll in inversen Buchstaben dargestellt werden.

2. Das Programm soll zu Beginn (bevor etwas geschrieben wird) alles löschen.

3. Laß vor jedem Namen ein Geräusch ertönen.

3. ANMERKUNGEN FÜR DEN LEHRER LIST, Speicherschachteln

In dieser Übung:

LIST, LIST 30

Speicherschachteln enthalten Zeilen

Eine Zeile aus dem Speicher löschen

Eine Zeile hinzufügen

Eine Zeile ersetzen

Zeichnen mit der PRINT-Anweisung

Eigentlich ist der Unterschied zwischen "Befehl" und "Anweisung" künstlich. Der BASIC-Interpreter unterscheidet nicht zwischen ihnen. Unsere Wünsche werden "Befehle" genannt, wenn wir uns im direkten Eingabemodus befinden. Innerhalb einer Programmzeile heißen sie "Anweisungen". Im ersten Teil des Buches werde ich die Bezeichnung "Befehl" verwenden und später erklären, was unter einer Anweisung zu verstehen ist (wenn wir über Doppelpunkte und mehrere Anweisungen in einer Ziele sprechen).

Im Augenblick genügt es, wenn der Schüler versteht, daß ein Programm auch dann, wenn es am Bildschirm nicht zu sehen ist, im Speicher steht und daß der LIST-Befehl das Programm auf dem Bildschirm sichtbar macht. Die speziellen Anwendungen von LIST wie LIST 100,300 behandeln wir später.

In diesem Buch entwickeln wir für den Arbeitsspeicher das Modell eines Regals voller Schachteln. Es dient den Schülern als Hilfsmittel, um Variable und die Arbeitsweise komplizierter Ausdrücke in einer Anweisung zu verstehen.

Es wird gezeigt, wie mit PRINT Bilder gezeichnet werden können. Das sollte am Ende jeder Übung geschehen. Das Zeichnen nach der vierten Übung soll helfen, das Editieren besser zu beherrschen.

Fragen:

1. Wie löscht man eine Zeile, die man nicht mehr braucht?

2. Lösche den Bildschirm. Wie zeigt man das Programm, das im Speicher steht, auf dem Bildschirm?

3. Wie kann man eine falsche Zeile durch eine korrigierte ersetzen?

4. Nimm an, Du willst eine Zeile zwischen zwei, die im Speicher stehen, einfügen. Wie machst Du das?

5. Erkläre, wie der Computer Programmzeilen in Speicherschachteln ablegt. Was steht auf der Vorderseite der Schachtel?

ÜBUNG 3: LIST, SPEICHERSCHACHTELN

Lösche den Bildschirm und den Speicher.

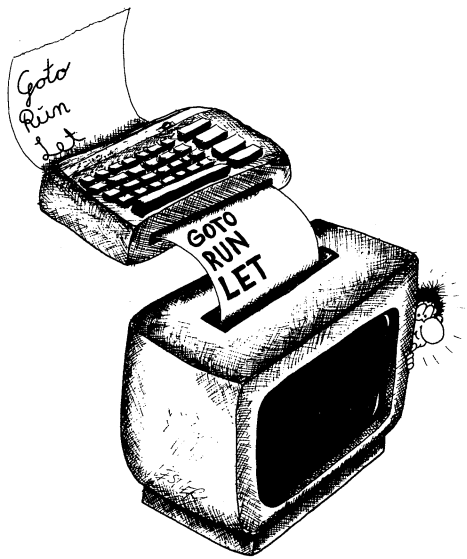
(Mache das bitte in Zukunft am Anfang jeder Übung.)

Gib ein:

```
10 REM HOEREN
20 PRINT "clear"
30 PRINT "HOER ZU"
40 PRINT "piepser"
50 PRINT "HAST DU ES GEHOERT?"
```

Laß dieses Programm aus fünf Zeilen laufen und lösche dann den Bildschirm.

Das Programm ist nun nicht mehr auf dem Bildschirm zu sehen.



Aber es ist nicht ganz verschwunden. Es steht noch im Speicher des Computers. Wir können den Computer bitten, uns das Programm wieder zu zeigen.

DAS PROGRAMM AUFLISTEN

Damit Dir der Computer das ganze Programm zeigt, mußt Du

```
LIST
```

eingeben. Willst Du nur eine Zeile sehen, mußt Du LIST und dann die gewünschte Zeilennummer eintippen:

```
LIST 20
```

DER SPEICHER

Der Speicher eines Computers ist wie ein Regal voller Schachteln. An der Vorderseite jeder Schachtel befindet sich ein freier Platz, auf den man einen Namen schreiben kann. Am Anfang sind alle Schachteln leer, und keine ist beschriftet.

Nach Deiner Eingabe von

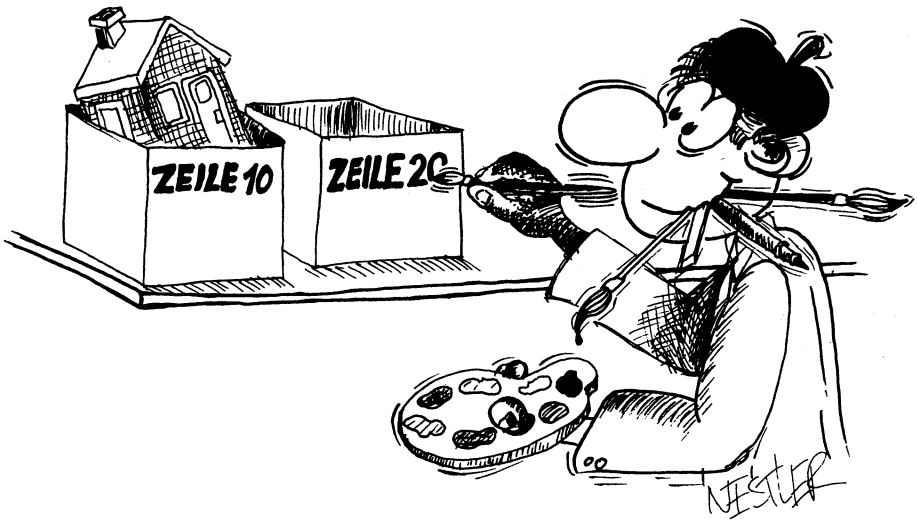
```
10 REM HOEREN
```

hat der Computer die erste freie Schachtel "genommen" und den Namen "Zeile 10" "draufgeschrieben". Dann hat er den Befehl REM HOEREN in die Schachtel "gelegt" und die Schachtel zurück ins Regal "gestellt".

Nach der Eingabe von

```
20 PRINT "clear"
```

hat der Computer die zweite Schachtel genommen und "Zeile 20" draufgeschrieben. Anschließend hat er die Anweisung PRINT "clear" hineingelegt und die Schachtel wieder ins Regal gestellt.



DAS LÖSCHEN EINER ZEILE AUS DEM SPEICHER

Um eine Zeile des Programms zu löschen, brauchst Du nur die Zeilennummer einzugeben. Willst Du zum Beispiel die Zeile 20 löschen, mußt Du

```
20
```

eingeben. Du siehst zwar die Zeile noch auf dem Bildschirm, aber wenn Du Dir mit LIST das Programm ausgeben läßt, siehst Du die Zeile 20 nicht mehr.

Gibst Du nur eine einzelne Zeilennummer ein, holt der Computer die Schachtel mit der Nummer, leert sie aus und radiert den Namen an der Vorderseite aus.

Wie löschst Du das ganze Programm? (Wenn Du die Antwort vergessen hast, sieh bitte in Übung 1 nach.)

Was macht der Computer mit den Schachteln, wenn Du den Befehl NEW eingibst?

DAS HINZUFÜGEN EINER ZEILE

Eine neue Zeile kann überall im Programm hinzugefügt werden, auch zwischen zwei schon vorhandene Zeilen. Such Dir eine Zeilennummer aus, deren Nummer dazwischenpaßt, und tippe Deine Zeile ein. Der Computer ordnet sie richtig ein.

Gib NEW und das Programm ein:

```
10 REM IMMER MEHR
20 PRINT "MEHR ZEILEN"
40 PRINT "HIER SIND SIE"
```

Liste das Programm auf und starte es. Füge die nächste Zeile hinzu:

```
15 PRINT "NOCH"
```

Liste und starte das Programm nochmals.

DAS VERBESSERN EINER ZEILE

Ist eine Zeile falsch, kann sie einfach überschrieben werden. Im obenstehenden Programm kann Zeile 40 so geändert werden:

```
40 PRINT "SIND ZU VERBESSERN"
```

Übung 3: LIST, Speicherschachteln

Was unternimmt der Computer mit der Schachtel, die den Namen "Zeile 40" hat, wenn Du die neue Zeile eingibst?

DER REM-BEFEHL

Mit dem REM-Befehl kannst Du Deinem Programm eine Überschrift geben.

Gib NEW und das Programm ein:

```
10 REM PROGRAMM 2
20 PRINT "clear"
30 PRINT "ZEILE 10 TUT NICHTS"
35 REM DIESE ZEILE TUT NICHTS
40 LIST
RUN
```

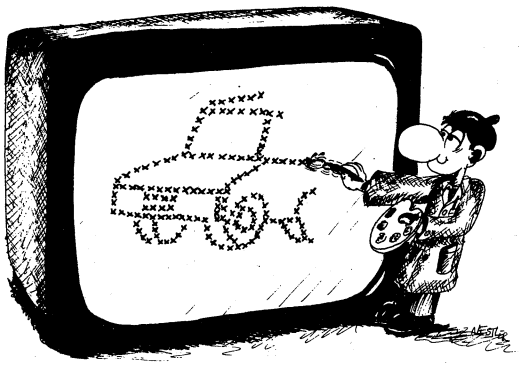
REM ist die Abkürzung für "remark" und heißt Bemerkung. Mit REM kannst Du Dir Notizen im Programm machen, damit Du oder ein anderer Leser es besser versteht.

BILDER ZEICHNEN

Du kannst mit dem PRINT-Befehl Bilder zeichnen. Hier findest Du das Bild eines Autos. Schreib die folgenden Zeilen ab:

```
10 REM TOLLES AUTO
20 PRINT "clear"
25 PRINT
30 PRINT " XXXXXX"
40 PRINT "XXXXXXXXXXXX"
50 PRINT " O         O"
```

Vergiß nicht die Leerräume in den Zeilen. Sie gehören zur Zeichnung.



AUFGABE 3

1. Welcher Befehl zeigt Zeile 10 Deines Programms auf dem Bildschirm an?

2. Wie sagst Du dem Computer, daß er das gesamte Programm am Bildschirm auflisten soll?

3. Was macht der Computer, wenn er den REM-Befehl sieht?

4. Zu welchem Zweck wird der REM-Befehl verwendet?

5. Zeichne drei fliegende Vögel auf dem Bildschirm. Verwende dazu REM, PRINT, "piepser" und "clear". Laß jeden Vogel einmal kreischen.

4. ANMERKUNGN FÜR DEN LEHRER Backspace, Cursor-Tasten, Einfügen, Löschen

Diese Übung befaßt sich mit den Cursor-Steuertasten, Der BACKSPACE-, INSERT- und DELETE-Taste.

Die BACKSPACE-Taste ist eine "Lösch"-Taste. Wird sie gedrückt, wird das Zeichen, das links vom Cursor steht, gelöscht. Der Cursor wird dann auf den leeren Platz gesetzt. Hält man die BACKSPACE-Taste fest, beginnt der Cursor nach etwa einer halben Sekunde fortwährend nach links zu laufen und löscht dabei alle Zeichen.

Tatsächlich haben fast alle Tasten eine Wiederholungsfunktion, wenn man sie länger festhält.

Mit den Pfeiltasten kann man den Cursor an jeden Punkt des Bildschirms stellen. Dazu muß allerdings die CONTROL-Taste gleichzeitig gedrückt sein.

Immer da, wo der Cursor anhält, kann ein neues Zeichen geschrieben werden.

Auch bei der INSERT- und der DELETE-Taste muß die CONTROL-Taste gedrückt sein. Mit der DELETE-Taste wird das Zeichen gelöscht, auf dem sich der Cursor gerade befindet. Mit der INSERT-Taste wird ein Leerraum links vom Cursor eingefügt. Anschließend wird der Cursor an die freie Stelle plaziert.

Fragen:

1. Was ist ein Cursor? Welchen Zweck hat er?

2. Lassen Sie sich von Ihren Schülern zeigen, wie man eine Zeile editiert. Um zur Zeile zu gelangen, sollten die Pfeiltasten verwendet werden. Die Zeichen sollten geändert, und mit RETURN sollte die Zeile abgelegt werden. (Die Tasten BACKSPACE, DELETE und INSERT sollten verwendet werden.)

ÜBUNG 4: BACKSPACE, CURSOR-TASTEN, EINFÜGEN, LÖSCHEN

BACKSPACE

Tippe eine lange Zeile ein. Drücke dann die BACKSPACE-Taste. Sie löscht ein Zeichen und stellt den Cursor anschließend eine Stelle zurück.

Halte jetzt die BACKSPACE-Taste fest. Warte. Der Cursor saust nun zurück und alles wird gelöscht.

DIE PFEILTASTEN

Es gibt vier Pfeiltasten. Sie sind in einem Quadrat auf der rechten Seite der Tastatur angeordnet. In der oberen Reihe befindet sich ein Taste mit nach oben gerichtetem Pfeil, bei der anderen ist der Pfeil nach unten gerichtet. In der unteren Reihe ist ein Pfeil nach rechts, der andere nach links gerichtet.

Halte die CONTROL-Taste fest.

Übung 4: Backspace, Cursor-Tasten, Einfügen, Löschen

Drücke eine der Pfeiltasten.

Der Cursor bewegt sich in die Richtung, in die der Pfeil zeigt.

Drücke dann eine andere Cursor-Taste.

ENTLANGFLITZEN

Halte die CONTROL-Taste fest. Drücke dann eine Pfeiltaste für längere Zeit. Der Cursor flitzt am Bildschirm entlang.

Das geht auch mit anderen Tasten. Probiers einmal mit der A-Taste. Nach einer Weile bekommst Du eine Zeile mit A's:

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

FEHLER VERBESSERN

Gib ein: 10 REM ZRACHE

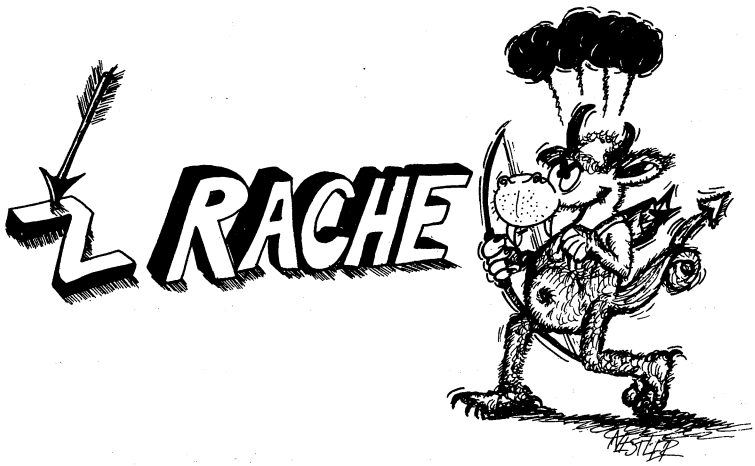
Gehe mit dem Cursor an den Buchstaben Z.

Tippe dort ein D.

Die Zeile ist jetzt richtig:

10 REM DRACHE

Drücke RETURN, damit die Zeile in den Speicher übernommen wird.



DIE INSERT-TASTE

Gib ein: 30 REM ATRI

Oh, da haben wir einen Fehler gemacht. Wir haben ein A vergessen. Geh mit dem Cursor an das R.

Halte die CONTROL-Taste fest und drücke die INSERT-Taste. Links vom Cursor erscheint jetzt ein Leerraum, und der Cursor wird auf den freien Platz gestellt. Tippe jetzt ein A ein.

20 REM ATARI

Wenn Du die CONTROL- und die INSERT-Taste länger festhältst, werden eine ganze Menge Leerräume eingefügt.

Übung 4: Backspace, Cursor-Tasten, Einfügen, Löschen

DIE DELETE-TASTE

Diese Taste ist das Gegenteil der INSERT-Taste.

Stell den Cursor über das zweite A von ATARI.

Halte die CONTROL-Taste fest und drücke die DELETE-Taste. Das A verschwindet, und Du siehst wieder:

```
20 REM ATRI
```

Ist doch wirklich leicht, oder?

Bei folgenden Tasten mußt Du die CONTROL-Taste festhalten:

Bei den Pfeiltasten, wenn Du an einen bestimmten Ort am Bildschirm gelangen willst.

Bei der DELETE-Taste, wenn Du löschen willst.

Bei der INSERT-Taste, wenn Du einen Buchstaben einfügen willst.

AUFGABE 4

1. Gib eine Zeile ein und verwende die Pfeiltasten, um den Cursor in dieser Zeile hin und her zu bewegen. Ändere Buchstaben in der Zeile. Drücke die RETURN-Taste, wenn Du fertig bist, damit die Zeile im Speicher abgelegt wird.

2. Gib eine Zeile ein. Gehe mit Hilfe der Pfeiltasten in die Mitte der Zeile. Lösche mit der DELETE-Taste ein paar Buchstaben. Drücke RETURN. Sieh Dir mit LIST dann das Programm an.

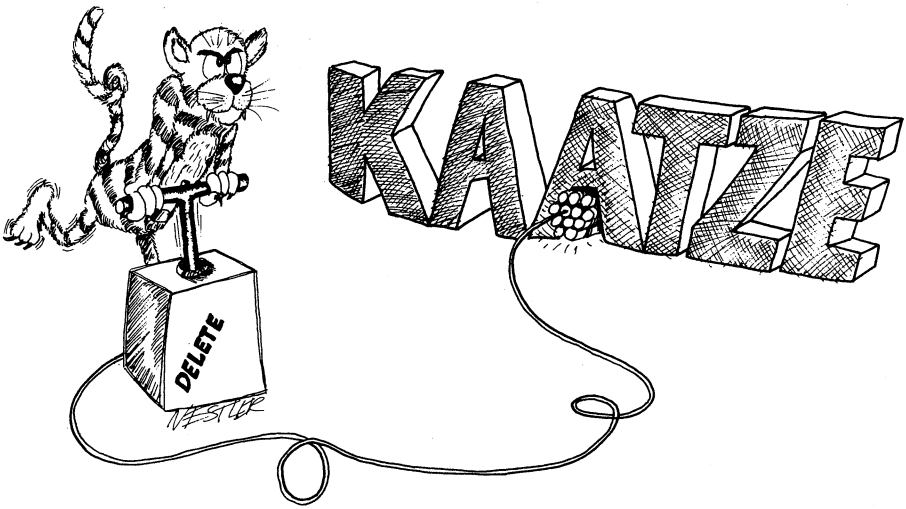
3. Gib eine Zeile ein und gehe mit den Pfeiltasten in die Mitte. Füge mit der INSERT-Taste einige Leerräume ein. Tippe dann etwas in den freigewordenen Platz. Drücke RETURN. Sieh Dir mit LIST das Programm an.

4. Zeichne ein lächelndes Gesicht.

5. Verbessere folgende Zeilen:

```
10 REM KAATZE            (KATZE)

10 REM TTIIGGEEEEEERR    (TIGER)
```



5. ANMERKUNGEN FÜR DEN LEHRER Zeichenkettenschachteln, DIM, LET

Der LET-Befehl wird wieder mit dem Schachtelmodell erklärt. DIM wird eingeführt.

Mit dem Schachtelmodell wird klar, daß LET ein Zuweisungsbefehl ist und nicht eine "Ist gleich"-Beziehung im mathematischen Sinn darstellt.

Mit der Schachtelvorstellung kann man die Konzepte "Variablennamen" und "Variablenwert" sehr gut unterscheiden. Der Name befindet sich außen an der Schachtel, der Wert ist in der Schachtel. Der Inhalt der Schachtel kann herausgenommen und durch einen neuen ersetzt werden.

Genauer gesagt, es wird eine Kopie des Inhalts angefertigt und verwendet. Der Originalinhalt bleibt erhalten.

Bis jetzt wurden folgende Befehle verwendet:

NEW, PRINT, REM, RUN, LIST, LET

Folgende Sondertasten wurden erklärt:

RETURN, SHIFT, CONTROL, ESC, BACKSPACE, INSERT, CLEAR, vier Pfeiltasten

Fragen:

1. Warum muß man den DIM-Befehl verwenden?

2. Laß dieses kleine Programm laufen:

```
10 DIM A$(25),B$(25)
20 LET A$="HALLO"
30 LET B$=A$
```

   Was befindet sich in Schachtel A$, was in B$?

3. Wie wird in diesem Programm:

```
10 DIM A$(5)
20 Q$="MUTTER"
```

   MUTTER genannt? Wie lautet der Name der Zeichenkettenvariable? Welchen Wert hat die Zeichenkettenvariable, nachdem Zeile 10 gelaufen ist? Welchen Wert hat sie, nachdem Zeile 20 gelaufen ist?

4. Was ist in diesem Programm falsch?

```
10 DIM H$(5)
20 LET H$="FETTE SOSSE"
```

ÜBUNG 5: ZEICHENKETTENSCHACHTELN, DIM, LET

ZEICHENKETTENKONSTANTEN

Zeichenkettenkonstanten kennst Du schon. Zum Beispiel:

```
"WICHTEL"
```

Du kannst die Zeichenkettenkonstante in einer PRINT-Anweisung verwenden:

Gib ein: 20 PRINT "WICHTEL"

SPEICHERSCHACHTELN

Wir wollen für jede Zeichenkette eine Speicherschachtel reservieren.

Zuerst müssen wir den Computer sagen, wie groß die Schachtel sein soll. Dazu verwenden wir den DIM-Befehl. Schreibe diese Zeilen ab:

```
10 REM ZEICHENKETTENSCHACHTEL
15 DIM W$(10)
```

Zeile 15 hat folgende Bedeutung:

> Hole eine Schachtel, die 10 Buchstaben aufbewahren kann.
> Schreibe den Namen W$ auf die Vorderseite.
> Laß die Schachtel vorerst leer.

DIM ist die Abkürzung für "dimension" und das heißt Größe oder Abmessung. Mit dem DIM-Befehl erhält die Schachtel einen Namen und die richtige Größe.

ZEICHENKETTENVARIABLEN

Der DIM-Befehl erzeugt eine Zeichenkettenvariable.

Ihr Name ist W$.

> Regel:
> Zeichenkettenvariable enden immer mit einem Dollarzeichen ($). Du kannst jeden Buchstaben als Namen wählen und hängst dann ein Dollarzeichen an.

W$ wird deshalb als Variable bezeichnet, weil zu verschiedenen Zeiten verschiedene Zeichenketten in die Schachtel gelegt werden können. Auf einmal kann die Schachtel allerdings nur eine Zeichenkette enthalten.

Die Speicherschachtel kann fast jede Größe annehmen: von einem Buchstaben bis zu ein paar Tausend Buchstaben, wenn der Speicher Deines Computers groß genug ist.

DAS AUFFÜLLEN DER SCHACHTEL

Mit dem LET-Befehl werden Schachteln mit Inhalt gefüllt. Gib ein und starte:

```
10 REM SPEICHERSCHACHTEL
12 PRINT "clear"
15 DIM W$(10)
20 LET W$="WICHTEL"
40 PRINT W$
```

Der Computer führt folgendes durch:

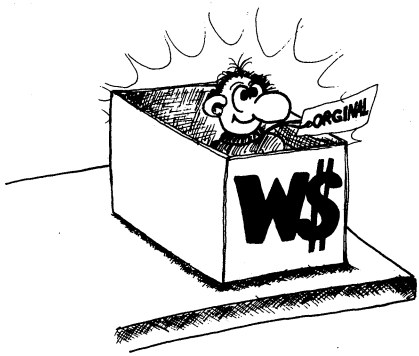
Zeile 12 Der Computer löscht den Bildschirm.

Zeile 15 Er nimmt eine Schachtel, die groß genug ist, daß 10 Buchstaben hineinpassen und schreibt den Namen W$ auf die Vorderseite.

Übung 5: Zeichenkettenschachteln, DIM, LET

Zeile 20 Der Computer nimmt die Zeichenkette "WICHTEL" und legt sie in die Schachtel mit dem Namen W$. (Die Schachtel kann 10 Buchstaben aufnehmen. WICHTEL paßt, denn das Wort hat nur 7 Buchstaben.)

Zeile 40 Der Computer erkennt, daß er ausdrucken soll, was in der Schachtel mit dem Namen W$ ist. Er fertigt vom Inhalt der Schachtel eine Kopie an. Das ist die Zeichenkette "WICHTEL". Diese Kopie schreibt er auf den Bildschirm. Die Zeichenkette "WICHTEL" befindet sich noch immer in der Schachtel.



NAMEN UND WERTE

Der Name der Variablen lautet W$. Der Wert der Variablen ist in der Schachtel. Im obigen Programm ist das die Zeichenkette "WICHTEL".

VIELE SCHACHTELN

Mit dem DIM-Befehl kann man mit einmal mehrere Schachteln vorbereiten. Zum Beispiel:

```
10 DIM A$(8), Z$(40), D$(10)
```

Zeile 10 erstellt drei Variable. Sie heißen A$, Z$ und D$.

REGELN FÜR SCHACHTELN

1. Bevor Du etwas in eine Schachtel geben kannst, mußt Du zuerst mit DIM eine erstellen.

2. Jede Schachtel muß einen anderen Namen haben. Sie darf nur einmal vorkommen.

3. Du kannst so viele DIM-Befehle angeben, wie Du willst, aber jede Schachtel muß einen anderen Namen haben.

NOCH EIN BEISPIEL

Gib ein und starte:

```
10 DIM A$(40)
12 DIM Z$(40), D$(40)
15 LET D$="SAURE GURKEN"
20 LET A$=" UND "
30 PRINT "WAS PASST ZU SAUREN GURKEN?"
35 LET Z$="SCHLAGSAHNE"
40 PRINT "clear"
50 PRINT D$;A$;Z$
```

Übung 5: Zeichenkettenschachteln, DIM, LET

Beschreibe die Ausführung jeder Zeile:

10 _____

12 _____

15 _____

20 _____

30 _____

35 _____

40 _____

50 _____

AUFGABE 5

1. Schreibe ein eigenes Programm, in dem der DIM- und der LET-Befehl vorkommt und erkläre, wie er Dinge in die "Schachteln" steckt.

2. Schreibe ein Programm, das eine Schachtel mit Namen N$ erstellt. Verwende dann LET, um Deinen Vornamen in die Schachtel zu geben. Laß dann Deinen Vornamen drucken (mit PRINT). Anschließend sollst Du den Namen Deines Freundes in dieselbe Schachtel legen. Laß auch ihn ausdrucken.

6. ANMERKUNGEN FÜR DEN LEHRER Der INPUT-Befehl

Diese Übung wiederholt das Modell der Speicherschachteln, die Zeichenkettenvariablen, den DIM- und den LET-Befehl. Der INPUT- und der SETCOLOR-Befehl für GRAPHICS-0-Darstellung werden erklärt.

Bis Übung 14 erklären wir die Arbeitsweise der Befehle nur in ihrer einfachsten Art. Auf diese Weise können dem Schüler relativ schnell die wesentlichsten Befehle beigebracht werden. So sieht er wenigstens einen Silberstreif am Horizont und kann schon sinnvolle Programme schreiben. Die folgenden Befehle sollten zum Schreiben interessanter Programme parat sein:

PRINT	Ausgabe
INPUT	Eingabe
GOTO	unendliche Schleifen
IF	Bedingungen und Abzweigungen
RND	Zufallszahlen für Spiele

Die Einführung der Zeichenkettenvariablen erfolgt wieder über das Schachtelmodell. Bei den Namen der Variablen beschränken wir uns auf einen Buchstaben. Dadurch vermeiden wir Verwirrung in kurzen Programmen und können den Text schneller eingeben.

Wir arbeiten möglichst lange mit Zeichenketten und vermeiden Zahlen, da Zeichenketten in Programmen weit häufiger Anwendung finden und weniger verwirrend beim Programmieren sind.

Mit dem SETCOLOR-Befehl kann man im GRAPHICS-0-Modus Bildschirm- und Bildschirmrandfarben ändern.

Fragen:

1. Welche unterschiedlichen Dinge legt der Computer in Schachteln ab?

2. Wie wird man in einem Programm aufgefordert, eine Eingabe zu machen?

3. Wie weiß man, daß der Computer auf eine Eingabe wartet?

4. Wie nennt man einen Buchstaben mit einem nachfolgenden Dollarzeichen ($)?

5. Schreibe ein kurzes Programm, in dem DIM, LET, PRINT und INPUT verwendet werden.

6. Wie änderst Du die Farbe des Bildschirms in Grün?

ÜBUNG 6: DER INPUT-BEFEHL

WIE MAN SCHACHTELN ERSTELLT

Bevor man eine Zeichenkette in eine Schachtel legen kann, muß man sie erst erschaffen.

Gib ein:
```
10 REM ZEICHENKETTENSCHACHTELN
15 DIM B$(30), C$(40)
```

Regel:

Bevor mit einer Variablen gearbeitet werden kann, muß sie mit DIM erstellt werden.

Vergißt Du das, gibt der Computer aus:

```
ERROR- 9
```

wenn Du das Programm laufen läßt.

WIE MAN DIE SCHACHTELN FÜLLT

Die Schachteln B$ und C$ sind noch leer.

Es gibt zwei Wege, wie man etwas in die Schachteln hineinlegen kann:

1. Du kannst den LET-Befehl verwenden.

2. Du kannst den INPUT-Befehl verwenden.

Übung 6: Der INPUT-Befehl

DER LET-BEFEHL

Schreibe diese Zeile dazu:

```
20 LET B$="SAG ETWAS"
```

Zeile 20 nimmt die Zeichenkette "SAG ETWAS" und legt sie in der Variable B$ ab.

B$ ist der "Name" der Zeichenkettenvariablen.

"SAG ETWAS" ist der der "Wert" der Zeichenkettenvariablen.

Der Wert hat 9 Zeichen. (Ein Leerraum und 8 Buchstaben.)

"SAG ETWAS" paßt in die Speicherschachtel B$ hinein, weil diese ja für 30 Zeichen ausgelegt ist. (Das hast Du bereits mit dem DIM-Befehl in Zeile 15 getan.)

DER INPUT-BEFEHL

Durch den INPUT-Befehl fragt der Computer nach etwas.

Der INPUT-Befehl ist der zweite Weg, mit dem man etwas in eine Speicherschachtel ablegen kann.

Bis jetzt sieht das Programm so aus:

```
10 REM ZEICHENKETTENSCHACHTELN
15 DIM B$(30), C$(40)
20 LET B$="SAG ETWAS"
```

Gib ein:

```
22 PRINT B$
25 INPUT C$
30 PRINT
35 PRINT "HAST DU "
40 PRINT "'";C$;"' GESAGT?"
```

Laß es laufen. Wenn Du ein Fragezeichen siehst, gib "HALLO" ein und drücke die RETURN-Taste.

Das Fragezeichen wurde von INPUT in Zeile 25 geschrieben. Der Cursor steht hinter dem Fragezeichen, und das bedeutet, daß der Computer erwartet, daß Du etwas eingibst.

Gibst Du ihm das Wort "HALLO", dann legt es der Computer in die Schachtel mit dem Namen C$.

Laß das Programm noch einmal laufen und gib diesmal etwas Lustiges ein.

FARBTÖPFE MIT 16 FARBEN

Der ATARI besitzt fünf Farbtöpfe, die mit 0, 1, 2, 3 und 4 numeriert sind.

Wir müssen die Töpfe 2 und 4 verwenden.

ATARI numeriert seine Farben von 0 (Grau) bis 15 (Orange).

Versuch das nächste Programm:

```
10 REM FARBIGER BILDSCHIRM
20 PRINT "FARBTOPF 2 IST DIE BILDSCHIRMFARBE"
30 PRINT "WELCHE FARBE MOECHTEST DU?"
33 PRINT "GIB ZAHLEN ZWISCHEN 0 UND 15 EIN"
36 INPUT F
40 PRINT "WIE HELL <0 BIS 14>"
45 INPUT H
50 SETCOLOR 2,F,H
60 GOTO 30
```

In Zeile 50 wird die Farbe und die Helligkeit von Farbtopf 2 festgelegt.

Es ist so ähnlich, wie wenn Du einen sauberen, leeren Farbeimer nimmst und eine Farbe (durch eine Zahl) hineinschüttest, mit der Du dann den Bildschirm bemalen kannst.

Übung 6: Der INPUT-Befehl

Laß das Programm laufen und probiere alle Farben von 0 bis 15 durch. Gib bei jeder Farbe für die Helligkeit gerade Zahlen zwischen 0 und 14 ein.

Schwarz ist Farbe 0 mit Helligkeit 0.

(Dieses Programm verwendet die Zahlenvariablen H und F. In Übung 12 erfährst Du mehr darüber.)

Zeile 60 enthält eine GOTO-Anweisung. In der nächsten Übung wird dieser Befehl erklärt.

FARBIGER BILDSCHIRMRAND

Mach im obigen Programm folgende Änderungen:

```
10 REM FARBIGER BILDSCHIRMRAND
20 PRINT "FARBTOPF 4 IST FUER DIE RANDFARBE"
30 PRINT "WELCHE RANDFARBE MOECHTEST DU?"

50 SETCOLOR 4,F,H
```

Die Zeilen 33, 36, 40, 45 und 60 darfst Du nicht verändern.

AUFGABE 6:

1. Schreibe ein Programm, das den Namen einer Person verlangt und gib einen dummen Spruch aus, der an die Person (mit Namen) gerichtet ist.

2. Schreibe ein Programm, in dem Du nach Deinem Lieblingsessen gefragt wirst (mit INPUT). Es soll unter dem Namen E$ in eine Schachtel abgelegt werden. Dann wirst Du nach Deinem Lieblingstier gefragt. Es soll ebenfalls unter E$ in einer Schachtel abgelegt werden. Anschließend soll E$ ausgedruckt werden. Was wird auf dem Bildschirm stehen? Starte das Programm und kontrolliere, ob Du recht hast.

7. ANMERKUNGEN FÜR DEN LEHRER Tricks beim Drucken

In dieser Übung:

PRINT mit einem Semikolon am Ende

PRINT mit Semikolons zwischen Ausdrücken

Der "unsichtbare" PRINT-Cursor

PRINT mit Kommas zwischen Ausdrücken

GRAPHICS-1,-2-Bildschirme, eine Einführung

Diese Übung befaßt sich mit dem Ausgabe-Cursor, der auf dem Bildschirm nicht zu sehen ist. Er legt den Platz fest, auf den das nächste Zeichen von einem PRINT-Befehl gesetzt wird. (Der Eingabe-Cursor ist ein Quadrat. Er ist bereits in der Übung über den Editiermodus und den INPUT-Befehl erklärt worden.)

Endet eine PRINT-Anweisung mit einem Semikolon, bleibt der Ausgabe-Cursor stehen, und das nächste PRINT-Kommando schreibt an dieser Stelle weiter.

Ohne Semikolon am Ende setzt der PRINT-Befehl zum Schluß den Cursor an den Anfang der nächsten Zeile.

Mit einer PRINT-Anweisung können mehrere Ausdrücke ausgegeben werden: Zeichenketten- und numerische Konstanten, Variablen und Funktionen. Numerische Konstanten und Variablen haben wir bisher noch nicht besprochen.

In einer Zeichenkette müssen alle Zeichen ohne Abstand hintereinanderstehen. Sollen Leerräume dazwischen sein, wie zum Beispiel in "SCHINKEN MIT SPECK", muß die Zeichenkette in Anführungszeichen stehen.

Die 11 Grafikmodi des ATARI erfordern einige aufwendige Befehle. Wir geben eine erste Einführung.

Fragen:

1. Welcher Cursor besteht aus einem Quadrat?

2. Welcher Cursor ist unsichtbar? Bei welchem Befehl wird er eingesetzt?

3. Wie stellst Du es an, daß zwei PRINT-Anweisungen in derselben Zeile ausgeführt werden?

4. Steht nach der Ausführung des Programms zwischen den beiden Worten ein Leerraum?

```
10 PRINT "HALLO";"DU!"
```

   Wenn nicht, wie bringst Du Leerzeichen dazwischen?

5. Was verursachen Kommas in Ausgabezeilen?

ÜBUNG 7: TRICKS BEIM DRUCKEN

EINE ZEILE ODER VIELE?

Gib dieses Programm ein:

```
10 REM MAHLZEIT
20 PRINT
30 PRINT "SCHINKEN"
40 PRINT "MIT"
50 PRINT "SPECK"
```

Starte es. Jede PRINT-Anweisung druckt eine eigene Zeile aus.

Gib jetzt ein:

```
30 PRINT "SCHINKEN ";
40 PRINT "MIT ";
```

(Ändere oder lösche die anderen Zeilen nicht.) Achte sorgfältig darauf, daß Du am Ende von SCHINKEN und von MIT jeweils einen Leerraum und ein Semikolon eintippst.

Laß es laufen.

Welchen Unterschied stellst Du zum vorigen Programm fest?

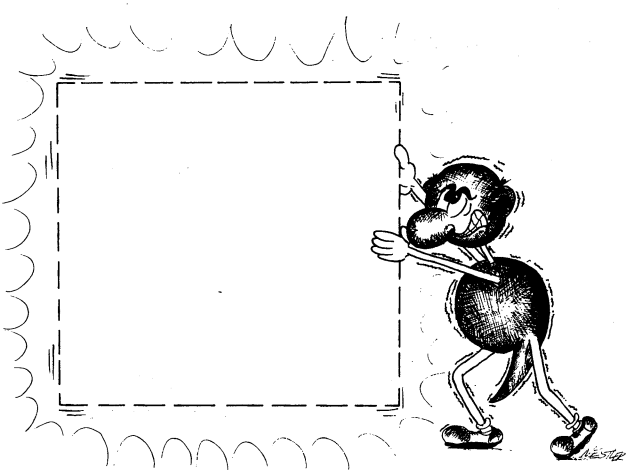
DER VERSTECKTE CURSOR

Erinnerst Du Dich noch an das blinkende Quadrat? Das ist der Eingabe-Cursor, der angibt, an welcher Stelle das nächste Zeichen am Bildschirm zu sehen ist.

Der PRINT-Befehl besitzt ebenfalls einen Cursor, aber er ist unsichtbar. Er kennzeichnet, an welcher Stelle der nächste Buchstabe erscheint, wenn der Computer druckt (PRINT).

Regel:

Das Semikolon veranlaßt den unsichtbaren PRINT-Cursor, an der letzten Stelle stehen zu bleiben. Mit dem nächsten PRINT-Befehl wird an dieser Stelle weitergeschrieben.



BERÜHMTE PAARE

Mit der PRINT-Anweisung können hintereinander mehrere Zeichenketten ausgegeben werden. Betrachte die Zeile 80:

```
10 PRINT "clear"
15 DIM A$(30),B$(30)
20 PRINT "GIB EINEN NAMEN EIN"
30 INPUT A$
35 PRINT "clear"
40 PRINT "UND NOCH EINEN"
50 INPUT B$
60 PRINT "clear"
70 PRINT "HIER IST DAS BERÜHMTE PAAR"
75 PRINT
80 PRINT A$;" UND ";B$
```

(Erinnere Dich: "clear" bedeutet, daß Du erst die ESC-Taste drükken und wieder loslassen, dann die SHIFT-Taste festhalten und die CLEAR-Taste drücken mußt.)

Vergiß die Leerzeichen vor und nach UND in Zeile 80 nicht.

ZUSAMMENGEDRÜCKT ODER AUSEINANDERGEZOGEN?

Gib NEW und dann die nächste Zeile ein:

```
10 PRINT "ROCK";"UND";"ROLL"
```

Nach der Ausführung tippe folgendes:

```
10 PRINT "ROCK ","UND","ROLL"
```

Regel:
Wenn Du in einer PRINT-Anweisung ein Komma zwischen zwei Ausdrücke schreibst, bringt der Computer den Text mit großem Abstand.

GROSSE BUCHSTABEN

Der ATARI-Computer kann auf 11 verschiedenen Arten den Bildschirm des Fernsehgeräts benutzen.

GRAPHICS 0	der Zustand nach dem Einschalten (Du arbeitest normalerweise damit).
GRAPHICS 1	breite farbige Buchstaben
GRAPHICS 2	große farbige Buchstaben

Laß das Programm laufen:

```
10 GRAPHICS 1
20 PRINT #6; "HALLO"
30 PRINT "clear MEIN FREUND"
```

Damit Du wieder zum Normalzustand zurückkehren kannst, mußt Du die RESET-Taste auf der rechten Seite drücken.

Verwende #6 mit PRINT, wenn Du große Buchstaben darstellen möchtest.

Verwende #6 nicht, wenn Du in dem kleinen unteren Bildfenster schreiben willst.

Versuch das Ganze noch einmal mit einer "2" anstelle einer "1".

DER AUFGETEILTE BILDSCHIRM

Sowohl im GRAPHICS-1- wie auch im GRAPHICS-2-Modus wird der Bildschirm aufgeteilt. Unter dem Bildschirm mit großen Buchstaben befindet sich ein zweites Bildschirmfenster mit normalen Buchstaben.

Willst Du, daß alle Buchstaben groß geschrieben werden, mußt Du folgendes eingeben:

```
10 GRAPHICS 1+16
20 PRINT #6; "NICHT UNTERTEILT"
30 GOTO 30
```

Versuchs auch mit GRAPHICS 2+16.

Zeile 30 ist eine unendliche Schleife. Sie sorgt dafür, daß das Programm nie aufhört zu laufen. Wenn das Programm beendet wird (das kannst Du mit der BREAK-Taste erreichen), kehrt der Bildschirm automatisch in den GRAPHICS-0-Modus zurück.

Den GOTO-Befehl besprechen wir in Übung 8.

AUFGABE 7

1. Schreibe ein Programm, das nach dem Namen einer Musikgruppe und nach einem ihrer Lieder fragt. Gib mit nur einem PRINT-Befehl den Namen der Gruppe und ihr Lied aus. Dazwischen soll "SPIELT" stehen.

2. Die gleiche Aufgabe soll mit drei PRINT-Anweisungen gelöst werden.

3. Gib nun den Text in großen Buchstaben aus (GRAPHICS 2).

8. ANMERKUNGEN FÜR DEN LEHRER Der GOTO-Befehl und die BREAK-Taste

Mit dem GOTO-Befehl können Schleifen erstellt werden, die sich unendlich oft wiederholen können. Nach Einführung des IF-Kommandos verbessert er in späteren Programmen auch den Befehlsfluß. Der Befehl macht es dem Schüler leicht, sich langsam mit der Vorstellung vertraut zu machen, daß es für den Befehlsfluß nicht nötig ist, daß das Programm stur die durchnumerierten Zeilen hintereinander durchläuft.

Im Augenblick besteht aber seine Hauptaufgabe darin, die Programme für eine gewisse Zeitdauer laufen zu lassen. Bei jedem Schleifendurchgang kann etwas geändert werden.

Das Problem ist nur, diesen Vorgang abzubrechen. Die BREAK-Taste ist dazu recht brauchbar.

Wir kennen jetzt schon drei der vier Hauptelemente, die ein echtes Programmieren ermöglichen. Diese sind die PRINT-, INPUT- und GOTO-Befehle. Was noch fehlt ist der IF-Befehl, der aus dem Computer, der bis jetzt nur eine Art Kassettenrecorder darstellte, eine Maschine macht, die Situationen abschätzen und entsprechende Entscheidungen treffen kann.

Fragen:

1. Was erscheint auf dem Bildschirm, wenn man dieses kleine Programm laufen läßt?

```
10 PRINT "HALLO"
20 GOTO 40
30 PRINT "LIEBER"
40 PRINT "VATI"
```

2. Und bei diesem?

```
10 PRINT "APFEL"
20 PRINT "TASCHE";
30 GOTO 20
```

3. Wie wird das Programm in Frage 2 abgebrochen?

4. Schreibe ein kurzes Programm, das nach Deinem Lieblingsfilmstar fragt und seinen Namen immer wieder druckt.

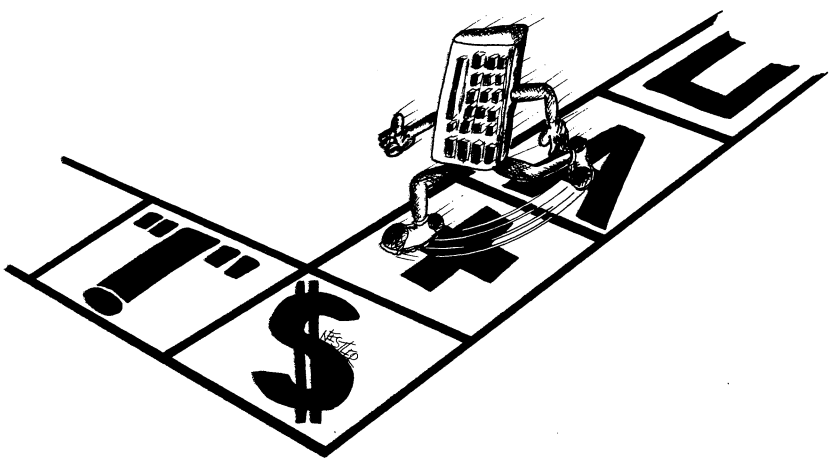
ÜBUNG 8: DER GOTO-BEFEHL UND DIE BREAK-TASTE

DAS HERUMSPRINGEN IN DEINEM PROGRAMM

Versuche dieses Programm:

```
10 REM DURCHLAUF
12 DIM N$(40)
15 PRINT "clear"
20 PRINT "DEIN NAME?"
25 INPUT N$
30 PRINT N$
35 PRINT
40 GOTO 30
```

Starte dieses Programm mit RUN. Es wird nie von selbst zu laufen aufhören! Du mußt die BREAK-Taste drücken, damit Dein Name nicht immer wieder an Deinen Augen vorbeisaust.



Zeile 40 verwendet den GOTO-Befehl. Es ist wie bei dem "Gehe ins Gefängnis" im Monopolyspiel. Jedesmal, wenn der Computer die Zei-

le 40 erreicht, muß er zurück zu Zeile 30 gehen und wieder Deinen Namen ausgeben.

Wir werden GOTO in vielen Programmen verwenden.

NOCH MEHR SPRÜNGE

Gib ein:
```
10 REM OHNE UNTERBRECHUNG
15 DIM S$(40)
20 PRINT "SAG ETWAS"
30 INPUT S$
40 PRINT "SAGTEST DU '";S$;"'?"
45 PRINT
50 GOTO 30
```

Starte das Programm. Gib jedesmal eine Antwort, wenn Du das "?" und den Cursor siehst. Drücke die BREAK-Taste, wenn Du das Programm abbrechen willst.

Beachte den Pfeil, der von Zeile 50 zu Zeile 30 geht. Er zeigt an, was GOTO bewirkt. Du solltest vielleicht auch in Deinen Programmlistings solche Pfeile einzeichnen.

WIE MAN SPRINGT

Es gibt nur zwei Möglichkeiten zu springen: vorwärts oder rückwärts.

Das Zurückspringen ergibt eine Schleife.

```
10 PRINT "HALLO"
20 GOTO 10
```

Der Weg durch das Programm sieht folgendermaßen aus:

```
10 PRINT "HALLO"

20 GOTO 10
```

Übung 8: Der GOTO-Befehl und die BREAK-Taste

Der Computer macht die Schleife immer wieder durch. Drücke die BREAK-Taste, um sie abzubrechen.

Beim Vorwärtsspringen werden Teile des Programms ausgelassen. Wir werden später beim IF-Befehl erfahren, wozu dies notwendig ist.

DIE BREAK-TASTE

Die BREAK-Taste ist ein "Lebensretter". Wenn Du Schwierigkeiten hast, mußt Du die BREAK-Taste drücken. Der Computer wird dann das Programm anhalten und auf Deinen nächsten Befehl warten. Dein Programm ist aber immer noch im Speicher.

Wenn Du große Schwierigkeiten hast, mußt Du die Taste RESET drükken. Sie befindet sich ganz rechts auf der Tastatur. Dein Programm ist auch in diesem Fall noch sicher im Speicher.

(Wenn der Computer auf gar nichts mehr reagiert, egal welche Taste Du drückst, sagt man: der Computer hat sich "aufgehängt".)

AUFGABE 8

1. Schreibe ein Programm, das Deinen Vornamen immer wieder ausgibt.

2. Wie hältst Du das Programm an?

3. Schreibe noch ein Programm, das zuerst Deinen Namen und dann den Deines Freundes auf getrennten Zeilen immer wieder ausdruckt. Gib jeden Namen in einer anderen Farbe aus. Brich das Programm mit der BREAK-Taste ab.

4. Schreibe ein Programm, das folgende Befehle benutzt: PRINT, INPUT, LET, GOTO. Es sollte auch zwei Zeichenketten (oder Strings) zusammenfügen.

9. ANMERKUNGEN FÜR DEN LEHRER Der IF-Befehl

In diesem Kapitel wird der IF-Befehl eingeführt. Es wird der Fall behandelt, in dem zwei Zeichenketten gleich oder nicht gleich sind.

IF ist ein sehr wichtiger Befehl, der den Computer zu einem denkenden Gerät macht. Er ist ein etwas schwieriger Befehl, bei dem der Schüler etwas mehr Hilfe brauchen wird.

Der IF-Befehl erfordert sowohl verbale als auch visuelle Vorstellungskraft. Die "Kuchen"- und die "Weggabelungs"-Zeichnungen illustrieren diese Idee. Bei der Einführung des GOTO-Befehls wurde festgestellt, daß der Datenfluß verändert werden kann. Hinzu kommt jetzt der Entscheidungstest: wenn ein Ausdruck wahr ist, geschieht etwas; wenn er falsch ist, etwas anderes.

Die "Bedingung A" steht für eine Behauptung, die auf ihren Wahrheitsgehalt hin untersucht wird. Der Ausdruck "Befehl C" wird verwendet, wenn die Behauptung richtig ist.

In den Behauptungen sind zwei Vorstellungsebenen enthalten. Die formale Ebene besteht aus "gleich" (=) und "nicht gleich" (<>).

Da die "Ungleich"-Zeichen (<>) für den Schüler wahrscheinlich ungewohnt sind, wäre es möglich, daß sie an diesem Punkt größere Schwierigkeiten bekommen. In diesem Fall sollte der IF-Befehl mit dem einfachsten Fall, dem positiven Vergleich (=) geübt werden. Wir werden später in diesem Buch noch einmal auf IF zurückkommen, und bis dahin sollte der Schüler mit diesem Befehl umgehen können. Der erweiterte Zeichensatz der Beziehungen:

< > = =< => <>

wird dort behandelt.

Fragen:

1. Wie bringst Du dieses Programm dazu, "DAS IST GUT" zu drucken?

```
10 DIM T$(5)
15 PRINT "TUT DEIN ZEH WEH?"
17 INPUT T$
20 IF T$="NEIN" THEN GOTO 90
40 IF T$="ETWAS" THEN GOTO 15
90 PRINT "DAS IST GUT"
```

2. Schreibe ein kurzes Programm, das Dich fragt, ob Du Schokoladen- oder lieber Vanilleeis magst? Die Antworten sollen entweder "S" oder "V" sein. Laß für das "S" "Lecker!" ausgeben. Für die "V"-Antwort soll der Computer "Mmmmmmmmm!" drucken.

ÜBUNG 9: DER IF-BEFEHL

Lösche den Speicher und gib ein:

```
10 PRINT "clear"
15 DIM A$(10)
20 PRINT "BIST DU HAPPY? (JA ODER NEIN)"
30 INPUT A$
40 IF A$="JA" THEN PRINT "DAS FREUT MICH"
50 IF A$="NEIN" THEN PRINT "SCHADE"
```

Laß das Programm einige Male laufen. Untersuche das Programm, indem Du "JA", "NEIN" oder "VIELLEICHT" eingibst. Was passiert?

JA _____

NEIN _____

VIELLEICHT _____

DIE IF-ANWEISUNG

Die IF-Anweisung besteht aus zwei Teilen:

```
10 IF    Bedingung A    THEN    Befehl C
```

Zuerst sieht sich der Computer "Bedingung A" an. Wenn sie richtig ist, führt der Computer "Befehl C" aus.

Wenn "Bedingung A" nicht richtig ist, geht der Computer zur nächsten Zeile über, ohne den "Befehl C" auszuführen.

Dies sieht etwa so aus:

```
10 IF Bedingung A wahr ist THEN führe Befehl C aus
und gehe zur nächsten Zeile
```

oder

```
10 IF Bedingung A falsch ist THEN
gehe zur nächsten Zeile
```

Übung 9: Der IF-Befehl

EINIGE BEISPIELE MIT IF

```
10 REM IF-TEST-PROGRAMM
50 IF   A$="JA"          THEN  PRINT "GUT"
55 IF   "JA"=A$          THEN  PRINT "GUT"
60 IF   A$=B$            THEN  LET C$="SO NICHT"
70 IF   N$="VOGEL"       THEN  PRINT "piepser"
75 IF   A$="FERTIG"      THEN  PRINT "ESSE KUCHEN"
```

Zeile 50 und 55 sind identisch.

AUFGABE 9A:

1. Füge die nächsten Zeilen zum Programm hinzu:

```
12 DIM A$(10), B$(10), C$(10), N$(10)
15 INPUT A$
17 LET C$="WAS?"
20 LET B$="BEZAHLE"
25 LET N$=A$
85 PRINT C$
99 GOTO 15
```

   Laß das Programm laufen und gib diese Wörter ein:

   JA, VOGEL, FERTIG, SO NICHT

   Laß Dir auch selbst noch welche einfallen. Sieh Dir an, was am Bildschirm ausgedruckt wird und vergleiche, ob der IF-Befehl so wirkt, wie Du es Dir vorgestellt hast. (Erinnere Dich, wenn "Bedingung A" wahr ist, wird die Anweisung nach THEN ausgeführt.)

2. Lösche den Speicher und schreibe ein Programm, das danach fragt, ob Du ein "JUNGE" oder ein "MAEDCHEN" bist. Lautet die Antwort "JUNGE", soll "DRACHEN STEIGEN" ausgedruckt, lautet sie "MAEDCHEN", soll "BALL SPIELEN" ausgegeben werden.

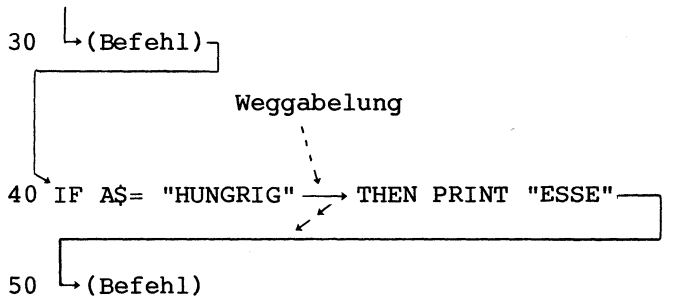
EINE WEGGABELUNG

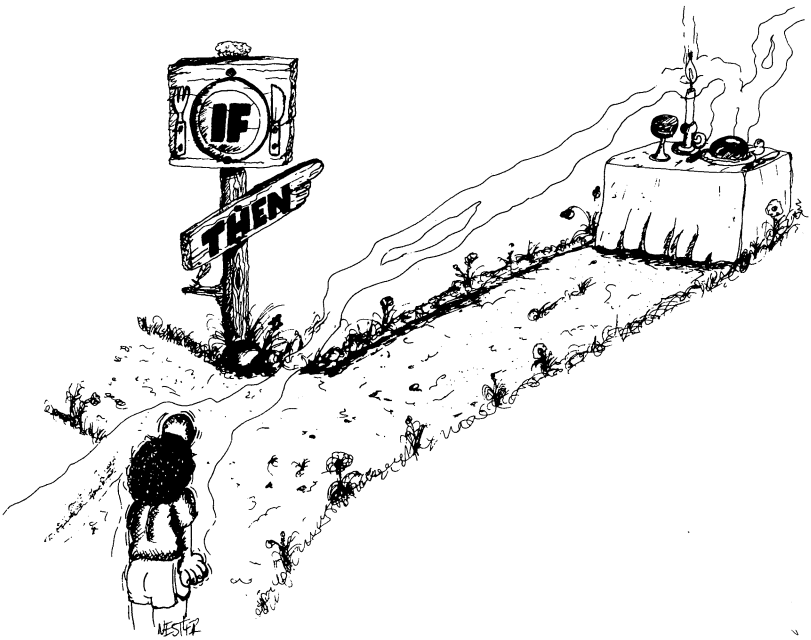
Wenn der Computer "IF" sieht, muß er sich entscheiden, welchen Weg er gehen soll.

Wenn die "Bedingung A" wahr ist, muß er in Richtung "THEN" gehen und den Befehl ausführen, den er dort findet. Dann geht er zur nächsten Zeile.

Wenn die "Bedingung A" falsch ist, geht er geradewegs zur nächsten Zeile.

Hier ist die Straßenkarte mit der eingezeichneten Weggabelung:





DAS "UNGLEICH"-ZEICHEN

Das Symbol "<>" bedeutet "ungleich". Es ist das Gegenteil des "="-Zeichens in einer Bedingung.

Um "<>" zu erzeugen, mußt Du die SHIFT-Taste gedrückt lassen und zuerst die "<"- und dann die ">"-Taste drücken.

```
40 IF    Bedingung A    THEN PRINT  "KEIN RAUCH"
```

Die "Bedingung A" muß WAHR (engl. TRUE) oder FALSCH (engl. FALSE) sein. Ist sie wahr, gibt der Computer "KEIN RAUCH" aus. Betrachte den Ausdruck, der für "Bedingung A" steht:

```
Q$<>"FEUER"
```

Gib ihn in die Zeile mit dem IF-Befehl ein:

```
40 IF Q$<>"FEUER" THEN PRINT "KEIN RAUCH"
```

Enthält die Q$-Schachtel das Wort "KALT", ist Q$ ungleich "FEUER" und der Ausdruck Q$<>"FEUER" ist WAHR. Der Computer druckt dann "KEIN RAUCH". Enthält die Q$-Schachtel "FEUER", ist der Ausdruck:

```
Q$<>"FEUER"
```

FALSCH, und der Computer druckt nichts.

So sieht das Ganze in einem Programm aus:

```
10 DIM Q$(5)
15 PRINT "BRENNT DEIN HAUS?"
20 PRINT "(GIB 'FEUER' ODER 'KALT' EIN)"
30 INPUT Q$
40 IF Q$<>"FEUER" THEN PRINT "KEIN RAUCH"
50 IF Q$="FEUER" THEN PRINT "HILFE"
```

AUFGABE 9B:

1. Schreibe ein "Pizza"-Programm. Laß das Programm abfragen, womit die Pizza belegt werden soll. Laß den Computer bei jeder Auswahl etwas Lustiges antworten. Du kannst zwischen Pilzen, Peperoni, Sardellen, Schinken und so weiter auswählen. Du kannst auch nach der Größe fragen lassen.

2. Schreibe ein Farbratespiel. Ein Spieler gibt mit INPUT eine Farbe in die Zeichenkette C$ ein. Ein anderer Spieler versucht dann die Farbe zu erraten, indem er seine Farbe zu der Zeichenkette G$ eingibt. Verwende zwei IF-Zeilen. Eine enthält "Bedingung A"

   G$<>C$

   wenn falsch geraten wurde. Die zweite Zeile enthält ein "="-Zeichen, für den Fall, daß richtig geraten wurde. Der "Befehl C" druckt "RICHTIG" oder "FALSCH" aus.

10. ANMERKUNGEN FÜR DEN LEHRER Die Zahlen

In diesem Kapitel werden Rechenvariablen und -operationen eingeführt. Die LET-, INPUT- und PRINT-Befehle werden wiederholt.

Die Vorstellung des Speichers als "Regal voller Schachteln" wird ausgebaut. Auch hier beschränken wir uns vorläufig darauf, daß Variablennamen nur aus einem Buchstaben bestehen.

Das "*"-Symbol für Multiplikation ist dem Schüler wahrscheinlich fremd. Da sich bei der Division Dezimalzahlen ergeben, sollte Ihr Schüler schon damit vertraut sein. Den größten Anteil werden Addition und Subtraktion ausmachen, die Multiplikation wird nur in geringem Umfang vorkommen.

Es wird für den Schüler etwas ungewöhnlich sein, daß die Zahlen, die in den Zeichenkettenkonstanten stehen, keine "Zahlen" sind, die direkt in der Arithmetik angewandt werden können. Die VAL- und STR$-Funktionen werden später in diesem Buch eingeführt. Sie erlauben die Umwandlung von Zahlen und Zeichenketten.

Mit Hilfe von PRINT kann eine Mischung aus String- und Zahlenwerten ausgedruckt werden.

Der unübliche Gebrauch von "=" in BASIC, in der es "ersetzen" und nicht "gleich" bedeutet, zeigt sich sehr deutlich im folgenden Beispiel:

```
LET N=N+1
```

Übung 10: Die Zahlen

Fragen:

1. Zähle die drei "Schachtel"-Arten im Speicher auf. (Das heißt, genannt nach den Dingen, die in die Schachteln gesteckt werden.)

2. Erkläre, warum für einen Computer "N=N+1" nicht das gleiche ist wie "5=5+1" in der Mathematik.

3. Gib ein anderes Beispiel "schlechter Mathematik" unter Verwendung eines LET-Befehls. Verwende die "*"- oder "/"-Symbole.

4. Erkläre, was man unter "Variablenname" und "Variablenwert" fÜr Zahlen- und Zeichenkettenvariablen versteht.

5. Wenn die Schachteln A und B die Zahlen 5 und 8 beinhalten, dann erhalten wir mit PRINT A;B den Ausdruck 58. Wie mußt Du den Computer programmieren, damit er 5 8 ausdruckt?

ÜBUNG 10: DIE ZAHLEN

INPUT, LET UND PRINT

Bis jetzt haben wir ausschließlich Zeichenketten verwendet. Es können auch Zahlen eingesetzt werden. Gib ein und starte folgendes Programm:

```
10 REM GROESSER
20 PRINT "GIB MIR EINE ZAHL"
30 INPUT N
40 LET A=N+1
45 PRINT
50 PRINT "HIER IST EINE GROESSERE"
60 PRINT A
```

SPEICHERSCHACHTELN

Zahlen kann man genauso wie Zeichenketten in Speicherschachteln ablegen. Aber es gibt einen Unterschied:

Zeichenketten:
Du mußt mit dem Befehl DIM die Größe der Schachtel angeben.

Zahlen:
Alle Zahlenschachteln haben die gleiche Größe. Der DIM-Befehl wird nicht verwendet.

Anschließend findest Du die Erklärung des GROESSER-Programms:

Zeile 10 Eine REM-Anweisung. Der Computer ignoriert sie.

Zeile 20 Mit PRINT wird die Zeichenkette ausgegeben.

Übung 10: Die Zahlen

Zeile 30 Der Computer wartet darauf, daß ihm eine Zahl eingegeben wird (INPUT). Er nimmt die Zahl und sucht nach einer Schachtel mit dem Namen N. Er findet keine, weil N bisher im Programm noch nicht verwendet wurde. Deshalb nimmt der Computer eine neue Schachtel, schreibt N auf die Vorderseite und legt die Zahl hinein.

Zeile 40 Der Computer nimmt die Zahl aus Schachtel N, zählt eins dazu und legt sie in eine andere Schachtel ab, der er den Namen A auf die Vorderseite schreibt.

Zeile 45 Es wird eine Leerzeile gedruckt.

Zeile 50 Es wird eine Meldung ausgegeben.

Zeile 60 Der Computer fertigt von der Zahl in Schachtel A eine Kopie an und gibt sie aus (PRINT). Die Original-Zahl ist noch in der Schachtel.

ARITHMETIK

Das Minus-, das Istgleich-, das Plus- und das Multiplikationszeichen befinden sich auf den Pfeiltasten.

Computer verwenden zum Multiplizieren statt eines "x"-Zeichens ein "*"-Zeichen.

Versuche folgendes: Ändere Zeile 40 so um, daß N mit 5 multipliziert wird.

Computer verwenden zum Dividieren ein "/"-Zeichen. Es befindet sich auf der "?"-Taste.

Die Antworten werden in Dezimalzahlen gedruckt.

VARIABLEN

Der Name einer Schachtel, die eine Zeichenkette enthält, muß mit einem Dollarzeichen enden. Beispiele: N$, A$, Z$.

Der Name einer Schachtel, die eine Zahl enthält, besitzt kein Dollarzeichen. Beispiele: N, A, Z.

Das, was in die Schachtel gesteckt wird, wird "Wert" der Variablen genannt.

ARITHMETIK UND DER LET-BEFEHL

```
10 LET A=2
20 LET B=3
30 LET C=B-A
40 PRINT A;" ";B;" ";C
```

Noch mehr Beispiele:

```
10 LET B=15
20 LET A=B/5
30 LET X=A*4+2
40 PRINT X;" ";A
```

VORSICHT!
Zahlen und Zeichenketten unterscheiden sich. Beispiel: "1984" ist keine Zahl. Sie ist eine Zeichenkettenkonstante, weil sie in Anführungszeichen steht.

Regel:

Auch wenn eine Zeichenkette aus Zahlenzeichen besteht, ist sie trotzdem keine Zahl.

Einige Zahlenkonstanten: 5, 22, 3.14, -50

Einige Zeichenkettenkonstanten: "HALLO", "7", "ZWEI", "3.14".

Übung 10: Die Zahlen

Regel:

Du kannst nicht mit Zahlen in Zeichenketten rechnen.

Richtig: `10 LET A = 3 + 7`

Falsch: `10 LET A$ = 3 + 7`

Falsch: `10 LET A = "3" + "7"`

Wenn Du eine dieser falschen Zeilen laufen läßt, druckt der Computer:

```
ERROR-
```

Es gibt zwei Arten von Variablen: Zahlen und Zeichenketten.

Du kannst nicht eine Zahl in eine Zeichenkettenschachtel oder eine Zeichenkette in eine Zahlenschachtel stecken.

Gib ein:
```
10 DIM B$(15)
15 LET A=5
20 LET B$="10"
30 LET C=A+B$
```

Die Zeilen 10, 15 und 20 sind in Ordnung. Zeile 30 ist falsch. Was passiert, wenn Du dieses kleine Programm laufen läßt?

Versuche zu erraten, was jede der folgenden Anweisungen drucken wird, wenn Du diese Zeilen eingibst:

`PRINT 5` _____

`PRINT "5"` _____

`PRINT "5 + 3"` _____

`PRINT "5" + "3"` _____

`PRINT 5 + 3` _____

DER "GEMISCHTE" PRINT-BEFEHL

Du kannst Zahlen und Zeichenketten im gleichen PRINT-Befehl drukken. (Vergiß aber nicht, daß Du damit nicht rechnen kannst.)

Richtig: PRINT A; "SIEBEN";"7"

```
PRINT A;B$
```

Starte: 10 PRINT 5/2;" IST GLEICH 5/2"

Was erhältst Du? _____

DAS SELTSAME AN DEM IST-GLEICH-ZEICHEN

Das "="-Zeichen beim Programmieren bedeutet nicht wirklich "ist gleich". Schau Dir dieses Programm an:

```
10 LET N=N+1
```

Dies ergibt mathematisch keinen Sinn. Nimm an, daß N gleich 7 ist. Dies würde bedeuten, daß:

```
7=7+1
```

ist, was aber nicht stimmt.

Beim Programmieren ist es aber richtig, N=N+1 zu sagen, weil das "="-Zeichen in Wirklichkeit "ersetze" bedeutet. Betrachte die Gleichung:

```
10 LET N=N+1
```

Der Computer geht zu der Schachtel, die mit N bezeichnet ist.

Er nimmt die Zahl 7 aus der Schachtel.

Er addiert 1 zu der 7, um 8 zu erhalten.

Dann steckt er die 8 in die Schachtel.

Man kann den Vorgang auch anders erklären:

```
10   LET   N   =   N   +   1
```

bedeutet: Setze (neues N) ist gleich (altes N) plus Eins

AUFGABE 10

1. Schreibe ein Programm, das nach Deinem Alter und dem Jahr fragt. Subtrahiere die beiden Zahlen und drucke Dein Geburtsdatum. Verwende PRINT-Anweisungen, um auszudrukken, was gewünscht wird und was die letzte Zahl bedeutet.

2. Schreibe ein Programm, das nach zwei Zahlen fragt und dann ihr Produkt druckt. (Das heißt, Du mußt sie multiplizieren).

11. ANMERKUNGEN FÜR DEN LEHRER Verzögerungsschleifen, Töne

Dieses Kapitel stellt Schleifen in einer leicht verständlichen Weise vor.

Die Verzögerungsschleife befindet sich komplett in einer Zeile (FOR...TO). Sie wird durch einen Doppelpunkt vom NEXT-Befehl abgetrennt. Die Anzahl der Schleifendurchläufe wird durch die Größe der Schleifenvariable definiert. Der Wert 500 ergibt eine Verzögerung von ungefähr einer Sekunde.

Die primäre Aufgabe einer Schleife besteht darin, bis zu einem bestimmten Wert zu zählen, bevor die nächste Anweisung ausgeführt wird. Hat man das einmal verstanden, bereitet es keine Schwierigkeiten mehr, mit Schleifen zu arbeiten.

Der SOUND-Befehl wird ebenfalls eingeführt. Von den möglichen vier Argumenten wird nur auf die Tonhöhe eingegangen. SOUND wird in einer späteren Übung vollständig behandelt.

Fragen:

1. Zeige, wie man eine Verzögerungsschleife schreibt, die etwa zwei Sekunden dauert.

2. Ist dies eine Verzögerungsschleife?

```
120 FOR Q=1000 TO 5000
122 NEXT Q
```

3. Schreibe ein Programm, das aus zwei Zeilen besteht und einen Ton erzeugt. Verwende eine Verzögerungsschleife, die eine Sekunde dauert. Gib für die Tonhöhe verschiedene Zahlen ein. Welche Zahlen erezugen hohe, welche tiefe Töne?

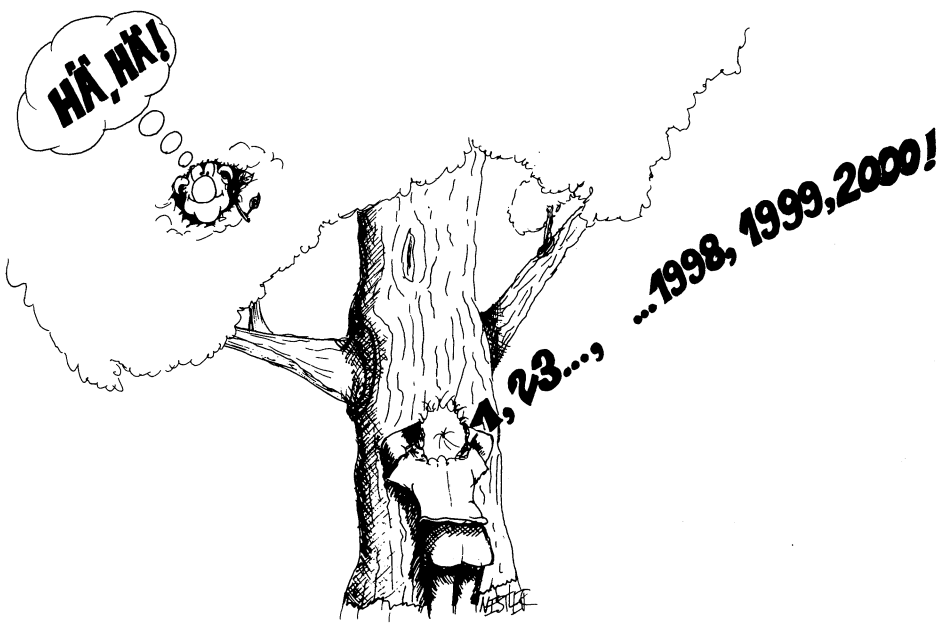
ÜBUNG 11: VERZÖGERUNGSSCHLEIFEN, KLANGEFFEKTE

VERZÖGERUNGSSCHLEIFEN

Hier ist eine Möglichkeit, Teile des Programms zu verlangsamen.

Starte folgendes Programm:

```
10 REM ICH ZAEHLE BIS 2000
20 PRINT "clear"
30 PRINT "VERSTECKEN!"
40 FOR I=1 TO 2000:NEXT I
50 PRINT "ICH KOMME!"
```



Zeile 40 ist die Verzögerungsschleife. Der Computer zählt von 1 bis 2000, bevor er zur nächsten Zeile übergeht. Es ist so, als ob Du derjenige wärst, der in dem Spiel "Ich zähle bis 2000" zählen muß.

Ändere die Zahl 2000 von Zeile 40 um.

Jede Zahl 500 in der Verzögerungsschleife hat den Wert von ungefähr einer Sekunde. Versuche folgendes:

```
10 REM -- TICK TACK --
20 PRINT "clear"
30 PRINT "WARTE WIE LANGE"
35 INPUT S
36 T=S*500
40 FOR Q=1 TO T:NEXT Q
45 PRINT
50 PRINT S;" SEKUNDEN DAUERN"
```

DEIN NAME WIRD GENANNT

```
10 PRINT "clear"
15 LET N=1
20 PRINT "DEIN VORNAME?"
30 INPUT W$
40 PRINT W$
45 FOR T=1 TO 10*N:NEXT T
50 LET N=N+1
60 GOTO 40
```

Halte das Programm mit der BREAK-Taste an.

AUFGABE 11A

1. Schreibe ein "Beleidigungs"-Programm. Es fragt nach Deinem Namen. Dann löscht es den Bildschirm und schreibt Deinen Namen. Anschließend wartet es zwei Sekunden und druckt eine Beleidigung.

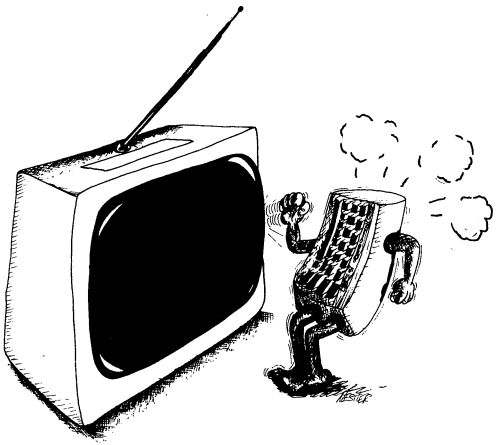
Übung 11: Verzögerungsschleifen, Klangeffekte

TÖNE

Laß das Programm laufen:

```
10 REM TOENE
15 REM ---------------------- ERSTER TON
20 PRINT "clear START MIT DEM MITTLEREN C"
30 SOUND 1,121,10,8
40 FOR T=1 TO 1000:NEXT T
45 PRINT "HALT"
50 SOUND 1,150,10,0
55 REM ---------------------- PAUSE
60 FOR T=1 TO 500:NEXT T
61 REM ---------------------- ZWEITER TON
65 PRINT "NOCHMAL, TIEFER"
70 SOUND 1,162,10,8
80 FOR T=1 TO 500:NEXT T
90 PRINT "ENDE"
```

Wenn Du keinen Ton hörst, mußt Du den Lautstärkregler des Fernsehgeräts aufdrehen.



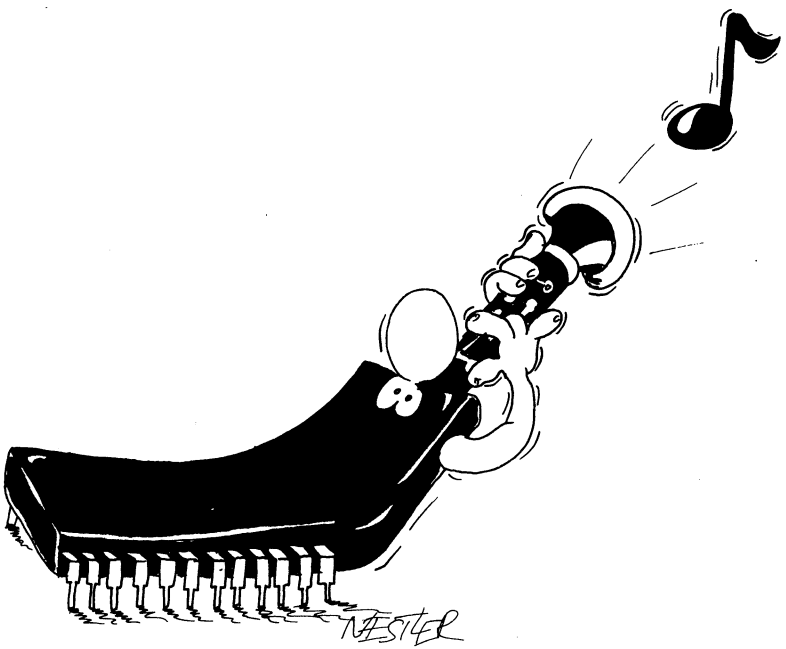
Nach dem SOUND-Befehl stehen 4 Zahlen.

Die zweite Zahl gibt die Tonhöhe einer Note an (die Zahl 121 in Zeile 30 und die Zahl 162 in Zeile 70). Verwende Zahlen von 1 bis 255.

Die vierte Zahl beeinflußt die Lautstärke. Die normale Lautstärke erhältst Du mit "8".

In Zeile 50 wird der Ton "abgedreht", weil die Lautstärke auf Null gesetzt wurde.

Wenn das Programm beendet wird, werden die Töne automatisch unterbrochen.



AUFGABE 11B:

1. Schreibe ein Klang-Programm, bei dem der Anwender selbst wählen kann, welcher Ton gespielt wird und wie lange er dauern soll.

2. Schreibe ein Digitaluhrenprogramm. Die Sekunden sollen mit einer Zeitschleife gezählt werden. Die gegenwärtige Zeit soll in Stunden, Minuten und Sekunden eingegeben werden. Die Uhr zählt und druckt dann die Sekunden. Wenn 60 Sekunden vergangen sind, soll es eine Minute hinzufügen und die Sekundenanzeige wieder auf Null stellen. Das gleiche soll mit den Stunden geschehen. Laß die Uhr eine Zeitlang laufen und stelle die Zeitschleife so ein, daß die Uhr genauer geht.

12. ANMERKUNGEN FÜR DEN LEHRER Die Zahlen und der IF-Befehl

Der IF-Befehl wird jetzt auch bei Zahlen verwendet. Die logischen Beziehungen, die in diesem Kapitel verwendet werden, sind:

= > < <>

Die Verwendung verschachtelter IF-Befehle wird besprochen.

Eine "selbstgebastelte" Schleife wird in dem Ratespielprogramm gezeigt, aber nicht erklärt. Die Schleife beginnt in Zeile 50 und läuft bis Zeile 80. Der Ausgangstest erfolgt in Zeile 57.

Fragen:

1. Welcher Teil des IF-Befehls kann WAHR oder UNWAHR sein?

2. Was folgt nach dem THEN in einem IF-Befehl?

3. Was wird sich in Schachtel D befinden, wenn dieses kleine Programm gelaufen ist?

```
10 LET D=4
15 IF 3 < 7 THEN LET D=9
```

4. Die gleiche Frage für 3 > 7.

ÜBUNG 12: DIE ZAHLEN UND DER IF-BEFEHL

Versuche folgendes:

```
10 REM *** TEENAGER ***
15 PRINT "clear"
20 PRINT "DEIN ALTER"
30 INPUT A
40 IF A<13 THEN PRINT "NOCH KEIN TEENAGER!"
50 IF A>19 THEN PRINT "SCHON ERWACHSEN!"
```

Dieser IF-Befehl ist der gleiche, den Du schon in Zusammenhang mit Zeichenketten verwendet hast. Auch hier haben wir

10 IF Bedingung A wahr ist THEN führe Befehl C aus

Die "Bedingung A" kann folgende Rechensymbole haben:

=	ist gleich
>	größer als
<	kleiner als
<>	ist ungleich

Jede "Bedingung A" wird in der "Mathe"-Sprache geschrieben, Du sollst sie aber laut aussprechen. Zum Beispiel:

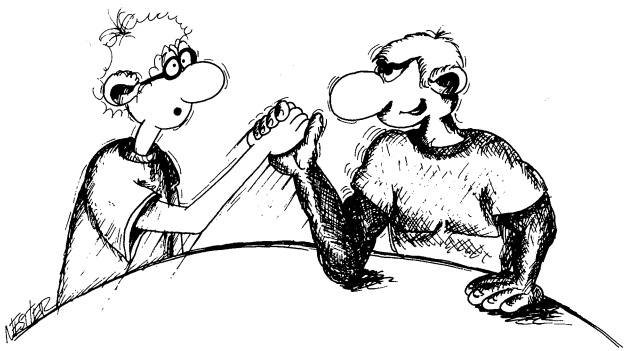
A <> B wird ausgesprochen: "A ist ungleich B"
5 < 7 wird ausgesprochen: "Fünf ist kleiner als Sieben"

SPRECHÜBUNG

Beispiele: LET A=7, LET B=5 und LET C=5

Sage jede "Bedingung A" laut auf und gib an, ob sie wahr oder unwahr ist:

A=B W U	A<B W U
A>C W U	B<C W U
A>B W U	A=C W U
B=C W U	B<>C W U



VERSCHACHTELTE IF-BEFEHLE

Dem obigen "Teenager"-Programm fehlt etwas. Füge hinzu:

```
60 IF A>12 THEN IF A<20 THEN PRINT "TEENAGER!"
```

Um dies zu verstehen, muß Du die Zeile in 2 Abschnitte teilen:

```
60 IF A>12 THEN   (Befehl C)    wobei

(Befehl C) lautet: (IF A<20 THEN PRINT "TEENAGER!")
```

Die erste Zeile fragt: "Bist Du älter als 12?"

Lautet die Antwort "ja", geht die Zeile über zur zweiten Frage: "Bist Du jünger als 20?" Wenn die Antwort wieder "ja" lautet, wird "TEENAGER!" gedruckt. Wenn die Antwort in beiden Fällen "nein" lautet, wird der PRINT-Befehl nicht erreicht und daher nichts gedruckt.

Übung 12: Die Zahlen und der IF-Befehl

AUFGABE 12A:

1. Zeichne das "Weggabelungsdiagramm" für die obige Zeile 60. Es müßten zwei Gabelungen sein.

EIN RATESPIEL

```
10 REM RATESPIEL
15 PRINT "clear"
20 PRINT "EIN SPIEL FUER ZWEI"
25 PRINT
30 PRINT "ERSTER SPIELER, GIB EINE ZAHL ZWISCHEN 1
UND 100 EIN"
35 PRINT "DER ZWEITE SPIELER DARF DABEI NICHT ZUSEHEN"
37 PRINT
40 INPUT N
45 PRINT "clear"
50 PRINT "RATE ";
55 INPUT G
60 IF G<N THEN PRINT "ZU KLEIN"
65 IF G>N THEN PRINT "ZU GROSS"
70 IF G=N THEN GOTO 90
80 GOTO 50
90 REM SPIELENDE
92 PRINT
95 PRINT  "DU HAST SIE ERRATEN!"
```

Wenn Du das Programm auf Band speichern willst, solltest Du Übung 14 lesen.

Im Normalfall schickt Dich Zeile 80 zu Zeile 50, so daß Du mehrmals raten kannst. Aber wenn in Zeile 70 G=N ist, springt das Programm zu Zeile 90 und druckt "DU HAST ES ERRATEN!".

AUFGABE 12B:

1. Was passiert in den Zeilen 50 bis 80:

   wenn G gleich 31 und N gleich 88 ist?

   50 _____

   55 _____

   60 _____

   65 _____

   70 _____

   80 _____

   wenn G gleich 88 und N gleich 88 ist?

   50 _____

   55 _____

   60 _____

   65 _____

   70 _____

   80 _____

2. Ein weiteres Programm: Was und wie oft wird es etwas drucken?

```
10 LET N=1
15 PRINT N;
20 IF N=13 THEN PRINT "PECH GEHABT!"
30 LET N=N+2
40 IF N>30 THEN GOTO 99
50 GOTO 15
99 PRINT "GESCHAFFT"
```

Was passiert, wenn Zeile 10 wie folgt geändert wird?

```
10 LET N=2
```

3. Schreibe ein Programm, das für jede Zahl zwischen 1 und 10 etwas aussagt. Der Spieler gibt eine Zahl ein und der Computer etwas für jede Zahl aus. Zum Beispiel: "Die drei Musketiere" für die Zahl 3, "Sieben auf einen Streich" für die Zahl 7 usw.

4. Schreibe ein Programm, das eine Spielkarte errät, die Spieler 1 eingegeben hat. Dann muß Spieler 2 die Farbe (Karo, Kreuz, Pik und Herz) und den Wert (zwischen 1 und 13) der Spielkarte eingeben. Zuerst errät der Spieler die Farbe und dann den Wert der Spielkarte. Speichere die Punktezahl.

13. **ANMERKUNGEN FÜR DEN LEHRER** Die Zufallszahlen und die INT-Funktion

Dieses Kapitel behandelt zwei Funktionen: RND und INT. Sie sind sehr wichtig beim Programmieren von Spielen.

Die RND-(Zufallszahl-)Funktion erzeugt Pseudo-Zufallsdezimalzahlen, die zwischen 0.0 und 1.0 liegen. Zwar könnte man mit diesem Zahlenbereich schon gut arbeiten, gewöhnlich werden aber größere Zahlenbereiche verwendet, zum Beispiel bei einem Würfel (1 bis 6) oder bei einem Kartenspiel (1 bis 13).

Man erweitert deshalb diesen Zahlenbereich dadurch, daß man die gewonnene Zufallszahl mit einer ganzen Zahl multipliziert oder addiert. Diesen Vorgang überläßt man dem Computer. Man möchte aber am Ende mit einer ganzen Zahl arbeiten. Die INT-(Ganzzahl-) Funktion macht dies, indem sie einfach das, was hinter dem Komma steht, wegläßt. (Bei negativen Zahlen ist der Fall etwas schwieriger. Dieser seltene Fall wird aber hier nicht behandelt.)

Das Konzept der Funktionen wird auch in diesem Kapitel wieder verwendet und weiter vertieft.

Das Verschachteln einer Funktion in die Klammern einer anderen wird bei der Verwendung von RND als Argument der INT-Funktion gezeigt.

Übung 13: Die Zufallszahlen und die INT-Funktion

Fragen:

1. Was druckt der Computer in jedem dieser Fälle ?

   ```
   10 PRINT INT(G)
   ```

   Die Schachtel G enthält: 2, 2.1, 2.95, 3.001, 67, 0, 0.2

2. Erkläre den Unterschied zwischen der INT()-Funktion und dem Runden von Zahlen. Was fällt Dir leichter?

3. Erkläre, wie die INT-Funktion eine Zahl rundet.

4. Was macht die Funktion RND(9)?

5. Wie kannst Du ganze Zufallszahlen zwischen 0 und 10 erzeugen? (Hinweis: INT(RND(9)*10) stimmt nicht ganz.)

6. Wie kannst Du ganze Zufallszahlen zwischen 5 und 8 erzeugen?

ÜBUNG 13: DIE ZUFALLSZAHLEN UND DIE INT-FUNKTION

DIE RND-FUNKTION

Beim Würfeln kannst Du nicht vorhersagen, welche Zahlen erscheinen werden.

Wenn Du Karten gibst, weißt Du nicht vorher, welche Karten jemand bekommen wird.

Du mußt einen Weg finden, damit der Computer "würfelt", "Karten gibt" oder andere unvorhersehbare Dinge macht.

Verwende dazu die RND-Funktion. RND ist eine Abkürzung für "random" (="Zufall").

Starte folgendes Programm:

```
10 REM ZUFALLSZAHLEN
20 PRINT "clear"
25 LET N=RND(9)
30 PRINT N
40 IF N<.95 THEN GOTO 25
```

Du erkennst eine Menge Dezimalzahlen auf dem Bildschirm. Sie wurden durch die RND-Funktion in Zeile 25 erzeugt.

Es spielt keine Rolle, welche Zahl innerhalb der Klammern bei der RND-Funktion steht. Sie muß allerdings positiv sein. Wir haben deshalb "9" gewählt, weil sich diese Zahl auf der Tastatur neben den Klammern "(" und ")" befindet.

RND erzeugt Dezimalzahlen, die zwischen 0 und 1 liegen. Um Zahlen größer als 1 zu erzeugen, brauchst Du nur zu multiplizieren.

Ändere das obige Programm wie folgt:

```
25 LET N=RND(9)*52
40 IF N<45 THEN GOTO 25
```

und laß es nochmal laufen.

Jetzt liegt der Zahlenbereich zwischen 0 und 52. Dieser Ausdruck kann verwendet werden, damit zwischen 52 Spielkarten ausgewählt werden kann.

Aber:

Normalerweise sind uns Zahlen wie 7 und 23 lieber als Dezimalzahlen wie 7.03 und 23.62. Du kannst Sie mit Hilfe der INT-Funktion erzeugen.

DIE INT-FUNKTION

Die INT-(Ganzzahl)-Funktion nimmt die Zahl, die in den Klammern steht, wirft den Teil hinter dem Komma weg und hinterläßt eine ganze Zahl. Verwende die INT-Funktion im nächsten kleinen Programm:

```
10 LET I=INT(6.3)
20 PRINT I
```

Und im nachfolgenden:

```
10 LET X=0.3
20 PRINT "X= ";X
30 PRINT "INT(X)= ";INT(X)
```

Und:

```
10 LET X=.3
20 LET Y=2.5
30 LET P=X+Y
40 LET Q=INT(X+Y)
50 PRINT P,Q
```

Sieh Dir das Ergebnis an. Der Dezimalteil ist nicht mehr vorhanden.

Versuche:

```
10 REM --- INT ---
20 PRINT "clear"
30 PRINT "DEZIMALZAHL? "
32 INPUT D
35 LET I=INT(D)
40 PRINT "DEZIMAL ";D;
45 PRINT " GANZZAHL ";I
50 IF I<>0 THEN GOTO 30
```

Mit der Eingabe 0 beendest Du das Programm.

EIN WÜRFELPROGRAMM

Die meisten Würfelspiele verwenden zwei Würfel. Hier ist ein Programm, das einen Würfel simuliert:

```
10 REM ///EIN WUERFEL///
15 DIM J$(1)
20 PRINT "clear"
30 LET R=RND(9)
40 PRINT "ZUFALLSZAHL ";R
50 LET S=R*6
55 PRINT "MAL 6 ";S
60 LET I=INT(S)
65 PRINT "GANZZAHLANTEIL ";I
70 LET W=I+1
75 PRINT "DER WUERFEL ZEIGT ";W
80 PRINT "NOCHMAL? (J/N) ";
82 INPUT J$
85 IF J$="J" THEN GOTO 20
```

Übung 13: Die Zufallszahlen und die INT-Funktion

WAS GEHÖRT IN DIE KLAMMERN ()?

Zahlen:
```
10 LET X=INT(34.7)
```

Variablen:
```
10 LET X=INT(J)
```

Ausdrücke:
```
10 LET X=INT(3*Y+2)
```

Funktionen:
```
10 LET X=INT(RND(9))
```

So kann man viel Platz sparen:

Statt
```
30 LET R=RND(9)
50 LET S=R*6
60 LET I=INT(S)
70 LET D=1+I
```

benutzte
```
70 LET D=1+INT(RND(9)*6)
```

AUFGABE 13

1. Schreibe ein Programm, das zwei Würfel (W1 und W2) "rollen" läßt. Die Augenzahl von W1 und W2 und die Gesamtsumme soll auf dem Bildschirm erscheinen. Die Variablen R, S und I aus dem obigen Programm brauchst Du nicht zu verwenden. Sie dienten nur dazu, zu zeigen, wie das Ergebnis gefunden wurde.

3. Schreibe ein "Papier, Schere und Stein"-Programm, das Du gegen den Computer spielen kannst. (Papier umwickelt Stein, Stein bricht Schere, Schere schneidet Papier). Der Computer wählt eine Zahl zwischen 1 und 3 mit Hilfe der RND()-Funktion aus: 1 ist Papier (P), 2 ist Stein (ST) und 3 ist Schere (S). Mit INPUT gibst Du P, ST und S ein. Der Computer gibt dann an, wer gewonnen hat und speichert die Punktezahl.

14. ANMERKUNGEN FÜR DEN LEHRER Das Speichern auf Band

Wir erläutern, wie man Programme auf Band speichern kann.

Die Befehle CSAVE und CLOAD werden erklärt.

Befehle, die wir in dieser Übung noch verwenden sind:

NEW, REM, LIST und PRINT.

Dieses Übung kann auch nach Übung 3 durchgearbeitet werden.

Wir bringen dieses Kapitel deswegen so spät, weil die meisten Programme bis jetzt ziemlich kurz und wenig interessant sind und es sich deshalb nicht lohnt, sie zu speichern.

Entscheiden Sie selbst, ob Sie das Kapitel zu einem früheren Zeitpunkt behandeln wollen. Ihre Schüler könnten zum Beispiel einige selbstgeschriebene Programme, auf die sie stolz sind, abspeichern.

Der ATARI-Kassettenrecorder funktioniert ähnlich wie ein normaler Recorder. Seine Besonderheit liegt darin, daß der Computer den Recorder ein- und ausschalten kann.

Es können normale Kassetten verwendet werden, die von bester Qualität sein sollten. Selbst die kleinste Unreinheit auf dem Band führt dazu, daß das aufgenommene Programm nicht mehr gelesen werden kann.

Kassetten mit kurzer Laufzeit sind am besten geeignet. Um ein Programm zu laden, braucht man zwischen 20 Sekunden und 5 Minuten. Man sollte entweder ein langes Programm oder einige kurze Programme auf eine Bandseite speichern.

Der ATARI-Computer unterstützt auch das Abspeichern von Programmen mit Dateinamen, aber das werden wir in diesem Buch nicht besprechen. Bei Interesse sollte im ATARI-Handbuch nachgesehen werden.

Übung 14: Das Speichern auf Band

Fragen:

1. Wenn Du den Befehl CSAVE eingibst, "brummt" der Computer zweimal. Was bedeutet das?

2. Mit welchem Befehl holt der Computer ein Programm von der Kassette?

3. Wie lange dauert es ungefähr, bis ein kurzes Programm abgespeichert ist?

4. Ist ein Programm, das auf eine Kassette abgespeichert wurde, noch im Speicher des Computers?

ÜBUNG 14: DAS SPEICHERN AUF BAND

EIN PROGRAMM EINGEBEN

Wenn Du gerade ein Programm bearbeitest, das Du abspeichern möchte, überspringe diesen Abschnitt und gehe zum nächsten (EIN PROGRAMM ABSPEICHERN).

Übung 14: Das Speichern auf Band

Gib sonst das nächste Programm ein:

```
NEW
10 REM :::HALLO:::
20 PRINT "HALLO"
```

EIN PROGRAMM ABSPEICHERN

Lege eine völlig neue Kassette in den ATARI-Kassetten-Recorder und spule sie zurück.

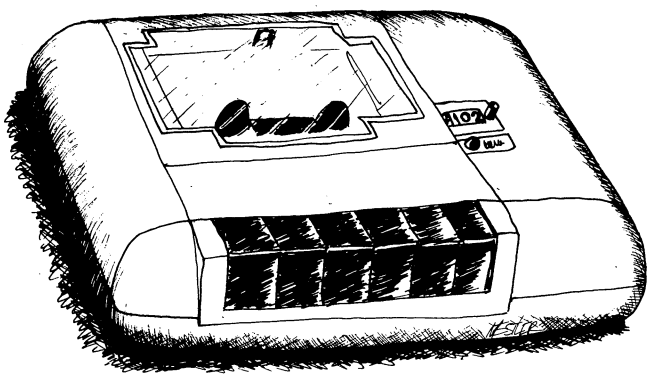
Drücke auf den Bandzählwerksknopf am Kassetten-Recorder. Damit wird das Zählwerk auf Null gestellt (genau auf 000).

Gib ein: CSAVE

Du hörst zweimal einen Summton. Das ist das Signal, daß Du jetzt zwei Tasten am Recorder betätigen sollst.

Drücke gleichzeitig die REC- und die PLAY-Taste.

Anschließend mußt die die RETURN-Taste am Computer drücken.



Der Computer schaltet den Motor des Recorders ein und nach etwa 10 Sekunden, in denen ein Vorspannband vorbeiläuft, fängt er an, das Programm auf das Kassettenband abzulegen.

Nach etwa 20 Sekunden (das gilt für dieses kurze Programm) ist der Computer fertig und schaltet den Motor des Kassettenrecorders wieder aus. Am Bildschirm steht:

READY

Das Zählwerk des Recorders sollte ungefähr auf 10 stehen.

EINE LISTE ERSTELLEN

Du solltest den Namen des Programms auf die Vorderseite der Kassette schreiben und ebenfalls den Zählwerksstand.

An dieser Stelle ist das Programm beendet. (Erinnere Dich, es fing mit 000 an.)

EINE WARNUNG!
Wenn Du ein sehr wichtiges Programm abspeicherst, solltest Du sicherheitshalber auf dasselbe Band eine zweite Kopie abspeichern. (Spule das Band nicht zurück. Tippe CSAVE ein und starte den Recorder.)

DAS LADEN EINES PROGRAMMS IN DEN COMPUTER

Den Ladevorgang wollen wir mit dem Programm ausprobieren, das Du gerade abgespeichert hast.

Gib ein: NEW

und: LIST

Das Programm ist aus dem Speicher gelöscht. (Sonst weißt Du ja nicht, ob das Programm wirklich von der Kassette geladen wurde oder noch im Speicher stand.)

Übung 14: Das Speichern auf Band

Spule das Band zurück.

Gib ein: CLOAD

Drücke: RETURN

Du hörst einen Summton. Das bedeutet, daß Du eine Taste am Recorder drücken sollst.

Drücke die PLAY-Taste am Recorder.

Drücke dann die RETURN-Taste an der Tastatur.

Der Computer schaltet den Recorder-Motor ein und sucht nach dem Programm.

Das Zählwerk des Recorders läuft wieder.

Nach etwa 20 Sekunden siehst Du folgendes auf dem Bildschirm:

READY

Gib ein: LIST

Jetzt solltest Du das Programm wieder sehen.

WIE VIELE PROGRAME HABEN AUF DER KASSETTE PLATZ?

Wenn Du mit dem Zählwerk des Recorders arbeitest und eine gute Liste führst, kannst Du viele Programme auf einer Kassette abspeichern. (Natürlich darfst Du während des Abspeicherns der einzelnen Programme nicht zurückspulen.)

Je mehr Programme Du auf ein Band ablegst, desto schwieriger ist es, sie wiederzufinden, auch wenn Du das Zählwerk zu Hilfe nimmst.

Es kann sein, das immer häufiger Fehler auftreten, oder Deine Programme sogar zerstört werden.

Wenn Deine Programm größer als 25 Zeilen sind, solltest Du pro Kassette immer nur ein Programm abspeichern.

AUFGABE 14

1. Schreibe ein kurzes Programm (4 Zeilen) und speichere es auf Kassette.

2. Gib NEW ein, schreibe noch ein kurzes Programm und speichere es ebenfalls ab.

3. Gib NEW ein. Lade dann jeweils ein Programm und laß es laufen.

2

Grafik, Spiele und noch mehr

15. ANMERKUNGEN FÜR DEN LEHRER Einige Abkürzungen

In dieser Übung erklären wir:

?	steht für PRINT
LET	wird ausgelassen
:	wird zwischen zwei Zeilen verwendet

Der erste Abschnitt ist erledigt. Wir sind bis zur RND-Funktion gekommen und haben gelernt, wie man Programme auf einer Kassette abspeichert.

Der Doppelpunkt wird dazu verwendet, Programme abzukürzen und klarer zu gestalten. Dies erreicht man, indem man mehrere Anweisungen in eine Zeile bringt. Eine Zeile sollte Anweisungen enthalten, die miteinander in Beziehung stehen.

Der Doppelpunkt kann ein Programm auch durcheinanderbringen. Einige Anweisungen werden über GOTO-Befehle angesprungen.

Gelegentlich wird ein anderer Fehler begangen: Eine Anweisung (durch Doppelpunkt getrennt) wird ans Ende einer Zeile gestellt. Enthält diese Zeile einen IF-Befehl und soll die letzte Anweisung immer ausgeführt werden, gibt es im Programmablauf einen logischen Fehler. Die Instruktion wird nur dann gestartet, wenn die IF-Bedingung WAHR ist.

Andererseits gestattet es der Doppelpunkt, kleine "Unterroutinen", die aus mehreren Anweisungen bestehen, hinter einen IF-Befehl zu setzen. Daraus ergibt sich, daß GOTO gar nicht gebraucht wird, um einen anderen Abschnitt zu erreichen. So entsteht ein kürzeres und klarer gegliedertes Programm. Deswegen kommt dem Doppelpunkt große Bedeutung bei der Programmgestaltung zu.

Übung 15: Einige Abkürzungen

Fragen:

1. Welche Abkürzung stellt das "?" dar?

2. Wie kannst Du erklären, daß das Wort LET in einem LET-Befehl fehlt?

3. Ein INPUT-Befehl enthält eine Meldung in Anführungszeichen. Was für ein Satzzeichen muß diesen Anführungszeichen folgen?

4. Wieso ist es manchmal gut, zwei durch einen Doppelpunkt getrennte Anweisungen auf die gleiche Zeile zu bringen?

5. Was stimmt nicht mit diesen Zeilen?

```
10 REM ANFANG:GOTO 1000
10 GOTO 50: S$="SCHNELL"
```

ÜBUNG 15: EINIGE ABKÜRZUNGEN

DIE PRINT-ABKÜRZUNG

Tippe statt PRINT ein Fragezeichen

Gib ein:
```
10 ? "HALLO"
LIST 10
```

Du siehst:

```
10 ? "HALLO"
```

Laß das Programm laufen. Das Ergebnis am Bildschirm sieht aus, als ob Du die Zeile 10 so geschrieben hättest:

```
10 PRINT "HALLO"
```

Tatsächlich ersetzt der Computer das Fragezeichen in seinem Speicher durch das Wort PRINT.

DIE LET-ABKÜRZUNG

Diese beiden Zeilen machen das gleiche:

```
10 LET A=41          und  10 A=41
```

auch diese beiden:

```
20 LET B$="HALLO"    und  20 B$="HALLO"
```

Du kannst das Wort LET bei der LET-Anweisung weglassen! Der Computer weiß, daß Du LET meinst, wenn die Zeile mit einem Variablennamen, gefolgt von einem "="-Zeichen, anfängt.

Übung 15: Einige Abkürzungen

EINE ABKÜRZUNG FÜR DEN LIST-BEFEHL

Es gibt 3 Möglichkeiten, den LIST-Befehl zu verwenden:

```
LIST        listet das gesamte Programm auf
LIST 48     listet Zeile 48 auf
LIST 50,75  listet alle Zeilen zwischen 50 und 75 auf
```

DIE DOPPELPUNKT-ABKÜRZUNG

Setze verschiedene Anweisungen in eine Zeile und trenne sie durch einen Doppelpunkt ":". Dies spart Platz.

Statt:
```
10 Q=17*3
20 R=Q+2
30 PRINT R
```

kannst Du schreiben:

```
10 Q=17*3:R=Q+2:? R
```

Im Speicher wirkt die Zeile so, als ob

```
10 Q=17*3:R=Q+2:PRINT R
```

stände.

WANN VERWENDET MAN DIE DOPPELPUNKT-ABKÜRZUNG?

Die Abkürzung wird verwendet,

1. um das Programm klarer aufzubauen.

   Setze verschiedene Anweisungen in die gleiche Zeile. Beispiel:

Statt:
```
10 X=0
12 Y=0
14 Z=0
```

schreibe:
```
10 X=0:Y=0:Z=0
```

2. um das Programm zu verkürzen.

3. um einen REM-Befehl an das Ende der Zeile zu setzen.

   Beispiel:
```
40 H=X+Y/66 : REM H IST DIE HOEHE
```

DER DOPPELPUNKT NACH EINEM IF-BEFEHL

Du kannst mit Hilfe des Doppelpunktes die IF-Anweisungen klarer gestalten.

Ohne:

```
50 IF A=0 THEN GOTO 80
60 B=Q
62 C=B*D
66 PRINT "FALSCH"
80 FOR ...
```

Mit Doppelpunkt:

```
50 IF A<>0 THEN B=Q:C=B*D:PRINT "FALSCH"
80 FOR ...
```

Alle Befehle auf dem Weg "A<>0 ist WAHR" stehen in der Zeile nach THEN.

VORSICHT!
Setze nicht etwas an das Ende der IF-Zeile, das nicht dazugehört.

Übung 15: Einige Abkürzungen

Beispiel:

```
35 IF A=B THEN PRINT "VERGLEICHBAR"
40 Q=R
```

ist nicht das gleiche wie:

```
37 IF A=B THEN PRINT "VERGLEICHBAR": Q=R
```

weil Q=R in der Zeile immer ausgeführt wird, egal, ob A=B wahr oder unwahr ist. Aber Q=R in Zeile 37 wird nur dann ausgeführt, wenn A=B wahr ist.

WEITERE FEHLER MIT DEM DOPPELPUNKT

REM und GOTO müssen die letzten Befehle einer Zeile sein. Alles, was nach ihnen kommt, wird nicht beachtet.

Richtig: `35 P=3:REM P IST DER PREIS`

Falsch: `35 REM P IST DER PREIS: P=3`

Der Computer läßt alles, was er nach dem REM-Befehl liest, außer acht.

Richtig:
```
40 R=P+1:GOTO 88
42 S=3
```

Falsch: `40 R=P+1:GOTO 88:S=3`

Der Computer geht zu Zeile 88 und kann nicht mehr zurückkehren, um den Befehl S=3 auszuführen.

BEFEHLE, ANWEISUNGEN UND ZEILEN

Befehle sagen dem Computer, daß er etwas tun soll. Bis jetzt haben wir diese verwendet:

PRINT, NEW, RUN, LIST, REM, INPUT, LET
GOTO, IF, CSAVE, CLOAD

Befehle, die in numerierten Zeilen verwendet werden, können "Anweisungen" genannt werden. Stehen sie allein, werden sie immer "Befehle" genannt.

Gib ein:

LIST	Wir sagen, wir haben "einen Befehl eingegeben".

Schreiben wir aber diese Zeile in einem Programm:

20 LIST	so sagen wir, daß Zeile 20 eine "Anweisung", den LIST-Befehl, enthält.

Übung 15: Einige Abkürzungen

Manche Zeilen enthalten mehrere Anweisungen, die durch Doppelpunkte getrennt werden.

```
30 DIM E$(20):PRINT:LET Z=55
```

ist eine Zeile mit drei Anweisungen.

AUFGABE 15:

1. Schreibe ein Programm, das wenigstens einmal jede dieser Abkürzungen verwendet.

2. Schreibe ein "Urlaubs"-Programm. Es fragt, wieviel Geld Du ausgeben willst. Dann sagt es Dir, wo Du hinfahren kannst oder was Du tun sollst.

3. Schreibe ein "verrücktes" Programm, das nach Deinem Namen fragt. Das Programm sagt Dir auf drei verschiedene Weisen, daß Du verrückt bist. Es wählt zufällig eine dieser Möglichkeiten und druckt sie hinter Deinen Namen.

16. ANMERKUNGEN FÜR DEN LEHRER Grafikzeichen, POSITION

Der POSITION-Befehl wird verwendet, um den Ausgabe-Cursor an eine beliebige Textstelle am Bildschirm zu bringen.

Diesen Befehl setzt man bei der Manipulation von Text und/oder Grafik ein.

Damit man mit dieser Anweisung besser arbeiten kann, sollten man sich den Bildschirm in ein Raster mit 40 Spalten und 24 Zeilen aufgeteilt vorstellen.

Die Grafikzeichen des ATARI-Computer kommen zum Vorschein, wenn man die CONTROL-Taste festhält und eine Taste mit Buchstaben oder Satzzeichen drückt.

Fragen:

1. Welchen Befehl mußt Du geben, wenn Du das nächste Wort in Zeile 12 ausgeben willst?

2. Welchen Befehl mußt Du geben, wenn Du das nächste Zeichen in Zeile 6 nach 20 Leerzeichen drucken willst?

3. Zeige, wie man die beiden Wörter "KLEINE" und "KATZE" in die gleiche Zeile schreibt, wenn "KATZE" mit Stelle 25 zuerst und dann "KLEINE" mit Stelle 5 ausgegeben werden soll.

ÜBUNG 16: GRAFIKZEICHEN, POSITION

DER GRAPHICS-0-MODUS

Auf dem Bildschirm kann man 24 Zeilen darstellen. Die Zahlen werden von 0 (oberste Zeile) bis 23 (unterste Zeile) durchnumeriert.

Jede Zeile kann 40 Zeichen enthalten. Sie werden von links nach rechts von 0 bis 39 numeriert.

Laß das Programm laufen:

```
10 REM POSITION DEMO
15 PRINT "clear"
20 POSITION 0,10:PRINT "ZUERST ZEILE 10"
25 FOR T=1 TO 500:NEXT T
30 POSITION 0, 1:PRINT "ALS NAECHSTES ZEILE 1"
35 FOR T=1 TO 500:NEXT T
40 POSITION 0,17:PRINT "ZULETZT ZEILE 17"
```

Die zweite Zahl nach dem Befehl POSITION gibt an, in welche Zeile der Cursor gehen soll.

DAS HERUMHÜPFEN AUF DEM BILDSCHIRM

Starte:

```
10 REM SPALTEN UND ZEILEN
15 PRINT "clear"
20 PRINT "WELCHE ZEILE";:INPUT Z
30 PRINT "WELCHE SPALTE";:INPUT S
40 POSITION S,Z:PRINT "*";
45 FOR T=1 TO 500:NEXT T
50 GOTO 20
```

Halte das Programm mit der BREAK-Taste an.

DAS LÖSCHEN VON TEXT

```
10 REM SPRINGENDES HIER
12 POKE 752,1:REM CURSOR AUSSCHALTEN
15 PRINT "clear":POKE 752,1
20 X=INT(RND(9)*36)
25 Y=INT(RND(9)*23)
30 POSITION X,Y:PRINT "HIER"
50 FOR T=1 TO 200:NEXT T
60 POSITION X,Y:PRINT "    "
80 GOTO 20
```

DER CURSOR WIRD UNSICHTBAR

Sieh Dir die Zeile 15 an. POKE legt eine Zahl in die Speicherschachtel 752. Diese Schachtel sagt dem Cursor, ob er sichtbar oder unsichtbar sein soll.

Enthält die Schachtel eine Null, ist der Cursor am Bildschirm zu sehen.

Enthält die Schachtel eine 1, ist der Cursor verschwunden.

GRAFIKZEICHEN VON DER TASTATUR

Du befindest Dich im GRAPHICS-0-Modus.

Du kannst direkt von der Tastatur Grafikzeichen eintippen.

Versuche einmal folgendes: Halte die CONTROL-Taste fest und drükke einen beliebigen Buchstaben, einen Strichpunkt, ein Komma oder einen Punkt. Auf dem Bildschirm erscheinen Grafiksymbole.

Drücke nun einmal die ATARI-Taste, halte dann wieder die CONTROL-Taste fest und tippe Buchstaben ein. Diesmal sind die Grafikzeichen invers dargestellt.

Insgesamt sind 58 Zeichen verfügbar.

Diese Zeichen kannst Du auch in eine PRINT-Anweisung schreiben und damit tolle Bilder malen.

Verwende POSITION, damit die Bilder auch an die richtige Stelle gestellt werden.

Laß das Programm laufen:

```
10 REM TUT-TUT
12 PRINT "clear"
20 POSITION 10,10
30 PRINT "u u lt lt invers j h normal"
32 POSITION 10,11
34 PRINT "u u u u u u"
36 POSITION 10,12
38 PRINT "t t lt lt t j "
```

Diesmal bedeuten die Kleinbuchstaben in den PRINT-Anweisungen, daß Du die CONTROL-Taste festhalten und die Taste mit den entsprechenden großen Buchstaben drücken mußt. Die Begriffe "invers" und "normal" sind Dir inzwischen sicher längst bekannt. Richtig, Du mußt bei beiden die ATARI-Taste einmal drücken, damit die Zeichen invers dargestellt werden. Der Begriff "lt" ist neu für Dich. Er wurde eingeführt, damit Du weißt, wann Du wirklich die Leertaste (lt) drücken sollst. Die anderen Leerräume sind nur der Übersichtlichkeit wegen eingeführt worden.

AUFGABE 16:

1. Verwende die RND()-Funktion, um Deinen Namen an irgendeine Stelle auf dem Bildschirm zu schreiben. Laß Deinen Namen über den ganzen Bildschirm verteilt darstellen.

2. Verwende POSITION, um mit den Buchstaben Deines Namens ein großes "X" zu bilden.

3. Zeichne den Zug fertig und verwende dazu die Grafikzeichen in einer PRINT-Anweisung.

17. **ANMERKUNGEN FÜR DEN LEHRER** FOR-NEXT-Schleifen

Eine Schleife besteht aus einer FOR- und einer NEXT-Anweisung. Diese Befehle können durch einige Zeilen getrennt werden, sind aber voneinander abhängig. Das kann Ihre Schüler verwirren. Die Verzögerungsschleife aus einer vorhergehenden Übung hilft bei der Vorstellung, daß FOR ... und NEXT miteinander verbunden sind. Es wird gezeigt, daß das Wiederholen einer Befehlssequenz, die sich innerhalb der Schleife befindet, sinnvoll ist.

Verschachtelte Schleifen werden anhand eines Beispiels vorgestellt, bei dem die innere Schleife eine Verzögerungsschleife darstellt.

In diesem Kapitel werden die Feinheiten nicht besprochen, die früher oder später auftauchen. Die Schleife muß wenigstens einmal durchlaufen werden, weil der Ausgangstest in der NEXT-Anweisung ausgeführt wird. NEXT kann nur dann erreicht werden, wenn die Schleife durchlaufen wird.

Die FOR-Anweisung wird nur einmal ausgewertet, und zwar dann, wenn man in die Schleife gelangt. Sie setzt den Anfangswert der Schleifenvariablen in den Variablenspeicher, wo sie wie jede andere Zahlenvariable behandelt wird. Der STEP-Wert, der Endwert und die Adresse der ersten Anweisung, die dem FOR-Befehl folgt, werden in einen Stapelspeicher gesetzt.

Ab hier läuft der Schleifenvorgang im NEXT-Befehl ab. Beim Erreichen von NEXT erhöht sich der Wert der Schleifenvariable um den Wert der STEP-Anweisung und wird mit dem Endwert verglichen. Ist die Schleifenvariable größer als der Endwert (oder kleiner, wenn STEP negativ ist), übergibt NEXT die Kontrolle des Ablaufs an die nächste Anweisung. Sonst wird sie an die nächste Anweisung, die nach dem FOR-Befehl kommt, abgegeben.

Da die Schleifenvariable von BASIC genau wie jede andere Variable behandelt wird, kann sie innerhalb der Schleife selbst verändert bzw. verwendet werden. Beim Ändern sollte man sehr umsichtig vorgehen, weil ja auch der NEXT-Befehl die Variable beeinflußt und auf das Ende der Schleife abfragt.

Übung 17: FOR-NEXT-Schleifen

Das Hineinspringen in die Mitte einer Schleife ergibt normalerweise eine kleine Katastrophe. Das Herausspringen aus einer Schleife ohne den natürlichen Abbruch durch NEXT ist allgemein üblich. In einigen Fällen kann es das Auffinden von Fehlern erschweren (besonders wenn Unterprogramme verwendet werden).

Fragen:

1. Schreibe eine Schleife, in der die Zahlen von 0 bis 20 ausgedruckt werden.

2. Schreibe eine Schleife, in der die Zahlen von 30 nach 20 in Zweierschritten heruntergezählt werden.

3. Schreibe ein Programm, in dem mit zwei verschachtelten Schleifen die Zahlen 100, 200, 300 und dazwischen die Zahlen 1, 2, 3, 4 und 5 in getrennten Zeilen ausgegeben werden.

ÜBUNG 17: FOR-NEXT-SCHLEIFEN

Erinnerst Du Dich noch an die Verzögerungsschleife? Der Computer zählte von 1 bis 2000 und machte dann weiter.

```
30 FOR T=1 TO 2000:NEXT T
```

Der Computer kann aber mehr. Er kann nämlich noch andere Dinge tun, während er zählt.

Starte:

```
10 REM ZAEHLEN
20 PRINT "clear"
30 FOR I=5 TO 20
40 PRINT I
50 NEXT  I
```

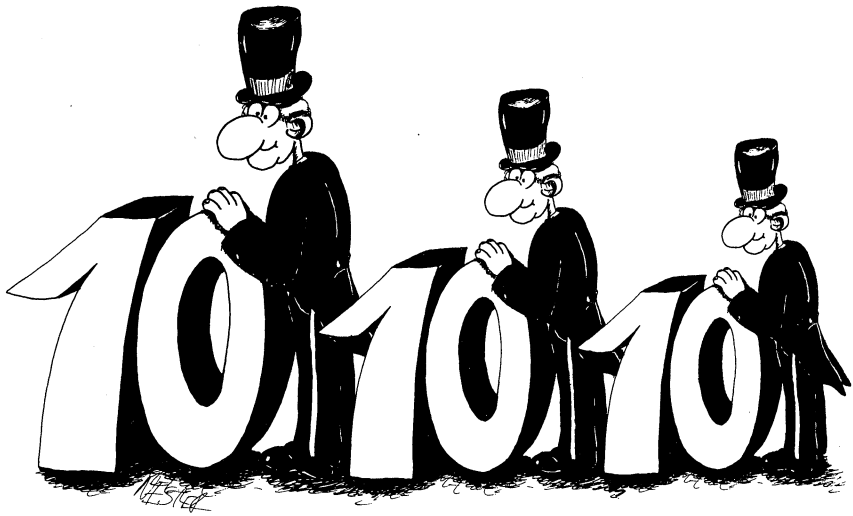
Die Schleife kann bei jeder Zahl anfangen und bei einer höheren Zahl aufhören. Versuche es, indem Du Zeile 30 folgendermaßen veränderst.

```
30 FOR I=100 TO 101
30 FOR 1=-7 TO 13
30 FOR I=1.3 TO 5.7
```

MARKIERE DEINE LISTINGS

Kennzeichne die Schleife durch Pfeile.

```
10 REM AUF PAPIER
20 PRINT "clr"
30 FOR I=0 TO 7
40 PRINT I
50 NEXT I
```



DER STEP-BEFEHL

Der Computer hat in den obigen Programmen in Einserschritten gezählt. Damit er in Zweierschritten zählt, ändere Zeile 30 wie folgt um:

```
30 FOR I=10 TO 30 STEP 2
```

AUFGABE 17A

1. Laß den Computer in Fünferschritten von 0 bis 100 zählen.

COUNT-DOWN-SCHLEIFEN

Mit der Verwendung eines negativen STEP-Befehls kann der Computer den "count down" eines Raketenstarts simulieren.

Versuche dieses:

```
10 REM **APOLLO 11**
20 PRINT "clear"
30 PRINT "T MINUS 12 SEKUNDEN UND ZAEHLEN"
35 FOR T=1 TO 500:NEXT T
40 FOR I=11 TO 0 STEP -1
50 PRINT I:PRINT "piepser"
60 FOR J=1 TO 500:NEXT J: REM ZEITSCHLEIFE
70 NEXT I
80 PRINT "ALLE MOTOREN LAUFEN. START"
81 FOR T=1 TO 500:NEXT T
82 PRINT "WIR HEBEN AB"
83 FOR T=1 TO 500:NEXT T
84 PRINT "32 MINUTEN SIND VERGANGEN."
85 FOR T=1 TO 500:NEXT T
86 PRINT "APOLLO 11 GESTARTET."
```

Zeile 60 enthält die Zeitschleife, die die Sekunden zählt. Die Zeilen 35, 81, 83 und 85 verlangsamen das Ausdrucken auf "Sprechgeschwindigkeit".

VERSCHACHTELTE SCHLEIFEN

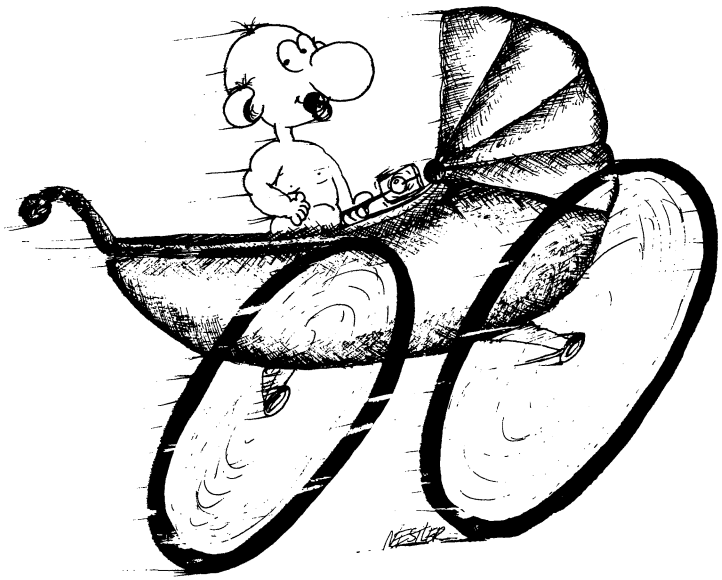
In diesem Programm ist eine Schleife in der anderen Schleife enthalten.

Die äußere Schleife fängt in Zeile 40 an und endet in Zeile 70.

Die innere Schleife befindet sich in Zeile 60.

Man nennt sie "verschachtelte Schleifen". Es ist wie bei Babyspielschachteln, die man ineinanderstecken kann.

Übung 17: FOR-NEXT-Schleifen



SCHLEIFENVARIABLEN

Um sicherzugehen, daß jeder FOR-Befehl weiß, welcher NEXT-Befehl zu ihm gehört, endet der NEXT-Befehl mit einem "Schleifenvariablen"-Namen. Siehe Zeile 60:

```
60 FOR J=1 TO 500:NEXT T
```

J ist die Schleifenvariable. Für die Schleife, die in Zeile 40 beginnt:

```
40 FOR I=12 TO 0 STEP -1

...

70 NEXT I
```

stellt I die Schleifenvariable dar.

SCHLECHT VERSCHACHTELTE SCHLEIFEN

Die innere Schleife muß ganz in der äußeren enthalten sein:

Richtig:

```
25 FOR X=3 T
30 FOR Y=3 TO 7
40 PRINT X*Y
50 NEXT Y
60 NEXT X
```

Falsch:

```
25 FOR X=3 T
30 FOR Y=3 TO 7
40 PRINT X*Y
50 NEXT X
60 NEXT Y
```

AUFGABE 17B:

1. Schreibe ein Programm, das Deinen Namen 15mal druckt.

2. Laß ihn jetzt jedesmal zwei Schritte nach rechts rücken. Er soll diagonal auf dem Bildschirm zu sehen sein.

3. Laß jetzt Deinen Namen 24mal schreiben. Er soll zuerst unten am Bildschirm erscheinen und sich dann nach oben bewegen.

4. Laß jetzt Deinen Namen in eine Zeile und den Namen Deines Freundes in eine andere schreiben. Setze diese Schreibweise fort, bis jeder Name fünfmal geschrieben wurde.

5. Laß die Lokomotive, die wir in der letzten Übung erstellt haben, über den Bildschirm "fahren". Du weißt doch noch: "Zeichnen, löschen, zeichnen, löschen, ...". Setze mit dem POSITION-Befehl jede Lokomotive auf den richtigen Platz. Denk auch an die Verzögerungsschleife! Lösche anschließend die Grafik, indem Du an dieselbe Stelle Leerzeichen schreibst. Zeichne dann das neue Bild. Verwende eine Schleife mit einer Variablen, die angibt, wo das Bild beginnen soll.

18. ANMERKUNGEN FÜR DEN LEHRER Editier- und RUN-Modus, der Rechner

In dieser Übung werden der Editier- und der RUN-Modus des Computers besprochen.

Wir behandeln diese Themen, trotz ihrer grundlegenden Bedeutung, ziemlich spät in diesem Buch, weil sie sehr abstrakt sind und weil wir nicht wollten, daß der Ablauf beim Erlernen der Grundbefehle von BASIC unterbrochen wird.

Natürlich kann diese Übung innerhalb des Kurses vorgezogen werden. Die folgenden beiden Befehle werden verwendet:

PRINT und RUN

Es gibt noch weitere Bezeichnungen für die Modi:

Editier-Modus:	Direkter Modus Befehls-Modus	Rechen-Modus
RUN-Modus:	Programm-Modus	Ausführungs-Modus

Als Computeranwender arbeitet man normalerweise im Editier-Modus. In diesem Modus werden Zeilen eingegeben. Die Zeichen werden im Eingabe-Puffer abgelegt.

Nachdem RETURN gedrückt wurde, kontrolliert der Computer, ob die Zeile mit einer Nummer beginnt. Ist das der Fall, legt er die Zeile im Programmspeicher ab.

Ist eine Zeile nicht numeriert, führt der Computer diese Zeile direkt aus dem Eingabe-Puffer aus. Gewöhnlich besteht eine Zeile nur aus einem einzigen Befehl, wie LIST oder RUN. Mit dem direkten Modus können aber auch größere Programme, die aus einer einzigen Zeile bestehen, unmittelbar ausgeführt werden. Mit dieser Möglichkeit werden zwei Programmierphasen erleichtert: die Entwurfsphase, weil man jederzeit arithmetische Ausdrücke ausprobieren kann und die Fehlersuchphase.

Übung 18: Editier- und RUN-Modus, der Rechner

Fragen:

1. Was macht der Computer im RUN-Modus?

2. Wie erkennst Du, daß der Computer im Editier-Modus ist?

3. Welche drei verschiedene Dinge kannst Du im Editier-Modus tun?

4. Du gibst eine Zeile mit einer Nummer davor ein. Was geschieht mit der Zeile?

5. Du gibst eine Zeile ohne Nummer ein. Was geschieht mit der Zeile?

ÜBUNG 18: EDITIER- UND RUN-MODUS, DER RECHNER

Gib NEW ein.
Drücke die SHIFT- und CLEAR-Tasten.
Nun kannst Du mit der Übung beginnen.

AUSFÜHREN UND LAUFEN LASSEN

"Ein Programm ausführen" bedeutet dasselbe wie "ein Programm laufen lassen".

DER DIREKTE MODUS

Hier ist ein kurzes Beispiel. Tippe die Worte ein (ohne Zeilennummer am Anfang):

```
PRINT "HALLO"
```

Der Computer gibt sofort HALLO aus. Er wartet nicht, bis Du RUN eingibst. Dies nennt man den "direkten Modus".

Regel für den direkten Modus:
Fängt eine Zeile nicht mit einer Nummer an, führt der Computer die Befehle sofort aus (sobald die RETURN-Taste gedrückt wurde).

Teste auch das nächste, etwas längere Beispiel:

```
FOR I=1 TO 20:PRINT I:NEXT I:PRINT "FERTIG"
```

Regel:

Ein Programm, das aus einer Zeile besteht und mehrere durch Doppelpunkt getrennte Anweisungen besitzt, kann im direkten Modus ausgeführt werden. (Aber der Computer vergißt das Programm sofort, wenn er fertig ist.)

DER AUSFÜHRUNGS-MODUS

Gib das nächste Programm ein und starte es:

```
10 PRINT "HALLO"
```

Auf diese Art erstellt man gewöhnlich Programme und läßt sie laufen. Man bezeichnet den Vorgang als Programmlauf. Im Ausführungs-Modus wartet der Computer, bis der RUN-Befehl eingegeben wird und führt dann das Programm aus.

Regel für den Ausführungs-Modus:

Fängt eine Zeile mit einer Nummer an, wird sie im Speicher abgelegt. Die Zeile wird Teil des Programms. Das Programm wird durch die Eingabe von RUN gestartet.

SCHLAFEN ODER WACH SEIN

Menschen tun das eine, wenn Sie wach sind und etwas anderes, wenn sie schlafen. Sie haben zwei "Operations-Modi".

Du kannst feststellen, daß sie schlafen, weil sie schnarchen. (Natürlich schnarchen nicht alle Leute, aber damit wir besser erklären können, daß sich Computer wie Menschen verhalten, wollen wir annehmen, daß jeder schnarcht.)

Auch der Computer hat zwei Arbeitsweisen. Man nennt sie den "Editier-Modus" und den "RUN-Modus".

DER EDITIER-MODUS

Drücke die BREAK-Taste.

Der Computer meldet sich mit READY. Man nennt dieses Wort "Bereitschaftsanzeige". Der Computer gibt es jedesmal aus, wenn er in den Editier-Modus von BASIC geht. "READY" ist das "Schnarchen" des Computers im Editier-Modus.

Das Quadrat nennt man den "Cursor". Er sagt uns, daß der Computer auf uns wartet, und daß wir etwas eingeben können. Der nächste Buchstabe, den wir eintippen, wird an die Stelle des Bildschirms geschrieben, an der der Cursor steht.

Übung 18: Editier- und RUN-Modus, der Rechner

Solange sich der Computer im Editier-Modus befindet:

- kannst Du Programme eingeben, indem Du Zeilen eintippst, die mit Nummern beginnen.
- kannst Du den Computer wie einen Taschenrechner benutzen. Einen Super-Taschenrechner!
- kannst Du Fehler in Programmen verbessern. Das nennt man "Editieren" eines Programms, und daher hat der "Editier-Modus" auch seinen Namen bekommen. Wir werden später im Buch mehr darüber hören.

DER RECHNER

Im Editier-Modus kannst Du rechnen. Probiers aus:

```
PRINT 3+7
```

Der Computer gibt die Antwort "10".

Natürlich kannst Du auch statt PRINT die Abkürzung "?" verwenden.

DER RUN-MODUS

Gib RUN ein. Damit verläßt Du den Editier-Modus und gehst in den RUN-Modus.

Solange der Computer im RUN-Modus ist:

- läuft das Programm im Speicher ab.

Ist das Programm abgearbeitet, geht der Computer automatisch in den Editier-Modus zurück.

AUFGABE 18:

1. Erkläre, was der "Direkte Modus" bedeutet. Verwende den Computer als Rechner, und löse damit einige arithmetische Aufgaben.

2. Erkläre, was der "Ausführungs-Modus" bedeutet. Schreibe ein Programm mit mehreren Zeilen. In einer Zeile soll "22 und 67 ist" ausgegeben und in einer anderen soll die Addition ausgeführt und das Ergebnis ausgedruckt werden.

3. Wie erkennst Du, daß der Computer im Editier-Modus ist?

4. Was macht der Computer im RUN-Modus?

5. In welchen Modus geht der Computer, nachdem er ein Programm ausgeführt hat?

6. Gib an, wo der nächste Buchstabe, den Du eintippst, am Bildschirm erscheint.

19. ANMERKUNGEN FÜR DEN LEHRER DATA, READ, RESTORE

Dieses Kapitel befaßt sich mit der DATA-Anweisung. READ holt die Daten aus den DATA-Anweisungen, und RESTORE setzt den Zeiger an den Anfang der ersten DATA-Anweisung zurück.

Bei der ersten Begegnung zeigt sich, daß das Speichern von Daten in die DATA-Anweisung einige verwirrende Elemente enthält. Sie sollten nicht versuchen, irgendeinen Teil der Daten in der Anweisung zu ändern, es sei denn, Sie schreiben das Programm neu. Natürlich können Sie mit READ die Daten in eine Variablenschachtel eingeben und sie dann dort ändern.

Sie müssen die Daten mit READ lesen, damit sie verfügbar sind. Sie werden in einer bestimmten Reihenfolge gelesen. Wollen Sie einige Daten auslassen, müssen Sie sie mitlesen und dann wegwerfen. (Dieser Vorgang wird in dem Kapitel nicht behandelt, kann aber dem Schüler erklärt werden, wenn er die anderen Vorstellungen von DATA beherrscht.)

In diesem Kapitel wird die Idee eines "Zeigers" verwendet. Die Benutzung eines Bleistifts, der auf die Abschnitte in den DATA-Anweisungen zeigt, hilft beim Verdeutlichen dieses Konzepts.

Die Anwendung von DATA erspart schwieriges Tippen, wenn eine Menge Daten vorhanden sind.

Es ist aber auch nützlich in den Fällen, in denen weniger Daten vorkommen, weil es die tatsächlichen Daten von den Verarbeitungsdaten klar trennt. Dies hilft bei der Fehlersuche.

Eine der häufigsten Anwendungen von DATA ist das Indizieren von Variablen mit Werten.

Fragen:

1. Was passiert, wenn Du versuchst, mehr Datenabschnitte zu lesen, als in den DATA-Anweisungen enthalten sind?

2. Welche Regel weist Dich darauf hin, wo Du die DATA-Anweisung in Programm einsetzen sollst? Wohin setzt Du die READ-Anweisungen?

3. Kannst Du numerische Daten und Zeichenkettendaten in die gleiche DATA-Anweisung geben?

4. Kannst Du die Abschnitte einer DATA-Anweisung ändern, während das Programm läuft?

ÜBUNG 19: DATA, READ, RESTORE

ZWEI ARTEN VON DATEN

Es gibt zwei Arten von Daten in Deinen Programmen:

1. Eine Art kannst Du über die Tastatur mit Hilfe von INPUT oder GET holen.

```
10 REM ERSTE DATENART
15 DIM P$(40)
20 PRINT "clear"
30 PRINT "WELCHES TIER MAGST DU NICHT?"
35 INPUT P$
40 PRINT "WIRKLICH!"
50 PRINT "DU MAGST KEIN(E)(N)"
60 PRINT P$
```

   In diesem Programm ist P$ das Datum, das vom Anwender während des Programmlaufs eingegeben wurde.

2. Die Art, die zu der Zeit gespeichert wird, während das Programm geschrieben wird.

```
10 REM DIE ZWEITE DATENART
20 PRINT "clear"
30 X=2
40 Y=3
50 PRINT X+Y
```

   In diesem Programm sind X und Y die Daten, die vom Programmierer beim Schreiben des Programms eingegeben wurden.

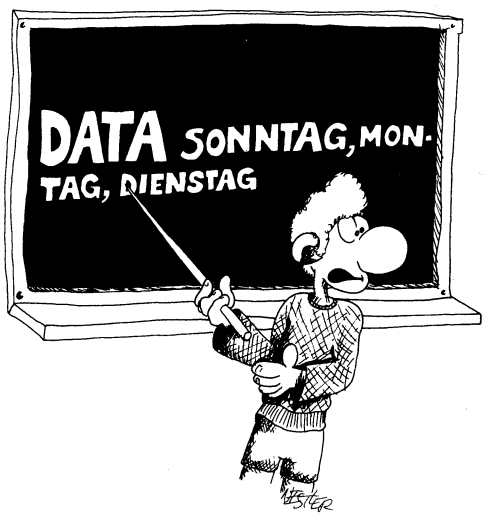
DAS SPEICHERN VON DATENMENGEN

Kleinere Datenmengen kann man schon in der LET-Anweisung speichern. Es ist aber ungeschickt, größere Datenmengen auf diese Weise zu speichern.

Verwende die DATA-Anweisung, um große Datenmengen zu speichern, und die READ-Anweisung, um Daten aus der DATA-Anweisung zu bekommen.

```
10 REM DATENMENGEN
15 DIM D1$(10),D2$(10),D3$(10),D4$(10)
20 PRINT "clear"
30 DATA SONNTAG,MONTAG,DIENSTAG,MITTWOCH,
DONNERSTAG,FREITAG,SAMSTAG
40 READ D1$
42 READ D2$
44 READ D3$
46 READ D4$
60 PRINT D1$,D2$
```

Wenn das Programm läuft, enthält Schachtel D1$ das erste Datenelement der DATA-Liste (SONNTAG), Schachtel D2$ enthält das zweite Datenelement (MONTAG), usw.



SELTSAME REGELN

1. Es ist egal, wo sich die DATA-Anweisungen im Programm befinden. Mache folgendes: Ändere Zeile 30 im obigen Programm um zu Zeile 90. Starte das Programm. Das Programm läuft genauso ab.

2. Es ist egal, wie viele DATA-Anweisungen es gibt.

   Mache folgendes: Teile die DATA-Anweisung auf:

   ```
   90 DATA SONNTAG, MONTAG, DIENSTAG
   91 DATA MITTWOCH, DONNERSTAG, FREITAG, SAMSTAG
   ```

   Laß das Programm laufen. Es läuft genauso ab.

DER READ-BEFEHL UND DER ZEIGER

READ verwendet einen Zeiger. Er zeigt immer auf den nächsten Abschnitt, der gelesen werden soll.

Während des Programmlaufs zeigt der READ-Zeiger auf den ersten Abschnitt der ersten DATA-Anweisung im Programm (das heißt, die DATA-Anweisung mit der niedrigsten Zeilennummer).

Jedesmal, wenn das Programm einen READ-Befehl ausführt, bewegt sich der Zeiger zum nächsten Abschnitt der DATA-Liste.

Erreicht der Zeiger das Ende der DATA-Anweisung, geht er automatisch zur nächsten DATA-Anweisung über(das heißt, zu der DATA-Anweisung mit der nächst höheren Zeilennummer).

Es spielt keine Rolle, daß viele Zeilen dazwischen liegen.

Mache folgendes: Ändere Zeile 90 wieder zu Zeile 30. (Laß Zeile 91 stehen.)

```
30 DATA SONNTAG, MONTAG, DIENSTAG

   ...

91 DATA MITTWOCH, DONNERSTAG, FREITAG, SAMSTAG
```

Starte das Programm. Es läuft genauso.

DAS HERUNTERFALLEN VON DEN "DATA-PLANKEN"

Erreicht der Zeiger den letzten Abschnitt der letzten DATA-Anweisung im Programm, gibt es keine Abschnitte mehr zu lesen. Versuchst Du es aber dennoch, erscheint eine Fehlermeldung:

```
ERROR-
```

ZURÜCK ZU FELD EINS

An jeder Stelle des Programms gibt es nur zwei Auswahlmöglichkeiten für den READ-Zeiger:

1. Du kannst einen weiteren Abschnitt mit READ lesen lassen: dann bewegt sich der Zeiger einen Abschnitt weiter.

2. Du kannst den Befehl RESTORE verwenden: dann wird der READ-Zeiger auf die erste DATA-Anweisung des Programms zurückgestellt.

Übung 19: DATA, READ, RESTORE

DATENMISCHUNGEN

Die DATA-Anweisung kann Zeichenketten und Zahlen in beliebiger Reihenfolge enthalten. Du mußt aber beim READ-Befehl aufpassen, daß die richtige Variablensorte mit der Datensorte übereinstimmt.

Richtig:
```
60 DIM B$(10)
70 DATA 77, STAUB
75 READ N
80 READ B$
```

```
Falsch:   60 DIM B$(10)
          70 DATA 77, STAUB
          75 READ B$          OK,die B$ Schachtel enthält "77"
          80 READ N           ERROR-
```

Du kannst "STAUB" nicht in eine Zahlenschachtel stecken.

AUFGABE 19

1. Schreibe ein Programm, das Deine Verwandten nennt. Fragst Du den Computer nach "ONKEL", soll er Dir die Namen all Deiner Onkel nennen. Die DATA-Anweisungen sollen Zuweisungspaare enthalten: Der erste Ausdruck soll eine Verwandtschaftsbezeichnung wie VATER oder COUSIN sein, der zweite soll den Namen der Person bezeichnen. Hast Du zum Beispiel mehrere Brüder, dann mußt Du für jeden Bruder ein DATA-Paar bilden.

20. ANMERKUNGEN FÜR DEN LEHRER Klangeffekte

Der SOUND-Befehl schaltet jeden der vier Tonkanäle ein oder aus. Tonhöhe, Lautstärke und "Verzerrung" jeder Stimme kann gesetzt werden.

Musiknoten benötigen einen "Verzerrungs"-Wert von 10 (das entspricht ungefähr einer Rechteckform). Die anderen Verzerrungswerte modulieren die Rechteckform und lassen die Töne zischend oder rauher klingen.

Ist der Ton einmal eingeschaltet, klingt er so lange, bis er wieder ausgeschaltet wird.

Bei Grafikspielen erhält man den stärksten Effekt, wenn man die SOUND-Befehle mit den "Grafik-Bewegungs"-Befehlen verwebt.

Um die Noten abzuspeichern, verwendet man am besten die DATA-Anweisung.

Fragen:

1. Welche Zahlen für die Tonhöhe ergeben einen tiefen Ton? Welche eine hohen?

2. Wie schaltest Du die Töne ab?

3. Wie erzeugst Du musikähnliche Geräusche?

4. Wie machst Du ein zischendes Geräusch?

5. Welche Zahl erzeugt das lauteste, welche das leiseste Geräusch?

6. Wie erzeugst Du ein Motorgeräusch?

ÜBUNG 20: KLANGEFFEKTE

Der ATARI besitzt vier Stimmen. Alle können gleichzeitig "singen".

Man kann sie für Musik und andere Effekte einsetzen.

Beispiele:

```
30 SOUND N,  T, V, L
30 SOUND 0,121,10, 8
```

Variable	Wert
N für "Tonkanal-Nummer"	0, 1, 2 oder 3
T für "Tonhöhe"	von 1 bis 255
V für "Verzerrung"	gerade Zahlen von 0 bis 14
L für "Lautstärke"	von 0 bis 15

STIMMEN

Alle Stimmen ähneln sich. Es spielt keine Rolle, welche Du verwendest. Insgesamt gibt es vier Stimmen, deshalb kannst Du zwei, drei oder alle vier zusammen erklingen lassen.

DIE TONHÖHE

Die Tonhöhe bestimmt, ob Du eine "hohe Note" oder eine "tiefe Note" hast. Um so größer die Zahl, um so tiefer ist die Tonhöhe. Anschließend findest Du eine Tonleiter:

Note	Zahl
C (tiefes C)	243
C #	229
D	216
D #	204
E	193
F	182
F #	172
G	162
G #	153
A	144
A #	136
B	129
C (mittleres C)	121
C #	115
D	108
D #	102
E	96
F	91
F #	86
G	81
G #	77
A	72
A #	68
B	64
C (tiefes C)	60

Diese Noten sind nicht ganz rein.

Der Grund dafür ist:

Im SOUND-Befehl sind nur Ganzzahlen erlaubt,

aber,

um einen reinen Ton zu spielen, müßte man Dezimalzahlen einsetzen.

Gibst Du für die Tonhöhe Zahlen in der Größenordnung von 10 ein, erhältst Du sehr hohe Töne. Wahrscheinlich hörst Du bei den Zahlen 2, 1 oder 0 überhaupt nichts.

Übung 20: Klangeffekte

VERZERRUNG

Das "Timbre" oder die Tonqualität kann man mit der Verzerrungszahl ändern.

Die Töne klingen am klarsten bei Verzerrung 10.

Um andere Toneffekte zu bekommen, kann man eine andere Verzerrung wählen.

Versuche auch andere geradzahlige Werte im unterstehenden Programm.

Kannst Du Donner, das Zischen einer Schlange, den Motor eines Bootes, das Geräusch einer Laserkanone nachmachen?

LAUTSTÄRKE

Bei Null hörst Du gar keinen Ton mehr.

Die normale Lautstärke erhältst Du mit der Zahl 8.

Am lautesten wird es bei 15.

Natürlich kannst Du auch den Lautstärkeregler am Fernsehgerät aufdrehen, damit es lauter wird.

> VORSICHT!
> Wenn zur gleichen Zeil mehr als eine Stimme erklingt, darf die Summe der Lautstärkezahlen 32 nicht übersteigen.

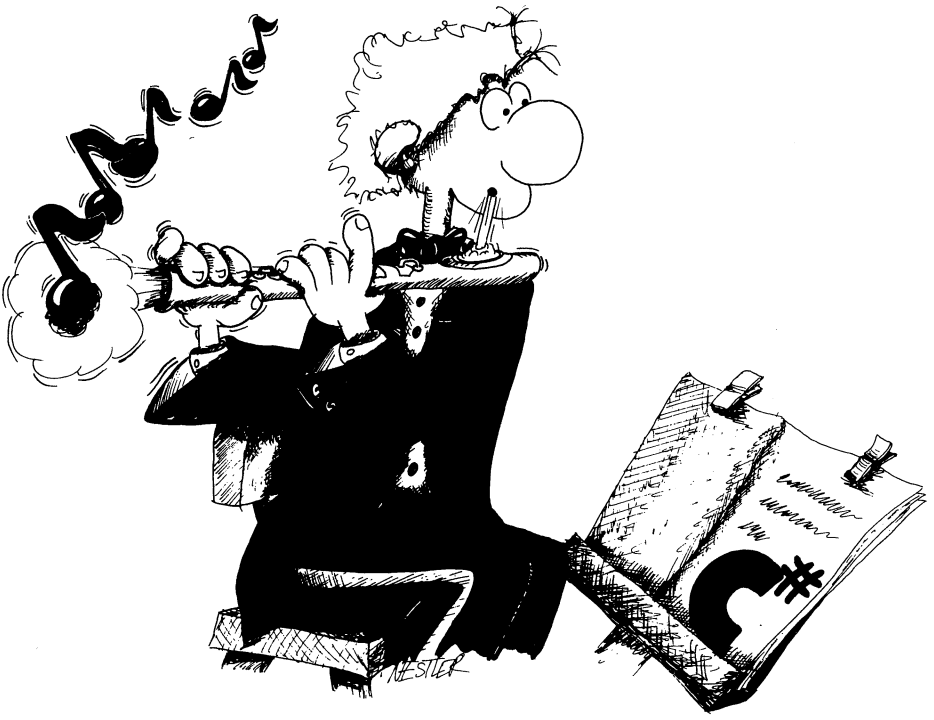
Ein Beispiel:

```
30 SOUND 0,200,10,9
32 SOUND 1, 35,8 ,11
34 SOUND 2,155,12,9
```

So ist es richtig, denn 9+11+9 ergibt erst 29.

DIE EINZELSTIMME

```
10 REM EINZELSTIMME
20 PRINT "STIMME NULL"
25 PRINT "WELCHE TONHOEHE <1 BIS 255>"
26 INPUT T
30 PRINT "WELCHE VERZERRUNG <0 BIS 14, GERADZAHLIG>"
31 INPUT V
35 PRINT "WIE LAUT <0 BIS 15>"
36 INPUT L
40 SOUND 0,T,V,L
45 FOR Z=1 TO 1000:NEXT Z
50 GOTO 20
```



Hast Du das Programm mit der BREAK-Taste angehalten, mußt Du den Befehl END eingeben, damit Du kein Geräusch mehr hörst.

Du kannst aber auch noch im Programm bei der Lautstärke den Wert Null (0) eingeben und dann erst auf die BREAK-Taste drücken.

EIN DUETT

```
10 REM DUETT
20 PRINT "STIMMEM 0 UND 1"
25 PRINT "TONHOEHE 0":INPUT T0
30 PRINT "TONHOEHE 1":INPUT T1
40 SOUND 0,T0,10,8
42 SOUND 1,T1,10,8
45 FOR Z=1 TO 1000:NEXT Z
50 GOTO 20
```

AUFGABE 20

1. Erstelle eine Liste der Geräusche, die jede Verzerrungszahl erzeugt. Versuche für jede Verzerrungszahl 0, 2, 4, 6, 8, 10, 12 und 14 die Tonhöhen 244, 122, 66, und 33. Schreib Dir die Art der Geräusche auf. Überlege Dir ein Spiel oder ein Programm, wo Du diese Geräusche einsetzen könntest.

2. Versuche, mit dem Computer eine kleine Melodie wie "Hänschen klein" oder "Alle meine Entchen" zu spielen. Du findest die Noten in einem Notenheft. Leg die Zahlen für die Tonhöhe in einer DATA-Anweisung ab.

21. ANMERKUNGEN FÜR DEN LEHRER Die Farbgrafik

In dieser Übung werden die Befehle GRAPHICS 3, SETCOLOR, COLOR, PLOT und DRAWTO behandelt.

Bei einen Schwarz/weiß-Monitor können nur die Grafikbefehle verwendet werden. Die Zeichnungen werden hier allerdings deutlicher.

Wird ein Farbmonitor eingesetzt, sollte der Schüler die Farben erst richtig einstellen. Am besten eignen sich dabei die Farben Lila und Goldgelb.

Im GRAPHICS-3-Modus werden ziemlich große Farbflecken als Bildpunkte verwendet. Sie haben rechteckige Form und sind so groß wie ein Buchstabe im Textmodus.

Im GRAPHICS-5- und GRAPHICS-7-Modus werden kleinere Bildpunkte verwendet. Beherrscht der Schüler erst einmal den GRAPHICS-3-Modus, wird es ihm ein Leichtes sein, die anderen Grafik-Modi einzusetzen.

Das punktweise Zeichnen von Bildern ist ziemlich langweilig. Besser ist es, wenn man den DRAWTO-Befehl dazu hernimmt.

Auf jeden Fall sollte zuerst die Zeichnung auf einem karierten Stück Papier entworfen werden. Ich schlage vor, eine Ecke oder die Mitte des Bildes mit einer Variablen zu bestimmen, und dann die anderen Punkte und Linien in der Zeichnung mit der Variablen plus Offset anzugeben. So kann man die ganze Figur bewegen oder an einen anderen Ort plazieren.

Übung 21: Die Farbgrafik

Fragen:

1. Was passiert, wenn Du den Befehl GRAPHICS 3 gibst?

2. Welchen Pinsel mußt Du für Topf 2 verwenden? Welcher Befehl "nimmt den Pinsel auf"?

3. Wie viele Farben kannst Du auswählen?

4. Wie viele Farben kannst Du mit einmal einsetzen?

5. Welcher Zahlenbereich ist für X und Y im Befehl PLOT X,Y erlaubt?

6. Was bringt diese Zeile auf den Bildschirm?

```
20 PLOT 1,5 : DRAWTO 1,20
```

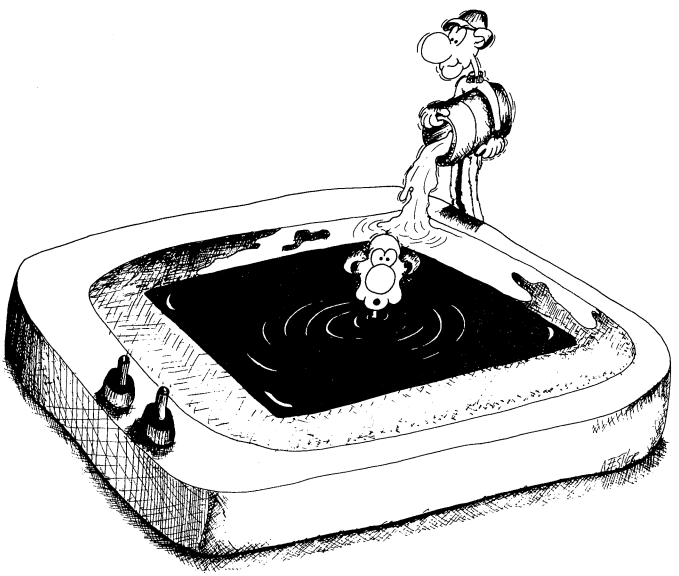
ÜBUNG 21: DIE FARBGRAFIK

Bevor Du mit der Farbgrafik loslegen kannst, mußt Du dem Computer erst klarmachen, mit welcher Art von Grafik Du arbeiten willst.

In Übung 29 zeigen wir, wie die restlichen Grafik-Modi eingesetzt werden.

FARBTÖPFE

Der ATARI hat fünf "Farbtöpfe", die von 0 bis 4 numeriert sind.



Bevor Du Farbe verwenden kannst, mußt Du erst mit dem SETCOLOR-Befehl "reine Farbe" in die Farbtöpfe geben.

```
SETCOLOR N,F,H
```

N ist die Nummer des Farbtopfs (von 0 bis 4)
F ist die Farbe, die in den Topf gegeben wird (von 0 bis 14)
H ist die Helligkeit der Farbe (gerade Zahlen von 0 bis 14)

Du kannst aus 15 Farben wählen, aber mit einmal kannst Du nur 5 Farben verwenden (Du hast nämlich nur 5 Farbtöpfe zur Verfügung).

Im GRAPHICS-3-Modus kannst Du nur die Töpfe 0, 1, 2 und 4 verwenden.

0 GRAU	8 BLAU
1 GOLDGELB	9 HELLBLAU
2 ORANGE	10 TÜRKIS
3 ROT-ORANGE	11 GRÜNBLAU
4 ROSA	12 GRÜN
5 LILA	13 GELBGRÜN
6 FLIEDER	14 ORANGE-GRÜN
7 BLAU	15 HELLORANGE

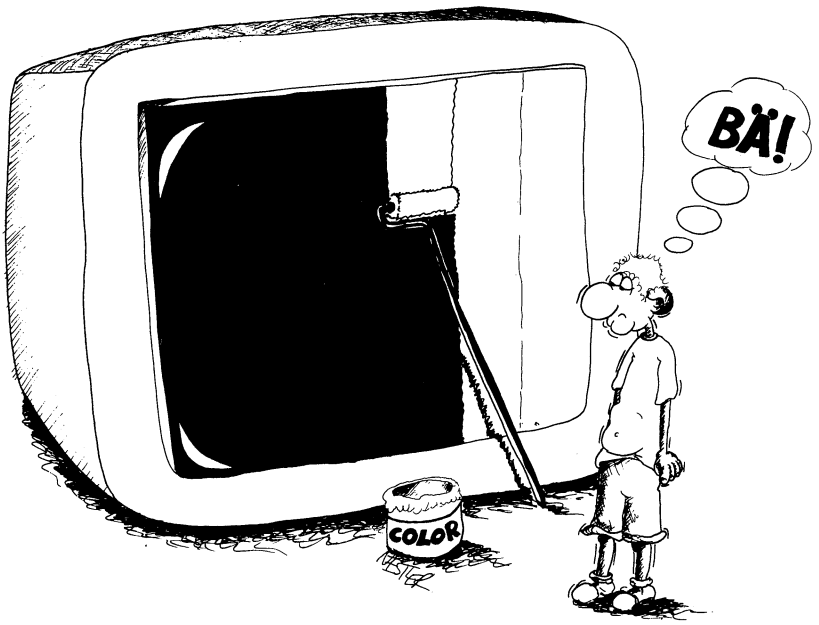
PINSEL

Du kannst 5 "Farbpinsel" in die Farbtöpfe eintauchen. Jeder Pinsel wird einem Farbtopf zugeordnet. Folgende Pinsel werden im GRAPHICS-3-Modus eingesetzt:

Pinsel 1 in Topf 0
Pinsel 2 in Topf 1
Pinsel 3 in Topf 2
Pinsel 0 in Topf 4

(Ich kann nichts dafür, daß die Nummern nicht übereinstimmen.)

Topf 4 dient einem besonderen Zweck. Die Farbe, die Du in Farbtopf 4 gibst, erscheint unmittelbar am Bildschirm als Hintergrundfarbe. Du kannst auch den Pinsel COLOR 0 hineintauchen und alle Punkte löschen, die Du früher gemalt hast.



Farbtopf	Farbpinsel
SETCOLOR 0,F,H	COLOR 1
SETCOLOR 1,F,H	COLOR 2
SETCOLOR 2,F,H	COLOR 3
SETCOLOR 4,F,H	COLOR 0 und Hintergrund

Mit dem COLOR-Befehl nimmst Du einen "Pinsel in die Hand", mit dem Du dann zeichnen kannst.

EIN PROGRAMM MIT ZWEI FARBEN

Laß das Programm laufen:

```
10 REM (((( LAECHELN ))))
15 DIM A$(1)
20 GRAPHICS 0
25 PRINT "HINTERGRUNDFARBE <0 BIS 15>":INPUT HF
26 PRINT "BILDFARBE <0 BIS 15>":INPUT F
30 GRAPHICS 3+16
32 SETCOLOR 4,HF,8
35 SETCOLOR 0,F,8
40 COLOR 1
42 PLOT 20,10
44 PLOT 21,11
46 PLOT 22,12
48 PLOT 23,12
50 PLOT 24,12
52 PLOT 25,11
54 PLOT 26,10
60 FOR Z=1 TO 1000:NEXT Z
90 PRINT "NOCH EINMAL <J/N>";:INPUT A$
95 IF A$="J" THEN GOTO 20
99 GRAPHICS 0
```

Speichere das Programm auf Kassette ab.

Mache den Hintergrund zuerst lila und das Lächeln goldfarben. Stelle dann Dein Fernsehgerät oder Deinen Farbmonitor so ein, daß Du deutliche Farben erhältst.

Was geschieht?

Zeile 20 Der Bildschirmmodus wird in den Textmodus gesetzt.

Zeile 25 Die Farbe für den Hintergrund wird eingegeben.

Zeile 26 Die Farbe für das Lächeln wird eingegeben.

Zeile 30 Der Bildschirm wird in den GRAPHICS-3-Modus geschaltet. Der Anhang "+16" bedeutet, daß der gesamte Schirm in diesen Grafik-Modus gesetzt wird. Sonst erscheinen vier Textzeilen am unteren Ende.

Zeile 32 Die Hintergrundfarbe wird gesetzt.

Zeile 35 Der Farbtopf 0 wird mit der eingegebenen Farbe "gefüllt".

Zeile 40 Pinsel 1 wird verwendet. (Er taucht in Topf 0.)

Zeile 42 Macht einen Farbklecks auf Spalte 20 und Zeile 10.

DER PLOT-BEFEHL

PLOT 4,13 Setzt einen Farbtupfer in die 4. Zeile und 13. Spalte.

Regel:

Der Befehl PLOT X,Y bedeutet: Mache einen Fleck am Bildschirm bei Punkt X von links aus und bei Punkt Y von oben aus gezählt. Im GRAPHICS-3-Modus ist X eine Variable im Bereich von 0 bis 39, Y eine Variable im Bereich von 0 bis 23.

DER DRAWTO-BEFEHL

DRAWTO kann man so übersetzen: zeichne nach.

Ändere Zeile 44 folgendermaßen:

```
44 DRAWTO 3,9
```

Damit wird zwar das Bild kaputtgemacht, aber man sieht, wie Linien gezogen werden.

Regel:

Der DRAWTO X,Y-Befehl zieht eine Linie von der Stelle, an der der Cursor gerade steht, bis zu einem Punkt X von links und einem Punkt Y von oben.

AUFGABE 21

1. Füge im "Lächel"-Programm die Zeilen 27 und 28 hinzu, damit auch noch die Helligkeit (Zeilen 32 und 35) geändert werden kann.

2. Zeichne im Programm noch eine Nase und Augen.

3. Verwende die Befehle PLOT und DRAWTO, um ein Bild aus 3 Farben zu malen (auch eine Hintergrundfarbe).

4. Vervollständige das Zahlenratespiel aus Übung 13 so, daß ein farbiger Stern anzeigt, wenn die Zahl richtig erraten wurde. Damit der Stern eine Zeitlang zu sehen ist, bevor das nächste Spiel beginnt, mußt Du eine Zeitschleife einführen.

5. Schreibe ein Programm, das "Sindbads Wunderteppich" zeichnet. Der Spieler soll die Farben des Teppichs selbst wählen können. Zeichne auch ein Muster auf den Bildschirm.

22. ANMERKUNGEN FÜR DEN LEHRER ASCII-Code

In dieser Übung behandeln wir den ASCII-Code für Zeichen und die Funktionen ASC() und CHR$(), mit denen man Zeichen in Zahlen und umgekehrt umwandeln kann.

Der ASCII-Code besteht aus Zahlen von 0 bis 127. Groß- und Kleinbuchstaben, Ziffern und Satzzeichen werden ASCII-Zahlen zugewiesen. Zeichen wie "bell" (=Tongeber), "line feed" (=Zeilenvorschub) und "carriage return" (=Wagenrücklauf) sind auch im ASCII-Code enthalten. Sie dienen als Steuerzeichen für Hardware.

Der ASCII-Code standardisiert in erster Linie den Austausch der Signale zwischen den einzelnen Hardware-Komponenten (Computer und Drucker, Computer und Terminals, Computer und Computer usw.). Auch innerhalb von Programmen ist es sinnvoll, ASCII-Zahlen zu verwenden. Die Buchstaben sind in aufsteigender Reihenfolge numeriert. Ordnet man die ASCII-Zahlen, erhält man das Alphabet. Auch die Zahlen erscheinen der Reihe nach, ebenfalls Satz- und Grafikzeichen.

Der ATARI-Computer verwendet einen geänderten ASCII-Zeichensatz, den man ATASCII nennt. Er schließt den normalen Buchstabencode ein, aber er enthält auch Grafikzeichen. Die ATASCII-Zahlen von 128 bis 255 entsprechen denen zwischen 0 und 127, allerdings sind sie invers dargestellt.

Übung 22: ASCII-Code

Fragen:

1. Gibt ASC(S$) eine Zeichenkette oder eine Zahl als Wert zurück?

2. Ist das Argument von ASC(S$) eine Zeichenkette oder eine Zahl?

3. Beantworte die beiden ersten Fragen auch für CHR$(N).

4. Welcher Buchstabe besitzt die höhere ASCII-Code-Zahl, B oder W?

5. Kennst Du den ASCII-Code für die Zahl "1"? Ist es die Zahl 1?

ÜBUNG 22: ASCII-CODE

NUMERIERTE BUCHSTABEN IM ALPHABET

"Das ist leicht", wirst Du sagen. "A ist 1, B ist 2, C ist 3 ..."

Du wirst Dich wundern, weil es ganz anders aussieht: A ist 65, B ist 66, C ist 67 ... .

Man nennt diese Zahlen den ASCII-Code für Zeichen. ASCII spricht man aus: "aski".

Auch die Satzzeichen, die Zahlen und die Grafikzeichen haben ASCII-Code-Zahlen.

ASCII- UND ATASCII-ZEICHEN

ASCII-Zahlen werden von allen möglichen Conmputern verwendet.

Der ATARI setzt eine besondere Art von ASCII-Code ein, den man ATASCII-Code nennt.

Für alle Buchstaben, Zahlen und Satzzeichen entspricht der ASCII-Code dem ATASCII-Code. Dieser Code liegt im Bereich von 32 bis 127 (außer 96).

Im Bereich von 0 bis 31 unterscheidet sich ATASCII von ASCII: Im ATASCII-Code werden da Grafikzeichen dargestellt.

Die ATASCII-Zahlen von 128 bis 155 bilden gewöhnlich die inverse Darstellung der Zeichen von 0 bis 127.

CHR$() WANDELT ZAHLEN IN ZEICHEN UM

Mit CHR$() wandelt man ATASCII-Code-Zahlen in eine Zeichenkette um, die ein Zeichen enthält.

Laß das Programm laufen:

```
10 REM ____ ALLE ATASCII-ZAHLEN ANZEIGEN ____
11 REM
20 PRINT "clear"
30 FOR I=0 TO 255
40 PRINT I,CHR$(I)
50 FOR T=1 TO 500:NEXT T
60 NEXT I
```

Wenn das Programm läuft, siehst Du die Zahlen und die dazugehörigen ATASCII-Zeichen. Bei den Zahlen 27 bis 31, 125 bis 127, 155 bis 159 und 253 bis 255 passiert etwas Merkwürdiges.

Diese Zahlen sind Steuerzeichen. Sieh in der Liste für die ATASCII-Zahlen im ATARI 400/800-BASIC-Handbuch ihre Bedeutung nach.

Die Zahl 253 erzeugt ein summendes Geräusch.

ASC() WANDELT ZEICHEN IN ZAHLEN UM

Mit der ASC()-Funktion kann man Zeichen in Zahlen umwandeln.

Starte:

```
10 REM * WELCHE ZAHL HAT DIE TASTE? *
15 DIM C$(1)
17 OPEN #1,4,0,"K:"
20 PRINT "clear"
25 PRINT "DRUECKE EINE TASTE,"
26 PRINT "DANN SIEHST DU DIE ASCII-ZAHL"
30 GET #1,C
35 C$=CHR$(C)
40 PRINT C$;"      ";ASC(C$)
50 GOTO 30
```

Probiere das Programm mit verschiedenen Buchstaben, Zahlen, Satzzeichen und Grafikzeichen aus. Versuch's auch einmal mit der RETURN-Taste.

Versuch das noch: Halte die CONTROL-Taste fest und drücke Tasten. Halte die SHIFT-Taste fest und drücke Tasten.

Halte das Programm mit der BREAK-Taste an.

GET und OPEN werden in der nächsten Übung erklärt.

DAS ALPHABET

Was kann man mit den ASCII-Zahlen alles anfangen? Zum Beispiel ein Alphabet erstellen.

```
Starte:   10 REM ALPHABETISCH
          15 DIM A$(1), B$(1)
          20 PRINT
          30 PRINT "TIPPE EINEN BUCHSTABEN";:INPUT A$
          36 PRINT
          40 PRINT "TIPPE NOCH EINEN";:INPUT B$
          45 A=ASC(A$):B=ASC(B$)
          47 REM GEORDNET NACH
          48 REM INHALT VON
          49 REM ASCII-ZAHLEN
          50 IF A>B THEN X=A:A=B:B=X: REM VERTAUSCHEN
          55 PRINT
          60 PRINT "DER REIHE NACH"
          65 PRINT:PRINT CHR$(A);"     ";CHR$(B)
```

Betrachte diese Funktionen: ASC() und CHR$().

Mit ASC() erhältst Du die ASCII-Zahl des **ersten** Buchstabens einer Zeichenkette.

CHR$() ist die Umkehrfunktion. Damit erhältst Du das Zeichen, das einer ASCII-Zahl entspricht.

AUFGABE 22

1. Schreibe ein Programm, das als Eingabe ein Wort verlangt. Anschließend sollen alle Buchstaben des Wortes in alphabetischer Reihenfolge wieder ausgegeben werden.

2. Schreibe ein Programm, das eine unverständliche Sprache "spricht". Aus einem eingegebenen Satz werden alle Vokale entfernt und der Rest dann ausgedruckt.

23. **ANMERKUNGEN FÜR DEN LEHRER** Geheimschrift und der GET-Befehl

Dieses Kapitel befaßt sich mit dem GET-Befehl.

Mit GET besteht die Möglichkeit, ein einzelnes Zeichen von der Tastatur zu holen.

Es wird durch den Bildschirm nicht angezeigt. Es wird weder eine Bereitschaftsanzeige noch ein Cursor gezeigt. Die erfolgte Tastenbetätigung wird am Bildschirm auch nicht bestätigt.

Das ist der Vorteil des GET-Befehls. Man kann zum Beispiel ein gesuchtes Wort durch eine Reihe von GET-Befehlen eingeben, ohne daß es die anderen mitbekommen.

Ein weiterer Vorteil gegenüber dem INPUT-Befehl liegt darin, daß die Betätigung der RETURN-Taste nicht erforderlich ist. Deswegen ist GET gut geeignet zum "anwenderfreundlichen" Programmieren von Menüs bzw. zur Erzeugung von Bewegung in Spielen.

Ehe man GET verwenden kann, muß man die Tastatur als Eingabegerät vereinbaren (im Fachjargon nennt man das Eröffnen, also Englisch: OPEN). Erst dann erkennt der GET-Befehl ein Zeichen von der Tastatur als ATASCII-Zahl. Mit CHR$() wandelt man die Zahl in ein Zeichen um.

Fragen:

1. Vergleiche die INPUT- und GET-Befehle. Ein Befehl holt einen Buchstaben, der andere ganze Wörter und Sätze. Einer besitzt einen Cursor, der andere nicht. Einer druckt die Eingabe am Bildschirm aus, der andere nicht. Bei einem muß die RETURN-Taste gedrückt werden, beim anderen nicht. Wer macht was?

ÜBUNG 23: GEHEIMSCHRIFT UND DER GET-BEFEHL

DER INPUT-BEFEHL

Beispiele:

```
10 INPUT A$
10 INPUT N
10 INPUT NAME$,ALTER,TAG,MONAT$,JAHR
```

Das Fragezeichen (?) und der Cursor zeigen Dir, daß Du einen Buchstaben, ein Wort, einen Satz oder eine Zahl eingeben sollst.

DER GET-BEFEHL UND DIE GEHEIMSCHRIFT

Der GET-Befehl unterscheidet sich vom INPUT-Befehl. Er übernimmt ein einzelnes Zeichen von der Tastatur.

nichts sieht man auf dem Bildschirm

kein Fragezeichen wird gezeigt

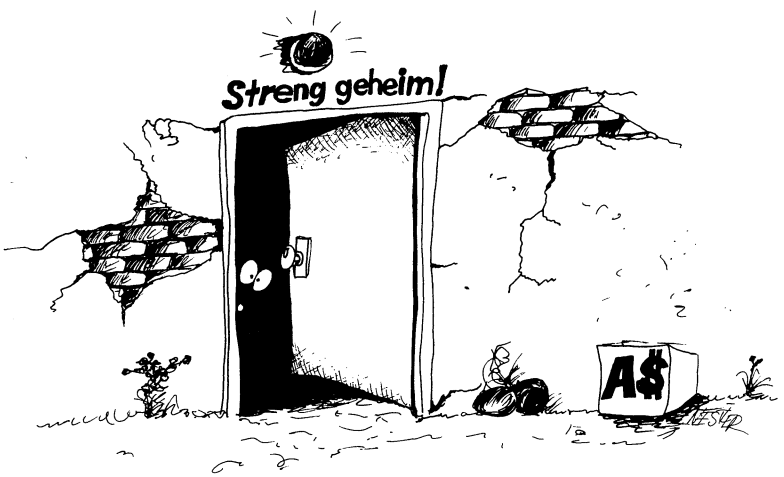
kein Cursor wird gezeigt

was Du eingegeben hast, wird nicht gezeigt.

In Ratespielen kannst Du GET verwenden, um ein Wort oder ein Zahl einzugeben, die geraten werden sollen, ohne daß sie der Spieler sehen kann.

Laß das Programm laufen:

```
10 REM ----- GET -----
15 DIM K$(1)
17 OPEN #1,4,0,"K:"
20 PRINT "clear"
30 PRINT "DRUECKE EINE TASTE"
40 GET #1,K
45 K$=CHR$(K)
47 FOR T=1 TO 500:NEXT T
50 PRINT "DIE TASTE, DIE DU GEDRUECKT HAST, IST ";K$
55 FOR T=1 TO 500:NEXT T
99 GOTO 20
```



In Zeile 17 steht der OPEN-Befehl. Mit OPEN wird dem Computer gesagt, daß er die Tastatur unter einem anderen Namen abfragen soll: "Ein/Ausgabegerät #1".

In Zeile 40 steht der GET-Befehl. GET #1,K heißt: "Sieh bei dem Ein/Ausgabegerät mit der Nummer 1 nach und lege das, was dort eingegeben wird, in die Varibalenschachtel K."

Die Zahl, die die Tastatur zur Schachtel K sendet, ist die ASCII-Zahl der Taste, die gedrückt wurde.

WIE MAN AUS BUCHSTABEN WÖRTER BILDET

Der GET-Befehl kann mit einmal immer nur einen Buchstaben annehmen. Damit daraus Wörter gebildet werden, müssen Zeichenketten verbunden werden.

```
10 REM EIN WORT HOLEN
15 DIM W$(40), L$(1)
17 OPEN #1,4,0,"K:"
20 PRINT "clear"
30 PRINT "GIB EIN WORT EIN. BEENDE ES MIT EINEM PUNKT."
35 I=1:W$="":REM ZEICHENKETTE IST LEER
40 GET #1,Q:L$=CHR$(Q):REM EINEN BUCHSTABEN HOLEN
50 IF L$="." THEN GOTO 80:REM EINGABEENDE?
60 W$(I)=L$:I=I+1:REM BUCHSTABEN ANFUEGEN
65 GOTO 40: REM NAECHSTEN BUCHSTABEN HOLEN
80 REM ENDE DES WORTES
85 PRINT W$
```

Wie weiß der Computer, daß das Wort vollständig eingegeben wurde? Er sieht einen Punkt am Ende des Wortes. Zeile 50 prüft, ob ein Punkt eingetippt wurde und beendet dann gegebenenfalls die Worteingabe. In Zeile 60 werden die Buchstaben "aneinandergeklebt", damit eine Zeichenkette (das Wort) entsteht. In Übung 26 erfährst Du mehr über das Verbinden von Zeichenketten.

AUFGABE 23

1. Schreibe ein Programm, das dem Anwender ein "Menü" zur Auswahl stellt. Der Anwender trifft seine Wahl durch Eintippen eines Buchstabens. Verwende GET, um den Buchstaben zu holen. Beispiel:

```
PRINT "WELCHE FARBE? <R=ROT, B=BLAU, G=GRUEN>"
```

2. Schreibe ein Programm, das Sätze erzeugt. Jeder Satz besteht aus einem Subjekt, einem Verb und einem Objekt. Der erste Spieler gibt ein Subjekt ein (wie "Der Esel"). Der zweite Spieler tippt ein Verb (wie "singt") ein. Der dritte Spieler tippt das Objekt (wie "der Zahnstocher") ein. Verwende GET, so daß kein Spieler die Wörter der anderen sehen kann. Du kannst das Spiel mit Adjektiven erweitern.

24. ANMERKUNGEN FÜR DEN LEHRER Nette Programme, GOSUB, RETURN und END

Dieses Kapitel befaßt sich mit Unterprogrammen. Der END-Befehl wird auch behandelt, weil das Programm normalerweise seine Unterprogramme bei den höheren Zeilennummern hat und deswegen bei den mittleren Zeilennummern ENDen muß.

Unterprogramme werden sinnvollerweise nicht nur in langen Programmen eingesetzt, sondern auch in kurzen, wo das Unterteilen der Aufgabe in einzelne Bereiche für größere Klarheit sorgt.

Mancher Schüler tut sich schwer (auch mancher Profi), sich an strukturiertes Programmieren zu gewöhnen.

Machen Sie den Schüler darauf aufmerksam, in welcher Weise strukturiert werden kann und welche Vorteile bezüglich der Nachvollziehbarkeit solches Programmieren hat. Die wichtigsten Verfahrensweisen, die in diesem Buch angeboten werden, sind das Schreiben von klaren REM-Anweisungen und die Verwendung des modularen Aufbaus von Programmen.

Mit GOSUB werden in BASIC Module erzeugt. In dieser Übung wird der modulare Aufbau am Beispiel des "Mann-hängt-am-Galgen"-Spiels gezeigt.

Fragen:

1. Was geschieht, wenn der END-Befehl ausgeführt wird?

2. Wie unterscheidet sich der GOSUB-Befehl vom GOTO-Befehl?

3. Was passiert, wenn RETURN ausgeführt wird?

4. Was passiert, wenn RETURN vor dem GOSUB-Befehl ausgeführt wird?

5. Was bedeutet "rufe das Unterprogramm auf"?

6. Wie viele END-Befehle darfst Du in einem Programm verwenden?

7. Wieso willst in Du in einem Programm Unterprogramme verwenden?

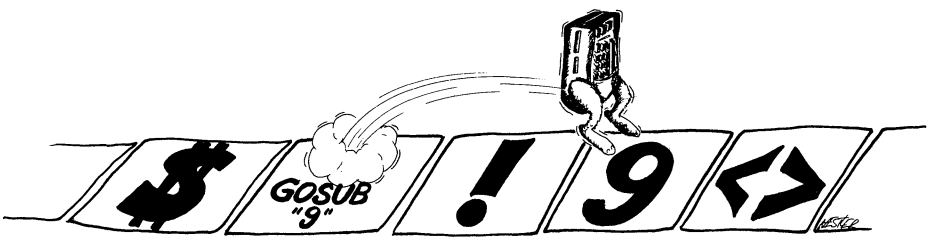
ÜBUNG 24: NETTE PROGRAMME, GOSUB, RETURN UND END

Laß dieses Programm laufen und speichere es dann ab:

```
100 REM HAUPTPROGRAMM
101 REM
110 PRINT "SPRINGE ZUM UNTERPROGRAMM"
120 GOSUB 200
130 PRINT "VOM UNTERPROGRAMM WIEDER ZURUECK"
133 PRINT
135 PRINT "WIEDER ZUM UNTERPROGRAMM"
140 GOSUB 200
150 PRINT "ENDLICH ZU HAUSE"
190 END
199 REM
200 REM UNTERPROGRAMM
201 REM
210 PRINT "IM UNTERPROGRAMM"
215 FOR T=1 TO 1000:NEXT T
217 PRINT "piepser"
290 RETURN
```

Dies ist das Gerüst eines langen Programms. Das Hauptprogramm beginnt in Zeile 100 und endet in Zeile 190.

Wo es PRINT-Befehle gibt, kannst Du viele weitere Programmzeilen einsetzen.



Der END-Befehl in Zeile 190 sagt dem Computer, daß das Programm beendet ist. Der Computer geht wieder in den Editier-Modus zurück.

Zeile 120 und Zeile 140 "rufen das Unterprogramm auf". Dies bedeutet, daß der Computer zu den Befehlen in dem Unterprogramm geht, diese ausführt und dann zurückkehrt.

Der GOSUB-200-Befehl verhält sich wie ein GOTO-200-Befehl mit dem Unterschied, daß der Computer sich daran erinnert, woher er gekommen ist, so daß er dann dorthin zurückkehren kann.

Der RETURN-Befehl sagt dem Computer, daß er zu der Anweisung zurückkehren soll, die dem GOSUB-Befehl folgt.

WIE SIEHT EIN GUTES UNTERPROGRAMM AUS?

In einem kurzen Programm ist es wenig sinnvoll.

In einem langen Programm macht es zwei Dinge:

1. Es erspart Dir viel Arbeit und Speicherplatz. Du brauchst die gleichen Programmzeilen in den verschiedenen Teilen des Programms nicht zu wiederholen.

2. Es macht das Programm leichter verständlich. Außerdem kann man das Programm schneller schreiben und leichter Fehler suchen.

DER END-BEFEHL

Ein Programm kann keinen, einen oder viele END-Befehle enthalten.

Regel:

Der END-Befehl sagt dem Computer, daß er das Programm abbrechen und in den Editier-Modus zurückkehren soll.

Das ist eigentlich alles, was er macht. Du kannst einen END-Befehl an jede beliebige Stelle des Programms setzen: Zum Beispiel nach dem THEN einer IF-Anweisung.

BEWEGLICHE BILDER

```
10 REM  ??? SPRINGENDES J ???
12 PRINT "clear"
13 GRAPHICS 3+16
14 SETCOLOR 0,6,8
18 SETCOLOR 4,0,0
20 X=15:Y=10:D=1
25 FOR J=1 TO 10
26 FOR I=1 TO 10
30 COLOR 1:GOSUB 100:REM ZEICHNEN
35 COLOR 0:GOSUB 100:REM LOESCHEN
45 Y = Y - D
50 NEXT I
55 D = -D
60 NEXT J
99 END
100 REM
101 REM ZEICHNE DAS J
102 REM
110 PLOT X,Y:DRAWTO X+6,Y
115 PLOT X+3,Y+1:DRAWTO X+3,Y+7
120 PLOT X,Y+7:DRAWTO X+2,Y+7
125 PLOT X,Y+6
190 RETURN
```

Speichere auf Band oder Diskette.

Die Zeichnung ist der Buchstabe J. Das Unterprogramm, das in Zeile 100 beginnt, zeichnet das J.

Das Unterprogramm zeichnet das J mit seiner linken oberen Ecke an der Stelle X,Y auf dem Bildschirm. Wenn Du X, Y (oder beides) vertauschst, wird das J an einer anderen Stelle auf dem Bildschirm erscheinen. Zeile 20 bestimmt, daß das erste J ungefähr in der Bildschirmmitte erscheint.

Die Variable D sagt Dir, wie weit das J sich von einer Zeichnung zur anderen weiterbewegt. Zeile 20 setzt D gleich 1, aber Zeile 55 ändert D in -1 um, nachdem sechs Bilder gezeichnet wurden.

Zeile 45 besagt, daß jedes Bild an der Stelle gezeichnet wird, an der Y um den Betrag D größer gegenüber dem letzten Y ist.

AUFGABE 24A

1. Gib das "Springende-J"-Programm ein und laß es laufen. Nimm dann folgende Änderungen vor:

   Ändere das Unterprogramm so, daß es Deinen eigenen Anfangsbuchstaben schreibt.

   Färbe Deinen Anfangsbuchstaben blau.

   Laß den Buchstaben nicht springen, sondern gleiten (so daß das J sich waagrecht und nicht senkrecht bewegt).

   Ändere den Anfangspunkt. Setze ihn an die rechte untere Ecke des Bildschirms.

   Ändere den Weg, den das J zurücklegt. Laß es 12 statt 10 Schritte gehen.

   Ändere die Schrittgröße von eins auf zwei.

   Ändere die Richtung, so daß der Buchstabe nach oben gleitet:

   ```
   X=X+D:Y=Y-D
   ```

   Ändere das Programm so, daß der Buchstabe in der Farbe von Grau (Farbe 0) nach Orange (Farbe 15) wechselt.

WIE MAN EIN LANGES PROGRAMM SCHREIBT

Nun wollen wir ein "Mann-hängt-am-Galgen"-Programm schreiben. Dabei handelt es sich um ein Ratespiel, bei dem jedesmal, wenn ein Buchstabe nicht erraten wurde, ein weiteres Teil eines Mannes am Galgen baumelt.

Mache Dir zuerst einen Programmentwurf. Das kannst Du auf dem Papier oder direkt auf dem Bildschirm erledigen. Hast Du Schwierigkeiten, dann probiere das Spiel auf einem Stück Papier aus, und verfolge den Ablauf. Das Programm muß in der gleichen Reihenfolge arbeiten.

Der Entwurf könnte so aussehen:

```
10 REM ------ MANN-HAENGT-AM-GALGEN-SPIEL
200 REM SPIELANWEISUNGEN
300 REM DAS ZU RATENDE WORT HOLEN
400 REM RATEVERSUCH
500 REM RICHTIG?
600 REM ZEICHNEN
700 IST DAS SPIEL ZU ENDE?
800 REM SPIELENDEMELDUNG
```

Ist der Entwurf erstellt, kann man ihn mit weiteren Einzelheiten ausfüllen.

```
10 REM *** MANN-AM-GALGEN-SPIEL ***
99 REM
100 REM HAUPTPROGRAMM
101 REM
115 DIM J$(1)
120 INPUT "SPIELANWEISUNGEN?<J/N>"
121 INPUT J$
122 IF J$="J" THEN GOSUB 200
130 GOSUB 300        :REM WORT HOLEN
132 STOP
135 GOSUB 400        :REM RATEN
140 GOSUB 500        :REM RICHTIG?
145 GOSUB 700        :REM SPIEL ZU ENDE?
190 GOTO 135         :REM NOCH EIN VERSUCH
199 REM
200 REM SPIELANWEISUNGEN
```

```
 ... schreibe die Spielanweisungen
290 RETURN
299 REM
300 REM DAS ZU RATENDE WORT HOLEN
 ... verwende  INPUT  für Eingabe eines  Wortes  durch
     Spieler 1
 ... zeichne Striche statt der zu ratenden Buchstaben
390 RETURN
399 REM
400 REM RATEVERSUCH
 ... Spieler 2 rät einen Buchstaben
490 RETURN
499 REM
500 REM RICHTIG?
 ... falsch? GOSUB 600 :REM Teil des Mannes zeichnen
 ... richtig? GOSUB 700:REM Ist Mann fertig gezeichnet?
590 RETURN
599 REM
600 REM ZEICHNEN
 ... nächsteN Teil des Mannes zeichnen
 ... feststellen, ob Zeichnung fertig
 ... wenn ja, GOSUB 800
690 RETURN
699 REM
700 IST DAS SPIEL ZU ENDE?
 ... Kontrolle, ob alle Buchstaben geraten wurden
 ... wenn ja, GOSUB 900
790 RETURN
799 REM
800 REM SPIELENDEMELDUNG
 ... Meldung, wenn Spieler verloren hat
890 RETURN
899 REM
900 REM SPIELENDEMELDUNG
 ... Meldung, wenn Spieler gewonnen hat
990 RETURN
```

Ich bin mir noch nicht im klaren darüber, wie ich das Spiel beenden soll. Deshalb stelle ich das zurück und schreibe und teste den ersten Teil des Programms. In Zeile 132 schreibe ich eine STOP-Anweisung, damit nur das erste Unterprogramm laufen kann. Ich beginne damit, das Unterprogramm ab 300 zu schreiben (DAS ZU RATENDE WORT HOLEN).

AUFGABE 24B

1. Schreibe ein kurzes Programm, das Unterprogramme verwendet. Es braucht nichts Nützliches zu machen, sondern kann einige Albernheiten drucken. Verwende drei Unterprogramme:

   Rufe eins von diesen vom Hauptprogramm zweimal auf.

   Rufe eins von diesen von einem der Unterprogramme einmal auf.

   Rufe eins von diesen durch eine IF-Anweisung auf.

2. Schreibe ein Programm, das Deine ersten drei Anfangsbuchstaben, jeden in einer anderen Farbe, auf dem Bildschirm zeigt. Bringe sie dann dazu, daß sie alle zusammen hinauf und hinunter springen.

3. Beende das "Mann-hängt-am-Galgen"-Programm. Weil das ein sehr großes Programm ist, solltest Du jetzt nur einen Teil bearbeiten, es dann abspeichern und später fertigstellen.

3

Fortgeschrittenes Programmieren

25. ANMERKUNGEN FÜR DEN LEHRER Die Tastatur, ON...GOTO

Das Byte 764 (dezimal) enthält eine Zahl von der Tastatur. In dieser Übung wird der Zweck untersucht.

Der ON...GOTO-Befehl wird erklärt.

Immer, wenn eine Taste gedrückt wird, übergibt die Tastatur eine Zahl in Byte 764. Die Zahl wird abgespeichert und weiteres Drükken anderer Tasten ändert diesen Wert nicht. Die Speicherschachtel kann für die Eingabe weiterer Tastenbetätigungen gelöscht werden, indem man POKE 255 hineinschreibt. Wird die Kombination SHIFT + Taste gedrückt, wird zum Code der Taste der Wert 64 dazuaddiert.

Ähnlich verhält es sich mit der CONTROL-Taste. In diesem Fall wird zum Code der gedrückten Taste der Wert 128 hinzugezählt. Der Vorteil, die Eingabe einer Taste auf diese Weise abzufragen, besteht darin, daß sich das Programm im Gegensatz zu INPUT oder GET nicht stehenbleibt, bis eine Taste gedrückt wird.

Ein Nachteil ist allerdings die Tatsache, daß der eingelesene Code in keiner Beziehung zum ASCII-Code steht.

Der ON...GOTO-Befehl ist ein einfaches Beispiel für die Anwendung der IM-FALLE-VON-Konstruktion. ON...GOSUB wird ebenfalls vom ATARI-BASIC unterstützt.

Übung 25: Die Tastatur, ON...GOTO

Fragen:

1. Welche Zahl steht auf der Vorderseite der Tastaturschachtel?

2. Warum muß die Tastaturschachtel geleert werden?

3. Wie erfährst Du, welche Zahl zu welcher Taste gehört?

4. Was geschieht bei ON V GOTO 200,300,400, wenn:

   V = 0 _____

   V = 1 _____

   V = 3 _____

   V = 8 _____

ÜBUNG 25: DIE TASTATUR, ON...GOTO

SPIELE UND DIE TASTATUR

Beim GET-Befehl wartet der Computer darauf, daß Du eine Taste drückst. Das Programm hält so lange an, bis Du eine Taste betätigt hast.

Es gibt noch einen anderen Weg, eine Eingabe über die Tastatur zu machen, ohne daß der Computer warten muß. Er wird bei Action-Spielen verwendet.

```
2 GOTO 1000:REM SCHLANGE
100 REM HAUPTSCHLEIFE
105 DR=PEEK(AR)
106 POKE AR,255
110 IF DR=R THEN D=D-1:IF D=0 THEN D=4
111 IF DR=LL THEN D=D+1:IF D=5 THEN D=1
113 FOR T=1 TO 50:NEXT T
115 ON D GOTO 120,122,124,126
120 Y=Y-1:GOTO 130
122 X=X-1:GOTO 130
124 Y=Y+1:GOTO 130
126 X=X+1
130 COLOR 3:PLOT X,Y
140 A=B:B=C:C=E:E=F:F=G:G=X
142 L=M:M=N:N=O:O=P:P=Q:Q=Y
145 COLOR 2:PLOT A,L
199 GOTO 100
999 END
1000 REM
1001 REM ----- SCHLANGE -----
1002 REM
1010 REM
1020 GOSUB 3000
2000 GRAPHICS 3+16
2010 SETCOLOR 0,8,8 :REM RAND
2020 SETCOLOR 4,0,4 :REM HINTERGRUND
2025 SETCOLOR 1,0,4 :REM LOESCHEN
2027 SETCOLOR 2,12,8:REM SCHLANGE
2030 COLOR 1
```

```
2040 PLOT 0,0
2042 DRAWTO  0,23
2044 DRAWTO 39,23
2046 DRAWTO 39, 0
2048 DRAWTO  0, 0
2100 LL=6          :REM LINKER PFEIL
2102 R=7           :REM RECHTER PFEIL
2105 AR=764        :REM TASTATURSCHACHTEL
2108 D=1           :REM STARTRICHTUNG
2110 X=20:Y=12     :REM STARTPOSITION
2115 A=X:B=X:C=X:E=X:F=X:G=X
2116 L=Y:M=Y:N=Y:O=Y:P=Y:Q=Y
2999 GOTO 100
3000 REM ANWEISUNGEN
3010 PRINT "clear"
3020 PRINT "LINKSHERUM: LINKE PFEILTASTE"
3030 PRINT "RECHTSHERUM: RECHTE PFEILTASTE"
3025 PRINT:PRINT "DU DARFST DEN RAND NICHT BERUEHREN"
2040 FOR T=1 TO 1000:NEXT T
3999 RETURN
```

DIE TASTATURSCHACHTEL

Wird eine Taste gedrückt, legt der Computer eine Zahl in eine spezielle Schachtel, die den Namen "764" auf der Vorderseite hat.

Damit man in die Schachtel sehen kann, verwendet man den PEEK-Befehl. Im SCHLANGEN-Programm steht er in Zeile 105:

```
105 DR=PEEK(AR)
```

wobei AR=764 (betrachte Zeile 2105).

```
105 DR=PEEK(764)
```

wäre einfacher, aber das Programm läuft schneller, wenn statt einer Zahlkonstanten eine Variable in PEEK verwendet wird.

In Zeile 105 wird in der Tastaturschachtel nachgesehen, ob seit dem letzten Mal, bei dem die Schachtel geleert wurde, eine neue Taste betätigt wurde.

Ist das der Fall, wird die Zahl aus der Tastaturschachtel geholt und in die Schachtel mit dem Namen DR gesteckt.

Diese Tastaturzahlen sind keine ATASCII-Zahlen. Es sind besondere Zahlen, die der ATARI-Computer verwendet.

In Zeile 110 wird überprüft, ob die Taste mit dem Rechtspfeil (Nummer 7: siehe Zeile 2102) gedrückt wurde.

Was macht Zeile 111? _____

In Zeile 106 wird der Computer aufgefordert, die Tastaturschachtel zu leeren. (Leeren heißt hier: die Zahl 255 eingeben.) Jedesmal, wenn Du eine Taste interpretieren willst, muß die Tastaturschachtel geleert werden, damit ein neuer Tastencode eingegeben werden kann.

```
106 POKE AR,255
```

DAS HINEINSEHEN IN DIE SCHACHTEL

Versuche:

```
10 V=PEEK(764)
20 PRINT V
30 GOTO 10
```

Drücke jede Taste. Sieh Dir die Zahlen an, die Du erhältst.

Halte die Tasten SHIFT oder CONTROL fest und drücke gleichzeitig eine Taste. Mit SHIFT wird die Zahl 64 dazuaddiert, mit CONTROL 128.

Übung 25: Die Tastatur, ON...GOTO

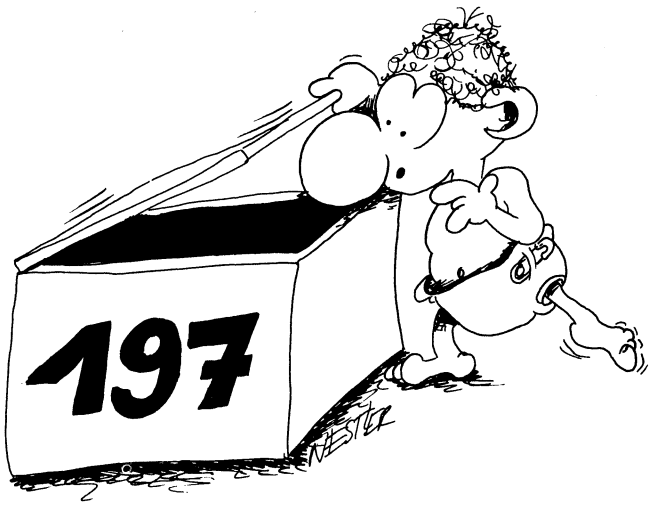
Der ON...GOTO-Befehl

```
115 ON D GOTO 120,122,124,126
```

bedeutet:

wenn	D	=	1	GOTO	120
			2		122
			3		124
			4		126
wenn	D	=	etwas anderes	GOTO	nächste Zeile

Nach einem GOTO-Befehl kannst Du eine, zwei oder eine beliebige Anzahl von Zeilennummern angeben, die sich irgendwo im Programm befinden.



AUFGABE 25

1. Mach Dir eine Tabelle mit den Zahlen, die in der Tastaturschachtel abgelegt werden, wenn Du eine Taste drückst. Du brauchst diese Tabelle, wenn Du Spiele programmierst.

2. Schreibe ein Programm, das ein Menü ausgibt. Der Buchstabe (A bis C), mit dem das Menü angewählt wird, soll mit GET eingegeben werden. Der Buchstabe wird in eine Zahl (1 bis 3) umgewandelt. Die Menüauswahl soll dann mit dem ON...GOTO-Befehl erfolgen.

26. ANMERKUNGEN FÜR DEN LEHRER Zeichenketten schneiden und verbinden

In diesem Kapitel behandeln wir:

die LEN-Funktion
Teile einer Zeichenkette
Zusammensetzen von Zeichenketten

Damit kann man Zeichenketten zerschneiden und sie in willkürlicher Reihenfolge wieder zusammensetzen.

Man sollte daran denken, daß die DIM-Anweisung die maximale Größe einer Zeichenkette angibt. Versucht man, einen längeren String abzulegen, wird er an der rechten Seite abgeschnitten.

Zeichenkettenteile können innerhalb der Klammern ein oder zwei Argumente haben.

Ist nur eine Zahl angegeben, heißt das, daß der Teil der Zeichenkette abgeschnitten wird, der vom numerierten Zeichen bis zum Ende geht.

Sind zwei Zahlen angegeben, heißt das, daß die Zeichenkette vom ersten bis zum letzten Zeichen der Zahl abgeschnitten wird.

Beispiele:

```
10 DIM G$(7)
12 G$="123456789"        nur 1234567 sind in der Schachtel
13 ? G$                  1234567 wird ausgegeben
15 ? G$(5)               567 wird ausgegeben
20 ? G$(2,4)             234 wird ausgegeben
25 ? G$(1)               1234567 (=G$) wird ausgegeben
```

Zusammensetzen: Es gibt viele mögliche Fälle. Die klarste Möglichkeit besteht darin, ein Loch in G$ zu machen, das die gleiche Größe wie der Teil von H$ hat, der eingebunden werden soll.

```
20 G$(3,5)=H$(8,10)
```

Die anderen Fälle werden in der Übung behandelt. Mit LEN wird ermittelt, wie lang eine Zeichenkette ist.

Fragen:

1. Wie speicherst Du "SONNE" von B$="SONNE UND MOND"?

2. Welche Funktionen und Argumente brauchst Du, um "UND" zu speichern?

3. Welche Funktion brauchst Du, wenn Du die Anzahl der Zeichen in einer Zeichenkette PQ$ zählen möchtest? Welches Argument?

4. Wie würdest Du in der Zeichenkette

   D$="ES DAUERT 2 MINUTEN"

   "2" in eine "5" verwandeln?

7. Schreibe ein kurzes Programm, das das Wort "COMPUTER" in "PUTERCOM" umwandelt?

ÜBUNG 26: ZEICHENKETTEN SCHNEIDEN UND VERBINDEN

WIE LANG IST EINE ZEICHENKETTE?

Starte:

```
10 REM --- LANGES SEIL ---
15 DIM N$(15)
20 PRINT "clear"
30 PRINT "GIB EINE ZEICHENKETTE EIN: ":INPUT N$
40 L=LEN(N$)
50 PRINT "DIE ZEICHENKETTE: '";N$;"' "
55 PRINT:PRINT "IST ";L;" ZEICHEN LANG"
```

Die erste Anwort: DONAU

Die zweite Antwort: RHEIN-MAIN-DONAU-KANAL

Mit dem DIM-Befehl haben wir 15 Zeichen zugelassen.

Wenn Du DONAU eingibst, hast Du 5 Zeichen gebraucht.

Wenn Du aber RHEIN-MAIN-DONAU-KANAL eingibst, füllt die Zeichenkette 22 Plätze aus. Deshalb wird der Teil U-KANAL weggeworfen.

ZEICHENKETTEN SCHNIPSELN

Es gibt zwei Wege, wie man einen Teil einer Zeichenkette abschneidet. Man nennt diesen Teil einen "Substring". String ist das gleiche Wort wie Zeichenkette, sub bedeutet: unter oder zwischen. Substring heißt also: eine Zeichenkettenteilstück.

1. Man schneidet den linken Teil der Zeichenkette weg und behält den rechten Teil.

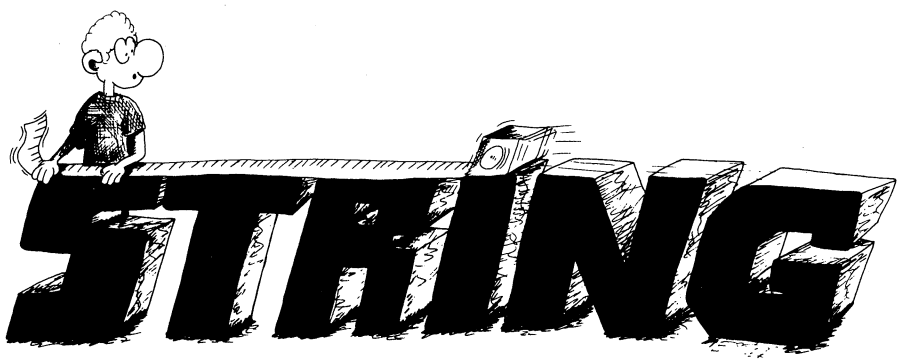
2. Man schneidet einen Teil aus der Zeichenkette heraus.

3. (Willst Du den linken Teil der Zeichenkette wegschneiden und den rechten Teil behalten, mußt Du die Regel 2 anwenden.)

Gib ein und starte:

```
10 REM > SCHERE >
15 DIM N$(40)
20 PRINT "clear"
30 N$="123456789"
40 PRINT N$(3)
50 PRINT N$(3,5)
```

Zeile 40 zählt drei Buchstaben von links ab, schneidet die Zeichenkette durch und behält das Stück einschließlich des dritten Buchstabens bis zum rechten Ende.



Übung 26: Zeichenketten schneiden und verbinden

Regel:

Steht Zahl innerhalb der Klammern, heißt das:

1. Die entsprechende Zahl von Buchstaben muß von rechts ab gezählt werden.

2. Der Teil einschließlich des erreichten Buchstabens wird bis zum Ende der Zeichenkette als neuer Teilstring abgespalten.

Merke Die, daß N$(1) dasselbe ist wie N$!

Zeile 50 schneidet ein Stück aus der Mitte von N$ aus. Die neue Zeichenkette beginnt mit dem 3. Buchstaben und endet mit dem 5.

Regel:

Sind zwei Zahlen innerhalb der Klammern, heißt das:

1. Die erste Zahl entspricht der ersten Stelle der Zeichenkette.

2. Die zweite Zahl entspricht der zweiten Stelle der Zeichenkette.

3. Die neue Zeichenkette besteht aus den Buchstaben, die in den angegebenen Plätzen eingeschlossen sind.

Du kannst auch nur einen Buchstaben herausschneiden.

Starte:
```
10 REM --- HERAUSSCHNEIDEN ---
12 PRINT "clear"
15 DIM W$(40)
20 PRINT "GIB MIR EIN WORT EIN":INPUT W$
25 L=LEN(W$)
30 PRINT "WELCHEN BUCHSTABEN WILLST DU?"
31 PRINT "(NENN MIR EINE ZAHL)"
35 INPUT N
40 IF L<N THEN PRINT "ZU GROSS":GOTO 30
45 PRINT W$(N,N)
```

Schreibe noch zusätzlich die nächsten Zeilen dazu und starte:

```
30 PRINT "WIE VIELE BUCHSTABEN WILLST DU?"
32 INPUT N
33 PRINT "WO SOLL ICH BEGINNEN?"
34 INPUT ST
35 ND=ST+N-1
40 IF ND>L THEN PRINT "ZU VIELE":GOTO 30
45 PRINT W$(ST,ND)
```

ZEICHENKETTEN ZUSAMMENKLEBEN

Jetzt wird gezeigt, wie man die Zeichenkettenteile wieder zusammenbringen kann.

```
10 REM >>> SCHERE UND LEIM >>>
15 DIM G$(10), H$(5)
20 G$="123456789"
30 H$="ABCDE"
40 G$(3,5)=H$(2,4)
50 PRINT G$
```

Zeile 40 In G$ wird vom dritten bis zum fünften Zeichen ein Platz vorbereitet, der leer ist.

Dann wird ein Stück aus H$ herausgetrennt.

Zuletzt wird das Loch in G$ mit dem Teilstück aus H$ ausgefüllt.

Experimentiere ein bißchen, indem Du Zeile 40 änderst.

`40 G$(3,4)=H$(2,4)`	Das Loch von G$ ist zu klein, und ein Teil von H$ wird weggeworfen.
`40 G$(3,6)=H$(2,4)`	Das Loch in G$ ist zu groß, und H$ füllt nur einen Teil dieses Lochs aus.

Übung 26: Zeichenketten schneiden und verbinden

`40 G$ =H$(2,4)`	Die alte Zeichenkette G$ gibt es nicht mehr. Die neue ist jetzt ein Teil von H$.
`40 G$(1) =H$(2,4)`	Wie oben.
`40 G$(3) =H$(2,4)`	Die neue Zeichenkette G$ besteht zur Hälfte (vorne) aus G$, zur anderen Hälfte (hinten) aus H$.

SCHAU MAL, KEINE ZWISCHENRÄUME

Gib ein:

```
10 REM <OHNE ZWISCHENRAEUME>
11 REM
15 DIM S$(40),L$(1),T$(40)
20 PRINT "clear":PRINT
30 PRINT "GIB EINEN LANGEN SATZ EIN":PRINT
35 INPUT S$
40 L=LEN(S$)
45 T$="":N=1
50 FOR I=1 TO L :REM SCHAUE JEDEN BUCHSTABEN AN
60 L$=S$(I,I)
70 IF L$<>" " THEN T$(N)=L$:N=N+1
80 NEXT I
90 PRINT:PRINT T$
95 PRINT:PRINT:PRINT
```

Zeile 60 schneidet jeweils nur einen Buchstaben aus der Mitte der Zeichenkette heraus.

Zeile 70 verbindet die Zeichen in T$, wenn kein Leerraum vorhanden ist.

AUFGABE 26

1. Schreibe ein Programm, das Geheimschrift erzeugt. Du gibst einen Satz ein und es gibt Dir an, wie lang er ist. Dann tauscht es den ersten Buchstaben mit dem zweiten, den dritten mit dem vierten und so weiter aus. Beispiel:

   DIES IST EIN TEST. wird:

   IDSEI TSE NIT SE.T

2. Schreibe ein Programm, das Fragen beantwortet. Du gibst eine Frage ein, die mit einem Verbum anfängt, und das Programm vertauscht das Verb mit dem Objekt. Damit wird auf die Frage geantwortet. Beispiel:

   BIST DU EIN HUHN?

   DU BIST EIN HUHN.

3. Schreibe ein "Geheimsprache"-Programm. Es soll nach einem Wort fragen. Dann nimmt es alle Buchstaben bis zum ersten Vokal und setzt sie an das Ende des Wortes, gefolgt von einem AI. Wenn das Wort mit einem Vokal anfängt, wird nur LAI hinzugefügt. Beispiele:

   HUHN wird UHNHAI
   ADAM wird ADAMLAI

27. **ANMERKUNGEN FÜR DEN LEHRER** Zahlen durch Zeichenketten ersetzen

Dieses Kapitel behandelt die Funktionen STR$ und VAL. Gleichzeitig werden die Funktionen wiederholt.

STR$ nimmt eine Zahl und wandelt sie in eine Zeichenkette um.

VAL macht genau das Gegenteil. Sie nimmt eine Zeichenkette und wandelt sie in einen Zahlenwert um.

Erhält VAL eine Zeichenkette, die es nicht in eine Zahl umwandeln kann, wird ERROR- 18 ausgegeben.

Diese Möglichkeit der gegenseitigen Umwandlung beider Hauptvariablentypen gibt Programmen große Flexibilität.

Funktionen und ihre Argumente werden in dieser Übung zusammengefaßt.

Fragen:

1. Wie kannst Du die Geschwindigkeit der zweiten Übungsaufgabe verlangsamen, wenn sie zu schnell läuft?

2. Dein Programm enthält die Zeichenkette "JOHANN WOLFGANG GOETHE WURDE 1749 GEBOREN". Du sollst einige Zeilen schreiben, damit die Frage "Wann wurde Goethe geboren?" beantwortet werden kann. (Du mußt das Geburtsdatum aus der Zeichenkette herausnehmen und es in eine Zahl verwandeln.)

3. Was ist ein Wert? Was versteht man unter "eine Funktion gibt einen Wert zurück"? Was kannst Du mit dem Wert machen?

4. Was ist das Argument einer Funktion?

5. Wo gehören die Befehle in einer Zeile hin? Kannst Du eine Funktion an den Anfang einer Zeile setzen?

ÜBUNG 27: ZAHLEN DURCH ZEICHENKETTEN ERSETZEN

In diesem Kapitel werden zwei Funktionen erklärt: VAL() und STR$().

DIE UMWANDLUNG VON ZEICHENKETTEN IN ZAHLEN

Es gibt zwei Arten von Variablen: Zeichenketten (oder Strings) und Zahlen. Wir können eine Art in die andere umwandeln.

Starte:

```
10 REM ZEICHENKETTE -> ZAHLEN
15 DIM L$(3),M$(3)
20 PRINT "clear"
30 L$="123"
40 M$="789"
50 L=VAL(L$)
60 M=VAL(M$)
70 PRINT L
72 PRINT M
74 PRINT "---"
76 PRINT L+M
```

VAL steht für "value". Das deutsche Wort dafür lautet: Wert. Die Funktion wandelt Zeichenketten in Zahlen um, wo immer es möglich ist.

DIE UMWANDLUNG VON ZAHLEN IN ZEICHENKETTEN

```
Starte:   10 REM ZAHLEN -> ZEICHENKETTEN
          11 REM
          15 DIM N$(10)
          20 PRINT "clear"
          25 INPUT "GIB MIR EINE ZAHL ":INPUT ZL
          30 Z$=STR$(ZL)
          35 PRINT
          60 PRINT "HIER IST DIE ZWEITE STELLE"
          65 PRINT: PRINT Z$(2,2)
```

STR$ steht für String (Zeichenkette). Die Funktion wandelt eine Zahl in eine Zeichenkette um.

NOCH EINMAL FUNKTIONEN

In diesem Buch verwenden wir folgende Funktionen:

RND(), INT(), LEN(), VAL(), STR$(), ASC(), CHR$()

Regeln von Funktionen:

1. Funktionen haben immer Klammern "()", die ein oder mehrere Argumente enthalten. Zum Beispiel:

```
ASC(D$)          ASC ist die Funktion
                 D   ist das Argument
```

   Die Argumente können Zahlen oder Zeichenketten sein.

   Das Argument kann eine Konstante, eine Variable, ein Ausdruck oder eine andere Funktion sein.

LEN("ZIEGE")	Das Argument ist eine Zeichenkettenkonstante.
LEN(M$)	Das Argument ist ein Zeichenkettenvariable.
STR$(1984)	Das Argument ist ein numerische Konstante.
STR$(DATUM)	Das Argument ist eine numerische Variable.
STR$(3+2*X)	Das Argument ist ein numerischer Ausdruck.
LEN(STR$(DATUM)	Das Argument von LEN ist eine Funktion.

2. Eine Funktion ist kein Befehl. Eine Anweisung kann nicht mit einer Funktion beginnen.

   richtig: `10 D=LEN$(CS$)`

   falsch: `10 LEN(CS$)=5`

3. Eine Funktion verhält sich genau wie eine Zahl oder eine Zeichenkette. Wir sagen: die Funktion gibt den Wert zurück. Dieser kann wie jede andere Zahl oder jede andere Zeichenkette in eine Schachtel gesteckt oder gedruckt werden. Die Funktion kann sogar ein Argument in einer anderen Funktion darstellen.

```
10 PRINT INT(3,14)
15 W$ = STR$(2+Q)
```

Die Argumente geben an, welche Werte übergeben werden.

(Vergiß nicht, daß Zeichenkettenwerte in die Zeichenkettenvariablenschachteln, die Zahlenwerte in die Zahlenschachteln gehören.)

AUFGABE 27

1. Gib für jede der folgenden Funktionen die Namen der Funktion und der Argumente an und sage, ob der Wert des Argumentes entweder eine Zahl oder eine Zeichenkette darstellt. Gib an, ob das Argument eine Konstante, Variable oder Funktion ist. Gib auch an, ob der Wert der Funktion eine Zahl oder eine Zeichenkette ist.

   RND(9) _____

   INT(Q) _____

   VAL("ER$") _____

   STR$(INT(RND(9))) _____

   LEN("FUSS") _____

   INT(22.4/V) _____

2. Jede dieser Zeilen enthält Fehler. Erkläre, was falsch ist.

   10 INT(Q)=65 _____

   10 D$=CHR(1) _____

   10 PW$=VAL$(F) _____

   10 PRINT CHR$ _____

Übung 27: Zahlen durch Zeichenketten ersetzen

3. Schreibe ein Programm, das eine Zahl verlangt. Erstelle dann eine zweite Zahl, die der Kehrwert der ersten sein soll. Addiere dann beide Zahlen miteinander. Laß die Zahlen genau wie bei einem Additionsvorgang (mit einem "+" Zeichen und einem Strich unter den Zahlen) erscheinen.

4. Schreibe ein Programm, das eine Zahl verlangt. Diese Zahl soll so über den Bildschirm laufen, daß die jeweils linke Ziffer nach rechts geschrieben wird. Laß diesen Vorgang mindestens so lange ablaufen, bis die Zahl in ihrer eingegebenen Form wieder erscheint.

28. **ANMERKUNGEN FÜR DEN LEHRER** Der Joystick und Action-Spiele

In dieser Übung behandeln wir die Befehle STICK und STRIG.

Normalerweise werden Joysticks in beweglichen Grafikspielen eingesetzt. In diesem Kapitel werden wir mit dem Joystick einen Punkt über den Bildschirm bewegen.

In der nächsten Übung wird mit dem Punkt ein Geschoß abgefeuert und auf Sterne gezielt.

Bevor der Schüler damit arbeiten kann, sollte er die Adressierung der Quadrate im 40x24-Raster des GRAPHICS-3-Bildschirms verstehen.

Wenn man bewegliche Objekte darstellen will, muß jedes alte Abbild gelöscht werden, bevor ein neues gezeichnet werden kann. Dabei sollte zuerst gelöscht werden und dann neu geschrieben, damit das Bild am Schirm möglichst wenig flackert.

Grafikspiele können lang und aufwendig werden. BASIC ist meist zu langsam für solche Spiele. Maximale Geschwindigkeit kann dadurch erreicht werden, daß zuerst der Arbeitsteil und dann erst der Definitionsteil programmiert wird. Dieser Definitionsteil wird am Programmanfang aufgerufen. Diese Struktur wird im Kapitel über anwenderfreundliches Programmieren behandelt.

Damit das Spiel bei maximaler Geschwindigkeit ablaufen kann, sollte während des Programmlaufs die Umwandlung numerischer Konstanten in Fließkommazahlen vermieden werden.

Statt

```
60 POSITION 33,20
```

schreibt man besser

```
60 POSITION A,B
```

wobei die Variablen A=33 und B=20 sind.

Wird die Zeile 60 nur einmal verwendet, ist es natürlich besser, die Zahlen direkt einzugeben, weil der Zuweisungvorgang von A und B zu lange dauert. Die Zeitersparnis erhält man nur, wenn POSITION A,B oft eingesetzt wird.

Fragen:

1. Welche Zahlen gibt die STICK-Funktion zurück?

2. Was bedeutet die "0" in STICK(0)?

3. Wie gibst Du an, daß der Knopf am Joystick festgehalten wird?

ÜBUNG 28: DER JOYSTICK UND ACTION-SPIELE

Stecke den Joystick in den vorderen Sockel. Er ist mit 1 beschriftet.

DER JOYSTICK UND DER JOYSTICK-KNOPF

Laß das Programm laufen:

```
10 REM === JOYSTICK ===
15 DIM C$(4)
16 POKE 752,1:REM CURSOR AUSSCHALTEN
20 PRINT "clear"
30 PRINT "BEWEGE DEN JOYSTICK"
35 PRINT:PRINT "UND DRUECKE AUF DEN KNOPF"
38 REM DEN JOYSTICK PRUEFEN
40 S0=STICK(0)
42 POSITION 10,5:PRINT "POSITION:  "
45 POSITION 10,5:PRINT "POSITION: ";S0
55 REM DEN KNOPF PRUEFEN
60 B0=STRIG(0)
70 C$="    ":IF B0=0 THEN C$="BUMM"
75 POSITION 10,7:PRINT C$;
80 FOR T=1 TO 20:NEXT T
99 GOTO 40
```

Beende das Programm mit der BREAK-Taste.

In Zeile 42 wird die alte Nummer gelöscht, bevor die neue eingegeben wird.

Übung 28: Der Joystick und Action-Spiele

DER JOYSTICK

An der Seite des ATARI 800 XL findest Du zwei Sockel, die mit 1 und 2 bezeichnet sind.

Die STICK()-Funktion gibt die Richtung an, in die der Joystick bewegt wurde.

DER JOYSTICK-KNOPF

Mit der STRIG(0)-Funktion erfährst Du, ob der Knopf an Joystick 0, der aber im Sockel mit der Nummer 1 sitzt, gedrückt worden ist. STRIG bedeutet "Stick trigger" und das heißt etwa: den Joystick drücken. (Gemeint ist natürlich der Knopf!)

Ist die Zahl 0, wurde der Knopf gedrückt. Sonst ist die Zahl 1.

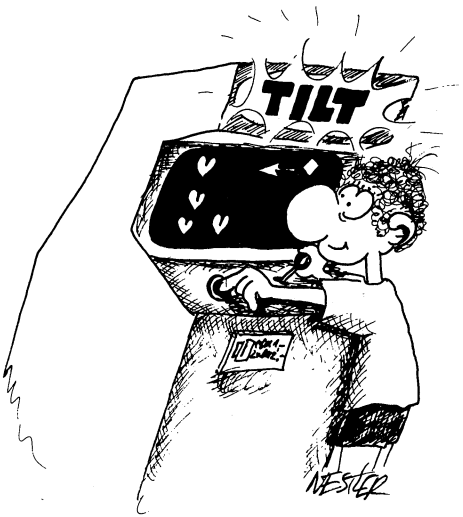
EINEN PUNKT ÜBER DEN BILDSCHIRM BEWEGEN

Starte:

```
10 REM EINEN PUNKT BEWEGEN
12 PRINT "clear":GRAPHICS 3+16
25 SETCOLOR 0,12,0
27 SETCOLOR 4,0,0
50 X=20:Y=12
60 GOSUB 900
64 COLOR 0:PLOT X,Y:REM ALTEN FLECK LOESCHEN
67 X=X+DX:Y=Y+DY
70 IF X<0 THEN X=0: REM AN DER LINKEN ECKE?
71 IF Y<0 THEN Y=0
72 IF X>39 THEN X=39
73 IF Y>23 THEN Y=23
80 COLOR 1:PLOT X,Y:REM FLECK AUF DEN SCHIRM BRINGEN
99 GOTO 60
900 REM JOYSTICK?
910 S=STICK(0)
915 IF S=15 THEN DX=0:DY=0:RETURN
920 IF S< 8 THEN DX=1:GOTO 950
```

```
925 IF S>12 THEN DX=0:GOTO 965
930 DX=-1
935 IF S=10 THEN DY=-1:RETURN
940 IF S=11 THEN DY= 0:RETURN
945 IF S= 9 THEN DY= 1:RETURN
950 IF S= 5 THEN DY= 1:RETURN
955 IF S= 7 THEN DY= 0:RETURN
960 IF S= 6 THEN DY=-1:RETURN
965 IF S=14 THEN DY=-1:RETURN
970 IF S=13 THEN DY= 1:RETURN
```

Halte das Programm mit der BREAK-Taste an. Speichere es auf Kassette.



LÖSCHEN UND WIEDERSCHREIBEN

"Löschen und schreiben, löschen und schreiben ... ." Jedesmal, wenn Du einen Punkt schreibst, mußt Du ihn löschen, bevor Du ihn an eine andere Stelle setzt. Sonst erhältst Du immer mehr Punkte. (Probier es einmal aus. Du mußt nur Zeile 64 löschen.)

In Zeile 64 werden die Punkte gelöscht.

In Zeile 80 wird der Punkt auf die Farbe 2 gesetzt und dann auf den Bildschirm ausgegeben.

AUFGABE 28

1. Fertige eine Zeichnung an, die alle Zahlen enthält, die der STICK-Befehl zurückgibt, wenn Du ihn in alle acht Richtungen bewegst. Die Tabelle wird Dir helfen, wenn Du mit dem Joystick Spiele machst.

2. Im Programm "EINEN PUNKT BEWEGEN" soll noch ein Rahmen hinzugefügt werden. Welche Änderungen mußt Du in den Zeilen 70 bis 73 vornehmen, damit der Rahmen nicht gelöscht wird?

3. Die Farbe des Punktes soll sich ändern, wenn er an den Rahmen stößt.

29. ANMERKUNGEN FÜR DEN LEHRER Sterne abschießen

Es wird gezeigt, wie man mit dem STRIG-Befehl ein Geschoß abfeuert, und wie man mit dem LOCATE-Befehl ein Ziel aufspürt.

Die Grafik-Modi unterscheiden sich voneinander in der Größe der Bildpunkte (das kleine Quadrat, das mit PLOT auf dem Bildschirm gezeichnet werden kann) und in der Anzahl der Farben.

An die meisten ATARI-Computer ist ein Fernsehgerät angeschlossen. Der ATARI 800 hat einen Monitorausgang und kann mit einem Farbmonitor verbunden werden. Damit erhält man eine bessere Darstellung der grafischen Bilder. (Die Auflösung ist größer.) Außerdem benötigt man den Anschluß eines Verstärkers und eines Lautsprechers, damit man Töne hören kann.

Auf jeden Fall sieht für die meisten Punkt-Hintergrund-Farbkombinationen ein einzelner Punkt oder eine senkrechte Linie nicht sehr gut aus. Waagrechte Linien und ausgefüllte Flächen sind besser darstellbar. Der Grund dafür ist in der nicht ausreichenden Bandbreite des Fernsehgeräts zu suchen.

Deshalb muß man, wie bei anderen Kunstformen auch, die Grenzen bei der Grafikdarstellung des ATARI abstecken. Mit etwas Übung kann man Farbkombinationen und Helligkeitsgrade herausfinden, die gut zusammenpassen.

Die Farbe "Schwarz" ist in Wirklichkeit Grau (Farbnummer 0) mit einer Helligkeit von 0.

"Weiß" ist Grau mit einer Helligkeit von 14.

Übung 29: Sterne abschießen

Fragen:

1. Welcher Punkt am Bildschirm wird in dem Befehl LOCATE 3,9,Z betrachtet? Was wird in die Variablenschachtel Z gegeben?

2. Wie erkennt das Programm, daß ein Stern getroffen wurde?

ÜBUNG 29: STERNE ABSCHIESSEN

MIT EINEM LASER SCHIESSEN

Lade das Programm "EINEN PUNKT BEWEGEN" und füge die nächsten Zeilen hinzu:

```
10 REM SCHIESSEN
28 SETCOLOR 2,4,8
62 IF STRIG(0)=0 THEN GOSUB 200
200 REM SCHUSS
205 IF Y=0 THEN RETURN
210 FOR I=Y-1 TO 0 STEP -1
215 COLOR 3:PLOT X,I
220 FOR T=1 TO 5:NEXT T
225 COLOR 0:PLOT X,I
250 NEXT I
299 RETURN
```

Laß das Programm laufen. Halte es mit der BREAK-Taste an. Speichere es auf Kassette.

In Zeile 62 wird abgefragt, ob der Knopf des Joysticks gedrückt worden ist. Ist das der Fall, wird im Unterprogramm 200 der Laser abgefeuert.

STERNE ABSCHIESSEN, DIE LOCATE-FUNKTION

Lade das Programm "SCHIESSEN" und schreibe zusätzlich noch die nächsten Zeilen dazu:

Übung 29: Sterne abschießen

```
10 REM *** STERNE ABSCHIESSEN ***
40 GOSUB 300
212 LOCATE X,I,L
213 IF L=3 THEN GOSUB 400
300 REM DIE STERNE SETZEN
310 COLOR 3
311 FOR I=1 TO 10
320 X=RND(9)*39
330 Y=RND(9)*23
350 PLOT X,Y
360 NEXT I
390 RETURN
400 REM STERN TREFFEN
410 SOUND 0,200,8,8
420 FOR T=1 TO 30:NEXT T
430 SOUND 0,0,0,0
490 RETURN
```

Laß das Programm laufen. Halte diesmal das Programm mit der RESET-Taste an. Speichere es auf Kassette.

DIE STERNE ZEICHNEN

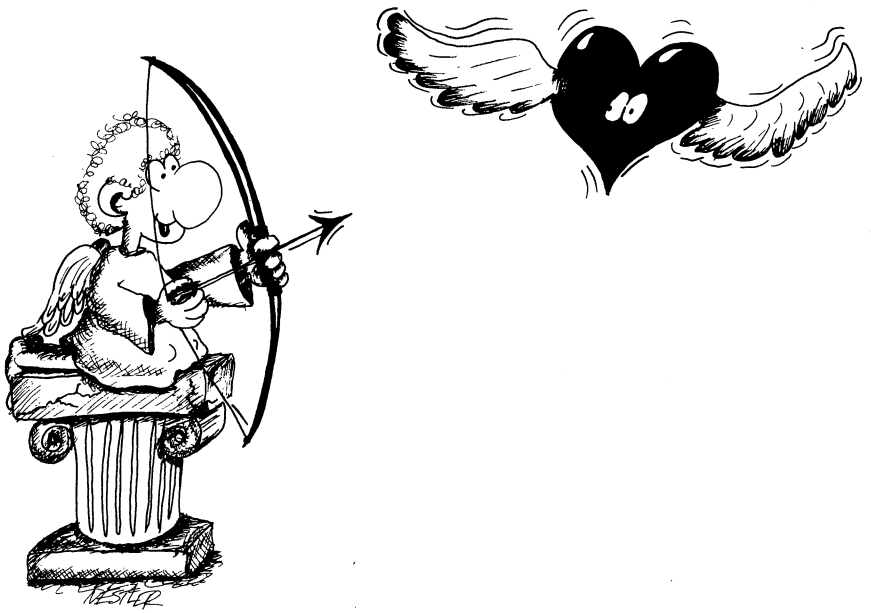
Das Unterprogramm ab Zeile 300 wird nur einmal aufgerufen. Es zeichnet 10 Sterne an zufälligen Stellen auf den Bildschirm.

HAT DER LASER EINEN STERN GETROFFEN?

Während das Programm den Laserstrahl zeichnet, untersucht es, ob das nächste Quadrat ein Stern ist (Farbe 3). Ist das der Fall, springt es in das Unterprogramm nach Zeile 400 und läßt ein Treffer-Geräusch erklingen.

In Zeile 212 wird das Quadrat mit Hilfe der LOCATE X,Y,L-Funktion gesucht. LOCATE betrachtet den Punkt X,Y am Bildschirm und gibt die Farbzahl, die es dort findet, in die Variable L.

In Zeile 213 wird bestimmt, daß in das Unterprogramm gesprungen wird, wenn Farbe 3 gefunden wurde.



WEITERE GRAFIK-MODI

Vier-Farben-Grafik

Diese Modi besitzen drei Farben und die Hintergrundfarbe.

Modus 3 ist ziemlich "grob" (die Punkte sind sehr groß).

Modus 5 hat eine höhere Auflösung und Modus 7 bringt ganz kleine Punkte am Bildschirm.

Übung 29: Sterne abschießen

MODUS	SENKRECHT	WAAGRECHT
GRAPHICS 3+16	0 BIS 39	0 BIS 23
GRAPHICS 5+16	0 BIS 79	0 BIS 47
GRAPHICS 7+16	0 BIS 159	0 BIS 95

Farbtöpfe und Pinsel, die für diesen Modus verwendet werden:

TOPF	PINSEL
SETCOLOR 0,F,H	COLOR 1
SETCOLOR 1,F,H	COLOR 2
SETCOLOR 2,F,H	COLOR 3
SETCOLOR 4,F,H	COLOR 0, Hintergrund und Rand

Der Buchstabe "F" steht für die gewünscht Farbe (0 bis 15).

Der Buchstabe "H" steht für die gewünschte Helligkeit.

Mit SETCOLOR 4 wird wird die Farbe und die Helligkeit für den Hintergrund bestimmt. In diesen Modi kannst Du keinen Rand mehr sehen. Er hat nämlich die gleiche Farbe wie der Hintergrund.

Den Pinsel COLOR 0 kannst Du in den Hintergrundfarbtopf tauchen und damit einen Punkt vom Bildschirm löschen.

Starte:

```
10 REM GRAFIK MIT VIER FARBEN
12 GRAPHICS 5+16
20 SETCOLOR 0,2,4:REM TOPF 0
21 SETCOLOR 1,9,8:REM TOPF 1
22 SETCOLOR 2,6,6:REM TOPF 2
24 SETCOLOR 4,6,4:REM HINTERGRUND
31 COLOR 1:PLOT 0,5:DRAWTO 20,5:REM PINSEL 1
32 COLOR 2:PLOT 0,7:DRAWTO 20,7:REM PINSEL 2
33 COLOR 3:PLOT 0,9:DRAWTO 20,9:REM PINSEL 3
40 FOR T=1 TO 1000:NEXT T
50 COLOR 0:PLOT 0,5:DRAWTO 10,5:REM LOESCHEN
99 GOTO 99:REM DAS BILD STEHEN LASSEN
```

Zwei-Farben-Grafik

Diese Modi haben für den Hintergrund und Punkte jeweils eine Farbe und Helligkeit.

Der Vorteil gegenüber den Modi 3, 5 und 7 besthet darin, daß sie weniger Platz im Speicher einnehmen.

MODUS	SENKRECHT	WAAGRECHT
GRAPHICS 4+16	0 BIS 79	0 BIS 47
GRAPHICS 6+16	0 BIS 159	0 BIS 95

Farbtöpfe und Pinsel, die für diesen Modus verwendet werden:

TOPF	PINSEL
SETCOLOR 0,F,H	COLOR 1
SETCOLOR 4,F,H	COLOR 0, Hintergrund und Rand

```
10 REM ZWEI-FARBEN-GRAFIK
12 GRAPHICS 6+16
15 SETCOLOR 4,12,8:REM GRUENER HINTERGRUND UND RAND
17 SETCOLOR 0,8,10:REM BLAUE PUNKTE
30 COLOR 1:PLOT 5,5:DRAWTO 20,5:REM LINIE
40 FOR T=1 TO 500:NEXT T
50 COLOR 0:PLOT 9,5:DRAWTO 17,5:REM LOESCHEN
99 GOTO 99:REM BILD STEHEN LASSEN
```

Grafik mit hoher Auflösung

Hier gibt es für den Hintergrund Farbe und Helligkeit, ebenso für den Rand und Punkte, die zwar die Hintergrundfarbe besitzen, aber eine andere Helligkeit.

GRAPHICS 8+16	0 BIS 319	0 BIS 191

Für den Hintergrund kann eine Farbe und die Helligkeit gewählt werden.

Die Punkte haben dann dieselbe Farbe wie der Hintergrund, aber die Helligkeit kann gewählt werden.

TOPF	PINSEL
SETCOLOR 1,0,H	COLOR 1
SETCOLOR 2,F,H	COLOR 0, Hintergrund
SETCOLOR 4,F,H	Randfarbe

Laß das Programm laufen:

```
10 REM GRAFIK MIT HOHER AUFLOESUNG
12 GRAPHICS 8+16
15 SETCOLOR 4,12,8              :REM GRUENER RAND
16 SETCOLOR 2,8,4               :REM BLAUER HINTERGRUND
17 SETCOLOR 1,0,10              :REM BLAUE PUNKTE
30 COLOR 1:PLOT 5,5:DRAWTO 20,5:REM LINIE
40 FOR T=1 TO 500:NEXT T
50 COLOR 0:PLOT 9,5:DRAWTO 17,5:REM LOESCHEN
99 GOTO 99
```

In Zeile 17 wird die Helligkeit der Punkt bestimmt. Die Farbe "0" oder "Grau" in SETCOLOR 1,0,10 wird nicht benutzt.

AUFGABE 29

1. Ergänze das "EINEN-STERN-ABSCHIESSEN"-Unterprogramm so, daß ein Riesenexplosion ausgelöst wird, wenn der Laserstrahl einen Stern trifft.

2. Ändere das Unterprogramm ab Zeile 300, so daß der Laserstrahl nach einem Treffer stoppt.

3. Laß die Sterne in zwei Farben, Rot und Blau, erscheinen (im Unterprogramm ab Zeile 300). Der Laserstrahl soll dann nur die roten Sterne (Unterprogramm ab 200) abschießen können.

4. Setze das Programm in den Grafik-Modus 5.

30. ANMERKUNGEN FÜR DEN LEHRER Matrizen

In diesem Kapitel werden die indizierten Variablen ("Arrays") vorgestellt. Die DIM()- und CLR-Anweisungen werden beschrieben.

Zuerst werden Matrizen mit nur einem Index beschrieben. Die Matrix wird mit einer Familie verglichen, wobei die einzelnen Elemente der indizierten Variablen die Familienmitglieder darstellen. Der Indexwert entspricht dabei dem "Vornamen" des Angehörigen.

Zweidimensionale Matrizen werden mit den Zahlen auf einer Kalenderseite für einen Monat verglichen.

Im ATARI-BASIC werden mehrdimensionale Matrizen (>2) nicht unterstützt.

Der Index einer Matrix beginnt mit "0", das heißt, daß eine Matrix immer ein Element mehr hat, als in der DIM-Anweisung als Zahl steht. (Die Matrix, die durch DIM(7) definiert ist, hat 8 Elemente.)

Jetzt wird auch deutlich, daß das ATARI-BASIC Zeichenkettenvariablen wie eine besondere Art der Matrizen behandelt. Trotzdem ist es besser, Matrizen und Zeichenketten voneinander getrennt zu behandeln.

Fragen:

1. Was bedeutet die Anweisung DIM AD(5)?
2. Wo setzt Du den DIM-Befehl in ein Programm ein?
3. Was verursacht der Befehl CLR?
4. Was bedeutet der "Index" einer Matrix?
5. Wie viele Schachteln legt der Befehl DIM SR(5,9) an?

ÜBUNG 30: MATRIZEN

MATRIZEN KENNENLERNEN

```
22 F(0)=3
24 F(1)=44
26 F(2)=12
```

Jedes Familienmitglied ist eine numerische Variable.

Die Familie hat einen "Nachnamen" wie F oder B.

Jedes Mitglied hat eine Zahl in Klammern (), die den Vornamen darstellt. Die Matrix beginnt immer mit dem Vornamen "0".

Anstatt Familie sagt man Matrix.

Anstatt Vorname sagt man Indexzahl.

VORSICHT!
Nur numerische Variablen können in solchen Familien zusammengefaßt werden.

Du weißt ja schon, da Zeichenkettenvariablen immer mit einem Index versehen werden müssen

A$(31) bedeutet: der Rest der Zeichenkette A$ ab dem 31. Buchstaben.

DER DIM()-BEFEHL RESERVIERT SCHACHTELN

Nehmen wir an, eine "Matrizen-Familie" geht ins Kino. Sie reserviert vorher die Plätze. Sie verwendet dazu einen DIM-Befehl.

Der DIM-Befehl sagt dem Computer, daß er eine Schachtelreihe für die Matrizen reservieren soll. DIM ist eine Abkürzung für "Dimension" und bedeutet "Größe".

```
18 DIM A(3)
30 DIM B(7),B$(4)
```

Zeile 18 sichert vier Speicherschachteln, jeweils eine für die Variablen A(0), A(1), A(2) und A(3). Diese Schachteln sind für Zahlen bestimmt und fangen mit der Zahl "0" an.

Zeile 30 reserviert acht Schachteln für die B()-Matrizen und eine Schachtel für die Zeichenkette B$(). B$ ist so groß, daß fünf Buchstaben hineinpassen und ist am Anfang leer.

Regel:

Setze die DIM()-Anweisung vorher im Programm ein, bevor die Matrix in irgendeiner anderen Anweisung verwendet wird.

Regel:

Im Programm darf der DIM-Befehl für eine Matrix oder eine Zeichenkette nur einmal gegeben werden.

EIN OFT GEMACHTER FEHLER

Irgendwann macht jeder einmal diesen Fehler: In einer Programmschleife wird eine DIM-Anweisung mehrmals durchlaufen. Das ist nicht erlaubt. Laß das Programm laufen:

```
10 DIM Q(15)
20 Q(3)=3/7
30 GOTO 10
```

Du erhältst:

```
ERROR- 9 AT LINE 10          (Fehler- 9 in Zeile 10)
```

Ändere deshalb Zeile 30:

```
30 GOTO 20
```

DER CLR-BEFEHL

Hast Du mit Zeichenketten und Matrizen gearbeitet und willst sie neu dimensionieren, mußt Du vorher den CLR-Befehl verwenden.

CLR bedeutet "clear" und das heißt: löschen. (Du erinnerst Dich sicher noch an unseren BASIC-"Befehl" "clear", der den Bildschirm löscht.) In diesem Fall wird der Speicher von allen Zeichenketten und numerischen Matrizen gelöscht. Die numerischen Variablen bleiben allerdings erhalten.

Schreibe die nächsten Zeilen zum obigen Programm dazu:

```
22 PRINT Q(3)
25 CLR
```

und laß es laufen. Du bekommst:

```
ERROR- 9 AT LINE 20
```

weil der Befehl CLR die Schachtel Q(), die mit der DIM-Anweisung erstellt wurde, löscht.

DAS ERSTELLEN EINER LISTE

```
Starte:   10 REM ++ DER REIHE NACH ++
          20 PRINT "clear"
          30 DIM A(5)
          35 PRINT "GIB EINE ZAHL EIN"
          40 FOR N=0 TO 5
          45 IF N>0 THEN PRINT "NOCH EINE"
          50 INPUT Q
          55 A(N)=Q
          60 NEXT N
          100 PRINT "DER REIHE NACH AUFGEFUEHRT"
          111 PRINT
          115 FOR I=0 TO 5
          120 PRINT A(I);"  ";
          130 NEXT I
```

Normalerweise werden Matrizen in einer Schleife behandelt, in der sich der Index laufend ändert.

EIN- UND ZWEIDIMENSIONAL

Die Matrizen, die einen Index besitzen, werden eindimensionale Matrizen genannt.

Matrizen können aber zwei oder mehrere Indizes enthalten. Zweidimensionale Matrizen setzen ihre "Familienmitglieder" in Vierecke wie die Tage eines Monates in einem Kalender.

```
10 REM +++ ZWEIDIM. MATRIX +++
18 PRINT "clear"
19 REM ------------------------- DIE MATRIX ERSTELLEN
20 DIM T(4,5)
30 FOR X=0 TO 4
40 FOR Y=0 TO 5
50 T(X,Y)=X+Y
60 NEXT Y:NEXT X
70 REM ------------------------- MATRIX AUSGEBEN
80 FOR J=0 TO 4
82 POSITION 0,2+3*J
85 FOR I=0 TO 5
87 PRINT " ";T(J,I);" ";
90 NEXT I
95 NEXT J
```

AUFGABE 30

1. Schreibe ein Programm, das eine Matrix einsetzt, um die Tage eines jeden Monats abzuspeichern. Also T(1)=31, T(2)=28 usw.

2. Schreibe mit Hilfe einer zweidimensionalen Matrix ein "Schulkalender"-Programm. Verwende zum Beispiel die Matrix KA(5,6), dann kannst Du für jeden Wochentag die entsprechenden Schulstunden eintragen.

31. ANMERKUNGEN FÜR DEN LEHRER Logik: AND, OR, NOT

Dieses Kapitel behandelt die AND-, OR- und NOT-Beziehungen sowie die numerischen Werte für WAHR und UNWAHR. Diese Beziehungen sind besonders bei IF-Anweisungen wichtig.

Im TEENAGER-Programm aus Übung 13 wird mit einer verschachtelten IF-Schleife die Meldung: "TEENAGER" ausgegeben. Es geht auch kürzer: mit der OR-Beziehung.

In diesem Kapitel kommen zwei abstrakte Ideen vor, die etwas Schwierigkeiten bereiten können. Die eine ist, daß WAHR und UNWAHR die numerischen Werte 1 und 0 besitzen. Die andere ist, daß der Computer manchmal jede Zahl, die nicht Null ist, als WAHR behandelt.

Fragen:

1. Gib bei jeder IF-Anweisung an, ob etwas gedruckt wird:

```
10 IF 3=3                THEN PRINT "HALLO"
10 IF NOT(3=3)           THEN PRINT "HALLO"
10 IF 3=3 OR 0=2         THEN PRINT "HALLO"
10 IF 3=3 AND 0=2        THEN PRINT "HALLO"
10 IF "A"="B"            THEN PRINT "HALLO"
10 IF NOT ("A"="B")      THEN PRINT "HALLO"
```

2. Welche Zahl wird jede dieser Zeilen drucken?

```
10 A=-1                  PRINT A, NOT A
10 A=0                   PRINT A, NOT A
10 A=-1:B=-1             PRINT A AND B
10 A=0:B=-1              PRINT A AND B
10 A=0:B=0               PRINT A AND B
10 A=0:B=-1              PRINT A OR B
10 A=0:B 0               PRINT A OR B
10 PRINT NOT 0
10 PRINT 2 AND 7
10 PRINT 3 AND 0
```

ÜBUNG 31: LOGIK: AND, OR, NOT

EIN WEITERES TEENAGER-PROGRAMM

```
10 REM <AND,OR,NOT>
15 DIM N$(20)
20 PRINT "clear"
30 PRINT " NAME";:INPUT N$
35 PRINT
40 PRINT " ALTER";:INPUT A
45 PRINT
50 IF (A>12) AND (A<20) THEN PRINT N$;" IST EIN
TEENAGER."
55 NFLAG =(A<13) OR (A>19)
60 IF NFLAG THEN PRINT N$;" IST KEIN TEENAGER!"
65 PRINT
70 IF (NOT NFLAG) AND (A=16) THEN PRINT "UND ";
N$;" IST SUESSE 16!"
```

Starte und speichere auf Band.

WAS BEDEUTET "AND" (="UND")?

Zwei Dinge über Teenager sind wahr: Sie sind über 12 Jahre alt und jünger als 20. Schau Dir Zeile 50 an.

> IF (Du über 12 bist) AND (Du jünger als 20 bist) THEN (bist Du ein Teenager).

WAS BEDEUTET "OR" (="ODER")?

In Zeile 55 wird OR verwendet. Zwei Aussagen gibt es: "Das Alter ist unter 13" und "das Alter ist über 19."

Nur eine dieser Aussagen muß wahr sein, damit Du "kein Teenager" bist.

IF (Du unter 13 bist) OR (Du über 20 bist) THEN (bist Du kein Teenager).

WAHR UND UNWAHR SIND ZAHLEN

Wie macht der Computer das? Er sagt, daß WAHR und UNWAHR Zahlen sind.

Regel:

WAHR ist die Zahl 1
UNWAHR ist Zahl 0

Um diese Zahlen zu sehen, gib folgendes im Editier-Modus ein:

```
PRINT 3=7
```

Der Computer prüft, ob 3 tatsächlich gleich 7 ist. Da es nicht stimmt, druckt er eine "0", was UNWAHR bedeutet.

Und:

```
PRINT 3=3
```

Der Computer prüft, ob 3=3 ist. Da es stimmt, druckt der Computer "1", was WAHR bedeutet.

WIE WERDEN WAHR UND UNWAHR IN DIE SCHACHTELN GESTECKT?

Die Zahlen WAHR und UNWAHR werden genau wie die übrigen Zahlen behandelt. Sie werden in Schachteln gesteckt, die mit Namen von Zahlenvariablen auf der Vorderseite benannt sind. Starte folgendes:

```
10 N= (3=22)
20 PRINT N
```

Die Zahl 0 wird in Schachtel N gesteckt, weil 3=22 UNWAHR ist.

Und:

```
10 N= "B"="B"
20 PRINT N
```

Die Zahl 1 wird in die Schachtel N gesteckt, weil die beiden Buchstaben in Anführungszeichen gleich sind, so daß die Aussage "B"="B" WAHR ist.

DER IF-BEFEHL ERZÄHLT KLEINE NOTLÜGEN

Der IF-Befehl sieht folgendermaßen aus:

```
10 IF     (Bedingung A)   THEN    (Befehl C)
```

Versuche dies im Editier-Modus:

```
IF 0 THEN PRINT "UNWAHR"
IF 1 THEN PRINT "WAHR"
```

Versuche nun:

```
IF 22 THEN PRINT "WAHR"
```

Regel:
In einem Befehl schaut sich der Computer die "Bedingung A" an.

Ist sie Null, erklärt der Computer: "Bedingung A ist UNWAHR" und läßt das, was nach THEN folgt, aus.

Ist sie nicht Null, erklärt der Computer: "Bedingung A ist WAHR" und gehorcht den Befehlen, die nach THEN folgen.

Der IF-Befehl erzählt kleine Notlügen. Bei WAHR wird angenommen, daß die Zahl "1" ist. Aber der IF-Befehl biegt die Wahrheit etwas um, indem er sagt, "WAHR ist alles, was nicht UNWAHR ist", das heißt, jede Zahl, die nicht Null ist, ist WAHR.

WAS BEDEUTET "NOT" (="NICHT")?

NOT verändert UNWAHR zu WAHR und WAHR zu UNWAHR. Versuche dies:

```
10 REM DOPPELTE VERNEINUNG
20 N=3
30 PRINT "N             "; N
40 PRINT "NICHT N       "; NOT N
50 PRINT "NICHT NICHT N"; NOT (NOT N)
60 REM DER COMPUTER WEISS,DASS "ICH NICHT NICHT HABE.."
61 REM BEDEUTET "ICH HABE..."
```

Speichere das Programm auf Band.

Übung 31: Logik: AND, OR, NOT

Das NOT erzählt kleine Notlügen:

N ist 3, und das wird als WAHR bezeichnet (eine kleine Notlüge).

Dann ist NOT 3 UNWAHR oder die Zahl 0.

Schließlich ergibt NOT(NOT3) das gleiche wie NOT(0) oder NOT(UNWAHR) oder WAHR oder die Zahl 1.

Versuche dies. Setze N=3 in das obige Programm ein. Du wirst erkennen, daß (NOT 3) nicht 0 gibt.

DIE ZEICHEN DER LOGIK

Du kannst diese sechs Symbole in der "Bedingung A" verwenden:

=	ist gleich
<>	ungleich
<	kleiner als
>	größer als
<=	kleiner gleich
>=	größer gleich

Du mußt zwei Tasten drücken, um das "<>"-Zeichen, das "<="-Zeichen und das ">="-Zeichen erzeugen zu können.

Die letzten beiden Zeichen sind neu. Schau Dir deshalb dieses Beispiel an, um den Unterschied zwischen < und <= zu sehen:

```
2<=3 ist WAHR        2<3 ist WAHR
3<=3 ist WAHR        3<3 ist UNWAHR
4<=3 ist UNWAHR      4<3 ist UNWAHR
```

Diese beiden "Bedingung A"-Bedingungen bedeuten das gleiche:

```
2<=Q                 (2<Q) OR (2=Q)
```

AUFGABE 31

1. Gib an, was in Schachtel N zu finden ist, wenn:

```
N=4=4
N="G"<>"S"
N=5>7
N=3>2 AND 3<2
N=4=3 OR 4=4
N=NOT 0
N=5>=4
```

2. Gib an, wann das Wort "GUMMIBAER" gedruckt wird:

```
IF 0                THEN PRINT "GUMMIBAER"
IF -1               THEN PRINT "GUMMIBAER"
IF 9                THEN PRINT "GUMMIBAER"
IF 3< >0            THEN PRINT "GUMMIBAER"
IF 0 OR -1          THEN PRINT "GUMMIBAER"
IF NOT -1           THEN PRINT "GUMMIBAER"
IF "A"="Z"          THEN PRINT "GUMMIBAER"
IF NOT(0) AND 2     THEN PRINT "GUMMIBAER"
IF NOT(0) OR -1     THEN PRINT "GUMMIBAER"
IF 4=5              THEN PRINT "GUMMIBAER"
```

3. Schreibe ein Programm, so daß in einem Satz eine doppelte Verneinung aufgespürt werden kann. Laß es nach negativen Wörtern wie nicht, nein, nichts, kein suchen, und zähle sie auf. Wenn es zwei solche Wörter gibt, handelt es sich um eine doppelte Verneinung. Überprüfe das Programm mit dem Satz "KEIN COMPUTER HAT KEIN GEHIRN".

32. ANMERKUNGEN FÜR DEN LEHRER Anwenderfreundliche Programme

Dieses Kapitel zeigt, wie man klar gegliederte Programme schreibt, die für den Anwender komfortabler zu benutzen sind.

Man sollte nicht durcheinander programmieren. Ein Schema zum Schreiben von Programmen wird in diesem Kapitel vorgestellt.

"Anwenderfreundlich" heißt, daß die Bildschirmanzeigen leicht verständlich sind, daß Tasteneingaben soweit wie möglich ohne RETURN erfolgen und daß Fehler leicht gefunden werden. Weiter bedeutet es, daß gefragt wird, ob die Eingaben in Ordnung sind und wenn nicht, daß eine Gelegenheit zum Ausbessern besteht.

Anweisungen und Hilfe sollen dem Schüler jederzeit zugänglich sein. Es sollten Gedächtnisstützen gegeben werden. Anfänger brauchen umfassende Hilfe, während fortgeschrittene Anwender nur wenig Unterstützung brauchen.

Das Schreiben von anwenderfreundlichen Programmen ist schwer beizubringen. Der Erfolg hängt hauptsächlich von der Einstellung des Programmierers ab. Der beste Ratschlag besteht darin, daß man sich sehr konzentriert, wenn man ein Programm schreibt.

Die meisten jüngeren Schüler werden keine großen Fortschritte im Bereich der anwenderfreundlichen Programmierung machen. Man hat schon viel erreicht, wenn man sie mit den Vorteilen der anwenderfreundlichen Programmierung vertraut gemacht hat und sie zu diesem Zweck einige einfache Verfahren kennengelernt haben.

Übung 32: Anwenderfreundliche Programme

Fragen:

1. Sollte Dein Programm Anweisungen geben, egal ob der Anwender sie will oder nicht?

2. Was ist eine "Gedächtnisstütze"? Gib zwei Beispiele dafür an.

3. Was versteht man unter "Scrollen"? Wie kannst Du ohne Scrollen auf dem Bildschirm schreiben lassen?

4. Welcher Befehl eignet sich am besten, wenn der Anwender nur einen Buchstaben eingeben soll? (Der Gebrauch der RETURN-Taste soll vermieden werden.)

5. Was ist eine "Fehlerfalle"? Wie würdest Du Fehler vermeiden, wenn eine Zahl zwischen 1 und 5 eingegeben werden soll?

6. In welchem Teil des Programmes befinden sich die meisten GOSUB-Befehle?

7. Warum setzt man den Einleitungsteil des Programmes ans Ende (bei den hohen Zeilennummern)?

ÜBUNG 32: ANWENDERFREUNDLICHE PROGRAMME

Es gibt zwei Anwendertypen:

1. Die meisten möchten das Programm laufen lassen. Sie wollen:

   - Anleitungen
   - Gedächtnisstützen
   - einen klar gegliederten Bildschirm
   - kein Durcheinander auf dem Bildschirm
   - daß nicht Benötigtes gelöscht wird
   - daß nicht zu viele Tasten gedrückt werden müssen
   - Schutz vor ihren eigenen Fehlern

2. Einige möchten das Programm ändern. Sie wollen:

   - ein Programm, das in Abschnitte geteilt ist
   - daß jeder Teil mit einer REM-Überschrift bezeichnet wird
   - Erklärungen im Programm

(Vergiß nicht, daß auch Du selbst Deine eigenen Programme anwendest! Sei nett zu Dir!)

Übung 32: Anwenderfreundliche Programme

PROGRAMME BESTEHEN AUS DREI TEILEN

"EINLEITUNGSTEIL": Am Anfang des Programmablaufs.

gibt dem Anwender Anleitungen
zeichnet eine Bildschirmanzeige
setzt die Variablen auf ihre Anfangswerte
fragt den Anwender nach Anfangsinformation

HAUPTSCHLEIFE:

- steuert die Reihenfolge, in der die Aufgaben bewältigt werden
- ruft Unterprogramme auf, um die Aufgaben zu bewältigen

UNTERPROGRAMME:

- erledigen Teile des Programmes

PROGRAMMGLIEDERUNG

```
1 GOTO 1000: REM          *** Programmname ***
---
100 REM HAUPTSCHLEIFE
---
---                       ruft Unterprogramme auf
---
199 END
1000 REM
1001 REM                  *** Programmname ***
1002 REM
---
---                       REMs, die eine Beschreibung des
---                       Programms, Variablennamen usw.
---                       geben.
1999 REM
2000 REM EINLEITUNGSTEIL
---
---                       fragt nach Anfangsinformationen
---                       setzt die Variablenwerte
---                       gibt Anleitungen
---
2999 GOTO 100
```

Übung 32: Anwenderfreundliche Programme

SETZE DIE HAUPTSCHLEIFE AN DEN ANFANG DES PROGRAMMS

Setze die HAUPTSCHLEIFE in die Nähe des Anfangs, weil sie dort schneller läuft.

SETZE DEN EINLEITUNGSTEIL AN DAS ENDE DES PROGRAMMS

Setze den EINLEITUNGSTEIL in die Nähe des Endes, weil er wahrscheinlich der größte Teil des Programms sein wird. Du kannst das Programm hier leichter erweitern, um es anwenderfreundlicher zu gestalten.

SETZE UNTERPROGRAMME AN DREI STELLEN

Setze die UNTERPROGRAMME an drei Stellen:

zwischen Zeile 2 und Zeile 99 für Unterprogramme, die schnell laufen müssen

nach Zeile 2999 für Einleitungsunterprogramme

zwischen Zeile 200 und Zeile 999 für den Rest der Unterprogramme.

INFORMATIONEN BITTE

```
380 PRINT "MOECHTEST DU ANLEITUNGEN? <J/N>"
```

Dies ermöglicht es dem Anfänger, Anleitungen zu lesen, während andere nein <N> sagen können.

EINEN KNOTEN INS TASCHENTUCH DES ANWENDERS MACHEN

Verwende eine Gedächtnisstütze, um die Anwender darauf aufmerksam zu machen, was sie wählen können.

Beispiel: <J/N> wobei die Wahl "J" für "JA" oder "N" für "NEIN" besteht.

Anfänger brauchen umfangreiche Gedächtnisstützen.

BEREITE DEM ANWENDER KEINE KOPFSCHMERZEN

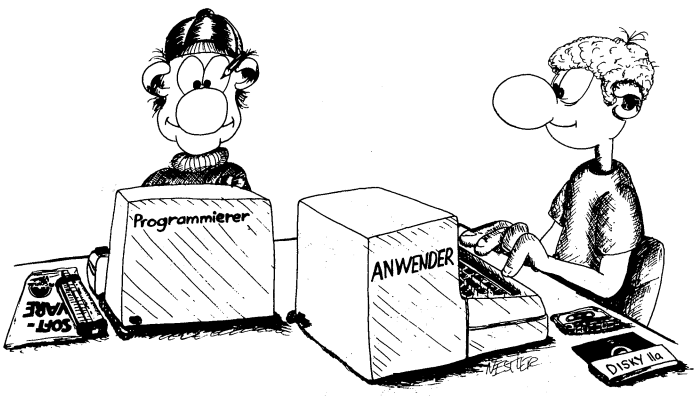
BASIC scrollt normalerweise. Es schreibt neue Zeilen am unteren Bildschirmrand und schiebt die alten Zeilen hoch.

Es ist vergleichbar mit den Leserollen der alten Römer. Sie rollten das Papier von der unteren Rolle zu oberen auf.

Übung 32: Anwenderfreundliche Programme

Vermeide das Scrollen! Verwende den POSITION-Befehl, um an jeder beliebigen Stelle drucken zu können. Lösche die Schrift, indem Du eine Zeichenkette von Leerräumen über die gleiche Stelle schreibst.

Verwende Verzögerungsschleifen, damit die Schrift auf dem Bildschirm stehenbleibt, während der Anwender sie liest.



AU! MEINE FINGER TUN WEH

Verwende für die Eingabe von einzelnen Buchstaben den GET-Befehl. Das erspart Dir das Drücken der RETURN-Taste.

```
100 OPEN #1,4,0"K:"
110 DIMA$(1)
380 PRINT "BRAUCHST DU ANLEITUNGEN? <J/N>"
382 GET #1,A:A$=CHR$(A)
384 IF A$="J" THEN 3400
```

FEHLERFALLEN

Beispiel: Füge diese Zeile in das obige Programm ein:

```
386 IF A$<>"N" THEN GOTO 380
```

Zeile 380 fragte nur nach zwei Wahlmöglichkeiten, J oder N. Wenn der Anwender eine andere Taste drückt, schickt ihn Zeile 386 zu Zeile 380 zurück.

Fallen machen Dein Programm "bombensicher". So kann es der Anwender nicht durcheinanderbringen.

AUFGABE 32

1. Betrachte das FARBFRESSER-Programm. Erkläre in REM-Zeilen das Programm. Ändere die verwendeten Farben und Zeichen.

2. Schreibe ein Geheimschriftprogramm. Der Anwender wählt ein Kennwort. Dies wird dazu verwendet, um ein chiffriertes Alphabet zu erzeugen:

   Ist das Kennwort DRACHENTIER, entferne die Buchstaben, die doppelt vorkommen, und Du erhältst DRACHENTI. Setze dieses Wort an den Anfang des Alphabets. Der Rest der Buchstaben folgt in der normalen Reihenfolge.

   DRACHENTIBFGJKLMOPQSUVWXYZ chiffriertes Alphabet

   ABCDEFGHIJKLMNOPQRSTUVWXYZ normales Alphabet

   Der Anwender hat dann die Möglichkeit, von einem Menü auf einen Text kodieren oder dekodieren zu lassen.

```
1 GOTO 1000:REM ***** FARBFRESSER *****
100 NC=0
101 FOR I=X-1 TO X+1
102 FOR J=Y-1 TO Y+1
104 IF I<0 THEN I=0
105 IF I>39 THEN GOTO 120
106 IF J<O THEN J=0
107 IF J>23 THEN GOTO 120
109 LOCATE I,J,CC
110 IF CC=C THEN X=I:Y=J:COLOR 0:PLOT X,Y:GOTO 100
112 IF CC<>0 THEN NC=1
115 NEXT J:NEXT I
120 C=C+1:IF C>3 THEN C=1
121 SOUND 1,0,0,0
122 SOUND 0,100+C*40,10,8
125 IF NC=0 THEN GOSUB 300
199 GOTO 100
300 REM DER FARBFRESSER HAT KEINE NAHRUNG
310 X=X+1:IF X>39 THEN X=1
320 SOUND 1,30,10,8
```

```
330 SOUND 0,0,0,0
399 RETURN
1000 REM BEGINN
2012 GRAPHICS 3+16
2013 SETCOLOR 0,5,8
2014 SETCOLOR 1,8,4
2015 SETCOLOR 2,12,4
2016 SETCOLOR 4,0,10
2020 FOR I=0 TO 39
2030 FOR J=0 TO 23
2040 COLOR INT(RND(9)*3)+1
2050 PLOT I,J
2060 NEXT J:NEXT I
2100 X=20:Y=12
2999 GOTO 100
```

33. **ANMERKUNGEN FÜR DEN LEHRER** Fehlersuche, STOP, CONT

Es ist sehr schwierig, jemanden das systematische Fehlersuchen beizubringen.

In diesem Kapitel wird eine Reihe von einfachen Techniken gezeigt und beschrieben. Dazu gehört auch die Technik des "STOP" und "CONT", die recht einfach zu verstehen ist. Nur durch ständiges Üben wird der Schüler in die Lage kommen, die Fehlersuche mit Zufriedenheit zu beherrschen.

Fragen:

1. Wie kannst Du den Computer dazu bringen

   STOPPED AT LINE 55

   während des Programmlaufs zu drucken?

2. Wodurch unterscheiden sich die STOP- und END-Befehle?

3. Worin unterscheidet sich der STOP-Befehl von der BREAK-Taste?

4. Was macht der CONT-Befehl?

5. Warum würdest Du STOP-Befehle in Dein Programm setzen?

6. Wie helfen Dir Verzögerungsschleifen bei der Fehlersuche in einem Programm?

7. Wie helfen Dir zusätzliche PRINT-Befehle bei der Fehlersuche in einem Programm?

8. Wieso nimmst Du die STOP- und zusätzlichen PRINT-Befehle aus dem Programm wieder heraus, nachdem Du die Fehler ausgebessert hast?

9. Kannst Du vorhersagen, in welcher Zeile die BREAK-Taste das Programm abbrechen wird? Kannst Du das auch mit dem STOP-Befehl?

ÜBUNG 33: FEHLERSUCHE, STOP, CONT

DER STOP-BEFEHL

Gib ein und starte:

```
10 REM GEHEIMES STOPPEN
20 PRINT "clear"
25 R=INT(RND(9)*200)
30 FOR I=0 TO 200
40 IF I=R THEN STOP
50 NEXT I
```

Das Programm wird stoppen, der Computer wird piepen und eine Meldung drucken:

```
STOPPED AT LINE 40
```

Was meinst Du, was der geheime Wert von I war?

Gib ein: PRINT I (keine Zeilennummer!)
und finde es heraus.

WIE STARTET MAN ES WIEDER

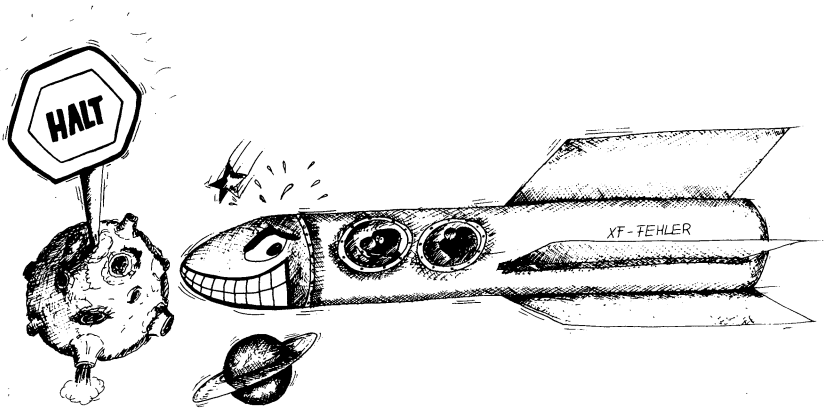
Gib den Befehl CONT ein.

"STOP" ist wie "END". STOP hält den Computer an. Er geht dann in den Editier-Modus.

Der Vorgang ist wie bei END. Allerdings wird die Zeilennummer, in der sich STOP befindet, mit ausgedruckt.

Du kannst beliebig viele STOP-Befehle in Deinem Programm verwenden.

STOP wird für die Fehlersuche in Deinem Programm verwendet.



EINE ANDERE MÖGLICHKEIT, DAS LAUFENDE PROGRAMM ANZUHALTEN

Du kannst das laufende Programm anhalten, wenn Du die BREAK-Taste drückst.

```
10 REM UNENDLICHES PROGRAMM
15 PRINT "clear"
20 PRINT "SCHILD "
21 GOSUB 40
22 PRINT "  KROETEN "
23 GOSUB 40
24 PRINT "    ALLER "
25 GOSUB 40
26 PRINT "      LAENDER ":PRINT
27 GOSUB 40
28 PRINT "        VEREINIGT "
29 GOSUB 40
30 PRINT "          EUCH "
31 GOSUB 40
39 GOTO 10
40 FOR T=1 TO 500:NEXT T
99 RETURN
```

Übung 33: Fehlersuche, STOP, CONT

Mit der BREAK-Taste hält das Programm an der Stelle an, an der es sich befindet. Es druckt:

STOPPED AT LINE XX und geht in den Editier-Modus.

(XX ist die Zeilennummer, in der es anhält.)

Mit dem Befehl CONT kann das Programm wieder an der gleichen Stelle gestartet werden.

Das obige Programm hält normalerweise in Zeile 40 an. Warum? Versuche, das obige Programm in einer anderen Zeile anhalten zu lassen.

WAS MACHST DU NACH DEM STOPPEN?

STOP setzt Du in den Teil des Programms, der nicht richtig funktioniert. Dann startest Du das Programm. Wenn es anhält, schaust Du nach, was passiert ist.

(Oder Du verwendest die BREAK-Taste, um das Programm abzubrechen. Aber es kann sein, daß es nicht an der Stelle hält, an der es Schwierigkeiten gibt.)

Denke jetzt richtig nach. Stelle Dir selbst Fragen über das, was während des Programmablaufs passiert.

Du bist jetzt im Editier-Modus. Du kannst:

- Teile des Programms auflisten lassen und sie studieren,
- den PRINT-Befehl verwenden, um nach Variablen zu schauen. Haben sie die Werte, die Du erwartet hast?
- mit dem Computer im "Rechenmodus" (ein anderer Name für den Editier-Modus) kleine Rechnungen machen, um nachzuprüfen, was der Computer macht.

Wenn Du das Problem gelöst hast, kannst Du Zeilen hinzufügen, verändern oder löschen.

DAS ERNEUTE STARTEN DES PROGRAMMS

Es gibt drei Möglichkeiten, ein Programm zu starten. Diese sind:

CONT	wenn Du das Programm nicht geändert hast
GOTO XX	wobei XX eine Zeilennummer ist
RUN	das kennst Du schon

Übung 33: Fehlersuche, STOP, CONT

Welche Unterschiede gibt es zwischen diesen drei Möglichkeiten?

CONT, GOTO XX	Dabei verändert sich der Wert der Variablenschachteln nicht. Der Inhalt bleibt wie nach dem letzten Programmlauf.
RUN	Diesmal wird der Inhalt sämtlicher Variablenschachteln zerstört. Das Programm beginnt von vorn.

Der CONT-Befehl setzt das Programm mit der nächsten Zeilennummer fort. Die Variablen behalten ihren Wert.

> VORSICHT!
> Das Programm geht nicht mit der nächsten Anweisung weiter, wenn mehrere Befehle in einer Zeile stehen. In diesem Fall würde das Programm nach CONT nicht korrekt abgearbeitet werden.

Deshalb solltest Du Dir vorsichtshalber mit LIST das Programm ansehen und den Befehl CONT nicht verwenden, wenn mehrere Anweisungen in einer Zeile stehen.

Du kannst das Programm an einer anderen Stelle anfangen lassen, indem Du (ohne Zeilennummer) den Befehl:

```
GOTO XX
```

eingibst. XX ist die Zeilennummer, von der aus Du weitermachen willst.

FEHLERSUCHE

Kleine Fehler in Deinem Programm werden in der Computerfachsprache "bugs" = "Wanzen" genannt.

Läuft Dein Programm nicht so, wie es sollte, kannst Du vier Sachen versuchen:

1. Wenn der Computer ERROR (=FEHLERMELDUNG) gemeldet hat, teilt er Dir die Zeile mit, in der er angehalten hat. Vorsicht, der Fehler kann sich in Wirklichkeit in einer anderen Zeile befinden!

2. Halte das Programm an, wenn es zwar läuft, aber nicht das Richtige macht. Füge einige PRINT-Zeilen ein, um die Werte der Variablen zu sehen.

3. Du kannst auch STOP-Befehle in das Programm setzen.

4. Wenn das Programm so schnell läuft, daß Du nicht erkennst, was passiert, kannst Du Verzögerungsschleifen einbauen.

Nimm die PRINT-Zeilen, die STOPs und die Verzögerungsschleifen aus dem Programm wieder heraus, nachdem Du es ausgebessert hast.

AUFGABE 33

1. Betrachte noch einmal das SCHLANGE-Programm und verbessere einige Fehler. Das Programm stürzt zum Beispiel ab, wenn die Schlange gegen eine Wand stößt. Gib der Schlange "Futter". Bewahre die erreichte Punktzahl auf. Beende das Programm, wenn die Schlange eine Wand berührt.

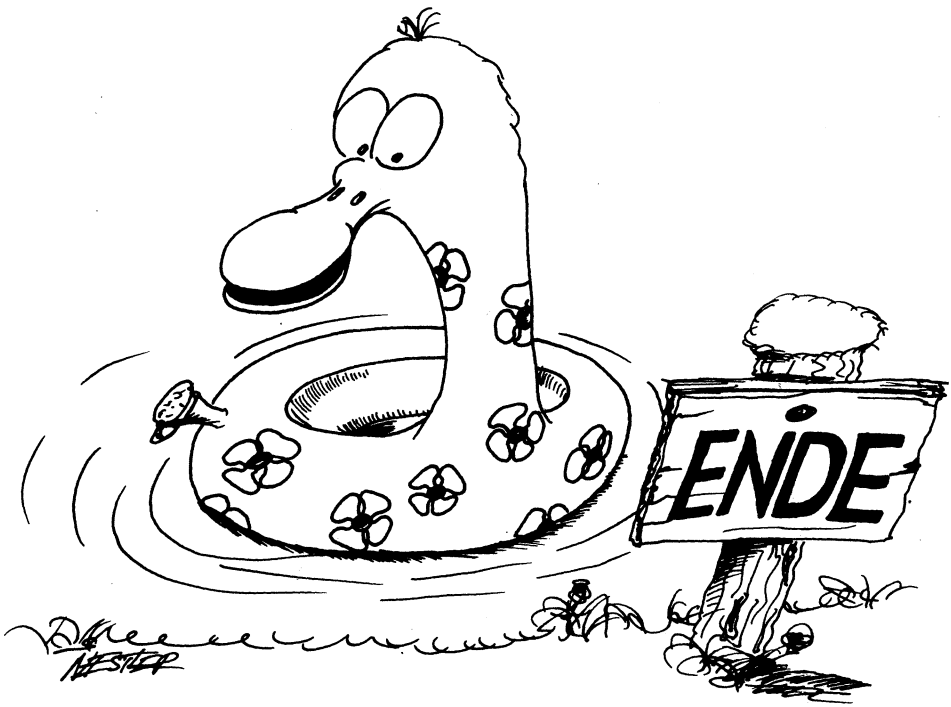
2. Überprüfe auch noch weitere Programme, die Du früher geschrieben hast, auf Fehler. Korrigiere sie.

4

Anhang

RESERVIERTE BEGRIFFE

ABS	ADR	AND	ASC	ATN
BYE				
CLOAD	CHR$	CLOG	CLOSE	CLR
	COLOR	COM	CONT	COS
	CSAVE			
DATA	DEG	DIM	DOS	DRAWTO
END	ENTER	EXP	FOR	FRE
GET	GOSUB	GOTO	GRAPHICS	
IF	INPUT	INT		
LEN	LET	LIST	LOAD	
	LOCATE	LOG	LPRINT	
NEW	NEXT	NOT	NOTE	
ON	OPEN	OR		
PADDLE	PEEK	PLOT	POINT	POKE
	POP	POSITION	PRINT	PTRIG
	PUT			
RAD	READ	REM	RESTORE	RETURN
	RND	RUN		
SAVE	SETCOLOR	SGN	SIN	SOUND
	SQR	STATUS	STEP	STICK
	STRIG	STOP	STR$	
THEN	TO	TRAP		
USR				
VAL	XIO			


ENDE

AUFGABENLÖSUNGEN

```
10 REM A1-3
15 REM *** NEUER FREUND ***
20 PRINT "HALLO DU"
30 PRINT "ATARI-COMPUTER"
```

```
1 REM A2-1
10 REM *** NAMEN ***
30 PRINT "ELKE"
35 PRINT "ANNE"
40 PRINT "invers KARL normal"
50 REM ERINNERE DICH: "invers" BEDEUTET: DRUECKE DIE ATARI-TASTE
60 REM "normal" BEDEUTET: DIE ATARI-TASTE NOCHMAL DRUECKEN
```

```
1 REM A2-3
10 REM *** NAMEN ***
20 PRINT "clear"
25 REM "clear" BEDEUTET: "ESC SHIFT CLEAR"
30 PRINT "piepser ELKE"
35 PRINT "piepser ANNE"
40 PRINT "piepser invers KARL normal"
50 REM WIE GIBST DU "piepser" EIN?
```

```
1 REM A3-5
10 REM *** VOEGEL ***
20 PRINT "clear"
30 PRINT
50 PRINT "piepser ---O---"
60 PRINT
70 PRINT
80 PRINT "          piepser ---O---"
90 PRINT
100 PRINT
110 PRINT "    piepser ---O---"
```

```
1 REM A4-4
10 REM *** LAECHELN ***
15 PRINT "clear"
20 PRINT
25 PRINT
30 PRINT
40 PRINT "       00    00"
50 PRINT
60 PRINT
70 PRINT
80 PRINT " *                * "
85 PRINT "    *           *"
90 PRINT "      *       *"
95 PRINT "        ****"
```

```
1 REM A5-2
10 REM DIE ZEICHENKETTENSCHACHTEL
20 PRINT "clear"
30 DIM N$(40)
40 LET N$="SABINE"
50 PRINT N$
60 LET N$="CLAUDIA"
70 PRINT N$
```

```
1 REM A6-1
10 REM FARBIGER NAME
20 DIM N$(40)
30 PRINT "clear"
40 PRINT "WIE HEISST DU?"
50 INPUT N$
60 SETCOLOR 2,4,8
70 SETCOLOR 4,12,6
80 PRINT N$
90 PRINT "MAGST DU DIE FARBEN?"
```

```
1 REM A6-2
10 REM *** LIEBLINGSSACHEN ***
12 DIM F$(40), T$(40)
15 PRINT "clear"
20 PRINT "WAS IST DEINE LIEBLINGSFARBE?"
25 INPUT F$
27 PRINT
30 PRINT "ICH STECKE DAS IN SCHACHTEL F$."
35 PRINT "NUN DEIN LIEBLINGSTIER?"
40 INPUT T$
43 F$=T$
50 PRINT "MAL SEHEN, WAS JETZT IN SCHACHTEL F$ STECKT"
55 PRINT "ES STECKT DRINNEN:"
56 PRINT
60 PRINT F$
```

```
1 REM A7-3
10 REM MUSIK
15 PRINT "clear"
17 DIM A$(40),B$(40)
20 PRINT "NENNE EINE MUSIKGRUPPE"
30 INPUT A$
50 PRINT "clear NENNE EINS IHRER LIEDER"
55 PRINT
60 INPUT B$
70 PRINT "clear"
80 PRINT
82 GRAPHICS 2+16
85 PRINT A$;" SPIELEN ";B$
90 GOTO 90
```

```
1 REM A8-3
10 REM *** FREUNDE ***
15 PRINT "clear"
20 PRINT "MARKUS"
25 PRINT
30 PRINT "MEIER"
40 GOTO 20
```

```
1 REM A9B-1
10 REM PIZZA
15 PRINT "clear"
20 PRINT "HALLO, ICH BIN MARIO, DER PIZZA-BAECKER"
22 PRINT
25 PRINT "SAG MIR, WAS DU AM LIEBSTEN MAGST"
27 PRINT
30 PRINT "UND ICH MACH DAS SCHON"
32 PRINT
35 PRINT "WIE GROSS SOLL DIE PIZZA SEIN (K/M/G)?"
37 DIM G$(1),W$(1)
40 INPUT G$
45 IF G$="K" THEN PRINT "DU MACHST WOHL EINE HUNGERKUR?"
50 IF G$="M" THEN PRINT "GUT GEWAEHLT!"
55 IF G$="G" THEN PRINT "DU SAMMELST SIE WOHL?"
60 PRINT "WILLST DU EINE DOPPELTE PORTION KAESE (J/N)?"
65 INPUT W$
70 REM FRAGE HIER NACH
75 REM DER GRUNDLAGE (GR$)
80 REM PILZE, SCHINKEN
85 REM SALAMI, PEPERONI
90 REM OLIVEN, USW.
94 PRINT "invers"
95 PRINT "OK, HIER IST DEINE PIZZA!"
100 PRINT "normal"
105 IF G$="K" THEN PRINT "EINE KLEINE PIZZA MIT ";
110 REM ZUTATEN AUFZAEHLEN
115 DIM GR$(1)
120 IF GR$="P" THEN PRINT "PEPERONI"
125 REM WEITER ABFRAGEN
130 PRINT
135 FOR J=1 TO 2000: NEXT J


10 REM A9A-2    JUNGEN UND MAEDCHEN
20 DIM A$(10)
30 PRINT "clear"
40 PRINT
50 PRINT "BIST DU EIN JUNGE ODER EIN MAEDCHEN?"
60 PRINT "GIB EIN 'JUNGE' ODER 'MAEDCHEN'"
70 INPUT A$
80 PRINT
90 IF A$="JUNGE"    THEN PRINT "DRACHEN STEIGEN"
95 IF A$="MAEDCHEN" THEN PRINT "BALL SPIELEN"
```

```
1 REM A9B-2
10 REM *** WELCHE FARBE ***
12 DIM C$(40),G$(40)
15 PRINT "clear"
17 PRINT
20 PRINT "SPIELER 1 SCHAU WEG"
25 PRINT "SPIELER 2 GIB EINE FARBE EIN"
40 INPUT C$
45 PRINT "clear"
47 PRINT
50 PRINT "SPIELER 1, DREH DICH UM UND RATE"
52 PRINT
55 INPUT G$
65 IF G$<>C$ THEN PRINT "FALSCH!"
70 IF G$=C$ THEN PRINT "RICHTIG"
75 PRINT
80 GOTO 55
```

```
1 REM A10-1
10 REM *** GEBURTSTAG ***
20 DIM J$(1)
25 PRINT "clear"
30 PRINT "WIE ALT BIST DU"
31 INPUT A
33 PRINT " WELCHES JAHR IST JETZT?"
40 INPUT J
45 LET B=J-A
49 PRINT "HATTEST DU HEUER"
50 PRINT "SCHON GEBURTSTAG?"
51 PRINT "<J> ODER <N>"
55 INPUT J$
60 IF J$="N" THEN LET B=B-1
70 PRINT "DU BIST ";B;" GEBOREN"
```

```
1 REM A10-2
10 REM *** MULTIPLIKATION ***
20 PRINT "clear"
22 PRINT
24 PRINT
26 PRINT
28 PRINT
30 PRINT " GIB MIR EINE ZAHL"
32 PRINT
34 INPUT A
36 PRINT
38 PRINT "GIB MIR NOCH EINE"
39 PRINT
40 INPUT B
45 LET C=A*B
47 PRINT
50 PRINT "HIER IST DAS PRODUKT: ";C
```

```
1 REM A11A-1
10 REM !"#$% BELEIDIGUNGEN %$#!
20 PRINT "clear"
22 DIM N$(40)
25 PRINT " HALLO DU! WIE HEISST DU?"
27 PRINT
29 PRINT
30 INPUT N$
35 PRINT "clear"
36 PRINT
37 PRINT
38 PRINT
39 FOR T=1 TO 1000:NEXT T
40 PRINT "BAH!"
50 PRINT
60 PRINT N$;", DEIN VATER ISST KNOBLAUCH!!!!"
70 PRINT "piepser"
```

```
1 REM A11B-1
10 REM TOENE
12 DIM J$(1)
15 PRINT "clear"
20 PRINT
21 PRINT
22 PRINT "TONHOEHE <VON 1 BIS 255>"
25 PRINT
26 INPUT T
27 PRINT
30 PRINT "LAENGE IN SEKUNDEN"
31 PRINT
32 INPUT S
33 PRINT
35 L=S*500
40 SOUND 1,T,10,8
45 FOR I=1 TO L:NEXT I
50 SOUND 1,T,10,0
60 PRINT "NOCHMAL?"
61 INPUT J$
65 IF J$="N" THEN END
99 GOTO 15
```

```
1 REM A11B-2
10 REM *** UHR ***
20 PRINT "clear"
23 PRINT "JETZIGE ZEIT:STD,MIN,SEK"
25 INPUT T,M,S
30 FOR I=1 TO 400: NEXT I
32 PRINT T;":";M;":";S
33 S=S+1
40 IF S<60 THEN GOTO 30
41 M=M+1
42 S=0
45 IF M<60 THEN GOTO 30
50 LET T=T+1
51 LET M=0
60 IF T>12 THEN LET T=1
90 GOTO 26
```

```
1 REM A12B-3
10 REM *** ZAHLEN ***
20 PRINT "clear"
21 PRINT
22 PRINT
24 PRINT "GIB MIR EINE ZAHL ZWISCHEN 0 UND 10"
25 PRINT
26 PRINT
28 INPUT Z
29 PRINT
30 IF Z=0 THEN PRINT "ICH HABE VIEL VON NICHTS"
31 IF Z=1 THEN PRINT "ICH BIN NUMMER EINS"
32 IF Z=2 THEN PRINT "ZWEI IST GESELLIG"
33 REM WEITER AUSFUELLEN
45 IF Z>10 THEN GOTO 99
50 FOR T=1 TO 2000:NEXT T
60 GOTO 21
99 PRINT "DAS WAR'S! "
```

```
1 REM A13-1
10 REM *** WUERFEL ***
15 DIM J$(1)
20 PRINT "clear"
22 LET W1=1+INT(RND(9)*6)
24 LET W2=1+INT(RND(9)*6)
30 LET W=W1+W2
35 PRINT "   DER ERSTE WUERFEL  ";W1
40 PRINT "   DER ZWEITE WUERFEL ";W2
45 PRINT "   DIE WUERFEL        ";W
47 PRINT
50 PRINT "NOCHMAL";
55 INPUT J$
57 PRINT
60 IF J$="J" THEN GOTO 20
```

```
1 REM A13-2
10 REM *** PAPIER,SCHERE UND STEINE ***
15 PRINT "clear"
17 PRINT
18 PRINT
19 DIM C$(1),Y$(1)
20 PRINT "SPIEL DAS"
21 PRINT
22 PRINT
25 PRINT "     P A P I E R"
26 PRINT "          S C H E R E"
28 PRINT "               S T E I N E"
29 PRINT
30 PRINT "           SPIEL GEGEN DEN COMPUTER"
34 PRINT
36 PRINT "DIE 'BREAK'-TASTE BEENDET DAS SPIEL"
38 PRINT
40 REM WAHL TREFFEN
45 PRINT "GIB EIN <P/S/ST> "
46 INPUT Y$
49 REM DER COMPUTER TRIFFT SEINE WAHL
50 LET C=INT(RND(9)*3)+1
51 IF C=1 THEN LET C$="P"
52 IF C=2 THEN LET C$="S"
53 IF C=3 THEN LET C$="ST"
55 IF C$<>Y$ THEN GOTO 60
56 PRINT "               GLEICHSTAND"
57 GOTO 46
60 REM WER GEWINNT?
62 IF C$="P" THEN IF Y$="S" THEN GOTO 90
64 IF C$="S" THEN IF Y$="ST" THEN GOTO 90
66 IF C$="ST" THEN IF Y$="P" THEN GOTO 90
70 REM DER COMPUTER GEWINNT
72 PRINT "     DER COMPUTER GEWINNT"
79 GOTO 46
90 REM DU GEWINNST
92 PRINT "        DU GEWINNST"
99 GOTO 46
```

```
1 REM A15-1
10 REM !!! URLAUB !!!
11 DIM M$(60),N$(69),O$(60),P$(60),Q$(60),R$(60),Z$(60)
12 PRINT "clear"
13 PRINT
14 PRINT
20 REM UEBERSCHRIFT
21 PRINT"URBLAUBS-WAHL-PROGRAMM"
23 PRINT
25 PRINT "WAEHLE DEINEN URLAUB"
26 PRINT "NACH DEINEM"
27 PRINT "GELDBETRAG AUS"
28 PRINT
30 REM ANWEISUNGEN
33 PRINT "GIB DEN GELDBETRAG IN DM AN "
40 PRINT
50 REM GELDBETRAG ERFASSEN
52 INPUT D
53 PRINT:PRINT
60 M$="WERFE MUENZEN MIT DEINEM BRUDER"
61 N$="VERBRINGE DEN NACHMITTAG IN EINEM
SCHOENEN SCHWEINESTALL"
62 P$="NIMM AN EINEN SAUREGURKEN WETTBEWERB TEIL"
63 REM UND SO WEITER
69 Y$="KAUFE DIR EINE SCHOENE JACHT UND KREUZE
DAMIT IN DER KARIBIK"
70 IF D<.5 THEN PRINT M$: GOTO 90
71 IF D<5 THEN PRINT N$:GOTO 90
72 IF D<50 THEN PRINT P$:GOTO 90
73 REM USW.
74 REM USW.
79 IF D<1000000 THEN PRINT Y$:GOTO 90
90 REM ENDE DES PROGRAMMS
```

```
1 REM A16-1
10 REM ZUFAELLIGER NAME
15 PRINT "clear"
20 DIM N$(40)
25 PRINT "WIE HEISST DU ";
30 INPUT N$
35 PRINT "clear"
40 POKE 752,1:REM CURSOR AUSSCHALTEN
45 X=INT(RND(9)*30)
50 Y=INT(RND(9)*23)
55 POSITION X,Y
60 PRINT N$
70 FOR T=1 TO 50:NEXT T
75 GOTO 40
```

```
1 REM A16-2
10 REM X NAME
15 PRINT "clear"
20 DIM N$(40)
25 POSITION 14,6:PRINT "S"
30 POSITION 26,6:PRINT "S"
35 POSITION 17,9:PRINT "E"
40 POSITION 23,9:PRINT "E"
45 POSITION 20,12:PRINT "F"
47 REM DEN MITTLEREN BUCHSTABEN NUR EINMAL
50 POSITION 23,15:PRINT "A"
55 POSITION 17,15:PRINT "A"
60 POSITION 14,18:PRINT "N"
65 POSITION 26,18:PRINT "N"
```

```
1 REM A17A-1
10 REM *** FUENFER SCHRITTE ***
15 PRINT "clear"
20 FOR I=0 TO 100 STEP 5
30 PRINT I
40 FOR T=1 TO 300:NEXT T
50 NEXT I
```

```
1 REM A17B-2
12 PRINT "clear"
13 DIM N$(40)
15 PRINT "DEIN NAME";:INPUT N$
17 ? "clear"
20 FOR I=1 TO 22
25 POSITION I,I
30 PRINT N$
35 FOR T=1 TO 300:NEXT T
40 NEXT I
```

```
1 REM A17B-3
10 REM *** KLETTERMAXE ***
12 PRINT "clear"
13 DIM N$(40)
15 N$="KLETTERMAXE"
20 FOR I=23 TO 1 STEP -1
25 POSITION 10,I
30 PRINT N$;
33 FOR T=1 TO 50:NEXT T
36 PRINT "clear"
40 NEXT I
```

```
1 REM A17B-4
10 REM FREUNDE
15 PRINT "clear"
20 DIM F$(20),Y$(20),B$(20)
25 PRINT "DEIN NAME":INPUT Y$
30 PRINT
35 PRINT "DER NAME DEINES FREUNDES":INPUT F$
40 B$="               "
45 PRINT "clear"
50 REM
55 REM HAUPTSCHLEIFE
60 REM
65 FOR I=1 TO 22
70 IF I>16 THEN GOTO 60
75 POSITION 2*I,12
80 PRINT B$
85 POSITION 2*I+2,12
90 PRINT Y$
95 POSITION 12,I
```

```
100 PRINT B$
105 POSITION 12,I+1
110 PRINT F$;
115 NEXT I
120 REM
125 FOR T=1 TO 1000:NEXT T


1 REM A19-1
10 REM *** FAMILIE ***
15 PRINT"clear"
18 DIM V$(40),T$(40),N$(40)
20 PRINT "WELCHER VERWANDTSCHAFTSGRAD?
25 PRINT
31 INPUT V$
32 PRINT
33 FLAG=0
35 READ T$:READ N$
40 IF T$=V$ THEN GOSUB 200
50 IF T$="ENDE" THEN GOTO 300
55 GOTO 35
100 DATA VATER,SIEGFRIED
101 DATA MUTTER,URSULA
102 DATA SCHWESTER, SABINE
103 DATA GROSSMUTTER,HELENE
104 DATA GROSSMUTTER,MARIA
105 DATA GROSSVATER,HEINZ
106 DATA TANTE,JUTTA
107 DATA TANTE,IRENE
108 DATA ONKEL,PETER
109 DATA COUSIN,BERND
110 DATA ENDE,ENDE
200 REM
201 REM AUSGEBEN
202 REM
210 PRINT V$;"         ";N$
260 FLAG=1
299 RETURN
300 REM
301 REM KEINE
302 REM
305 IF FLAG=0 THEN PRINT "DU HAST KEINE(N) ";V$
320 PRINT
330 FOR T=1 TO 1000:NEXT T
```

```
350 RESTORE
399 GOTO 20


1 REM A20-1
10 REM EIN LIED
15 PRINT "clear"
20 FOR I=1 TO 10
25 READ T:READ S
30 SOUND 1,T,10,8
35 FOR Z=1 TO S*100:NEXT Z
40 SOUND 1,0,0,0
45 NEXT I
50 DATA 121,3,121,3,121,2,108,1,96,2,108,1,96,2,91,1,81,6


1 REM A21-5
2 GOTO 1000:REM SINDBADS WUNDERTEPPICH
12 PRINT "clear"
198 REM
199 REM HAUPTSCHLEIFE
200 REM
201 GRAPHICS 3+16
202 SETCOLOR 4,FH,HH
203 SETCOLOR 0,F1,H1
204 SETCOLOR 1,F2,H2
205 SETCOLOR 2,F3,H3
206 A=RND(9):B=RND(9)
207 C=RND(9)
210 FOR I=1 TO 19:FOR J=1 TO 19
211 L=INT(I*24/40)
212 K=I+J:Q=INT(K*24/40)
216 X=X+A*(I+3*B)/(J+3)+C*(20-I-J)/53
218 IF X>3 THEN X=X-3
219 IF X<1 THEN X=1
225 COLOR X
230 PLOT I,Q:PLOT K,L
232 PLOT 39-I,Q:PLOT K,23-Q
234 PLOT 39-K,L:PLOT I,23-Q
236 PLOT 39-I,23-Q:PLOT 39-K,23-L
290 NEXT J:NEXT I
800 FOR T=1 TO 2000:NEXT T
999 END
1000 REM
```

```
1001 REM SINDBADS FLIEGENDER TEPPICH
1002 REM
2000 PRINT "clear"
2010 PRINT "ERSTE FARBE,          HELLIGKEIT":INPUT F1,H1
2011 PRINT "ZWEITE FARBE,         HELLIGKEIT":INPUT F2,H2
2012 PRINT "DRITTE FARBE,         HELLIGKEIT":INPUT F3,H3
2013 PRINT "HINTERGRUNDFARBE,     HELLIGKEIT":INPUT FH,HH
2999 GOTO 200
```

```
1 REM A22-1
10 REM ALPHABETISCH
15 PRINT "clear"
16 K=1
17 DIM W$(40),H$(40)
20 PRINT "DIESES PROGRAMM ORDNET BUCHSTABEN"
25 PRINT "IN ALPHABETISCHER REIHENFOLGE"
27 POSITION 0,6
30 PRINT "GIB MIR EIN WORT:"
31 PRINT:INPUT W$
32 PRINT
35 L=LEN(W$)
38 REM BUCHSTABEN TESTEN
40 FOR I=65 TO 65+26
45 FOR J=1 TO L
50 G=ASC(W$(J,J))
55 IF G=I THEN H$(K)=CHR$(G):K=K+1
60 NEXT J:NEXT I
70 PRINT "IN ALPHABETISCHER"
81 PRINT "REIHENFOLGE: "
84 PRINT
85 PRINT "      ";H$
```

```
1 REM A22-3
10 REM *** UNFUG ***
12 PRINT "clear"
15 DIM S$(99),L$(1),PL$(99)
25 PRINT
30 PRINT "GIB MIR EINEN SATZ :":INPUT S$
32 PRINT
35 L=LEN(S$)
38 K=1
40 FOR I=1 TO L
```

```
42 L$=S$(I,I)
44 IF L$="A" THEN GOTO 55
45 IF L$="E" THEN GOTO 55
46 IF L$="I" THEN GOTO 55
47 IF L$="O" THEN GOTO 55
48 IF L$="U" THEN GOTO 55
50 IF T THEN GOTO 60
52 PL$(K)=L$
53 K=K+1
55 NEXT I
65 ?
70 ? PL$
```

```
1 REM A23-1
10 REM EIN MENUE
12 PRINT "clear"
15 DIM F$(1)
17 OPEN #1,4,0,"K:"
19 POSITION 0,3
20 PRINT "WELCHE FARBE MOECHTEST DU?"
21 PRINT
22 PRINT "       <L>   LILA"
23 PRINT "       <O>   ORANGE
24 PRINT "       <G>   GRUEN"
25 PRINT "       <B>   BLAU"
26 PRINT
30 GET #1,F:F$=CHR$(F)
35 IF F$="L" THEN F=4
36 IF F$="O" THEN F=2
37 IF F$="G" THEN F=12
38 IF F$="B" THEN F=8
40 SETCOLOR 4,F,8
```

```
1 REM A23-2
10 REM *** ALBERNE SAETZE ***
12 PRINT "clear"
13 OPEN #6,4,0,"K:"
15 DIM J$(1),L$(1),S$(99)
16 PRINT "ALBERNE SAETZE ":PRINT
17 PRINT "ANLEITUNG ERWUENSCHT  <J/N>?":?:GET #6,J
18 J$=CHR$(J)
19 IF J$="J" THEN GOSUB 100
```

```
20 K=1
21 PRINT "DAS SUBJEKT:"
22 PRINT " MIT PUNKT BEENDEN."
24 GET #6,J:L$=CHR$(J)
26 IF L$="." THEN 30
28 S$(K)=L$:K=K+1
29 GOTO 24
30 S$(K)=" ":K=K+1
32 PRINT "DAS VERB:":PRINT
34 GET #6,J:L$=CHR$(J)
36 IF L$="." THEN 40
38 S$(K)=L$:K=K+1
39 GOTO 34
40 S$(K)=" ":K=K+1
42 PRINT "DAS OBJEKT: ":?
44 GET #6,J:L$=CHR$(J)
46 IF L$="." THEN 60
48 S$(K)=L$:K=K+1
49 GOTO 44
60 S$(K)="."
70 PRINT
85 PRINT S$
99 END
100 REM ANLEITUNG
105 PRINT "clear"
110 PRINT "DREI SPIELER GEBEN DEN"
115 PRINT "TEIL EINES SATZES EIN":?
125 PRINT "KEINER SIEHT,"
128 PRINT "WAS DER ANDERE EINGIBT.":?
130 PRINT "DER ERSTE GIBT DAS SUBJEKT,"
135 PRINT "DER ZWEITE DAS VERB UND"
140 PRINT "DER DRITTE DAS OBJEKT EIN."
150 FOR T=1 TO 4000:NEXT T
199 RETURN


1 REM A24B-1
10 REM *** UNTERPROGRAMME ***
15 PRINT "clear"
20 REM HAUPTPROGRAMM
21 GOSUB 100
22 GOSUB 200
25 GOSUB 300
26 IF RND(9)<.5 THEN GOSUB 400
```

```
99 END
100 REM
101 REM DAS ERSTE UNTERPROGRAMM
102 REM
110 PRINT " JETZT IST ES SOWEIT"
199 RETURN
200 REM
201 REM DAS ZWEITE UNTERPROGRAMM
202 REM
210 PRINT " DASS AUS DEINEM"
299 RETURN
300 REM
301 REM DAS DRITTE UNTERPROGRAMM
302 REM
310 PRINT "ATARI-COMPUTER ROTER RAUCH"
320 GOSUB 900
330 PRINT " HERAUSQUILLT!"
399 RETURN
400 REM
401 REM VIELLEICHT
402 REM
420 PRINT "<ICH MEINE ES NICHT SO>"
450 GOSUB 900
499 RETURN
900 REM
901 REM ZEITSCHLEIFE
902 REM
910 FOR T=1 TO 400:NEXT T
999 RETURN
```

```
1 REM A25-2
10 REM "ON...GOTO"-BEISPIEL
12 OPEN #6,4,0,"K:"
20 REM EIN MENUE
22 PRINT "clear"
25 POSITION 0,3
30 PRINT "TREFFE DEINE WAHL:"
31 PRINT:PRINT "  <A>  EIN NICKERCHEN MACHEN"
32 PRINT:PRINT "  <B>  EINEN APFEL ESSEN"
33 PRINT:PRINT "  <C>  EINEN FREUND ANRUFEN"
40 PRINT: GET #6,X:PRINT
41 X=X-64
50 ON X GOTO 60,70,80
```

```
52 GOTO 22
60 PRINT "DEIN BETT IST NICHT GEMACHT!"
61 END
70 PRINT "DEINE SCHWESTER HAT DEN LETZTEN GEGESSEN"
71 END
80 PRINT "DEIN VATER TELEFONIERT!"
81 END
```

```
1 REM A26-1
10 REM *** TEXT VERSCHLUESSELER ***
12 PRINT "clear"
15 DIM S$(99),P$(2),Q$(2),L$(99)
20 PRINT "TEXT VERSCHLUESSELER"
25 PRINT "GIB EINEN SATZ EIN"
30 INPUT S$
33 PRINT
35 L=LEN(S$)
36 S$(L+1)=" "
40 FOR I=1 TO L STEP 2
45 P$=S$(I,I+1)
50 Q$=P$(2)
51 Q$(2)=P$(1,1)
55 L$(I)=Q$
60 NEXT I
64 PRINT
65 PRINT "HIER IST DER VERSCHLUESSELTE SATZ: ":PRINT
70 PRINT " ";L$
```

```
1 REM A26-2
10 REM *** FRAGEBEANTWORTER ***
12 PRINT "clear"
15 DIM F$(99),C$(1),S$(40),V$(40),P$(99)
18 POSITION 0,3
20 PRINT " GIB EINE FRAGE EIN "
25 INPUT F$
28 L=LEN(F$):PRINT
30 REM '?' WEGNEHMEN
33 F$(L)="."
36 REM WORTENDE SUCHEN
39 FOR I=1 TO L
40 C$=F$(I,I)
45 IF C$=" " THEN S1=I:I=L
```

```
46 NEXT I
50 FOR I=S1+1 TO L
52 C$=F$(I,I)
54 IF C$=" " THEN S2=I:I=L
56 NEXT I
58 REM VERTAUSCHE DAS ERSTE WORT
59 REM MIT DEM ZWEITEN
60 S$=F$(S1+1,S2)
62 V$=F$(1,S1)
63 PRINT
65 PRINT S$;V$;F$(S2+1,L)
```

```
1 REM A26-3
10 REM GEHEIMSPRACHE
15 DIM W$(40),L$(40)
17 PRINT "clear"
18 POSITION 0,3
19 PRINT "GEHEIMSPRACHENPROGRAMM":PRINT
20 PRINT "GIB MIR EIN WORT:":PRINT
30 INPUT W$
35 L=LEN(W$)
40 PRINT
45 REM DEN ERSTEN VOKAL FINDEN
50 FOR I=1 TO L
60 IF W$(I,I)="A" THEN 72
61 IF W$(I,I)="E" THEN 72
62 IF W$(I,I)="I" THEN 72
63 IF W$(I,I)="O" THEN 72
64 IF W$(I,I)="U" THEN 72
70 NEXT I
72 IF I<>1 THEN 80
75 L$=W$
77 L$(L+1)="LAI"
79 GOTO 90
80 REM GEFUNDEN
82 L$=W$(I)
84 L$(L-I+2)=W$(1,I-1)
86 L$(L+1)="AI"
90 PRINT " ";L$
94 FOR T=1 TO 1000:NEXT T
98 GOTO 17
```

```
1 REM A27-3
10 REM *** RUECKWAERTS ***
11 DIM N$(20),B$(20),A$(20),L$(10)
12 PRINT "clear"
15 POSITION 0,3
20 PRINT" GIB MIR EINE ZAHL "
25 INPUT N
30 N$=STR$(N)
35 L=LEN(N$)
40 FOR I=1 TO L
45 B=L-I+1
47 B$(I,I)=N$(B,B)
50 NEXT I
55 B=VAL(B$)
57 PRINT
60 PRINT "    ";N
61 PRINT "  +";B
62 L$="----------"
65 PRINT "    ";L$(1,L+1)
70 A=N+B
72 A$=STR$(A)
75 IF LEN(A$)=L THEN PRINT "    ";A
76 IF LEN(A$)=L+1 THEN PRINT "  ";A
80 END
```

```
1 REM A27-4
10 REM LAUFENDE ZAHLEN
12 DIM B$(10),Z$(10)
15 PRINT "clear"
17 POSITION 0,3
20 PRINT "GIB MIT EINE ZAHL "
21 PRINT
22 INPUT Z
25 Z$=STR$(Z)
26 L=LEN(Z$)
35 POSITION 0,12
36 PRINT Z$
40 FOR I=1 TO 39-L
42 POSITION I-1,12
43 B$=Z$(2,L)
44 B$(L)=Z$(1,1)
45 Z$=B$
46 PRINT "  ";Z$
```

```
50 FOR T=1 TO 100:NEXT T
60 NEXT I
```

```
1 REM A28-2
10 REM MACHE DIESE AENDERUNGEN,
11 REM DAMIT DU EINEN PUNKT BEWEGEN KANNST
70 IF X<1 THEN X=1:REM AN DER LINKEN ECKE?
71 IF Y<1 THEN Y=1
72 IF X>38 THEN X=38
73 IF Y>22 THEN Y=22
74 REM SCHREIBE DIESE ZEILEN DAZU
29 GOSUB 500:REM RAND
500 REM RAND
510 SETCOLOR 1,4,8
511 COLOR 2
515 FOR I=0 TO 39
520 PLOT I,0
521 PLOT I,23
530 NEXT I
540 FOR I=0 TO 23
545 PLOT 0,I
546 PLOT 39,I
550 NEXT I
599 RETURN
```

```
1 REM A29-2
10 REM STERNE ABSCHIESSEN
11 REM FUEGE DIESE ZEILEN EIN
213 REM AENDERE ZEILENNUMMER 213 IN 216:
216 IF L=3 THEN GOSUB 400
401 REM
440 COLOR 0:PLOT X,I:REM STERN LOESCHEN
450 I=1:REM ENDE DER SCHLEIFE
```

```
1 REM A30-1
10 REM MATRIZEN
12 DIM D(12)
15 PRINT "clear":PRINT:PRINT:PRINT
20 FOR I=1 TO 12
22 READ D
24 D(I)=D
29 NEXT I
30 PRINT "MONAT? <1 - 12>"
31 PRINT:INPUT M
32 PRINT
35 PRINT "DER MONAT MIT NUMMER ";M;" HAT ";D(M);" TAGE."
90 DATA 31,28,31,30,31,30,31,31,30,31,30,31
```

```
1 REM A31-3
10 REM *** DOPPELTE VERNEINUNG ***
12 DIM S$(99),W$(40),N$(40),L$(1)
15 PRINT "clear":PRINT:PRINT:PRINT
19 REM --------------------------- EINEN SATZ HOLEN
20 PRINT " GIB EINEN SATZ EIN"
22 PRINT:INPUT S$:PRINT
25 L=LEN(S$)
30 REM --------------------------- ENDE DES SATZES
31 L$=S$(L,L)
32 C=ASC(L$)
34 IF C<35 OR C>90 THEN S$(L,L)=" "
40 NN=0:REM                         ANZAHL NEGATIVE WOERTER
42 S2=1
45 FOR I=1 TO L:REM --------------- WOERTER IM SATZ?
50 L$=S$(I,I)
55 IF L$=" " THEN S1=S2:S2=I+1:GOSUB 200
60 NEXT I
65 PRINT
69 REM
70 IF NN=0 THEN PRINT "KEINE NEGATIVEN WOERTER "
72 IF NN=1 THEN PRINT "EIN NEGATIVER SATZ "
74 IF NN=2 THEN PRINT "DOPPELT NEGATIV"
76 IF NN>2 THEN PRINT" VIELE NEGATIVE "
99 END
100 REM
101 REM ------------------------- EINIGE TESTSAETZE
102 REM
111 REM ICH MAG KEINEN KOHL
```

```
112 REM ICH HABE NOCH NIE NICHT KOHL GEGESSEN
113 REM ICH HABE NOCH NIE NICHT KEINEN KOHL GEMOCHT
200 REM
201 REM ------------------------- NEGATIVES WORT?
202 REM
205 RESTORE
210 W$=S$(S1,S2-2):REM              EIN WORT AUS DEM SATZ
220 READ N$
223 IF  N$="ENDE" THEN RETURN:REM  LISTE FERTIG
224 IF N$=W$ THEN NN=NN+1
230 GOTO 220
300 REM
301 REM ------------------------- NEGATIVE WOERTER
302 REM
310 DATA NEIN,NICHT,KEIN,KEINES,NIEMALS
311 DATA NICHTS,NIE,OHNE,UEBERHAUPTNICHTS
312 DATA NEGATIV,UEBERHAUPTNICHT,KEINE
313 DATA ENDE
```

```
1 REM A32-2
10 GOTO 1000
100 REM HAUPTPROGRAMM
110 GOSUB 400
115 PRINT:PRINT "CODIEREN ODER DECODIEREN?<C/D>"
116 GET #6,J:J$=CHR$(J)
120 IF J$="C" THEN 500
130 IF J$="D" THEN 600
140 GOTO 115
400 REM GEHEIMWORT
405 PRINT "GIB DAS GEHEIMWORT EIN ":PRINT
406 INPUT PW$
408 F$=PW$(1,1)
410 FOR I=2 TO LEN(PW$)
411 L1$=PW$(I,I)
412 FOR J=1 TO LEN(F$)
415 L2$=F$(J,J)
420 IF L1$=L2$ THEN 430
421 NEXT J
422 F$(LEN(F$)+1)=L1$
430 NEXT I
431 PW$=F$
433 PRINT:PRINT "ABGEKUERZT:":PRINT
434 PRINT "  ";PW$
```

```
435 REM SCHLUESSELWORT ENTFERNEN
436 PW=LEN(PW$)
438 K=1
440 FOR I=1 TO 26:L1$=A$(I,I)
441 FLAG=0
442 FOR J=2 TO PW:L2$=PW$(J,J)
445 IF L1$=L2$ THEN FLAG=1
455 NEXT J
456 IF FLAG=0 THEN F$(K,K)=L1$:K=K+1
460 NEXT I
462 REM CODIERALPHABET
465 A$=PW$
470 A$(PW+1)=F$
475 PRINT:PRINT "ALPHABET: ":PRINT
480 PRINT A$;" CHIFFRIERT"
485 PRINT B$;" NORMAL"
499 RETURN
500 REM CODIER-MELDUNG
505 PRINT:PRINT "EINGABE DER MELDUNG, ENDE MIT '*'":PRINT
507 K=1
510 GET #6,J:J$=CHR$(J)
515 IF J$="*" THEN 590
520 IF J<65 OR J>90 THEN P$(K,K)=J$:GOTO 540
525 Q=J-64
530 P$(K,K)=A$(Q,Q)
540 K=K+1:PRINT J$
589 GOTO 510
590 PRINT:PRINT P$
595 END
600 REM DECODIER-MELDUNG
610 PRINT:PRINT "MELDUNG EINGEBEN"
612 PRINT:PRINT "BEENDE SIE MIT '*'"
615 GET #6,J:J$=CHR$(J)
617 IF J$="*" THEN 690
620 FOR I=1 TO 26
625 IF J$=A$(I,I) THEN PRINT B$(I,I):GOTO 615
630 NEXT I
635 PRINT J$;
640 GOTO 615
690 END
1000 REM START
1010 PRINT "clear":PRINT
1015 OPEN #6,4,0,"K:"
1016 DIM A$(40),B$(40)
```

```
1020 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
1030 B$=A$
1040 DIM J$(1),PW$(30),F$(30)
1050 DIM L1$(1),L2$(1),P$(99)
1999 GOTO 100
```

BEGRIFFSLEXIKON

ANWEISUNG

der kleinste vollständige Abschnitt eines Programms. Sie fängt mit einem Befehl an. Der Befehl kann Ausdrücke enthalten.

ARGUMENT

die Variable, Zahl oder Zeichenkette, die zwischen den Klammern einer Funktion erscheint. Wie:

```
INT(N)    hat  N    als Argument
LEN(W$)   hat  W$   als Argument
```

ASCII

steht für American Standard Code for Information Interchange (= Amerikanischer Standard-Code für Informationsaustausch). Jedem Zeichen ist eine ASCII-Zahl zugeordnet.

ATASCII

ein Code mit 256 Zahlen, der vom ATARI-Computer verwendet wird und der dem ASCII-Code ähnelt. Jeder Buchstabe, jede Ziffer, jedes Satzzeichen und die Grafikzeichen werden durch eine Zahl identifiziert.

AUFLISTUNG

eine Liste aller Zeilen eines Programms.

AUFRUF

Die Verwendung von GOSUB ruft ein Unterprogramm auf. Das Setzen einer Funktion in eine Anweisung ruft diese Funktion auf. Unter Aufruf versteht man, daß der Computer die Befehle im Unterprogramm oder den Rechenvorgang für die Funktion aufruft und dann an die Stelle des Aufrufs zurückkehrt.

AUSDRUCK

der Teil der Anweisung, der einen einzelnen Wert - entweder eine Zahl oder eine Zeichenkette - enthält. Beispiele:

```
7*X=1                          3*LEN(D$)=RND(9)
"APFEL"<>N$
```

AUSFÜHRUNG

Ein Programm laufen lassen oder einen einzelnen Befehl oder eine Anweisung ausführen.

BASIC

Beginners All-purpose Symbolic Instruction Code. Eine Computersprache, die von John Kemeny und Thomas Kurtz von der Dartmouth Universität in den frühen sechziger Jahren entwikkelt wurde.

BEDINGUNG

ist in diesem Buch eine Bedingung, die für eine Behauptung in einer IF-Anweisung steht. Siehe Behauptung. Beispiel:

```
IF A>4 THEN 500          A>4 ist "Bedingung"
```

BEFEHL

In BASIC bringt ein Befehl den Computer dazu, etwas zu tun, wie zum Beispiel das Löschen des Speichers mit dem NEW-Befehl. Siehe Anweisung, Ausdruck. Einige Befehle benötigen Ausdrücke, um vollständig zu sein. Zum Beispiel:

```
COLOR 2+I
```

BEHAUPTUNG

der Inhalt einer Bedingung, der WAHR oder UNWAHR sein kann. Die "Bedingung A" in einer IF-Anweisung ist eine Behauptung. Eine Behauptung hat einen Zahlenwert von 0 oder -1. Siehe Ausdruck, WAHR, UNWAHR, Logik, IF, Bedingung A. Beispiel:

```
die Behauptung      "A"<>"B"  ist WAHR
die Behauptung      3 = 4     ist UNWAHR
```

BEMERKUNG

eine kurze Anmerkung, die Du ins Programm mit Hilfe der REM-Anweisung setzt. Zum Beispiel:

```
REM DIE NAECHSTEN 7 ZEILEN SIND EIN UNTERPROGRAMM
```

BILDPUNKT

ein Bildelement. Ein Bildpunkt (engl. pixel) ist der in einem bestimmten Grafik-Modus kleinste Punkt auf dem Bildschirm.

BILDSCHIRM

der Fernsehbildschirm oder der eines Monitors, der mit dem Computer verbunden ist. Siehe Monitor.

BOOT

bedeutet, daß nach dem Einschalten des Computers ein Programm abläuft, das die notwendigen Schritte einleitet, damit der Computer mit dem Anwender "reden" kann. Eine leichte Sache für moderne Computer. Früher war das ein komplizierter Vorgang. Heute bedeutet es normalerweise das Einlesen des Betriebssystems von einer Diskette (englisch: Disk Operating System oder abgekürzt DOS).

CRLF

Abkürzung für carriage return (=Wagenrücklauf) gefolgt von line feed (=Zeilenvorschub). Das entspricht dem "Wagenrücklauf" einer Schreibmaschine. Siehe Wagenrücklauf, Zeilenvorschub.

CURSOR

ein blinkendes Quadrat, das zeigt, wo das nächste Zeichen auf dem Bildschirm erscheint oder in den Pufferspeicher geschrieben wird. Cursor bedeutet "Läufer". Der Cursor läuft über den Bildschirm, während Du tippst. Der VC-20 hat zwei Cursorarten:

INPUT-Cursor	ein blinkendes Feld auf dem Bildschirm
PRINT-Cursor	unsichtbar, "zeigt" die an die Stelle, an der das nächste Zeichen gedruckt wird

DATEI

eine Ansammlung von Daten, die sich auf einer Kassette oder im Speicher des Computers befindet.

DATEN

BASIC verwendet zwei Arten von Daten: Zahlen und Zeichenketten. Die Logikdaten (WAHR, UNWAHR) sind eine Unterart der numerischen Daten.

EDITIEREN

(Korrigieren, Ausbessern). Es gibt zwei Arten: eine Zeile editieren und ein Programm editieren. In jedem Fall mußt Du Teile neu tippen, um sie auszubessern.

EDITIER-MODUS

Du befindest Dich im Editier-Modus, wenn Du die Meldung "READY." und den Cursor siehst. Der Computer erwartet eine Eingabe von Befehlen oder Programmzeilen.

EIN/AUSGABE

Eingabe von der Tastatur, vom Kassetten-Recorder usw. Ausgabe zum Bildschirm, zum Drucker, zum Kassetten-Recorder usw.

EINGEBEN

Eingeben heißt, Informationen in den Computer eintippen und dann die RETURN-Taste drücken. Die Information geht beim Tippen in den Eingabe-Speicher. Wenn RETURN gedrückt wird, verwendet der Computer die Informationen.

EINFACHE VARIABLE

eine Variable, die nicht indiziert ist.

EINLEITUNGSTEIL

ist die in diesem Buch verwendete Bezeichnung für den Initialisierungsteil eines Programms. Er enthält REM-Anweisungen, um Programme zu beschreiben, die Eingabe von Initialwerten der Variablen, das Aufstellen der indizierten Variablendimensionen, das Zeichnen der Bildschirmgrafik und alles weitere, was nur einmal am Programmanfang benötigt wird.

FEHLERFALLE

Teil des Programms, der nach Fehlern in der vom Anwender eingegebenen Information sucht. Es überprüft auch, ob das Ergebnis des Rechenvorgangs sich in annehmbaren Grenzen befindet.

FEHLERSUCHE

bedeutet, ein Programm laufen zu lassen, um die Fehler aufzuspüren und auszubessern. Du besserst die Fehler aus, indem Du das Programm editierst. Siehe Editieren.

FUNKTION

BASIC enthält eine Anzahl von eingebauten Funktionen. Jede Funktion hat einen Namen, gefolgt von Klammern. In den Klammern sind ein oder mehrere Argumente enthalten. Die Funktion hat einen einzelnen Wert (Zahl oder Zeichenkette), der durch das Argument bestimmt wird. Siehe Wert, Argument. Die in diesem Buch behandelten Funktionen sind:

ASC, CHR$, INT, LEN, RND, STR$, VAL, TAB, PEEK

GANZZAHLEN

Sie können positiv, negativ und Null sein.

GEDÄCHTNISSTÜTZE

Eine kleine Mitteilung, die Du mit einem INPUT-Befehl auf den Bildschirm bringst und die den Anwender daran erinnert, was für eine Antwort Du erwartest.

GRAFIK

bedeutet Bilder zeichnen.

GRAFIK-STEUERZEICHEN

die Zeichen, die Du bekommst, wenn Du CONTROL festhältst und eine Buchstabentaste drückst. Sie können für Grafikdarstellungen im Grafikmodus 0 und 1, und im Modus 2 eingesetzt werden. (Dann muß CHBAS=756 dezimal verändert werden.)

HINTERGRUND

Der Teil des Bildschirms, der leer ist und keine Zeichen enthält.

INDEX

Ein anderes Wort für Index ist Kennziffer. Dem Namen einer Matrix folgen eine oder mehrere Zahlen oder numerische Variable in Klammern. Jede Zahl stellt einen Index dar.

JOYSTICK

ein Gerät, das bei Spielen eingesetzt wird. Man kann ihn sich etwa wie die Steuerknüppel in Flugzeugen vorstellen. Man kann damit acht verschiedene Richtungen ansteuern.

KASSETTE

der Datenträger (Band) für einen Kassettenrecorder.

KONSTANTE

Eine Zahl oder eine Zeichenkette, die sich nicht verändert, während das Programm läuft. Sie wird direkt in der Programmzeile und nicht in einer etikettierten Schachtel gespeichert. Siehe Zeile.

LADEN

bedeutet, daß eine Datei von einer Kassette oder Diskette zum Speicher des Computers unter Verwendung des LOAD- oder CLOAD-Befehls gebracht wird.

LEERZEICHEN

das Zeichen, das einen Abstand darstellt.

LÖSCHEN

Information im Speicher zerstören oder Leerzeichen auf den Bildschirm schreiben.

LOGIK

Teil des Programms, der Zahlen oder Zeichenketten vergleicht. Es werden die Beziehungen AND, OR, NOT, =, <>, <, >, <=, und >= verwendet. Siehe Behauptung, Bedingung A, AND, OR, NOT.

MARKE

eine Zahl oder eine Zeichenkette, die ein bestimmtes Ereignis markieren. Beispiel: Die Variable S wird Null, wenn wieder ein Pfeil abgeschossen werden kann. S wird ungleich Null, wenn ein Pfeil bereits fliegt.

MATRIX

ein Satz von Variablen, die den gleichen Namen haben. Die Elemente einer Matrix sind durchnumeriert. Die Zahlen erscheinen nach dem Variablennamen in Klammern. Beispiele:

```
A(0)       ist das erste Element der Matrix A
B$(7)      ist das achte Element der Matrix B$
CD(3,M+1) ist ein Element der Matrix CD
```

MENÜ

eine Auswahlliste, die auf dem Bildschirm gezeigt wird. Bei jeder Auswahlmöglichkeit steht ein Zeichen. Der Programmbenutzer drückt die entsprechende Taste, um seine Auswahl zu treffen.

MITTEILUNG

eine PRINT-Anweisung, die Auskunft gibt, was in einer INPUT-Anweisung erwartet wird. Beispiel:

```
61 PRINT "ALTER";:INPUT A
```

MONITOR

hat zwei Bedeutungen. Erstens bezeichnet das Wort ein besonderes Fernsehgerät, das mit dem Computer verbunden wird. Er zeigt Texte und Grafik, kann aber keine Fernsehprogramme empfangen. Zweitens bedeutet das Wort in der Maschinensprache: Kontrollprogramm.

PADDLE

ein Gerät für Spiele, das einen drehbaren Knopf besitzt. In diesem Buch ist es nicht vorgestellt worden.

PFEILTASTEN

Der ATARI-Computer besitzt vier Pfeiltasten. Sie bewegen den Eingabe-Cursor nach rechts, nach links, nach oben und nach unten.

PROGRAMME

Es gibt zwei verschiedene Arten von Programmen. Das normale Programm besteht aus einer Liste von durchnumerierten Zeilen, die Anweisungen enthalten. Der Computer führt die Anweisungen (Befehle) der Reihe nach aus, wenn RUN eingegeben

wird. Das Programm wird in einem besonderen Teil des Speichers aufbewahrt. Es kann nur jeweils ein Programm gespeichert werden.

Im Editier-Modus kann ein Ein-Zeilen-Programm eingegeben werden. Es besitzt keine Zeilennummer und startet, sobald die RETURN-Taste gedrückt wird. Es wird nicht innerhalb eines gespeicherten Programms abgelegt. Es kann aber während des Programmlaufs die Variablen ändern, mit denen das abgespeicherte Programm arbeitet.

PSEUDO-ZUFALLSZAHL

eine Zahl, die vom Computer mit der RND-Funktion ausgerechnet wird. Sie wird normalerweise "Zufallszahl" genannt. Diese Bezeichnung betont aber, daß die Zahl nicht wirklich zufällig ist (weil sie mit einem bekannten Vorgang ausgerechnet wird). Sie ist aber für den Computeranwender nicht vorhersehbar.

PUFFER

ein Bereich im Speicher zur vorübergehenden Speicherung von Informationen, die vom Computer ein- oder ausgegeben werden.

RAND

der äußerste Bereich des Bildschirms bei einem Fernsehgerät oder Monitor.

REIHE

waagrecht geordnete Dinge.

RECHENOPERATION

In der Arithmetik: Addition, Subtraktion, Multiplikation und Division, mit den Symbolen +, -, * und /.

RESERVIERTE BEGRIFFE

eine Liste von Wörtern und Abkürzungen, die BASIC als Befehle, Anweisungen oder Funktionen erkennt. Die reservierten Wörter dürfen nicht als Variablennamen verwendet werden.

RUN-MODUS

Führt ein Computer gerade ein Programm aus, so befindet er sich im RUN-Modus. Vom Editier-Modus gelangt man in den RUN-Modus durch die Eingabe von RUN. Nach Ablauf des Programms kehrt der Computer wieder in den Editier-Modus zurück.

SATZZEICHEN

die Zeichen Punkt (.), Komma (,), /, ?, !, $, usw.

SCHLEIFE

ein Teil des Programms, der immer wieder ausgeführt wird. Es gibt zwei Schleifenarten: FOR...NEXT-Schleifen und Schleifen, die IF-Befehle mit einer Schleifenvariablen und GOTO-Befehlen verwenden.

SCHLEIFENVARIABLE

die Zahl, die sich ändert, wenn die Schleife wiederholt wird. Zum Beispiel:

```
40 FOR I= 0 TO 5
50 NEXT I           I ist die Schleifenvariable
```

SCROLLEN

die Art, in der der Computer normalerweise auf einen vollen Bildschirm schreibt. Er setzt eine neue Zeile auf den unteren Bildschirmrand und rückt die alten Zeilen nach oben. Dies nennt man in der Computerfachsprache Scrollen.

SPALTE

senkrecht geordnete Dinge.

SPEICHER

der Teil des Computers, in dem Informationen aufbewahrt werden. Der Speicher besteht aus Halbleiterchips. Wir stellen ihn uns aber als Schachteln mit Etiketten auf der Vorderseite und Informationen als Inhalt vor.

SPEICHERN AUF KASSETTE

Ein Programm, das sich im Speicher des Computers befindet, wird mit dem Befehl CSAVE auf eine Kassette gespeichert:

```
CSAVE
```

SPRINGEN

Der GOTO-Befehl läßt den Computer auf eine andere Zeile im Programm springen und führt nicht die nächste Zeile aus.

STAPEL

(Engl. stack) ist ein Datentyp, der bei der Maschinenspracheprogrammierung verwendet wird. Die Daten werden so in eine Spalte gebracht, daß das zuletzt Eingegebene als erstes wieder weggenommen wird.

STRING

Siehe Zeichenkette.

SYNTAX

bedeutet die Art und Weise, wie eine Anweisung in BASIC ausgedruckt wird. "Syntax error" heißt, daß ein Befehls- oder Variablenname falsch buchstabiert wurde oder daß die Satzzeichengebung oder die Reihenfolge der Satzteile falsch ist.

TIPPEN

das Drücken der Computertasten.

TONGEBER

Ein Computer erzeugt damit einen Ton (er piepst), um den Programmierer auf etwas hinzuweisen. Beim ATARI klingt das eher wie ein Brummen. Der Ton wird mit ESC CONTROL 2 in einer PRINT-Anweisung erzeugt.

ÜBERSCHRIFT

der Name eines Programms oder Unterprogramms. Wird in eine REM-Anweisung geschrieben.

UNTERPROGRAMM

ein Teil eines Programms, der mit einer von einem GOSUB-Befehl aufgerufenen Zeile beginnt und mit einem RETURN-Befehl endet. Es kann von mehreren Stellen des Programms aufgerufen werden.

UNWAHR

die Zahl 0. Siehe Logik, Wahr, Behauptung.

VARIABLE

ein Name für eine Speicherschachtel. Die Schachtel enthält einen Wert. Wenn der Computer einen Variablennamen in einem Ausdruck bemerkt, geht er zur Schachtel, bringt eine Kopie des Schachtelinhalts zurück zum Ausdruck, ersetzt sie durch den Variablennamen und fährt mit der Auswertung des Ausdrukkes fort. Siehe Variablennamen.

VARIABLENNAME

Eine Variable stellt entweder eine Zahlen- oder eine Zeichenkettenvariable dar. Der Name kennzeichnet dies. Zeichenkettenvariablen bestehen aus Namen, die mit einem "$"-Zeichen enden. Zahlenvariablen hingegen nicht. Der Variablenname kann jede beliebige Anzahl von Zeichen enthalten, aber nur die ersten beiden werden vom Computer registriert, wenn er seine Variablentabelle erstellt. Das erste Zeichen muß ein Buchstabe sein, der Rest kann aus Buchstaben oder Zahlen bestehen.

VARIABLE, EINFACHE

Siehe einfache Variable.

VERKETTUNG

Bedeutet das Zusammenfügen zweier Zeichenketten.

VERSCHACHTELN

wenn eine Sache in einer anderen steckt. In einem Programm verschachteln wir Schleifen. Innerhalb einer Anweisung können wir Ausdrücke oder Funktionen verschachteln.

```
L=INT(LEN(A$)+3,J)        verschachtelte Funktion
X=5*(6+(7*(8+K)))         verschachtelte Klammern
```

VERZÖGERUNGSSCHLEIFE

ein Teil des Programms, der nichts anderes macht, als Zeit zu verbrauchen. Beispiel:

```
30 FOR T=1 TO 2000:NEXT T
```

VERZWEIGUNG

eine Stelle in einem Programm, in der es eine Wahlmöglichkeit gibt, welche Anweisung als nächstes ausgeführt werden soll. Eine IF-Anweisung ist eine Verzweigung. Das gleiche gilt auch für eine ON...GOTO-Anweisung. Eine Verzweigung ist nicht das gleiche wie ein Sprung. Siehe Springen.

WAGENRÜCKLAUF

Auf einer Schreibmaschine drückst Du den Hebel, der den papiertragenden Wagen so bewegt, daß man mit einer neuen Zeile beginnen kann. Beim Computer bedeutet es, daß der Cursor an den Anfang der Zeile, aber nicht zur nächsten Zeile bewegt wird. Siehe Zeilenvorschub.

WAHR

hat den Wert -1. Siehe Logik, Unwahr, Behauptung.

WEGGABELUNG

beschreibt einen Verzweigungspunkt im Programm. Siehe Verzweigung.

WERT

Der Wert einer Variable ist die Zahl oder die Zeichenkette, die in die zur Variablen gehörigen Speicherschachtel gesteckt wird. Siehe Variable.

WERTAUSTAUSCH

Wenn eine Funktion verwendet (aufgerufen) wird, wird ihr Feld in dem Ausdruck durch einen Wert (eine Zahl oder Zeichenkette) ersetzt. Dies nennt man Wertaustausch.

ZAHL

ist eine Informationsart von BASIC. Die andere Art ist die Zeichenkette. Die Zahlen sind normalerweise Dezimalzahlen. Siehe ganze Zahlen, Zeichenketten.

ZEICHEN

Buchstaben, Zahlen, Interpunktion und Leerzeichen sind Zeichen. Auch die Grafikzeichen, die auf einigen Tasten abgebildet sind, sind Zeichen.

ZEICHENKETTE

eine Art von Daten im BASIC. Sie besteht aus einer Reihe von Zeichen. Siehe Zahl.

ZEIGER

eine Zahl im Speicher, die angibt, wo Du Dich innerhalb der DATA-Listen gerade befindest. Dieser Zeiger ist für den Programmierer nicht sichtbar, er hilft aber, ein Programm besser zu verstehen.

ZEILE

In BASIC gibt es zwei verschiedene Arten von Zeilen: Die, die mit einer Nummer beginnen, werden im Speicher abgelegt. Zeilen ohne Nummer werden sofort ausgeführt (siehe Ausführung). Eine Zeile kann eine oder mehrere Anweisungen, die dann durch Doppelpunkt getrennt sind, enthalten.

```
16 IF 7<=INT(Z) THEN PRINT LEN(Q$)+2;"RATTE":GOTO 40
```

16	Zeilennummer
IF 7<=INT(Z) THEN PRINT ..."RATTE"	Anweisung
GOTO 40	Anweisung
7<=INT(Z)	eine Behauptung
7<=INT(Z)	ein Ausdruck
LEN (Q$"R")+2;"RATTE"	ein Ausdruck
INT(Z)	eine Funktion
LEN(Q$)	eine Funktion
Z	Argument
Q$	Argument
7, 2, "RATTE"	Konstanten
<=, +	Operationen
IF, INT, THEN, PRINT, LEN, GOTO	reservierte Begriffe

ZEILENEDITIERUNG

das Übertippen von Zeilenabschnitten zur Korrektur. Dazu stellt man den Cursor an die Stelle, die man ändern will und gibt dann die richtigen Zeichen ein.

ZEILENNUMMER

die Zahl am Anfang einer Programmzeile. Die Zeilennummer sagt dem Computer, wo er diese Zeile im Programm anordnen soll. Es gibt auch Zeilen, die keine Nummer besitzen (siehe Ausführung).

ZEILENPUFFER

der Speicherraum, der die eingetippten Zeichen empfängt. Siehe Pufferspeicher.

ZEILENVORSCHUB

den Cursor geradewegs zur nächsten Zeile hinabbewegen. Die ASCII-Zahl 10 signalisiert diesen Befehl dem Bildschirm oder dem Drucker.

ZUFALLSZAHL

Zahlen, die nicht vorhergesagt werden können. Wie die Zahlen, die beim Würfeln erscheinen oder die Seite einer Münze, die erscheint, wenn man sie zehnmal wirft.

FEHLERMELDUNGEN

Fehler-Code-Nummer	Fehler-Code-Meldung und Bedeutung
2	**Speicher reicht nicht aus** (memory insufficient) Dein Programm ist zu groß und hat keinen Platz mehr im Speicher.
3	**Zahlenbereich falsch** (value error) Du hast eine negative Zahl eingeben, aber es sollte eine positive sein.
4	**Zu viele Variablen** (to many variables) Du darfst maximal 128 Variablen verwenden.
5	**Zeichenkette zu lang** (string length error) Du hast versucht, eine Zeichenkette zu bilden, die größer war als in der DIM-Anweisung vereinbart.
6	**Zu wenig Daten** (out of data error) Du hast versucht, mehr Daten mit READ einzulesen als in der DATA-Anweisung vorhanden waren.
7	**Zahl größer als 32767** (number greater then 32767) Du verwendest eine negative Zahl anstelle einer positiven, oder Deine Zahl übersteigt den Wert 32767.

Fehler-Code-Nummer	Fehler-Code-Meldung und Bedeutung

8 **Falscher INPUT-Befehl** (input statement error)

Du hast versucht, bei INPUT einen Buchstaben oder ein Satzzeichen einzugeben, obwohl der Computer eine Zahl erwartet hat.

9 **DIM-Fehler** (array or string DIM error)

Du hast einen der folgenden Fehler gemacht: den DIM-Befehl vergessen, DIM zweimal verwendet, in DIM eine Zahl verwendet, die größer ist als 32767, in DIM eine negative Zahl verwendet, eine größere Zahl in einer Variablen verwendet als durch DIM festgelegt, eine zu lange Zeichenkette verwendet.

10 **Stapelspeicherüberlauf** (argument stack overflow)

Du hast zu viele verschachtelte GOSUB-Befehle verwendet oder einen zu großen Ausdruck.

11 **Zahlenbereich verlassen** (floating point overflow or underflow error)

Du hast versucht, durch Null zu dividieren. Oder Deine Zahl war zu groß.

12 **Zeile nicht gefunden** (line not found)

Du hast mit GOTO, GOSUB oder THEN eine Zeile angegeben und hast vergessen, sie ins Programm zu schreiben.

Fehler-Code-Nummer	Fehler-Code-Meldung und Bedeutung

13 **Kein passender FOR-Befehl** (no matching FOR statement)

Dein Programm erreicht eine Next-Anweisung, bevor es über einen FOR-Befehl gekommen ist.

14 **Zeile zu lang** (line too long error)

Du hast eine Zeile geshrieben, die zu lang oder zu kompliziert ist.

15 **GOSUB/FOR nicht vorhanden** (GOSUB or FOR line deleted)

Dein Programm erreicht eine RETURN-Anweisung, aber Du hast inzwischen die Zeile mit der GOSUB-Anweisung gelöscht. Oder das Programm erreicht einen NEXT-Befehl und Du hast inzwischen den FOR-Befehl gelöscht.

16 **RETURN-Fehler** (RETURN error)

Dein Programm erreicht einen RETURN-Befehl, bevor es auf GOSUB kommt.

17 **Syntax-Fehler** (garbage error)

Das Programm versteht eine Eingabe nicht. Die Bytes können von einem POKE-Befehl kommen, der auf einen falschen Platz zugreift. Am besten stellst Du den Computer wieder in den Grundzustand, indem Du RESET drückst und dann den POKE-Befehl entfernst. Läuft das Programm auch jetzt noch nicht, solltest Du den Rechner aus- und wiedereinschalten. Das Programm ist dann verloren.

Fehler-Code-Nummer	Fehler-Code-Meldung und Bedeutung
18	**Ungültige Zeichenkette** (invalid string character) Du hast eine Zeichenkette verwendet, deren erstes Zeichen ungültig ist. Oder Du hast in einer VAL-Funktion eine Zeichenkette eingesetzt, die nicht vollständig aus Ziffern besteht.

STICHWORTVERZEICHNIS: BEFEHLE UND FUNKTIONEN

STICHWORTVERZEICHNIS: TASTENBELEGUNG

STICHWORTVERZEICHNIS DER GRÖSSEREN PROGRAMME

STICHWORTNVERZEICHNIS

O

P

R

S

T

U

V

W

Z

Aus dem M&T-Buchverlag

W. Pest: **Computerchinesisch für Einsteiger**
Juli 1984, 107 Seiten
Ein praxisnahes Lexikon, das Personal Computer-Benutzern und solchen, die es werden wollen, das Lesen von Fachzeitschriften, Büchern, Bedienungsanleitungen und Datenblättern erleichtert · über 1000 häufig benötigte Fachbegriffe klar und verständlich erläutert · mit zahlreichen Abbildungen.
Best.-Nr. MT 690, DM 28,— (Sfr. 25,90/öS 218,40)

Personal Computer Lexikon
1982, 136 Seiten
Über 1000 Suchbegriffe aus Hard- und Software · deutsch/englisch · ausführlicher Artikel zu jedem Suchbegriff · englisch/deutsch Register im Anhang · der ideale Einstieg ins Homecomputing · das unentbehrliche Nachschlagewerk für den Profi.
Best.-Nr. MT 390, DM 19,80 (Sfr. 18,50/öS 154,40)

M. J. Winter: **Lehrspielzeug Computer: Atari**
Juli 1984, 139 Seiten
Das neue Computer-Kinderbuch für den Atari 400, 800 und 1200 · Spielprogramme und grafische Darstellungen für Kinder ab 8 Jahren · viele Rechenaufgaben für den kleinen Einstein · so macht Lernen Freude!
Best.-Nr. MT 696, DM 24,80 (Sfr. 23,—/öS 193,40)

H. Glicksman/K. Simon: **Spiele für den Atari**
September 1984, 216 Seiten
Eine unterhaltsame Einweisung in die Atari-BASIC-Programmierung anhand von bereits bewährten sowie raffinierten neuen Computerspielen · Wie man ein Programm strukturiert · Einsatz von Unterprogrammen · Tabellenverarbeitung · Bewegte Grafiken · Testen von Programmen · Noch nie hat Home-Computing so viel Spaß gemacht!
Best.-Nr. MT 678, DM 32,— (Sfr. 29,50/öS 249,60)

H. L. Schneider/R. Bichler: **Das Atari-Buch, Bd. 1**
1984, 158 Seiten
Die grundlegenden Programmiermöglichkeiten für Ihren Atari · mit einem Spiel zum Eingewöhnen · Erstellung von Text und Grafik · Player Missiles · Basic-Besonderheiten · ausführliche Assemblerlistings im Anhang.
Best.-Nr. MT 703, DM 32,— (Sfr. 29,50/öS 249,60)

T. Bridge: **Atari-Abenteuerspiele**
1984, 148 Seiten
Alles über die Anfänge der Abenteuerspiele · Textabenteuer mit vielen Rätseln · Schatzsuche · Kampf mit Monstern · Das Auge des Sternenkriegers · mit hilfreichen Anregungen zum Schreiben Ihrer eigenen Spieleprogramme.
Best.-Nr. MT 727, DM 29,80 (Sfr. 27,50/öS 232,40)

J. White
Strategische Computerspiele für Ihren Atari
1984, 148 Seiten
Aufbau eines Spielfeldes · der Bewegungsablauf · Mustereröffnungen · das Endspiel · Dame, Schach, Warp Trog als Beispiele strategischer Spiele · Anleitung zur systematischen Fehlersuche · Grundkenntnisse in Atari-Basic erforderlich.
Best.-Nr. MT 681, DM 32,— (Sfr. 29,50/öS 249,60)

H. Kohl/T. Kahn et al.: **Spiel und Spaß mit dem Atari**
1984, 338 Seiten
Einfache Programme in Basic · wie man ein Spiel entwickelt · Lernstoff trainieren · Zahlen und Logik · Grafik · Farben · Töne und Musik · den Atari-Computer spielend erforschen.
Best.-Nr. MT 672, DM 42,— (Sfr. 38,60/öS 327,60)

H.L. Schneider/W. Eberl
Das Commodore 64-Buch, Bd. 1
1984, 270 Seiten
Der Commodore 64 und seine Handhabung · Einführung in die Grafik · Balkendiagramme · Einführung in die Spritetechnik · Basic-Erweiterungen in Assembler · Ein Leitfaden für Erstanwender.
Best.-Nr. MT 591 (Buch)
DM 48,— (Sfr. 44,20/öS 374,40)
Best.-Nr. MT 592 (Beispiele auf Diskette)
DM 58,— (Sfr. 58,—/öS 522,—)

H.L. Schneider/W. Eberl
Das Commodore 64-Buch, Bd. 2
1984, 181 Seiten
Spiele nicht nur zum Abtippen · Programmlisting · Programmbeschreibung · Variablenübersicht · Programme nach Anleitung frei ergänzbar · das ideale Buch, um Programmieren spielend zu lernen.
Best.-Nr. MT 593 (Buch)
DM 38,— (Sfr. 35,—/öS 296,40)
Best.-Nr. MT 594 (Beispiele auf Diskette)
DM 58,— (Sfr. 58,—/öS 522,—)

H. L. Schneider/W. Eberl
Das Commodore 64-Buch, Bd. 3
1984, 206 Seiten
Alles über Sprites · Wissenswertes über Multi-Color-Grafik · Assembler/Disassembler · jede Menge Basic-Erweiterungen · Umgang mit dem Soundgenerator · ein Leitfaden für Fortgeschrittene.
Best.-Nr. MT 595 (Buch)
DM 38,— (Sfr. 35,—/öS 296,40)
Best.-Nr. MT 596 (Beispiele auf Diskette)
DM 58,— (Sfr. 58,—/öS 522,—)

H. L. Schneider/W. Eberl
Das Commodore 64-Buch, Bd. 4
1984, 261 Seiten
Einführung in Maschinenprogrammierung · Verknüpfung von Maschinenprogrammen mit Basic-Programmen · alles über Assembler/Disassembler · der Leitfaden für Systemprogrammierer.
Best.-Nr. MT 597 (Buch)
DM 38,— (Sfr. 35,—/öS 296,40)
Best.-Nr. MT 598 (Beispiele auf Diskette)
DM 58,— (Sfr. 58,—/öS 522,—)

H. L. Schneider/W. Eberl
Das Commodore 64-Buch, Bd. 5
Juli 1984, 322 Seiten
Ein Leitfaden durch Simon's Basic · ausführliche Besprechung aller Befehle · viele erklärende Beispiele · mit kommentierter Assembler-Listing · das richtige Nachschlagewerk für den geübten Commodore 64-Benutzer.
Best.-Nr. MT 599 (Buch)
DM 38,— (Sfr. 35,—/öS 296,40)
Best.-Nr. MT 600 (Beispiele auf Diskette)
DM 58,— (Sfr. 58,—/öS 522,—)

Die angegebenen Preise sind Ladenpreise

Sie erhalten M&T-Bücher in guten Buchhandlungen, Computershops und Fachabteilungen der Kaufhäuser.

Sollten Sie diese Bücher ausnahmsweise im Handel nicht beziehen können, so bestellen Sie bitte bei:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Str. 2, 8013 Haar, Telefon: 089/4613-220

Aus dem M&T-Buchverlag

H.L. Schneider/W. Eberl
Das Commodore 64-Buch, Bd. 6: Spiele
1984, 190 Seiten
Programmieren auf dem Commodore 64 spielend gelernt · leicht verständliche Spielanleitungen · Programmlisting mit anschließender Programmbeschreibung · Variablenübersicht · Tips zum Ändern und Ergänzen des Programms.
Best.-Nr. MT 619 (Buch)
DM 38,— (Sfr. 35,—/öS 296,40)
Best.-Nr. MT 620 (Beispiele auf Diskette)
DM 58,— (Sfr. 58,—/öS 522,—)

H. L. Schneider: **Das Commodore 64-Buch, Bd. 7**
August 1984, 110 Seiten
Der Commodore 64 als Klaviatur · Noten schreiben mit hochauflösender Grafik · relative Dateien am Beispiel einer kleinen Adreßverwaltung · Benutzung des Joysticks und der Paddles · Grafikspeicher unter Kernal · Interrupt-Manager · eigene Zeichen definieren · Bildschirmrollen · für Profis.
Best.-Nr. MT 731, DM 38,— (Sfr. 35,—/öS 296,40)

Computerspiele & Wissenswertes — Commodore 64
1984, 156 Seiten
Eine Sammlung von interessanten und nützlichen Maschinenprogrammen · schnelle binäre Artithmetik · Basic-Erweiterungen · mit unterstützendem Assembler-Listing · für den fortgeschrittenen Programmierer.
Best.-Nr. MT 601 (Buch)
DM 29,80 (Sfr. 27,50/öS 232,40)
Best.-Nr. MT 602 (Beispiele auf Diskette)
DM 38,— (Sfr. 38,—/öS 342,—)

F. Ende
Das große Spielebuch — Commodore 64
1984, 141 Seiten
46 Spielprogramme · Wissenswertes über Programmiertechnik · praxisnahe Hinweise zur Grafikherstellung · alles über Joystick- und Paddleansteuerung · das Spielebuch mit Lerneffekt.
Best.-Nr. MT 603 (Buch)
DM 29,80 (Sfr. 27,50/öS 232,40)
Best.-Nr. MT 604 (Beispiele auf Diskette)
DM 38,— (Sfr. 38,—/öS 342,—)

Dr. P. Albrecht: **Commodore 64 — Multiplan**
1984, 230 Seiten
Multiplan jetzt auch für den Commodore 64 · der volle Leistungsumfang der 16-Bit-Version · Einführung in die Arbeitsweise von Tabellenkalkulationsprogrammen · praxisnahe Beispiele · Beschreibung alle Befehle und Funktionen · nicht nur für Anfänger.
Best.-Nr. MT 655, DM 48,— (Sfr. 44,20/öS 374,40)

E. H. Carlson: **Basic mit dem Commodore 64**
1984, 320 Seiten
Ein Basic-Lehrbuch für den jugendlichen Anfänger · übersichtlich gegliederte Lernprogramme · Alles über INPUT-GOTO · Let-Befehle · Editorfunktionen · POKE-Befehle für die Grafik · geeignet auch als Leitfaden für Lehrer und Eltern.
Best.-Nr. MT 657, DM 48,— (Sfr. 44,20/öS 374,40)

R. E. Williams: **CalcResult richtig eingesetzt**
1984, 236 Seiten
Ein Übungsbuch speziell für Anwender des CalcResult-Computerprogramms · zahlreiche Einsatzmöglichkeiten im täglichen Leben · Kreditrückzahlung · Rabattberechnung · Kostendeckung · Inventur · Finanzierung und Ankauf eines Hauses und vieles andere mehr.
Best.-Nr. MT 671, DM 48,— (Sfr. 44,20/öS 374,40)

W. B. Sanders: **Einführungskurs: Commodore 64**
1984, 276 Seiten
Die Programmiersprache Basic · Einsatzgebiete des Commodore 64-Basic: Grafik, Musik, Dateiverwaltung · mit vielen Beispielprogrammen, häufig benötigten Tabellen und nützlichen Tips · für Einsteiger und Fortgeschrittene.
Best.-Nr. MT 685, DM 38,— (Sfr. 35,—/öS 296,40)

J.W. Willis/D. Willis
Commodore 64 — leicht verständlich
1984, 154 Seiten
Informationen für den Computer-Neuling · Installation und Inbetriebnahme · Programmieren in Basic · Grafik und Töne · Auswahl von Hardware und Zubehör · Software für Ihren Computer · die ideale Einführung in das Arbeiten mit Ihrem Commodore 64.
Best.-Nr. MT 700, DM 29,80 (Sfr. 27,50/öS 232,40)

G. Beekman: **Ihr Heimcomputer Commodore 64**
August 1984, 296 Seiten
Alles Wissenswerte im Umgang mit dem Commodore 64 · Planung, Kauf und Inbetriebnahme der Anlage · Einsatz fertig gekaufter oder selbst erstellter Programme · Schwächen und Stärken der altbewährten und neuesten Programmiersprachen · die gängigsten Software-Angebote für jeden Einsteiger ·
Best.-Nr. MT 701, DM 38,— (Sfr. 35,—/öS 296,40)

M. J. Winter
Das Commodore 64-LOGO-Arbeitsbuch
September 1984, 225 Seiten
Kinder lernen auf dem Commodore 64 mit der Schildkröte als Lehrer: Bilder malen · Grafikeffekte erzeugen · Wörter verarbeiten · Prozeduren und Variablen · Umgang mit Begriffen wie: Längenmaß, Winkel, Dreieck, Quadrat.
Best.-Nr. MT 720, DM 34,— (Sfr. 31,30/öS 265,20)

J. Mihalik
35 ausgesuchte Spiele für Ihren Commodore 64
September 1984, 141 Seiten
Programmieren Sie selbst 35 faszinierende Spiele · geschrieben in Commodore 64-BASIC · mit Farbe, Grafiken und Ton · Vorschläge zur Programmabwandlung · für kreative Computerfans, die Ihre Programmierkenntnisse vertiefen wollen!
Best.-Nr. MT 774, DM 24,80 (Sfr. 23,—/öS 193,40)

M.J.Winter: **Lehrspielzeug Computer: C 64/VC-20**
Juli 1984, 139 Seiten
Speziell für Kinder entwickelt führt dieses Buch spielerisch in die Basic-Welt des C 64/VC-20 ein · mit vielen lehrreichen Spielprogrammen und Grafikmöglichkeiten · kleinere Kinder benötigen die Hilfe ihrer sachkundigen Eltern.
Best.-Nr. MT 695, DM 24,80 (Sfr. 23,—/öS 193,40)

Die angegebenen Preise sind Ladenpreise

Sie erhalten M&T-Bücher in guten Buchhandlungen, Computershops und Fachabteilungen der Kaufhäuser.

Sollten Sie diese Bücher ausnahmsweise im Handel nicht beziehen können, so bestellen Sie bitte bei:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Str. 2, 8013 Haar, Telefon: 089/4613-220

Aus dem M&T-Buchverlag

S. Urute
Grafik & Musik auf dem Commodore 64
Oktober 1984, 336 Seiten
68 gut strukturierte und kommentierte Beispielprogramme zur Erzeugung von Sprites und Klangeffekten · Sprite-Tricks · Zeichengrafik · hochauflösende Grafik · Musik nach Noten · spezielle Klangeffekte · Ton und Grafik · für fortgeschrittene Anfänger, die alle Möglichkeiten des C64 ausnutzen wollen.
Best.-Nr. MT 743, DM 38,— (Sfr. 35,—/öS 296,40)

H. L. Schneider
Commodore 64 Listings — Band 1: Spiele
Oktober 1984, 199 Seiten
Mit ausführlicher Dokumentation · Spielanleitung · Variablen für die Änderung der Spiele · vollständige Listings für: Bürger Joe · Nibbler · Zingel Zangel · Universe · Würfelpoker · Maze-Mission · der magische Kreis · Todeskommando Atlantik · Enterprise
Best.-Nr. MT 748, DM 24,80 (Sfr. 23,—/öS 193,40)

E. H. Carlson: **Lerne Basic auf dem VC-20**
August 1984, 320 Seiten
Das neue Basic-Lehrbuch für den Commodore VC-20 · einfach erklärte Basic-Befehle mit Übungen · viele heiße Actionspiele · nützliche Programmiertricks · mit ausführlichem Begriffslexikon · der Renner für junge Computer-Freaks!
Best.-Nr. MT 691, DM 38,— (Sfr. 35,—/öS 296,40)

P. Rädsch:
Programme und Tips für VC-20
1983, 152 Seiten
Nützliche Hilfsprogramme für die Arbeit mit dem VC-20 · kommerzielle Anwendung in der Textverarbeitung, Fakturierung und Lagerverwaltung · Möglichkeiten hochauflösender Grafik über eine Assemblerroutine · unterhaltsame Spielprogramme.
Best.-Nr. MT 513, DM 38,— (Sfr. 35,—/öS 296,40)

M. Hegenbarth/M. Schäfer: **Das VC-20-Buch**
1983, 351 Seiten
Eine Sammlung gut erklärter Programme · viele Spielebeispiele · einfache kommerzielle Anwendungen
Best.-Nr. MT 516 (Buch)
DM 49,— (Sfr. 45,10/öS382,20)
Best.-Nr. MT 581 (Kassette)
DM 19,90 (Sfr. 19,90/öS 179,10)
Best.-Nr. MT 582 (Diskette)
DM 29,90 (Sfr. 29,90/öS 269,10)

N. Hampshire: **Grafik mit dem VC-20**
1984, 202 Seiten
38 vollständige Programme · zahlreiche grafische Darstellungen · alles über hochauflösende Grafik und Multicolor-Modus · praktische Anwendungen und Simulationen von Kunst über Videospiele, Mathematik, Naturwissenschaften bis hin zum kaufmännischen Bereich.
Best.-Nr. MT 644, DM 32,— (Sfr. 29,50/öS 249,60)

R. Zamora/D. Inman et al.: **Basic mit dem VC-20**
1984, 364 Seiten
Eine schrittweise Einführung in das Gebiet von VC-20-Basic · Geräusch- und Musikerzeugung · Drucken von grafischen Schriftzeichen · Erstellen eines lauffähigen VC-20-Programms · Arbeiten mit Zeichenvariablen, einfachen Federvariablen, READ- und DATA-Befehlen · Zeichentricks.
Best.-Nr. MT 649, DM 38,— (Sfr. 35,—/öS 296,40)

D. Laine
Maschinencode-Programme für den ZX Spectrum
1984, 204 Seiten
Nützliche Maschinencode-Programme mit Ihrem ZX Spectrum · Sortierung von Fließkommazahlen · Übernahme von Parametern direkt von einem Basic-Programm · Flußdiagramme · für Profis und solche, die es werden wollen.
Best.-Nr. MT 702, DM 32,— (Sfr. 29,50/öS 249,60)

M. Gavin
Astronomie-Programme für den ZX-Spectrum
September 1984, 268 Seiten
Eine phantastische Reise in die Welt des Kosmos mit Ihrem ZX-Spectrum: Der Julianische Kalender · Die Mondphasen · Eigene Satelliten starten · Kepler's Umlaufbahnen · Die Umlaufbahn Plutos · Interessant nicht nur für Hobby-Astronome.
Best.-Nr. MT 732, DM 32,80 (Sfr. 27,50/öS 232,40)

T. Bridge/R. Carnell: **ZX-Spectrum Abenteuerspiele**
September 1984, 208 Seiten
Die Entstehungsgeschichte der Abenteuerspiele mit repräsentativen Beispielen für jede »Epoche«. Ein Programm speziell für Ihren ZX-Spectrum: »Das Auge des Sternenkriegers«, ein Grafik-Abenteuerspiel, das Sie in Atem hält!
Best.-Nr. MT 712, DM 29,80 (Sfr. 27,50/öS 232,40)

J. Cassidy/P. Katz et al.: **Im Land der Abenteuer**
1984, 146 Seiten
Eine Hilfestellung für zahlreiche Computerspiele · Tod in der Karibik · Transsylvanien · Unternehmen Asteroid · Das geheimnisvolle Haus · Zauberer und Prinzessin · Das goldene Vlies · Zeitzone · Der dunkle Kristall · mit Lösungen.
Best.-Nr. MT 699, DM 29,80 (Sfr. 27,50/öS 232,40)

J. R. Brown: **Basic für Einsteiger**
1984, 239 Seiten
Ein Arbeitsbuch für den absoluten Anfänger · Basic-Anweisungen Schritt für Schritt erklärt und anhand von einfachen Beispielen erläutert · das beliebte Arbeitsmittel für Lehrkräfte und für den interessierten Computerfan.
Best.-Nr. MT 680, DM 32,— (Sfr. 29,50/öS 249,60)

Ch. Langfelder: **Basic ohne Probleme, Bd. 1**
1983, 226 Seiten
Eine Unterweisung in Basic mit CBM-Rechnern (CMB 8032) · Grundlagen des Betriebssystems · Funktionsweise des Interpreters · mathematische Programme · Verarbeitung von Texten und Zeichen · Glossar der wichtigsten Fachbegriffe.
Best.-Nr. MT 480, DM 36,— (Sfr. 33,10/öS 280,80)

Ch. Langfelder: **Basic ohne Probleme, Bd. 2**
1982, 119 Seiten
Für alle CBM 8032-Rechner · ausgewählte Routinen und Programme · drei allgemeine Routinen · fünf kommerziell-technische Anwendungen · zwei Statistikprogramme · zwei Mathematikprogramme · drei Lehr- und Spielprogramme.
Best.-Nr. MT 490, DM 26,— (Sfr. 24,10/öS 202,80)

Die angegebenen Preise sind Ladenpreise

Sie erhalten M&T-Bücher in guten Buchhandlungen, Computershops und Fachabteilungen der Kaufhäuser.

Sollten Sie diese Bücher ausnahmsweise im Handel nicht beziehen können, so bestellen Sie bitte bei:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Str. 2, 8013 Haar, Telefon: 089/4613-220

Aus dem M&T-Buchverlag

H. L. Schneider: **Basic ohne Probleme, Bd. 3**
1983, 256 Seiten
Von der Problemanalyse über Programmentwurf zur Programmiertechnik · Beschreibung allgemeiner, immer wiederkehrender, wichtiger Programmsequenzen · einiges zum Thema Datenverwaltung · mögliche Dateiformen · Zugriffsverfahren auf Dateien, z.B. Binärbäume.
Best.-Nr. MT 500, DM 44,— (Sfr. 40,50/öS 343,20)

H. L. Schneider: **Basic ohne Probleme, Bd. 4**
1983, 428 Seiten
Eine komplette Dateiverwaltung · dateibeschreibende Variablen · Index-sequentielle Schlüsselverwaltung · verkettete Listen · variable Drucklisten · variabler Etikettendruck.
Best.-Nr. MT 514, DM 53,— (Sfr. 48,80/öS 413,40)

D.A. Brain: **Basic-Dialekte im Vergleich**
1984, 105 Seiten
Konvertierung von Apple-, Commodore-und TRS-80-Programmen · Grundlagen der jeweiligen Betriebssysteme · Untersuchung verschiedener Basic-Dialekte · alphabetische Auflistung aller Befehle für die verschiedenen Anpassungsrichtungen.
Best.-Nr. MT 564, DM 32,— (Sfr. 29,50/öS 249,60)

M. Waite/M. Pardee: **Basic-Programmier-Handbuch**
1984, 506 Seiten
Grundlagen · Basic und seine Dialekte · geschäftliche- und wissenschaftliche Anwendungen · Spiele · Lernprogramme · alles über Programmsteuerung · Schleifen und Verzweigungen · Amortisationsprogramm · numerische Funktionen · Stringfunktionen · Variationen mit PEEK und POKE · der Zauberwürfel.
Best.-Nr. MT 658, DM 78,— (Sfr. 71,80/öS 608,40)

J.J. Purdum: **Basic-80 und CP/M**
1983, 296 Seiten
Wo anfangen? · Daten innerhalb eines Programms · Schleifenprogrammierung und Sprünge · Zahlen · sequentielle Datenzugriffsverfahren · Sortieren und Suchen · CP/M und Basic-80 · Fehlersuche · Beispiele.
Best.-Nr. MT 525, DM 48,— (Sfr. 44,20/öS 374,40)

K. Knecht: **Microsoft-Basic**
1984, 204 Seiten
Eine Übersicht der Version 5.0 von Microsoft-Basic · umfangreiche Beispiele für CP/M-Systeme und TRS-80 · Programmieren mit Sprüngen und Schleifen · Umgang mit Zeichenketten und Matrizen · die Arbeitsweise des Editors · Aufbau verschiedener Dateitypen.
Best.-Nr. MT 650, DM 48,— (Sfr. 44,20/öS 374,40)

Otmar Feger: **Ein-Chip-Mikrocomputer-Handbuch**
1983, 389 Seiten
Grundlagen · technisch Entwicklung · Trends · Marktübersicht und Eigenschaften der wesentlichen 4-, 8- und 16-Bit-Typen und ihrer Peripherie · Funktion und Arbeitsweise der wichtigsten mitintegrierten Schaltungskomponenten · Phasen und Durchführung eines Mikrocomputerprojektes.
Best.-Nr. MT 517, DM 58,— (Sfr. 53,40/öS 452,40)

K.H. Heß
Basic-Programme für CBM/VC-20-Computer
1984, 150 Seiten
Programmanwendungen für die Serien CBM 2000, 3000, 4000, 8000 und VC-20 · Analyse verschiedenster Aufgabenstellungen · allgemeingültige Lösungswege in CBM-Basic konvertiert · von Programmanpassungen für VC-20 · für Laien und Profis.
Best.-Nr. MT 501, DM 32,— (Sfr. 29,50/öS 249,60)

H. Stein: **Logo — Grafik, Sprache, Mathematik**
1984, 257 Seiten
Eine Einführung in Logo als Lehr- und Lernsprache · Grafikprozeduren · Zeichenkettenmanipulationen · Probleme der Rekursivität · Sprachbildung und Sprachforschung · Grundlagen der Arithmetik · mit umfassendem Glossar.
Best.-Nr. MT 648, DM 42,— (Sfr. 38,60/öS 327,60)

K. Knecht: **Einführung in Forth**
1984, 218 Seiten
Ausführliche Informationen über die MMS Forth-Version der Computersprache Forth · syntaktische Grundlagen · zahlreiche Programmierbeispiele · der richtige Einstieg in das Programmieren mit Forth.
Best.-Nr. MT 635, DM 58,— (Sfr. 53,40/öS 452,40)

J. Purdum: **Einführung in C**
1984, 304 Seiten
Die grundlegende Charakteristik von C · Operatoren, Variablen und Schleifen · Erstellung eigener Funktionen · Ein-und Ausgabeoperationen in C · Anlegen einer Adreßkartei · Einsatzmöglichkeiten in nahezu allen Bereichen · für Einsteiger und Fortgeschrittene.
Best.-Nr. MT 561, DM 69,— (Sfr. 63,50/öS 538,20)

W. Maaß: **Software-Schnellkurs: CP/M**
1984, 85 Seiten
Was man von CP/M unbedingt kennenlernen muß · die wichtigsten Befehle des 8-Bit-Standard-Betriebssystems und ihre Handhabung · die wichtigsten Befehle für den täglichen Umgang.
Best.-Nr. MT 605, DM 37,— (Sfr. 34,—/öS 288,60)

W. Maaß: **Software-Schnellkurs: CP/M-86**
1984, 93 Seiten
Der tägliche Umgang mit dem Betriebssystem · wie man Dateien anlegt, kopiert, sichert, löscht.
Best.-Nr. MT 615, DM 37,— (Sfr. 34,—/öS 288,60)

W. Pest: **Hardware-Auswahl leicht gemacht**
3. völlig überarbeitete und aktualisierte Ausgabe
1984/85, 485 Seiten
Die wichtigsten Daten von über 200 Personal-Computersystemen sowie die wichtigsten Peripheriegeräte · ausführliche Begriffserläuterungen · Checklisten für den Geräteeinkauf · Trendberichte und Bezugsquellen.
Best.-Nr. MT 350, DM 58,— (Sfr. 53,40/öS 452,40)

Die angegebenen Preise sind Ladenpreise

Sie erhalten M&T-Bücher in guten Buchhandlungen, Computershops und Fachabteilungen der Kaufhäuser.

Sollten Sie diese Bücher ausnahmsweise im Handel nicht beziehen können, so bestellen Sie bitte bei:
Markt & Technik Verlag Aktiengesellschaft, Hans-Pinsel-Str. 2, 8013 Haar, Telefon: 089/4613-220

Lerne BASIC auf dem ATARI

Dieses BASIC-Lehrbuch ist besonders für jugendliche Anfänger gedacht und erlaubt durch seinen Aufbau den Einsatz zum Selbststudium. Kinder und Erwachsene lernen, wie man Actions-Spiele, Brettspiele und Wortspiele programmiert. Hinweise, Erklärungen, Übungen und Wiederholungen werden in einer amüsanten, leicht verständlichen Art präsentiert.

Mit diesem Buch wird das Geheimnis des Computers aufgedeckt und führt in die Grundlagen von ATARI-BASIC ein. Jedes Kapitel besteht aus einem Hinweisblock, in dem die BASIC-Befehle genauer erklärt werden und einem Übungsteil, der auch ohne Vorkenntnisse für jedermann verständlich geschrieben ist. Erklärt werden unter anderem:

- die Befehle PRINT, RUN, INPUT, GOTO, FOR-NEXT,
- Zeichenkettenverarbeitung,
- Cursor-Bewegung, ASCII-Zeichen,
- Unterprogrammtechnik,
- Klangeffekte,
- Fehlermeldungen.

Am Schluß des Buches erklärt ein Begriffslexikon die wichtigsten Fachausdrücke.

Markt&Technik Verlag Aktiengesellschaft
ISBN 3-89090-007-0