Mein LASER Home-Computer

Tips und Tricks
für Einsteiger

SANYO Video Vertrieb, Hamburg

Mein

LASER

Home-Computer

Tips und Tricks
für Einsteiger

Einführung in das Programmieren
mit Schaltbildern

SANYO Video Vertrieb, Hamburg

Impressum:

COPYRIGHT 1984 by SANYO Video Vertrieb, Hamburg.
Verlegerische Betreuung: Roland Löhr, Ahrensburg.
Fotosatz: Telesatz Infora GmbH, Norderstedt bei Hamburg.
Buchumschlag und Skizzen gestaltet von Cathrin Utescher, Hamburg.
Druck und Herstellung: Kuncke Druck GmbH, Ahrensburg.

Der Text dieses Buches wurde auf einem Personal Computer erfaßt. Ein Terminal-Computer übertrug ihn weiter zur Lichtsatzmaschine, deren Computer u.a. auch die Silbentrennung vornahm. Von kleinen Unebenheiten abgesehen, für die wir um Verständnis bitten, hat das Zusammenspiel von Menschen und Computern dabei gut geklappt.

Allen Freunden und Besitzern der Home-Computer LASER und des VZ 200 soll der Einstieg in das Arbeiten mit dem Computer in leicht verständlicher Weise nahegebracht werden.

Für den LASER 110 (4 KByte RAM, monochrom), den LASER 210 (8 KByte RAM, 8 Farben) und den neuen LASER 310 (8 KByte RAM, 8 Farben, QWERTY-Schreibmaschinentastatur) sowie für den VZ 200 (4 KByte RAM, 8 Farben) gibt es mittlerweile ein breit gefächertes Sortiment von Anwendersoftware und eine große Auswahl an Peripherie, wie 64 KByte RAM Erweiterungsmodul, Lightpen, Floppy-Disk-Laufwerk und vieles mehr, was die Anwendung so reizvoll macht.

In diesem Buch mußten wir uns darauf beschränken, den BASIC-Teil zu beschreiben und zu erläutern. Die Reihe wird jedoch fortgesetzt mit einer Ausgabe, die sich intensiv mit der Technik der Geräte auseinandersetzt, die Möglichkeiten der Programmierung in Maschinensprache behandelt und eine Vielzahl von speziellen Schaltbildern für den Fortgeschrittenen enthält.

Nun aber 'ran an die Arbeit und viel Spaß mit Ihrem Home-Computer!

Hamburg, im Juni 1984

Inhaltsverzeichnis

1 Die Vorfahren des Computers

Ist es nicht aufregend? Sie haben gerade Ihren neuen Heim-Computer nach Hause gebracht. Und das erste, was Sie wissen wollen, ist, wie man ihn gebraucht. Nun, dazu kommen wir gleich. Zunächst sollen Sie wissen, was Ihre neue Maschine eigentlich darstellt.

Jeder weiß, daß Computer Elektronengehirne genannt werden, stimmt's? Nein falsch!
Eigentlich haben sie überhaupt kein Gehirn.

Vergessen Sie das, was Sie in Kinofilmen darüber gesehen haben, erstaunliche Maschinen, die intelligente Gespräche führen, Ideen haben oder Pläne zur Übernahme der Weltherrschaft schmieden. Ein Computer ist eher wie ein dressierter Papagei. Sie können ihm beibringen, Tricks zu machen oder Befehlen zu gehorchen und sogar mit ihm zu "reden". Aber er kann ganz gewiß nicht so denken, wie Sie und ich! Das ist also das, was ein Computer nicht ist. Zur Erklärung, was ein Computer ist, betrachten wir einmal die Vorgänger Ihres Computers.

Seit Beginn der Geschichte hat der Mensch nach Mitteln und Wegen gesucht, die ihm das Leben einfacher machen. Er entdeckte, daß es Grenzen seiner körperlichen Leistungsfähigkeit gab. Also begann er Wege zu suchen. die es im ermöglichten, seine physischen Fähigkeiten zu erweitern. Das geschah mit Werkzeugen.

Die Menschheit verläßt sich heutzutage auf viele verschiedene Werkzeuge. Das Kraftfahrzeug beispielsweise ist ein Werkzeug, welches uns eine viel schnellere Fortbewegung erlaubt, als es mit unseren Beinenmöglich ist.

Das Telefon ist ein Werkzeug, das es uns erlaubt, mit jemandem zu sprechen, der viel weiter weg ist, als unsere Stimme hörbar ist.

Ein Mikroskop ist ein Werkzeug, unter dessen Zuhilfenahme wir Dinge betrachten können, die mit dem bloßen Auge nicht erkennbar sind.

Der Computer schließlich ist ein Werkzeug, mit dessen Hilfe wir uns viel merken und Berechnungen viel schneller durchführen können, als es unser Gehirn alleine vermag. In seinem Falle begann es vor tausenden von Jahren mit dem chinesischen Abacus.

Was wir eigentlich sagen wollen ist: Vor Ihrem Computer brauchen Sie keine Angst zu haben.

Sie müssen sich nur in Erinnerung rufen, daß es sich um ein Werkzeug zur Erweiterung Ihres geistigen Fähigkeiten handelt. Ohne Ihre Hilfe kann er

nicht denken; er hat keinen eigenständigen Verstand; und er wird Sie sicherlich nicht auslachen, wenn Sie etwas falsch machen!

Apropos erinnern, hier ist noch ein Punkt, den sie sich merken sollten: Egal was Sie in Ihren Computer eingeben, Sie können ihn nicht verletzen. Sie können damit herumspielen, soviel Unsinn hineintippen, wie Sie wollen. Sie werden dadurch absolut nie in eine ausweglose Situation geraten. Denn wenn es zum Schlimmsten kommt, können sie ganz einfach das Gerät ausschalten und von neuem beginnen. Deswegen experimentieren Sie soviel Sie wollen. Warum sich darüber Gedanken machen, was passieren könnte, wenn Sie diese, statt die andere Taste drücken. Versuchen Sie's und schauen Sie, was dabei herauskommt?!

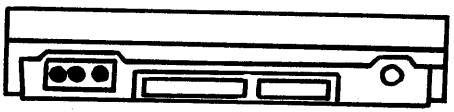
2 Inbetriebnahme des Rechners

Um Ihren Computer anschließen zu können, müssen sie Ihr Fernsehgerät ausschalten. Auf der Rückseite des TV-Gerätes werden Sie den Antennenanschluß finden. Über diesen Anschluß werden Bilder vom Fernsehsender empfangen.

Aber im Augenblick wollen wir, daß Ihr TV von Ihrem Computer Informationenempfängt!

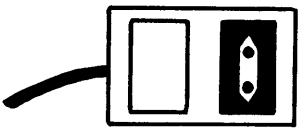
Also: Entfernen Sie das Antennenkabel aus der Anschlußbuchse. Stattdessen verwenden sie das dicke Kabel, das in der Verpackung Ihres Computers liegt. Das dickste Ende dieses Kabels wird in die Anschlußbuchse Ihres TV-Gerätes gesteckt.

Soviel zu dem einen Ende des Kabels. Das andere Ende paßt in die äußerste rechte Buchse auf der Rückseite Ihres Computers. (Falls Sie genau hinsehen, erkennen Sie direkt unterhalb dieser Buchse die Buchstaben "TV".)

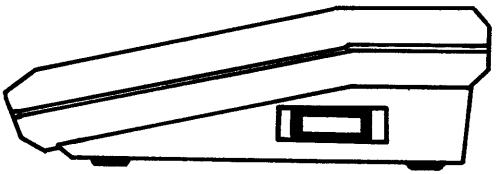


In der Computer-Verpackung befindet sich auch ein schwarzes Gerät mit einem dünnen Kabel. Dieses Gerät nennt man Stromversorgungsteil (Trafo), und zwar weil er den Netzstrom eines gewöhnlichen Hausanschlußes (220v, 50 Hz) in eine für Ihren Computer geeignete Stromart umwandelt. Zuerst stecken Sie den üblichen Netzstecker in die Steckdose. Gut, dann werden Sie am Endes des langen, dünnen Kabels einen kleinen runden

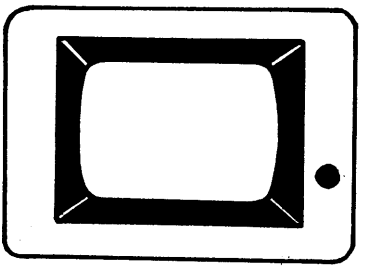
Stecker sehen. Stecken Sie diesen in die ganz linke Buchse auf der Rückseite Ihres Computers. (Unterhalb dieser Anschlußstelle am Computer steht "DC9V".



Der Computer wird eingeschaltet, indem sie den Kippschalter betätigen, der sich an der rechten Seite befindet. (Falls alles in Ordnung ist, wird das kleine rote"Power"-Lämpchen oben auf dem Computer aufleuchten.)



Schalten Sie Ihr TV-Gerät wieder ein und wählen Sie sodann Kanal 1.
Das war's, schon fertig!
Jetzt ist alles betriebsbereit. Und schauen Sie. Wenn Sie alles richtig gemacht haben, wird Ihr Computer schon begonnen haben, mit Ihnen zu sprechen. Er sagt Ihnen, daß er "READY" (bereit) ist, auf alles zu hören, was Sie ihm sagen. Der Cursor, das blinkende Viereck auf der dritten Zeile, sollte ebenfalls sichtbar sein. Der Cursor hat die Funktion, Ihnen zu sagen, wo sich die nächste Schreibstelle gerade auf dem Bildschirm befindet.



Falls Sie keine READY-Nachricht von Ihrem Computer erhalten, überprüfen Sie nocheinmal die vorgenommenen Anschlüsse. Möglicherweise liegt der Fehler nur an einem losen Kontakt oder an einem falschen Anschluß.

Ein großer Unterschied zwischen dem Computer und dem Menschen

Wenn Menschen miteinander sprechen, benötigen sie zwei Dinge:
Ohren um zu hören, was gesagt wird, und einen
Mund, um zu antworten, Fragen zu stellen u.s.w.

Betrachten Sie den Computer. Sicher hat er weder Ohren noch einen Mund, aber Computer sind dennoch in der Lage, "Gespräche" zu führen! Die Tastatur des Computers sind gewissermaßen seine "Ohren". Wenn sie Fragen oder Anweisungen eingeben, "hört" er Sie. Und wenn der Computer mit Ihnen "sprechen" will, bedient er sich des Bildschirms (in diesem Fall Ihr TV-Gerät.) Können Sie sich vorstellen, was das bedeutet? Richtig - um Gespräche mit einem Computer führen zu können, müssen Sie Ihre Sinne etwas anders als sonst zur Anwendung bringen.

Mit anderen Worten, wenn sie mit einem Computer sprechen, werden Ihre Finger zum Mund und Ihre Augen zu Ohren!

"Aber wie verwende ich die Tastatur?"

Sollten Sie zu diesem Zeitpunkt von der Vielzahl einzelner Tasten verwirrt sein, so stellt das kein Problem das. Erstens wollen wir den ganzen Tastenbereich "die Tastatur" nennen. Zweitens wollen wir erklären wie einfach Tastaturen zu bedienen sind.

Es gibt viele Buchstaben, Ziffern, Symbole, Kommandos u.s.w., die Ihr Computer versteht. Natürlich hätten wir jedem Zeichen eine Taste zuordnen können. Aber dies hätte die Tastatur des Computers ziemlich groß und unübersichtlich werden lassen!

Es war besser, die einzelnen Tasten mit mehreren Funktionen auszustatten, die sogenannten *Multi-Funktionen*. Einige dieser Multi-Funktionen können bis zu vier verschiedene Zweckanwendungen haben!

Um Ihnen zu zeigen, wie sich dies auswirkt, wählen wir eine beliebige Taste, z.B. die "P"-Taste. Das sieht auf der Tastatur folgendermaßen aus:

Wenn Sie den Buchstaben "P" schreiben wollen, dann drücken Sie einfach die Taste. Wenn Sie die eckige Klammer haben wollen, müssen Sie zuerst die "SHIFT"-Taste suchen. (Sie befindet sich in der linken unteren Ecke der Tastatur.) Haben Sie's? Halten Sie nun die SHIFT-Taste gedrückt, bis Sie die P-Taste betätigt haben.

Oberhalb und unterhalb der "P"-Taste sehen Sie zwei weitere Symbole.

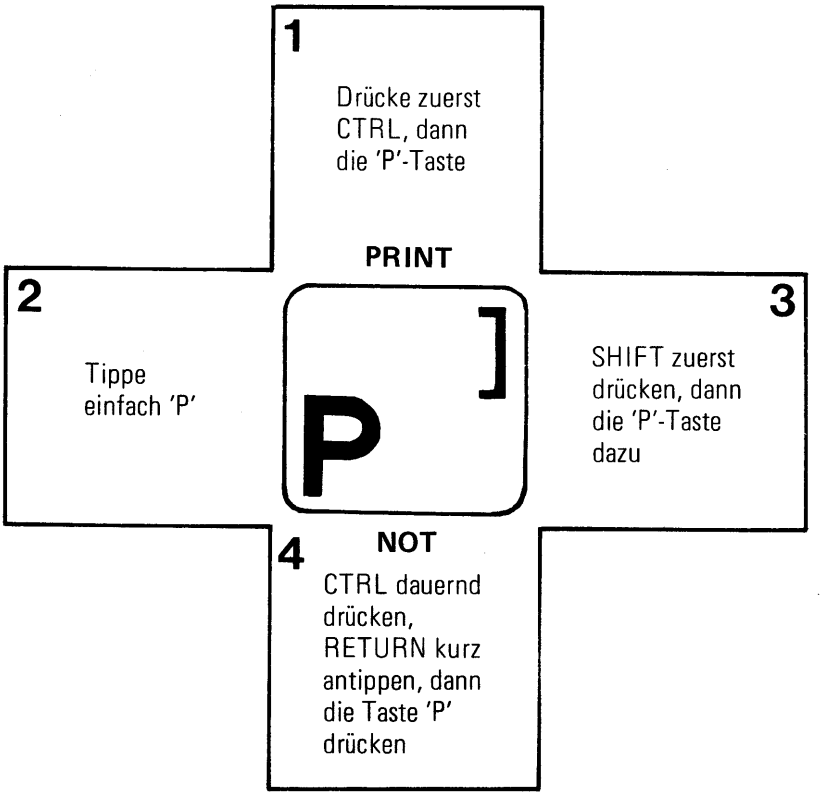
Um "PRINT" zu erhalten, halten Sie die "CTRL"-Taste (sie befindet sich gleich oberhalb von SHIFT) gedrückt, während Sie die P-Taste betätigen.

Wenn "NOT" gewollt ist, muß zweierlei vorgenommen werden:

1. CTRL ist solange gedrückt zu halten, bis RETURN gedrückt und losgelassen wurde.

Danach die P-Taste betätigen.

Die eben genannten Regeln sind bei jeder Multi-Funktionstaste zu beachten.



1. Anwendung: Während der Betätigung die Taste CTRL gedrückt halten für PRINT
2. Anwendung: Lediglich die Taste drücken für P
3. Anwendung: SHIFT bis zur Betätigung der Taste gedrückt halten
4. Anwendung: CTRL gedrückt halten, Return betätigen, die Taste betätigen für NOT

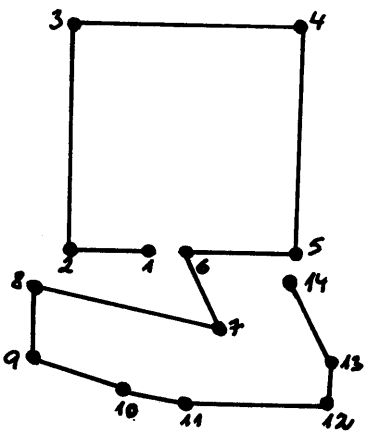
Übrigens, die eigenartigen schachtelförmigen Zeichen auf einigen Tasten werden Graphik-Zeichen genannt. Sie kommen später bei der graphischen Darstellung auf dem Bildschirm zur Sprache.

3 Programmierung kurz gefaßt!

Wir sagen: "Computer-Programmierung"
Sie denken: "HILFE!" Nun, das ist eine ganz verständliche Reaktion auf etwas, worüber man noch nichts weiß.

Aber Moment, sehen sie nicht gleich schwarz! Es ist bei weitem nicht so kompliziert, wie Sie dachten. Im Gegenteil, es ist in der Tat sehr einfach zu verstehen. Besonders, wenn Sie es sich so vorstellen:

Nehmen wir an, sie wollten ein *Zahlen-Verbindungs-Spiel* lösen, wie man es in Illustrierten findet (jeder weiß, wie einfach das ist!) Man beginnt natürlich beim ersten Schritt (das ist der Punkt mit der Ziffer 1). Dann zieht man den Strich bis zum Punkt mit der Ziffer 2, von dort zum Punkt 3 u.s.w., bis man bei der letzten Ziffer ankommt.



Daran ist eigentlich nichts schwer, oder? Sie haben dabei lediglich auf eine bereits vorgegebene Schreibfolge zu achten. Dabei ist ein Bild entstanden, obwohl Sie vorher nicht wußten, wie das Bild am Ende aussehen würde. Mit anderen Worten: *Sie haben ein Programm befolgt!*
Also besteht Ihre Tätigkeit als Programmierer lediglich darin, eine Schrittabfolge für Ihren Computer vorab festzulegen.

Hierbei gibt es etwas wichtiges zu beachten: Was wohl würde passieren, wenn die Punkte des Zahlen-Spiels falsch beziffert wären? Oder wenn

einige Punkte überhaupt fehlen würden? Genau, es käme ein Gebilde dabei heraus, das den falschen oder keinerlei Sinn ergäbe!

Damit sich aus dem Programm für den Computer ein Sinn ergibt, *müssen Sie darauf achten, daß die einzelnen Schritte in der richtigen Reihenfolge eingegeben sind.*So, und nun wissen Sie schon wesentlich mehr über die "mysteriöse" Computer-Programmierung. (Die Sache ist doch eigentlich gar nicht so wild, oder?)

In nur wenigen Minuten haben wir die Grundregeln gelernt, wie man den Computer dazu bringt, das zu tun, was wir wollen:

* Er benötigt die einzelnen *Schritte*, die er befolgen muß, um die Aufgabe zu erledigen
* und die *Reihenfolge* der abzuarbeitenden Schritte.

So weit so gut! Aber es gibt noch etwas, das man wissen muß, um dem Computer zu sagen, was er tun soll, nämlich, wie man ihn *anzusprechen* hat!

Die Überwindung der Sprach-Hürde

Erinnern Sie sich an die Ausführungen des letzten Kapitels über die Unterschiede zwischen Computer und Mensch. Hier ist noch ein Unterschied: Computer sind nicht intelligent genug, unsere komplexe Sprache zu verstehen! *Zur Überwindung dieser "Sprach-Hürde" verwenden Menschen eine spezielle Sprache namens "BASIC".*

Sie nehmen wahrscheinlich an, daß eine Sache, die *"BASIC"* heißt, ziemlich *einfach* sein muß. Nun, so verhält es sich wirklich. Man kann diese Sprache sehr gut mit Englisch vergleichen, nur verfügt sie über einen wesentlich geringeren Wortschatz.

Einem Mitmenschen kann man die verschiedensten Dinge auftragen. Z.B.: "Mach' mir etwas zu essen!" oder "Klettere auf den Fahnenmast!" Wir benötigen natürlich viele verschiedene Worte, um solche Anweisungen zu geben. Demgegenüber ist ein Computer nur dazu fähig, relativ wenige Tätigkeiten zu verrichten. Deswegen kann er auch nur wenige Worte verstehen. (Versuchen Sie einmal Ihrem Computer zu sagen, er solle Ihnen etwas zum Essen zubereiten, es wird Sie nicht verstehen).

Und nun etwas Computer-Psychologie

Wie bereits erwähnt, ist ein Computer kein elektronisches Gehirn, sondern er hat überhaupt kein Gehirn. Der Computer ist eigentlich nur dazu fähig, sich zu *erinnern* (mit Hilfe des Speichers) und *Befehle auszuführen* (mit seinem Rechner).

Menschen sind jedoch mit einem Gehirn ausgestattet. Es erlaubt uns zu denken, Ideen zu entwickeln, Problemlösungen auszuarbeiten, Initiative zu zeigen und eine Unmenge anderer, intelligenter Dinge zu tun. Wenn Sie also mit dem Computer ein "Gespräch" führen wollen, dann ist es wichtig, daß Sie sich dieser Unterschiede immer bewußt sind. Denn der Computer wird freudig jede ihm übertragene Aufgabe genau durchführen, und zwar sehr, sehr schnell. *Aber er wird ohne Ihre Veranlassung nichts tun!*

Dort könnten Sie Schwierigkeiten begegnen, denn schließlich sind die meisten Menschen daran gewöhnt, mit anderen zu sprechen, und dabei vergißt man sehr leicht, wie dumm der Computer wirklich ist.

Ein Beispiel: Nehmen wir an, Sie würden einen Freund 2 und 2 addieren lassen. Ihr Freund würde einen Moment überlegen, und dann würde er antworten. Warum? Weil er automatisch annehmen würde, daß Sie die Antwort wissen wollen. Zum Unterschied: Fragen Sie Ihren Computer, so würden sie den ganzen Tag auf eine Antwort warten, ohne eine zu erhalten. *Ihr Computer wird Ihnen so lange keine Antwort geben, bis Sie ihm sagen, daß er das tun soll! Dies liegt daran, daß er anders, als Ihr Freund nicht in der Lage ist, Annahmen zu machen.* Ihr Computer wird Sie nicht verstehen, solange Sie sich nicht klar ausdrücken.

Zweites Beispiel: Nehmen wir an, Sie sagen zu Ihrem Freund: "bitte geh' und hol' mir meine Meßlatte. Sie ist in einer der Schubladen dort drüben." Wahrscheinlich würde Ihr Freund begreifen, daß es das Lineal ist, was Sie suchen. Und er würde in den Schubladen danach suchen, bis er es fände.

Aber wenn das Gehirn Ihres Freundes wie ein Computer arbeiten würde, wäre er gar nicht in der Lage herauszufinden, was Sie von ihm wollen! Stattdessen würde er etwa sagen: "Ich verstehe nicht! Was ist eine Meßlatte? Wie komme ich zu den Schubladen? In welcher Schublade soll ich suchen? Oder er würde dasitzen und Sie mit leerem Blick anschauen!

Beachten Sie eines: Computer sind zwar nicht so intelligent wie wir, aber sie machen sehr selten Fehler. Was ist also zu tun, wenn der Computer ein Programm nicht zu dem Resultat führt, das Sie sich wünschen? Oder sich sogar weigert, überhaupt etwas zu tun? Nun, eines sollten Sie nicht tun, nämlich, ihn mit wüsten Beschimpfungen eindecken. Nicht weil er etwa empfindlich ist, sondern weil er höchst wahrscheinlich nicht schuld ist.

> Die Wahrscheinlichkeit, daß Sie - der Programmierer - gepatzt haben, steht 1 : 1 000 000 gegen sie, denn der Computer arbeitet nahezu fehlerfrei.

Sie müssen also streng darauf achten, daß die gegebenen Anweisungen

bis ins Detail ausgearbeitet sind. Denn wenn auch nur eine Kleinigkeit versäumt wurde, wird Ihr Computer nie herausfinden was es war.

4 Unser erstes Programm

Da Sie jetzt schon ziemlich viel über Computer wissen, wollen wir jetzt etwas programmieren. Die Programmierung Ihres Computer ist so einfach, wie "Hallo" zu sagen. Und um das zu beweisen, wird "HALLO" das erste Wort sein, das wir dem Computer beibringen.

Die Verwendung einer Zeilenbezifferung

Erinnern Sie sich an das Zahlenverbindungs-Spiel. Jedem Punkt war eine Ziffer zugeordnet. Diese Ziffern wiesen die Reihenfolge der Schritte an. Ihr Computer muß wissen, in welcher Reihenfolge er sich die einzelnen Programmschritte vorzunehmen hat. Also geben wir ihm auch für jeden Schritt (oder Zeile) eine Ziffer an.

Man könnte natürlich einfacherweise die Ziffern 1, 2, 3, u.s.w. verwenden, aber ein guter Plan ist das nicht. Denn wenn längere Programme erstellt werden, kann es durchaus vorkommen, daß die eine oder andere Zeile aus Versehen ausgelassen wird, (Das passiert jedem einmal). Mit den Ziffern 1, 2, 3, u.s.w. hat man aber keine Möglichkeit, die vergessenen Zeilen später einzufügen.

Da ist es besser, die Anfangsbezifferung mit größeren Abständen zu versehen, z.B.: 10, 20, 30 u.s.w.. Dadurch erhält man neun nicht belegte Zwischenzeilen (für alle Fälle).

OK, Computer sage "HALLO"!

Geben Sie die Zeilen, die Sie nachfolgend sehen, genauso ein, wie Sie diese vorfinden. (Wie bitte? Sagten Sie, Sie wissen nicht wie? Also gut, für den Moment genügt es, wenn Sie sich die Zeilen vor Augen halten. Wir gehen Zeile für Zeile, Taste um Taste vor, damit Sie sehen, wie einfach es ist.

10 PRINT "HALLO" RETURN-Taste drücken
20 GOTO 10 RETURN-Taste drücken

So, das erste, was in der obersten Zeile stehen soll, ist die Ziffer 10 (das ist, weiter oben erklärt, die Zeilennummer). Drücken sie einfach die Taste mit der 1 darauf und dann die Taste mit der 0 darauf. Nun sehen Sie die 10 auf dem Bildschirm. Und der kleine blinkende Cursor befindet sich unmittelbar hinter der 0. Er markiert die Stelle, an der das nächste Zeichen erscheinen wird.

Übrigens: Haben Sie nach jedem Eintasten einen Piepton gehört? Das ist die Meldung Ihres Computers, die besagt: "Ja, ich habe den Tastendruck angenommen."

Was kommt nun als nächstes in unsere erste Programmzeile? Das Wort PRINT, nicht wahr? Also nein,nicht ganz. Zunächst muß die Eingabe einer Leerstelle erfolgen. *Beim Computer werden Leerstellen mit einer entsprechenden Sondertaste, nämlich der Space-Taste eingegeben.* (Sehen Sie diese Taste? Sie befindet sich in der unteren rechten Ecke der Tastatur.) Versuchen Sie's jetzt. Drücken Sie die Taste einmal und lassen Sie sie wieder los. Sie werden erkennen, daß der Cursor um eine Stelle nach rechts gesprungen ist.

Nun ist PRINT an der Reihe. Zur Eingabe von PRINT gibt es verschiedene Möglichkeiten. Mit der gewöhnlichen Methode wird Buchstabe für Buchstabe eingetippt. Aber es gibt eine *Abkürzung*, die Sie vorziehen könnten, um sich die Tipparbeit zu ersparen, nämlich folgende: auf der P-Taste sehen Sie das Wort PRINT. Und Sie wissen bereits, wie man die Abkürzung eintippen kann. (Wie bitte? Das haben Sie schon vergessen? Wenn ja, dann gehen Sie zurück zum Ende des 2. Kapitels.) Wir probieren die Abkürzung und halten die CTRL-Taste gedrückt, während wir P betätigen.

Also, das ist doch recht praktisch. So schnell wie Ihr Computer piepsen konnte, erschien das Wort PRINT auch schon im Bildschirm. Es sieht vielleicht aus wie Zauberei, aber es stellt nur eine Vereinfachung dar. Ihr Computer bietet Ihnen eine Vielzahl solcher Abkürzungsmöglichkeiten. Bei fast allen Tasten (darüber oder darunter) Ihrer Tastatur finden Sie Worte, die recht nützlich sind.

Um PRINT vom nächstfolgenden Wort zu trennen, verwenden wir wieder die Space-Taste. Ist der Cursor um eine Stelle nach rechts gesprungen? Gut! Ihnen fällt bestimmt auf, daß "Hallo" innerhalb von *Anführungszeichen* (" ") steht. Das entsprechende Zeichen befindet sich in der oberen Ecke der 2-Taste. Zur Eingabe halten Sie SHIFT gedrückt, während Sie die 2 betätigen. Haben Sie's? Gut! Und jetzt tippen Sie Hallo ein, dann nochmal ein Anführungszeichen zum Schluß. Fertig!

Zu guter Letzt steht ein RETURN am Ende der ersten Programmzeile. Dies ist wirklich das Einfachste. Betätigen Sie die entsprechende Taste und beobachten Sie, was passiert. Auf dem Bildschirm erscheint nichts Neues, aber der Cursor ist an den Anfang der nächsten Zeile gesprungen.

Da wir Sie von der Einfachheit der Handhabung nunmehr überzeugt haben, können Sie die nächste Zeile in Angriff nehmen, und zwar ohne unsere Hilfe:

Tippen Sie 20 dann GOTO, gleich oberhalb der G-Taste, dann 10 und schließlich die RETURN-Taste.

Wissen Sie eigentlich, was Sie eben getan haben? Es ist wirklich aufregend. Sie haben gerade mit Ihrem Computer gesprochen und zwar mittels Tastatur. Und Sie haben zwei besondere Worte aus der BASIC-Sprache verwendet, die in Kapitel 3 erwähnt wurden.

Wir übersetzen jetzt das bisher Programmierte und zeigen Ihnen genau, was Sie veranlaßt haben.

> PRINT bedeutet: "Schreib' das auf den Bildschirm." (Vergessen Sie nie die Anführungszeichen, die das zu Druckende beidseitig umfassen müssen.)
> GOTO bedeutet: "Gehe weiter zu der Zeile, die ich gleich angeben werde." Dieser Befehl wird immer von einer Zeileziffer gefolgt, denn Sie müssen dem Computer immer genau sagen, wo er die Bearbeitung fortsetzen soll. (In diesem Fall wollten wir, daß er zurück nach Zeile 10 geht. Also sagten wir: "GOTO 10". Alles klar?
> Die Betätigung der RETURN-Taste sagt dem Computer einfach: Das ist alles was in diese Zeile gehört. Jetzt sehe dir die nächste an. Diese Taste schließt fast jede Zeile ab, die Sie in Ihrem Computer eintippen.

Wir beglückwünschen Sie! Ihr erstes Programm sitzt im Augenblick im Speicher Ihres Computers. Fantastisch, mögen Sie sagen. Aber warum passiert nichts? Nun, es ist gleich soweit...sobald wir Ihnen noch einige BASIC-Worte beigebracht haben.

> Die Worte, die Sie gleich lernen werden, nennt man *direkte Kommandos*. Sie benötigen keine Zeilennummer (obwohl sie auch dann vom Computer ausgeführt würden). Wenn Ihr Computer ein direktes Kommando liest, weiß er, daß Sie eine bestimmte Ausführung erwarten, und zwar sofort.

Das erste direkte Kommando ist LIST und bedeutet: "Zeige mir das Programm genauso, wie ich es eingetippt habe." Wenn Sie LIST tippen, wird das ganze Programm im Bildschirm erscheinen. Versuchen Sie das jetzt. (Ist nichts passiert? Wenn nicht, dann wahrscheinlich weil Sie es versäumt haben, RETURN einzugeben. Und bis Sie das tun, wird Ihr Computer denken, daß Sie mit Ihrer Kommando-Eingabe noch nicht fertig sind.)

Das zweite direkte Kommando ist RUN und bedeutet: "Führe die Programmanweisung aus." Wenn Sie Ihrem Computer zunächst RUN und dann RETURN eingeben, wird er das ihm eingegebene Programm ausführen.

Sind Sie bereit zu sehen, was Ihr kleines Programm tun wird? O.K. Dann geben Sie RUN und RETURN ein und beobachten sie was passiert. Toll! Wenn Sie es sich nicht schon denken konnten was passieren würde, dann werden Sie von den vielen auf dem Bildschirm erscheinenden "HALLOS" ganz schön überrascht worden sein. Aber, es ist eigentlich gar nicht so überraschend, wenn man sich ansieht, wie das Programm funktioniert:

Sie erinnern sich, daß der Computer mit der niedrigsten Zeilenziffer beginnt und normalerweise die nächstfolgende abarbeitet, dann, wenn er nicht auf ein GOTO trifft.

> Der Computer nahm sich also zuerst Zeile 10 vor. Diese Zeile veranlaßte ihn, "HALLO" zu drucken. Dann ging er weiter zur nächsten Zeile (20). Zeile 20 sagt dem Computer, daß er zurück zu Zeile 10 gehen soll. Und das läßt das Ganze wieder von vorne beginnen.

Diesen schlauen Trick nennt man *Looping" (Programmschleife)*. Warum? Weil ein "Loop" (Schleife) endlos ist, was bei dieser Art Programm auch der Fall ist. Das ist ungefähr so, als würden Sie Ihren Computer auf eine nicht endende Karussell-Fahrt schicken.

> Falls Ihr Computer sich weigerte, HALLO zu sagen, so nehmen Sie es ihm nicht übel. Tippen Sie LIST noch einmal ein (Sie haben nicht schon vergessen, was das bedeutet, oder?) und überprüfen Sie, ob das Programm, das sich auf dem Bildschirm befindet, auch genau mit dem in diesem Buch übereinstimmt. Möglicherweise entdecken Sie einen Fehler. Überprüfenn Sie auch, ob alle Verbindungskabel richtig sitzen.

"Aber wie halte ich ihn wieder an?"

Während wir uns zwischenzeitlich beschäftigt haben, hat sich der Computer zu einem richtigen "HALLO-dri" entwickelt (er hört nicht mehr auf, "HALLO" zu drucken). Und das so lange nicht, bis wir das Programm "abbremsen" oder abbrechen. Hier ist noch ein BASIC-Kommando, das genau dies veranlassen wird. Es wird "BREAK" (abbrechen) genannt und es bedeutet: "Bleib' stehen, wo du gerade bist." Die BREAK-Taste finden Sie in der obersten Tastenreihe ganz rechts. Zur Anwendung betätigen Sie die BREAK-Taste während Sie die CTRL-Taste gedrückt halten. Und nun kommt eine Überraschung: Nach BREAK muß RETURN nicht getippt werden. Kommandos, die zur Eingabe RETURN nicht benötigen, sind sehr dünn gesät (äußerst dünn, denn BREAK ist der einzige Befehl, bei dem RETURN nicht notwendig ist!).

Mit BREAK brechen Sie garantiert ab. Auf Ihrem Bildschirm sollten Sie die letzten HALLOS vorfinden, gefolgt von: BREAK IN LINE 10 (oder BREAK IN LINE 20, das kommt darauf an, wo der Computer gerade stand, als Sie "abbremsten").

Wenn Sie an den HALLOs Freude gehabt haben und noch weitere sehen wollen, können Sie dem Computer CONT und RETURN eingeben. CONT ist eine Abkürzung für "continue" (fortsetzen) und weist den Computer an, dort fortzufahren, wo Sie das Programm unterbrachen.

Natürlich können Sie Ihrem Programm leicht ein Ende setzen, indem Sie eine andere BASIC-Anweisung eingeben, und zwar END. Ein Programm, mit dem Sie Ihren Computer nur ein einziges Mal "HALLO" drucken lassen können, sieht so aus:

```
10 PRINT "HALLO"
20 END
```

END bedeutet: "Dies ist das Ende des Programms".

Sind Sie bereit weiterzugehen? Dann lernen wir jetzt zwei neue BASIC-Kommandos, die ebenfalls die HALLOs abbrechen würden: CLS und NEW

CLS bedeutet: "Bildschirm löschen"

Versuchen Sie es jetzt

```
CLS RETURN
```

Sehen Sie? Der Bildschirm ist glöscht und der Computer sagt Ihnen, daß er wieder READY (bereit) ist.

NEW bedeutet: "Gespeichertes Programm löschen".

Dieses Kommando löscht den Bildschirm und den Speicher. Das ist vergleichbar mit sofortigem Gedächtnisverlust.

Also tippen sie einmal

```
NEW RETURN
```

Im Bildschirm wird sich nichts ändern, aber wenn Sie jetzt versuchen RUN anzuweisen, wird der Computer Ihnen nur sagen, daß er READY ist.

BASIC-Kommandos, die wir bis jetzt gelernt haben:

PRINT"...." Zeige das an, was innerhalb der "...." ist, auf dem Bildschirm.

GOTO gehe bis zu der Zeile (vergessen Sie nie eine Zeilennummer unmittelbar nach diesem Kommando anzugeben!)

END Ende des Programms

LIST Zeige mein Programm auf dem Bildschirm an

RUN Führe das Programm aus

BREAK Unterbreche das Programm

CONT Setze das Programm fort

CLS Lösche den Bildschirm

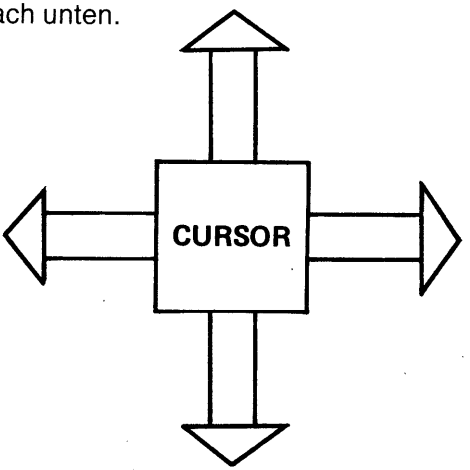
NEW Lösche das Programm

Und vergessen Sie nie: Tippen Sie am Ende jeder Zeile die RETURN-Taste, damit der Computer weiß, daß Sie ein Kommando abschließen.

Wie man Tipp-Fehler berichtigt: Das Editieren

Egal,wie vorsichtig Sie beim Tippen waren, Sie werden früher oder später einen Tippfehler machen, indem sie die falsche Taste tippen. Verzweifeln Sie aber nicht! Patzer sind schnell passiert. Sie wären kein Mensch, würden Sie keine Fehler machen. Und außerdem ist Ihr Computer derart clever, daß es genauso einfach ist, Fehler auszubessern, wie sie zu machen!

Wenn Sie die letzten vier Tasten in der untersten Reihe der Tastatur betrachten, was sehen Sie? Auf diese Tasten kommt es an. Oberhalb jeder ist ein Pfeil abgebildet. Ein Pfeil zeigt nach links, einer nach rechts, einer nach oben und einer nach unten.



Sie können diese Pfeile dazu verwenden, den Cursor am Bildschirm zu steuern. Die Idee an der Geschichte ist, den Cursor direkt auf den Fehler zu plazieren.

"Ja, aber wie funktionieren die Pfeiltasten?"

Wenn Sie die CTRL-Taste gedrückt halten, wird jede einzelne Betätigung der Pfeil-Tasten den Cursor des Computers um eine Stelle in Richtung des Pfeils bewegen. (Nach oben, unten, links, rechts, es kommt darauf an, welchen Pfeil Sie wählen.)

Wenn Sie nicht damit zufrieden sind, jeweils nur um eine Stelle weiterzurücken, dann versuchen Sie einmal, Ihren Finger länger auf der Pfeil-Taste zu belassen. Der Cursor wird sich (piepsend) über den Bildschirm bewegen, bis Sie die Taste wieder loslassen.

Eine kleine Warnung: Falls Sie diesen Trick versuchen, könnten Sie über das Ziel hinausschießen und damit die angepeilte Stelle verfehlen. Aber keine Sorge, verwenden Sie einfach die entgegengesetzte Pfeil-Taste, um den Cursor wieder auf Ihr Ziel zu bringen. (Den Cursor rund um den Bildschirm zu steuern ist ungefähr wie Autoscooter zu fahren, Sie können fahren, wie Sie wollen.)

Eine Warnung: Sie dürfen nicht vergessen, die CTRL-Taste während der ganzen Zeit, die Sie die Pfeil-Taste verwenden, gedrückt zu halten. Denn wenn Sie es nicht tun, wird der Cursor auf dem Bildschirm alles komplett durcheinander bringen.

Es gibt verschiedene Möglichkeiten Patzer zu machen, aber glücklicherweiseauch verschiedene Möglichkeiten, diese zu beheben.

"Ich habe einen Buchstaben zuviel eingetippt!"

Nehmen wir einmal an, daß Sie folgendermaßen die Zeile 10 getippt hätten:

```
10 PRINT "HAALLO"
```

Das sieht etwas komisch aus! Das überflüssige "A" müssen wir löschen. Einen Buchstaben auszuradieren (oder ein Leerzeichen, eine Zahl oder irgend ein Zeichen) ist einfach, denn dafür gibt es ein besonderes Kommando, nämlich: RUBOUT. Dieses Kommando finden Sie oberhalb der ";"-Taste.

Die Fehler-Behebung:

Verwenden Sie die Pfeile, um den Cursor auf das überflüssige "A" zu setzen (es spielt keine Rolle, auf welches). Jetzt halten Sie die CTRL-Taste gedrückt und betätigen RUBOUT. Der Cursor "verschluckt" den darunter liegenden Buchstaben.

"Ich habe einen Buchstaben ausgelassen"

Auch gibt es ein Kommando, mit dem man ein vergessenes Zeichen zwi-

schen zwei anderen einfügen kann. Dieses Kommando wird: INSERT genannt. Sie finden es oberhalb der "L"-Taste.

Was wäre, wenn Sie die Zeile Zeile 10 so geschrieben hätten:

10 PRINT "ALLO"

Die Behebung eines solchen Fehlers:

Setzen Sie den Cursor über das Zeichen, welches *nach* dem ausgelassenen folgt. (In diesem Falle, wäre es der Buchstabe "A"). Wenn Sie die CTRL-Taste gedrückt halten und INSERT drücken, wird der Cursor den darunter liegenden Buchstaben nach rechts schieben. (Warum? Um Platz zu machen für das Zeichen, das Sie einfügen wollen.) In die entstandene Leerstelle setzen Sie das vergessene "H" ein.

> Leider können Sie mit INSERT keine Abkürzungen machen. Um mehrere Zeichen einzufügen, müssen Sie jeweils vor dem Einfügen INSERT drücken.

"Diese Zeile habe ich wirklich vermurkst!"

Manchmal ist eine Befehlszeile total verschandelt. Zum Beispiel könnten Sie zum Schluß folgendes haben:

1& NELOO"

Welch' eine Katastrophe! In einem solchen Fall, ist es am einfachsten, die ganze Zeile neu zu schreiben. Bewegen Sie dazu den Cursor an den Anfang der Zeile und tippen Sie einfach über das Ganze hinweg und dann RETURN.

Nachdem Sie eine derartige Korrektur vorgenommen haben, wäre es recht nützlich, Ihr Programm anzuLISTen. Warum sollten Sie so etwas tun? Einfach um nach noch vorhandenen falschen Zeilen zu suchen, die vielleicht noch in Ihrem Computer gespeichert sind.

Falls wenigstens die Zeilen-Ziffer richtig war, gibt es eine geeignetere Lösung für die vermurkste Zeile. Tippen Sie einfach die ganze Zeile neu, indem Sie dieselbe Zeilen-Ziffer verwenden. (Aber diesmal ohne Fehler!) Ihr Computer weiß, daß es nicht zwei Zeilen mit derselben Zeilen-Ziffer geben kann. Also wird er die erste Zeile durch die zweite (richtige) Zeile ersetzen.

> Wir haben Ihnen eine wichtige Regel mitzuteilen: *Nach Korrekturen jeder Art (ob RUBOUT, INSERT oder dem Wiedereintippen einer ganzen Zeile), müssen Sie RETURN drücken. Dabei spielt es keine Rolle, an welcher Stelle Sie in der Zeile sind.* Denn es ist nämlich so, daß Sie mit RETURN Ihrem Com-

puter sagen, daß er die eben gemachten Korrekturen zu beachten hat.

5 Und nun etwas Mathe-Magie!

Wunderbar! Jetzt haben Sie sich Ihr erstes Programmierstück geleistet. Und es hat funktioniert. Dieses Programmiergeschäft ist doch sicherlich recht nützlich - oder? Aber Ihr Computer hat auch noch eine andere Seite, und die ist mindestens genauso nützlich.

Es ist nur eine Sache von Augenblicken, und Ihr Computer entwickelt sich zum

Super-Rechner!

Dazu brauchen Sie ihn nicht einmal auf die Universität zu schicken, sondern Sie müssen lediglich die Anweisungen etwas anders eingeben.

Als wir unser erstes Programm schrieben, hatte jeder Schritt eine Zeilennummer. Für Ihren Computer bedeutet das Setzen einer Nummer am Anfang jeder Zeile etwa: "Merk' Dir diese Schritte, aber folge ihnen noch nicht." Der Computer übernimmt Ihre Anweisungen in seinen Speicher, damit er nicht vergißt, was Sie wollten.

Solche Befehle nennt man bedingte Instruktionen.

Darüber hinaus gibt es eine andere Art von Befehlen, die Sie Ihrem Computer geben können. Diese drücken folgendes aus: "Führe diesen Schritt aus, und zwar sofort!"

Solche Befehle nennt man unbedingte Instruktionen.

Wenn wir diese unbedingten Instruktionen anwenden, dann sind wir im Tischrechner-Modus und können nun wie mit einem Taschenrechner rechnen. *Befehle im Tischrechnermodus werden ohne vorangestellte Zeilennummer geschrieben und nach Tippen der RETURN-Taste sofort ausgeführt.*

Jetzt haben wir Ihnen gesagt, wie Sie aus Ihrem Computer einen Super-Rechner machen. Vielleicht wäre es jetzt angebracht, Ihnen als nächstes zu zeigen, wie man damit rechnet.Stellen Sie sich vor, Sie wollten zwei Zahlen zusammenziehen. Wir sind es gewöhnt, zur Addition die Zahlen auf ein Blatt Papier zu schreiben. Und das bedeutet, daß die meisten von uns gewöhnt sind, folgende Symbole für mathematische Funktionen zu sehen:

Addition oder plus, dargestellt durch +
Subtraktion oder minus, dargestellt durch -

Multiplikation, dargestellt durch X
Division, dargestellt durch :

Wenn Sie mit Ihrem Computer rechnen wollen, werden Sie bei der Addition und auch bei der Subtraktion keine Probleme haben, da sind die Funktionszeichen gleich. Aber was wäre, wenn Sie das übliche Multiplikationszeichen "X" in einer Eingabe machen würden? Richtig, der Computer würde das Zeichen für den Buchstaben "X" halten und würde dadurch völlig durcheinander kommen. Wenn Sie also mit dem Computer eine Multiplikation durchführen wollen, dann müssen Sie folgendes Zeichen verwenden: "*" (Der Stern sieht ja auch ein wenig wie ein "X" aus).

Das Divisionszeichen des Computers sieht auch etwas anders aus, nämlich so:"/" (Schrägstrich). Wenn Sie beispielsweise die Zahl 56 durch 8 durch den Computer geteilt haben wollten, würde das so aussehen: 56/8.

Ihr Computer kann sogar potenzieren! Nun, die meisten von Ihnen wissen, wie man mit Bleistift und Papier potenziert. Z.B. wird die vierte Potenz von zehn so geschrieben:

$$10^4$$

Wobei 10 die Basis und 4 der Exponent ist. (Dies ist ja eigentlich nur eine Abkürzung für 10 X 10 X 10 X 10.)

Für den Computer wäre es schwierig, eine solche Rechenoperation auf dem Bildschirm darzustellen. Also verwenden wir ein besonderes Zeichen, um die Rechnenoperation abgekürzt darzustellen, nämlich: "↑" (Hochpfeil). Dieses Symbol wird zwischen der Basis und dem Exponenten eingefügt (in unserem Beispiel also zwischen der 10 und der 4).

Durch Eintippen folgender Zeichen wird der Computer veranlaßt, die vierte Potenz von 10 zu rechnen: 10 ↑ 4. (Stellen Sie es sich so vor: Der Pfeil dient dazu, der zweiten Zahl zu sagen, wo sie hingehört.)

Wir haben erwähnt, daß sich der Computer zu einem Super-Rechner entwickeln läßt. In der Tat, er ist fähig, "super"-komplizierte Rechnungen durchzuführen. Aber bei der Lösung komplexer Rechenoperationen geht er nach einer bestimmten Reihenfolge der Einzeloperationen vor:

Als erstes: Er potenziert und zwar von links nach rechts.
Z.B.: bei der Operation 3 ↑ 2 ↑ 2
rechnet der Computer zuerst 3 ↑ 2 (das ergibt 9);
dann rechnet er 9 ↑ 2 (das ergibt dann 81).

Als zweites: Er multipliziert oder dividiert, falls solche Operationen vorkommen (und zwar wiederum von links nach rechts).

Z.B.: Bei Operation 6 + 3*4 + 6/3
rechnet der Computer zuerst 3*4 (ergibt 12) und dann 6/3 (ergibt 2). Anschließend (!) addiert er 6 + 12 + 2 (und erhält als Endsumme 20).

Übrigens würden Sie vielleicht mit Klammern arbeiten wollen. Wenn ja, dann wird das, was innerhalb der Klammer steht, als eine in sich geschlossene Teiloperation aufgefaßt und wird immer (!) vor allen anderen Operationen durchgeführt.

Z.B.: Bei der Operation 18/(3 + 3).
Der Computer rechnet zuerst die Klammer aus (3 + 3) und erhält 6. Die verbliebene Operation ist somit 18/6 (ergibt 3).

Wie Sie Ihren Computer veranlassen, Ihnen die Ergebnisse zu nennen

Im letzten Kapitel haben wir den Computer darum gebeten, etwas auf dem Bildschirm anzuzeigen. Wissen Sie noch das BASIC-Wort dafür? Es ist PRINT, und wir setzten damals Anführungszeichen vor und hinter das, was wir angezeigt haben wollten.

Versuchen wir's einmal mit einer einfachen Rechnung, mit 2 + 4. Als erstes tippen Sie:

PRINT "2 + 4" RETURN

Sehen Sie, was passiert ist? Ihr Computer hat genau das getan, was Sie wollten. Er hat das gedruckt, was innerhalb der Anführungszeichen steht. Aber das Problem ist, wir wollen ja nicht die Aufgabe sondern das Ergebnis sehen!

Um den Computer dazu zu bringen, die Rechenoperation auszuführen und das Ergebnis anzuzeigen, verwenden sie einfach die PRINT-Anweisung ohne(!) die Anführungszeichen.

Wollen Sie's versuchen? Geben Sie CLS ein, um den Bildschirm zu löschen (das ist fast so, als würde man einen Tafel-Schwamm verwenden.) Jetzt tippen Sie Ihre Operation noch einmal ein und zwar so:

PRINT 2 + 4 RETURN

Auf dem Bildschirm ist jetzt eine 6 erschienen, und das ist natürlich Ihr Ergebnis.

Wollen Sie etwas Eleganz in die Lösung Ihrer mathematischen Probleme bringen? Das können wir nämlich schon, indem wir beide Arten der PRINT-Anweisung verwenden. Da ist es nur eine Sache des Einsetzens Ihrer

Operation in ein kurzes Programm, beispielsweise wie das folgende:

```
10 PRINT "2 + 4="
20 PRINT 2 + 4
30 END
```

Können Sie sich denken, was das bewirken wird? Tippen Sie es in Ihren Computer. Vergessen Sie aber dabei nicht, die SPACE-Taste für Leerstellen und die RETURN-Taste am Ende jeder Zeile anzuwenden!

Haben Sie's? Wenn Sie Ihrem eigenen Tippen nicht trauen, dann geben Sie LIST ein, um es zu überprüfen, (Ist es nicht praktisch, diese hilfreichen Kommandos so griffbereit zu haben?). Wenn alles klar ist, dann können Sie mit RUN das Programm starten. (Ach ja, wenn Sie vorher CLS (Bildschirm löschen) eingeben wollen, dann bitte. Viele Leute machen das gerne, denn es sieht etwas schöner aus.)

Jetzt müßte der Bildschirm folgendes anzeigen:

```
READY
RUN
2 + 4 =
6
READY
```

Das Programm hat beim Computer folgendes veranlaßt:

> Zeile 10 sagte: "Zeige den Text zwischen den Anführungszeichen genau so, wie er ist, auf dem Bildschirm an." Zeile 20 sagte: "Führe die Rechenoperation aus und zeige das Ergebnis an." Zeile 30 sagte: "Das war's für heute."

Das ist doch recht schneidig! Aber, glauben Sie, daß es einen Weg gibt, sowohl die Aufgabe als auch das Ergebnis auf einer Zeile erscheinen zu lassen? Glauben Sie, daß wir das erwähnt hätten, wenn es nicht möglich wäre?

DamitIhr Programm wieder auf dem Bildschirm erscheint, geben Sie wieder LIST ein. Gut! Und können Sie sich noch an die Editier-Pfeile erinnern, von denen wir gesprochen haben? Wenden Sie sie jetzt dazu an, den blinkenden Cursor *an das Ende der Zeile 10 hinter die zweiten Anführungszeichen zu bringen.* (Vergessen Sie dabei nicht, die CTRL-Taste während der ganzen Zeit, die Sie den Cursor bewegen, gedrückt zu halten!)

Wenn der Cursor dort angelangt ist, wo Sie ihn haben wollen, dann tippen Sie einen Strichpunkt ein (das sieht so aus: ";"). Fertig? Jetzt, geben Sie RETURN ein. Der Cursor sollte dann an den Anfang der Zeile 20 gesprungen sein. Benutzen Sie wieder Ihre Pfeile, um den Cursor an die richtige

Stelle zu bringen, unter die unterste Zeile (READY).

Und nun die CLS-Taste kurz antippen und Sie sind bereit, Ihr frisiertes Programm mit RUN abzufahren. Sobald Sie das tun, sieht's so aus:

```
READY
RUN
2 + 4 = 6
READY
```

Jawohl, das ist schon viel besser. Wer hätte gedacht, daß ein einfacher Strichpunkt ein so wichtiges Hilfsmittel sein könnte? Er hat den Computer veranlaßt, nach der Anzeige der Zeile 10, in derselben Zeile weiter zu schreiben.

Jetzt packt und die Abenteuerlust

Je mehr wir Ihnen über diesen wunderbaren kleinen Computer erzählen, desto mehr werden Sie selbst neue Anwendungen ausprobieren wollen. Versuchen wir doch, gemeinsam dasselbe Mathe-Problem noch einmal zu lösen, aber diesmal auf eine wesentlich ausführlichere Art und Weise.

Tippen Sie NEW, um das gespeicherte Programm zu löschen. Und nun tippen Sie dieses Programm ein:

```
10 PRINT "COMPUTER, ADD 2 AND 4"
20 PRINT
30 PRINT "SELBSTVERSTÄNDLICH, MEISTER!"
40 PRINT "DAS ERGEBNIS IST";
50 PRINT 2+4
60 END
```

Löschen Sie den Bildschirm (nur der Ordnung wegen), tippen Sie RUN und halten Sie sich fest! Wer hätte gedacht, daß ein Computer so höflich sein kann? Schauen Sie, was der Computer jetzt sagt:

```
READY
RUN
COMPUTER, ADD 2 AND 4
SELBSTVERSTÄNDLICH, MEISTER!
DAS ERGEBNIS IST 6
READY
```

Haben Sie den Streich bemerkt, den wir Ihnen gespielt haben? Dort, wo die Zeile 20 aktiv war, ist nichts gedruckt. Als Sie die Zeile 20 eingetippten, sah es so aus:

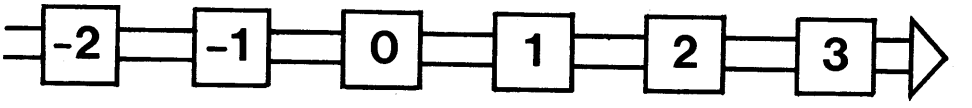
20 PRINT In anderen Worten, Sie haben den Computer dazu veranlaßt, nichts (!) in dieser Zeile anzuzeigen, und genau das hat er getan. Er hat eine leere Zeile erscheinen lassen. Sie können diesen Trick dazu verwenden, um dem Druckbild eine schöne Form zu geben.

6 Konstante und Variable

Früher oder später wird jeder Programmierer den Unterschied zwischen konstanten und variablen Dingen kennen müssen. Wir sollten ruhig daran gehen, es etwas früher zu wissen.

Was ist eine Konstante?

Wenn man etwas als konstant bezeichnet, so meint man damit, daß es sich nicht verändert: Der Wert des ganzen bleibt immer der gleiche. Hier ein kleines Beispiel zu besseren Veranschaulichung:



Diese Zahlen befinden sich auf einem kurzen "Zahlenpfeil", und zwar von minus drei bis plus drei.

Sie merken, daß jede Zahl einen angestammten Platz auf der Zahlenlinie hat. Wo sie sitzen, kommt es auf ihren jeweiligen Wert an, und sie können ihren Platz niemals verlassen! Nehmen wir die Zahl 1 zum Beispiel. Die 1 hat immer denselben Wert und sitzt immer genau dort, wo sie jetzt sitzt. Daran ist nichts zu ändern.

Vielleicht werden Sie sagen: "Ich kann den Wert schon ändern, ich brauche ja nur zu der 1 noch eine 1 hinzu addieren." Also gut, schauen wir einmal, was passiert, wenn wir es versuchen.

Die Darstellung "1+1" ist eigentlich nur eine Abkürzung für: "Fange bei Zahl 1 an und schreite um ein Zahl entlang der Zahlenlinie." Wenn man das macht, kommen Sie natürlich bei der Zahl 2 an.

Sie haben aber deswegen aus der 1 noch lange keine 2 gemacht. Sie haben sich lediglich entlang der Linie zu einer neuen Zahl bewegt. Die 1 sitzt immer noch da und hat sich überhaupt nicht verändert.

Konstante kann man addieren, subtrahieren, mulitplizieren und dividieren. Man kann eigentlich sämtliche Rechenoperationen mit ihnen durchführen. Die Konstanten ändern sich dadurch aber nicht, es ergeben sich nur neue Konstante!

Und die Variablen?

Haben Sie schon erraten, was wir Ihnen über Variable berichten werden? *Eine Variable ist etwas, dessen Wert sich sehr wohl ändern kann. Und Sie* sind es, der diese Änderungen vornimmt, so oft Sie wollen.

Stellen Sie sich vor, daß der Speicher Ihres Computer aus vielen einzelnen Abstellplätzen besteht, wie bei der Tiefgarage eines Hochhauses. Nehmen wir an, jeder würde in der Tiefgarage sein Auto abstellen dürfen, wo er es wollte. Das würde natürlich zu einem Gerangel un die besten Abstellplätze (bei denen der Ausgang am nächsten liegt) führen. Um dies zu vermeiden und um niemanden zu benachteiligen, sind für gewöhnlich die einzelnen Abstellplätze markiert, d.h. den einzelnen Inhabern zugewiesen. Mit der Markierung der Abstellplätze ist also festgelegt, wo welches Auto geparkt wird.

Wenn Sie eine Variable anlegen, tun Sie folgendes:

1. Sie *markieren* den Abstellplatz im Speicher des Computers, indem sie ihm eine *Bezeichnung* zuweisen.
2. Sie parken etwas auf dem Abstellplatz.

Die Markierungen für die Variablen können sein: + Irgendwelche Buchstaben (von A-Z).
+ Eine Kombination mehrerer Buchstaben von AB, IQ, MT usw.)
+ Eine Kombination von einem Buchstaben und einer Ziffer von 0-9.
(A3, Y0, R2 usw.) Dabei ist lediglich zu beachten, daß der Buchstabe als erstes in der Reihenfolge steht. 4U, 2Z und dergleichen werden von einem Computer nicht(!) aufgenommen.

Die Markierung einer Variablen kann zwar länger sein als zwei Zeichen, aber der Computer wird nur zwei lesen und unterscheiden.

> Wenn Sie beispielsweise mit SCHÖN markieren, wird der Computer nur SC lesen. Die Variable wird also vom Computer als SC markiert. Weitere Beispiele: Markierungen wie PETER, ANNA oder ABRAKADABRA gehen als PE, AN oder AB in den Computer ein.

Achtung: Der Computer hat eine bestimmte Anzahl "reservierter" Wörter. Diese besonderen Wörter, wie bei reservierten Tischen, sind bereits für

andere Anwendungen belegt. Die Verwendung einer solchen Bezeichnung ist nicht zulässig! (Im Anhang C finden Sie eine Liste der bereits belegten Bezeichnungen.)

Wir empfehlen Ihnen, sich einzelner Buchstaben zu bedienen, wenn Sie nicht eine immense Fülle verschiedener Variablen haben. Es ist wirklich auch viel leichter dadurch.

Das Belegen von Speicherplätzen beim Computer

Wir verwenden ein neues BASIC-Wort namens "LET". Betrachten Sie folgendes Beispiel:

LET A=4 Der erste Teil dieser Anweisung sagt: "Markiere einen Speicherplatz mit Namen A". Der zweite Teil besagt: "Speichere den Wert von 4 in Speicherplatz A".

Erinnern Sie sich an das bereits Gesagte über Variable? Ihr Wert läßt sich jederzeit ändern. Der Wert von A läßt sich folgendermaßen ändern, beispielsweise auf 7:

LET A=7

IhrComputer wird den Speicherplatz namens A in seinem Speicher aufsuchen. Im Speicherplatz ist nur Platz für jeweils einen Wert. Also wird der Computer den neuen Wert (7) einspeichern, den alten aber vergessen.

Einfach, nicht wahr? *Der Wert einer Variablen läßt sich also jederzeit ändern, in dem man ihm einen neuen Wert zuteilt.*

Und jetzt einige überraschende Neuigkeiten: Es gibt einige Abkürzungen für einen Programmierer und hier ist eine davon:

Wenn der Computer Anweisung erhält, eine Variable zu erstellen, *ist das Kommando LET gar nicht unbedingt notwendig.* Es kann vernachlässigt werden. Der Computer wird Sie auch ohne LET verstehen. LET ist nur dazu da, uns Menschen zu sagen, was das Kommando bewirkt. Es sind also gleichwertig:

LET A=7 A=7

7 Mehr Mathe-Magie mit INPUT

Schön, daß Sie's durch das letzte Kapitel geschafft haben. Es freut uns, das zu hören, denn nun wissen wir, daß Sie sich mit der LET-Anweisung auskennen. Diese Anweisung kommt nicht nur bei der Markierung von Speicherplätzen im Speicher des Computers zur Anwendung, sondern dient auch der Wertzuweisung in den jeweiligen Speicherplatz.

Das ist nur eine Möglichkeit, eine Variable zu erstellen. In diesem Kapitel werden wir etwas über eine andere BASIC-Anweisung hören, die etwas ähnliches macht. Nur die Art und Weise wie sie es macht, ist verschieden zu der bisher behandelten Art. Der Ausdruck für diese Anweisung ist INPUT. Das ist der Unterschied:

LET A=7 besagt:1. "Markiere einen Speicherplatz mit A." 2. "Weise Speicherplatz A den Wert 7 zu, sofort."

INPUT A besagt:1. "Markiere einen Speicherplatz mit A". 2. "Mit der Wertzuweisung an Speicherplatz A warten, warte dort, wo du bist, bis von der Tastatur ein Wert eingetippt wird."

Erkennen Sie den Unterschied zwischen den beiden? Wenn der Computer auf eine INPUT-Anweisung stößt während er gerade ein Programm fährt (RUN), wird er anhalten und wartet (ziemlich geduldig), bis er von Ihnen über die Tastatur die Wertzuweisung für A erhält.

Die Anweisung zu kennen, ist extrem nützlich. Wollen Sie wissen warum? Dann zeigen wir es Ihnen jetzt!

Ein Tanz mit dem Einmaleins

Fast jeder, den wir kennen, haßt es, das Einmaleins aufsagen zu müssen. Sie erinnern sich, das sieht so aus:

1 x 4 = 4
2 x 4 = 8
3 x 4 = 12
4 x 4 = 16 usw..

Allgemeines Stöhnen. Aber keine Angst! Ihr treuer Computer steht einsatzbereit zur Stelle. Nachdem wir dieses Kapitel durchgearbeitet haben, werden Sie nie mehr ein Einmaleins rechnen müssen, denn wir werden inzwischen Ihrem Computer beigebracht haben, dies für Sie zu erledigen.

Also, fangen wir an! Das entsprechende Programm ist überhaupt nicht schwierig. Wir gehen Schritt für Schritt mit Ihnen gemeinsam vor. Bevor wir anfangen, benötigen wir einen Multiplikator. Nehmen wir doch zunächst etwas einfaches, beispielsweise die 4.

Das erste für unser Programm wird sein, eine Zahl zu haben, die mit dem Multiplikator (4 in diesem Fall) multipliziert wird. Wäre es nicht schön, wenn Ihr Computer Sie recht höflich danach fragen würde? Das dachten wir uns schon. Vielleicht sollte die erste Zeile des Programms ungefähr so aussehen:

```
10 PRINT "WELCHE ZAHL, BITTE"
```

Jetzt zur Zeile 20: Nun ist es Zeit, unsere neue BASIC-Anweisung "INPUT" zu aktivieren. Tippen Sie die folgende Zeile ein:

```
20 INPUT A
```

(Bitte merken: Wenn das Programm im Computer gestartet wird, kommt bei dieser Zeile ein Stop, bei dem wir später eine Wertzuweisung für A geben.

Die nächste Zeile soll die Operation und das Ergebnis auf dem Bildschirm anzeigen. Das ist doch einfach! Das haben wir bereits in Kapitel 5 getan. Tippen Sie folgendes in Ihren Computer ein:

```
30 PRINT A;" X 4 =";
```

In dieser Zeile sind zwei Strichpunkte. Wenn sie Ihnen aufgefallen sind, um so besser. Wir wissen, was der zweite bewirkt, aber was ist mit dem ersten? Entspannen Sie sich, es ist nur wieder ein Trick, um die Sache zu vereinfachen. Es ist nämlich so, in dieser Zeile soll der Computer folgendes anzeigen (mit PRINT):

Die Zahl, die Sie an A per INPUT zugewiesen haben.
Die Kleinigkeit, die, von den Anführungszeichen umklammert, danach folgt. Indem Sie nun einen Strichpunkt zwischen diese beide setzen, können Sie dieselbe PRINT-Anweisung für beide verwenden, womit sie nacheinander in der gleichen Zeile erscheinen werden.

Wahrscheinlich wissen Sie, was als nächstes kommt, ohne daß wir es Ihnen sagen müssen. Es ist die Zeile, die Ihren Computer dazu veranlaßt, zu rechnen und Ihnen das Ergebnis mitzuteilen. Und das sieht so aus:

```
40 PRINT A*4
```

Natürlich sagt dies nichts anderes als "multipliziere das, was im Abstellplatz A ist, mit 4 und zeige mir das Ergebnis auf dem Bildschirm."

Dies soll aber ein komplettes Einmaleins werden und wir werden uns mit nur einem Ergebnis nicht zufrieden geben! Wahrscheinlich werden Sie andere Zahlen auch mit 4 multiplizieren wollen. Warum sollten wir also nicht den Computer veranlassen, uns nach der nächsten Zahl zu fragen? Das machen wir so:

```
50 PRINT "NOCH EINE ZAHL"
```

Wie bekommen wir die neue Zahl nun in den Speicherplatz A? Gesagt, getan, die nächste Zeile unseres Programms wird das erledigen. In der Zeile 60 werden wir unseren alten Freund, die GOTO-Anweisung, anwenden. Tippen Sie:

```
60 GOTO 20
```

Diese GOTO-Anweisung ist wirklich etwas wunderbares, nicht? Sie schickt Ihren Computer blitzschnell wieder zurück zu Zeile 20 und dort wartet er, bis Sie sich eine neue Zahl ausdenken, die in Speicherplatz A zu setzen ist.

> Vergessen Sie nicht was passiert, wenn Sie Ihren Computer anweisen, einen neuen Wert in eine bestimmte Speicherstelle zu übernehmen. Als erstes bildet er einen neuen Wert für die Variable. Dann entfernt er den bisherigen Wert und ersetzt ihn mit dem neuen.

Jetzt ist es soweit! Ihr eigenes Programm zur Berechnung des 4er Einmaleins. Aber, warum sollten Sie sich auf unser Wort verlassen und glauben daß es funktioniert, wenn Sie sich selbst davon überzeugen können, indem sie mit RUN starten. Dann mal los, versuchen Sie's (und vergessen Sie nicht, vorher den Schirm mit CLS zu löschen).

Moment mal, was ist das? Das einzige, was man auf dem Bildschirm sieht, ist:

```
READY
RUN
WELCHE ZAHL, BITTE
?
```

Wir haben Sie aber gewarnt! Das Fragezeichen in der zweiten Zeile Ihres Programms stellt nur die INPUT-Nachricht da, die Sie beachten sollen. Ihr Computer soll nichts weiter, als daß Sie ihm eine Zahl geben, die er mit 4 multiplizieren will. Nun denn, es gibt keinen Grund zu warten. Denken wir uns eine Zahl aus. Wie wärs's mit 2? Tippen Sie's ein und drücken Sie RETURN.

```
READY
RUN
WELCHE ZAHL, BITTE
? 2
2 X 4 = 8
NOCH EINE ZAHL
?
```

Es funktioniert also doch! Überlegen Sie einmal, was sich abgespielt hat. Das Programm hat eine Schleife zurück zu der INPUT-Nachricht in Zeile 20 gezogen und jetzt fragt der Computer nach einer neuen Zahl, die er mit 4 multiplizieren will. Und er wird immer weiter Schleifen ziehen, dabei immer

nach einer weiteren Zahl fragen, bis Sie des Spiels müde sind. (Oder, bis das Abendessen ruft).

Versuchen Sie also noch ein paar Zeilen, nur so zum Spaß. Vielleicht möchten Sie ein anders Einmaleins probieren, das Einmalneun beispielsweise. Leichter getan als gesagt. Dazu muß man nur einige Zeilen des Programms ändern.

Zuerst tippen Sie BREAK, das stoppt den Computer sofort. Dann LISTen Sie Ihr Programm auf dem Bildschirm. Jetzt verwenden Sie die Editier-Pfeile, um das "X 4=" in der Zeile 30 in "X 9=" umzuwandeln. Dann ändern Sie Zeile 40 so ab, daß A*9 dort steht, statt A*4.

Vergessen Sie nicht, nach jeder Zeilenänderung die RETURN-Taste zu betätigen.

Alles erledigt? Bewegen Sie den Cursor wieder nach unten (so, daß er unter READY blinkt). Tippen Sie CLS und dann RUN. Dann wird Ihr neues Programm den Computer dazu veranlassen, jede von Ihnen eingegebene Zahl mit 9 zu multiplizieren.

Nun ein kleines Programm für die oft erfindungsreichen, trägen Typen. Sie wissen schon, wen wir meinen. Das sind diejenigen, die denken, es sei zu viel Aufwand, immer wieder eine neue Zahl mit INPUT eingeben zu müssen. Ob Sie es glauben oder nicht, dieses kleine Programm wird Ihren Computer dazu bringen, den Wert von A bei jeder erneuten Schleife von sich aus zu erhöhen. Und zwar vollautomatisch.

```
10 A=1
20 PRINT A; "X 4=";
30 PRINT A*4
40 A=A+1
50 GOTO 20
```

Wenn Sie dieses Programm mit RUN laufen lassen, müssen Sie selbst überhaupt nichts mehr tun! (Außer, natürlich sich zurückzulehnen und erstaunt dreinzuschauen.) Und so funktioniert es:

Zeile 10 speichert den Wert 1 in eine Variable namens A.
Zeile 20 zeigt die Rechenaufgabe im Bildschirm an.
Zeile 30 führt die Operation aus, indem A (anfangs = 1) mit 4 multipliziert wird, und zeigt das Ergebnis an.
Zeile 40: hier befindet sich das Geheimnis unseres automatischen Einmaleins. Diese Zeile sagt: "Addiere 1 zu dem in A gespeicherten jeweiligen Wert."

Zeile 50 führt das Programm in einer Schleife wieder zurück zu Zeile 20.

Beim ersten Erreichen der Zeile 20 addiert Ihr Computer 1 mit 1, womit A dann den neuen Wert 2 erhält. Beim nächsten Mal wird der Wert von A auf 3 geändert. Dann auf 4, dann auf 5, dann auf 6 ...u.s.w.. Der Computer wird immer weiter den Wert von A jeweils um 1 erhöhen, bis Sie mit BREAK das Programm abbrechen.

Da haben Sie Ihrem Computer aber einen ganz praktischen Trick beigebracht.Egal mit welcher Zahl Sie ein Einmaleins erstellen wollen, es wird immer funktionieren und Sie dürfen stolz auf sich sein, daß Sie's gemeistert haben, Aber nun ist's genug, dieses Kapitel war wirklich ziemlich schwierig. Wahrscheinlich wollen sie sich jetzt etwas ausruhen (und ein wenig mit Ihren neugewonnenen Programmierkenntnissen spielen). Also lassen wir Sie einen Moment in Ruhe.

8 Entscheidungsfindung

Jetzt ist es an der Zeit, daß wir uns etwas sehr aufregendem zuwenden. Wir werden eine der mächtigsten BASIC-Anweisungen, die Sie in Ihren Programmen verwenden können, kennenlernen. Diese Anweisung wird IF-THEN-ELSE (Wenn-Dann-Sonst) genannt.

Beilhrer Kenntniserweiterung in Sachen Programmierung werden Sie feststellen, daß diese Anweisung ganz erstaunliche Möglichkeiten bietet.

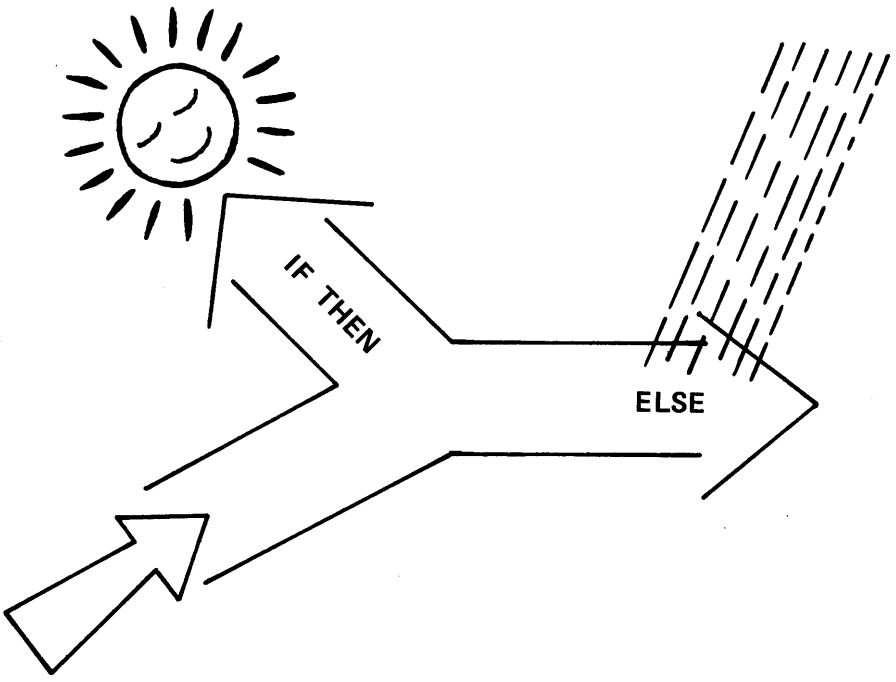
DasSpeichern eines Programms in Ihren Computer ist etwa so, als würden Sie einen Jogging-Pfad abstecken, auf dem Ihr Computer dann losläuft. Und wenn Sie ihm sagen, daß er loslaufen soll (RUN), dann tut er genau das. Er beginnt beim Programm-Anfang (erinnern sie sich? Das ist die Zeile mit der kleinsten Ziffer.) und läuft und läuft, bis er am Programmende ankommt.

WennSie eine IF-THEN-ELSE-Anweisung in Ihr Programm einfügen, erkennt der Computer an der Stelle eine Gabelung des Pfades auf dem er joggt. Nun ist das für den Computer etwas verwirrend, Plötzlich existieren mehrere Wege, auf denen er weiter joggen könnte und wie soll er dabei wissen, auf welchem er weiterlaufen soll? Eben das weiß er von sich aus nicht. Sie müssen ihm da schon mit Informationen weiterhelfen, damit er eine 'Entscheidung' treffen kann.

Hier ist ein einfaches Beispiel, anhand dessen die ganze Sache leicht verständlich wird. Nehmen wir an, Sie spazieren durch eine Vorortstraße, auf dem Weg zu einem Freund. Die Gegend ist aber noch unbekannt und Sie

müssen sich deswegen an die Straßenbeschilderung halten, um sich zu orientieren und um den richtigen Weg zu finden.

Also, Sie spazieren so dahin. Aber plötzlich stehen Sie vor einer Weggabelung, bei der es nach links in einen Feldweg geht und nach rechts in eine geteerte Staße. Welcher Weg ist der richtige? Um das entscheiden zu können, benötigen Sie zusätzliche Informationen. Unglücklicherweise haben Sie keine Ahnung und müssen von irgendwo her sich Informationen besorgen. Keine Angst! Denn direkt an der Weggabelung steht ein Schild, auf dem folgendes steht:



WENN (IF) es nicht regnet, DANN (THEN nehme den Feldweg.
SONST (ELSE) nehme die geteerte Staße.

Und da ist Ihre Antwort! Um den richtigen Weg zu finden, müssen Sie nur die Umstände untersuchen, in diesem Fall die Wetterverhältnisse. Also, wie untersucht man die Umstände? Natürlich indem man Sie prüft!

Wenn Sie sich die Nachricht auf der Tafel genau ansehen, werden Sie erkennen, daß sie in drei Teile geteilt ist. Sehen Sie sie? Der eine Teil beginnt mit IF (WENN), der andere mit THEN (DANN) und der letzte mit ELSE (SONST).

IF (WENN) es nicht regnet,
THEN (DANN) begehe (=GOTO) den Feldweg,
ELSE (SONST) begehe (=GOTO) die geteerte Straße.

Das Problem, welchen Weg Sie beschreiten sollten, lösen Sie so:

Erster Teil:
IF es nicht regnet...

Sie sollen sich fragen: "Werde ich gerade naß?" Wenn nicht, dann fällt der-Schluß nicht schwer, daß es nicht regnet. Und wenn es nicht regnet, dann ist dieser Teil der Nachricht richtig. Also müssen Sie direkt weitergehen zu:

Zweiter Teil: ... THEN begehen (GOTO) Sie den Feldweg.

Und Sie tun genau das, was dieser Teil Ihnen vorschreibt.

Aber was wäre, wenn Sie sich fragen würden: "Werde ich gerade naß?" Und plötzlich wird Ihnen bewußt: "Mensch, das stimmt! Es muß ja regnen!" Also, wenn es wirklich regnet, dann kann der erste Teil der Nachricht nicht richtig sein. Und wenn es so ist, dann dürfen Sie die Anweisung im zweiten Teil nicht befolgen, Sie müssen SONST etwas tun, nämlich:

Dritter Teil: ... ELSE (SONST) begehe (=GOTO den geteerten Weg.

Und Sie müssen dann stattdessen das tun, was Ihnen dieser Teil sagt.

Bis jetzt haben wir den Fall behandelt, bei dem ein Mensch sich auf dem gegabelten Weg befindet. Wir haben einen guten Frund, Ihnen dies zu erklären. Wir wollen nichts anderes sagen, als daß der Computer ähnlich reagieren würde (mit einigen Ausnahmen), befände er sich in derselben Situation.

Unterschied Nr.1: Wenn wir die IF-THEN-ELSE-Anweisung benutzen wollen, um Ihren Computer dazu zu zwingen, eine Wahl zu treffen, dann müssen wir ihm zwei Dinge gegen, zwischen denen er einen Vergleich ziehen kann. Diese Dinge können entweder Variable oder Zahlen sein. Computer-Leute nennen sie 'Ausdrücke'.

Unterschied Nr.2: Als Sie auf dem Weg zu Ihrem Freund waren, hatten Sie die Wetterumstände geprüft. Es gibt eine Vielzahl von anderen Bedingungen, die Ihre Entscheidung in anderen Situationen beeinflussen könnten, die Auswahl kann unendlich groß sein.

Der Computer hat aber dem gegenüber eine sehr begrenzte Auswahlmöglichkeit. Wir nennen Sie 'Relationen' (Verglei-

che), denn was Ihr Computer wirklich tut, ist, daß er Vergleiche zwischen zwei Dingen anstellt. Hier ist eine Liste seiner Vergleichsmöglichkeiten:

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

Das Gefühl der Macht ist wundervoll! Jetzt haben Sie ein Mittel in der Hand, mit dem Sie Ihren Computer zwingen können, 'sich über etwas klar zu werden', ohne daß er Sie zuerst fragt. Wie Sie vielleicht schon erraten haben, dient dieses Mittel auch dazu, Sie etwas zu entlasten, während der Computer mehr Arbeit übernimmt. Wozu sind elektronische Sklaven denn sonst gut?

Wennman in ein Programm eine Gabelung einbaut, dann sind IF und THEN absolut notwendig. *Mit ELSE jedoch, verhält es sich anders. Dieses Statement kann ausgelassen werden.* Die ELSE-Anweisung gehört in dieselbe Programmzeile unmittelbar hinter IF-THEN. Sollte sich nämlich der IF-Teil des Statements als falsch erweisen, dann wird der Computer den THEN-Teil ignorieren und statt dessen fällt er hinunter zur nächsten Zeile des Programms und wird dort mit der Bearbeitung fortfahren.

Das funktioniert so:

```
10 IF es nicht regnet THEN GOTO den Feldweg
20 GOTO den geteerten Weg
```

Ruhen Sie sich jetzt etwas aus und lesen sie dieses Kapitel nochmal durch-,nur um sicher zu gehen, daß Sie die IF-THEN-ELSE-Anweisung verstanden haben. Denn im nächsten Kapitel werden wir Ihnen zeigen, wie Sie Ihre hinzugewonnene Macht zur Entfaltung bringen können!

9 Spielereien

Wenn Sie den ganzen bisherigen Weg mit uns gegangen sind, dann haben Sie schon sehr, sehr viel über Ihren kleinen Computer gelernt. Wahrscheinlich betrachten Sie Ihren Computer inzwischen nicht mehr als Mysterium, sondern als Freund. Jedenfalls hoffen wir das. Nun, wozu braucht man Freunde? Zum einen kann man mit ihnen sehr viel Spaß haben. Also werden wir Ihnen in diesem Kapitel sagen, wie Sie mit Ihrem Computer spielen können.

Wir müssen allerdings zugeben, daß es viele Spiele gibt, bei denen der Computer ziemlich schlecht spielt. Beispielsweise würden Sie nie einen Computer beim Ein-Hockey oder beim Ping-Pong sehen. Monopoly ist etwas zu schwer für ihn. Aber es gibt ein Spiel, welches der Computer mit Vorliebe spielt, und zwar ist es ein Ratespiel.

Wähle eine Zahl, igendeine Zahl

Endlich eine Chance, die neue mächtige IF-THEN-Anweisung zu versuchen (nein, diesmal brauchen wir das ELSE nicht). Bevor wir weitergehen, denken Sie sich eine Zahl aus, irgendeine Zahl. Haben Sie's? Gut, behalten Sie sie für sich. Dies wird unsere Geheimnummer sein und niemand, außer Ihrem Computer und und Ihnen selbst, wird sie niemand kennen. (Mögen Sie nicht auch Geheimnisse?!)

Und weiter im Text! Wir werden ein Programm schreiben, bei dem einer Ihrer Freunde Ihre Geheimnummer erraten soll. Betrachten sie das untenstehende Programm. Da sind keine Fallgruben drin! Jedes Kommando, welches dort vorgefunden wird, haben wir schon besprochen. Vielleicht erkennen Sie schon mit einem Blick, wie es funktionieren wird. Wir werden Zeile um Zeile vorgehen. Tippen Sie jede Zeile wie angegeben.

```
10 A=7
20 PRINT "ERRATE MEINE ZAHL"
30 INPUT B
40 IF B=A THEN GOTO 70
50 PRINT "FALSCH! VERSUCH'S NOCHMAL"
60 GOTO 30
70 PRINT "JAWOHL, DAS IST SIE"
80 END
```

Schon fertig? Sie werden immer schneller mit der Tastatur, nicht wahr? Nun, weil dies das bisher längste Programm ist, das Sie schreiben, wäre es vielleicht eine gute Idee, es zu LISTen. Das machen Sie jetzt.

Hier ist noch ein praktischer Rat. Bis jetzt haben Sie immer CLS gedrückt,um nach dem aufLISTen Ihres Programms auf dem Bildschirm, diesen wieder zu löschen. Wäre es nicht besser, wir würden das Kommando CLS in die erste Zeile Ihres Programms setzen? Auf diese Weise wird der Computer als erstes den Bildschirm löschen, sobald Sie das Programm mit RUN starten. Ja, wir dachten schon, daß Ihnen das gefallen würde!

Tippen Sie die folgende Zeile unter die bereits auf Ihrem Bildschirm befindliche Zeile 80:

```
5 CLS
```

Sehen Sie jetzt, warum wir Lücken zwischen den einzelnen Zeilen lassen? 5 ist kleiner als 10, also wird unsere neue Zeile die erste Zeile unseres Programms werden. Sie glauben das nicht? Dann drücken Sie halt LIST und schauen Sie sich's an. Sehen Sie? Ganz oben, in der ersten Zeile unseres Programms steht CLS. Wenden wir uns nun wieder unserem Spiel zu.

Programmanalyse

Zeile 10: In dieser Zeile "flüstern" wir dem Computer unsere Geheimnummer zu und sagen ihm, daß er sie in den Speicherplatz A abstellen soll.

Zeile 20: Druckt eine Nachricht in den Bildschirm, der Ihren Freund auffordert, die Geheimnummer zu erraten.

Zeile 30: Der Computer markiert einen zusätzlichen Speicherplatz mit B. Dann wartet er, bis Ihr Freund sich eine Zahl ausgedacht hat. Wenn er die Zahl dann eintippt, speichert sie der Computer im Speicherplatz B ab.

Zeile 40: Die IF-THEN-Anweisung wird aktiv! Im Speicher des Computers befinden sich zwei Variable. In A befindet sich die Geheim- nummer, B enthält die geratene Zahl Ihres Freundes. Wenn der Computer Zeile 40 erreicht, bleibt er stehen, um den Inhalt von A mit dem von B zu vergleichen. Merken Sie sich: Die IF-THEN-Anweisung bildet eine Gabelung im Programm und der Computer muß sich entscheiden, in welche Richtung er zu gehen hat. Wenn B gleich A ist, wird der Computer die Bear- beitung des Programms in Zeile 70 weiterführen, wobei eine Nachricht auf dem Bildschirm erscheint, die besagt, daß richtig geraten wurde.

Wenn aber wenn B ungleich A ist, dann ist der IF-Teil des IF-THEN-Statements falsch und der Computer wird den THEN-Teil ignorieren und in der Programmzeile 50 fortfahren.

Zeile 50: Druckt eine Nachricht auf dem Bildschirm, der Ihrem Freund sagt, er habe falsch geraten und fordert ihn gleichzeitig auf, es nocheinmal zu versuchen.

Zeile 60: Diese Zeile bringt das Programm auf einer Schleife zurück zur Zeile 30, bereit für die nächste Eingabe einer geratenen Zahl, die wieder in den Speicherplatz B genommen wird. Wenn Ihr Freund immer wieder falsch rät, wird diese Zeile immer wieder eine Schleife zurück zu Zeile 30 ziehen, bereit für den nächsten Versuch. Ihr Freund kann so oft raten, wie er will.

Wenn er endlich das große Los zieht und die Zahl in Speicher- platz B gleich der in Speicherplatz A ist, springt (GOTO) der Computer zu Zeile 70, druckt die entsprechende Nachricht und beendet dann das Programm in Zeile 80 und gleichzeitig das Spiel.

Sie können's nicht abwarten, dieses Spiel mit jemanden zu versuchen? Machtnichts, wir haben nicht erwartet, daß Sie warten würden. Sie sollten versuchen einen Freund (oder auch einige) aufzutreiben, die das Spiel ausprobieren wollen.Wenn Sie dann mit dem Spiel fertig sind, werden wir noch hier sein. Die IF-THEN-Anweisung hat aus diesem Programm wirklich etwas Tolles gemacht. Und dabei haben wir uns nur eine der Vergleichsmöglichkeiten, die Ihr Computer besitzt, angewendet, die er benutzen kann, um seinen Weg zu bestimmen. Wie wär's, wenn wir noch einige Verbesserungen in das Ratespiel brächten? Wir haben schon das bestimmte Gefühl, daß Sie zustimmen werden. Drücken Sie LIST, damit Ihr Programm wieder auf dem Bildschirm erscheint.

Zuerst: Ändern Sie Zeile 50 folgendermaßen ab

```
50 IF B>A THEN GOTO 56
```

(Zur Änderung müssen Sie lediglich den Cursor zum Anfang der bereits vorhandenen Zeile 50 bewegen und dann einfach darüber tippen. Vergessen Sie das RETURN zum Schluß der Zeile nicht.)

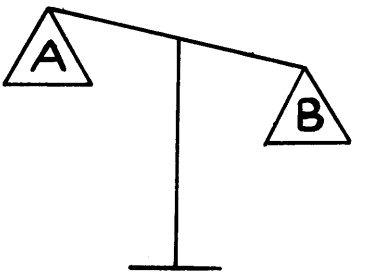
Sobald Sie das haben, bewegen Sie den Cursor hinab bis unter die letzte Zeile und tippen folgende Zeilen ein:

```
52 PRINT "ZU NIEDRIG!"
54 GOTO 30
56 PRINT "ZU HOCH!"
```

Schon wieder erweisen sich die verbliebenen Zwischenräume zwischen den Zeilennummern als sehr praktisch. Wie wir's gesagt hatten. Tippen Sie wieder LIST ein. Ihr Computer wird die Zeilen entsprechend Ihrer Numerierung sortieren, so daß Ihr neues Programm folgendermaßen aussehen sollte:

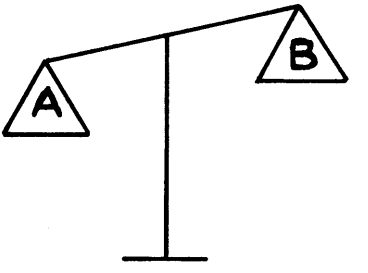
```
5 CLS
10 A=7
20 PRINT "ERRATE MEINE ZAHL"
30 INPUT B
40 IF B=A THEN GOTO 70
50 IF B>A THEN GOTO 56
52 PRINT "ZU NIEDRIG!"
54 GOTO 30
56 PRINT "ZU HOCH!"
60 GOTO 30
70 PRINT "JAWOHL! DAS IST SIE!"
80 END
```

Dieses Programm läuft genauso wie das letzte Programm, und zwar bis zur Zeile 50. Wenn die geratene Zahl in B gleich der Geheimnummer in A ist, wird Ihr Computer sofort zu Zeile 70 sausen. Aber, wenn B ungleich a ist, geht der Computer zur nächsten Programmzeile. Und hier wird's interessant! Da wir das Programm etwas veränderten, haben wir eine neue modifizierte Zeile 50.



In unserem hoch-frisierten Programm enthält Zeile 50 noch eine IF-THEN-Gabelung und der Computer muß noch einen Vergleich zwischen B und A vornehmen, um seine Weg-Entscheidung zu treffen. Dieses Mal wollen wir überprüfen, ob die Zahl in B größer ist als die in A. WENN ja, DANN fährt der Computer mit Zeile 56 fort. 56 druckt eine Nachricht, daß die Wahl *zu hoch* ausgefallen ist. Dann geht er eine Zeile weiter zur Zeile 60. 60 führt eine Schleife zu 30 durch, um auf die neue Wahl Ihres Freundes zu warten.

Aber wenn B nicht größer ist als A, dann ist der IF-Teil der zweiten IF-THEN-Anweisung falsch. Also wird der Computer seinen Weg zu Zeile 52 fortsetzen.



Nun wissen wir, daß B ungleich A ist. Und daß B auch nicht größer ist als A. Wenn Ihr Computer also mit Zeile 52 fertig ist, kann B im Verhältnis zu A nur eines sein, nämlich kleiner als A. In dieser Zeile wird dem Computer gesagt, daß er eine Meldung anzeigen soll, die besagt, daß die Wahl *zu niedrig* war.

Was kommt nach Zeile 52? Zeile 54. Und diese Zeile bringt den Computer auf einer Schleife zurück zu 30, bereit, die nächste Eingabe entgegenzunehmen.

Jetzt haben Sie ein Programm für ein Rate-Spiel, welches dem Ratenden sogar mit kleinen Hinweisen weiterhilft!

10 Mehr Spielereien

Egal wieviele Freunde Sie haben, es wird immer wieder so sein, daß Sie niemanden finden, der mit Ihnen und Ihrem Computer spielen will. Na ja, Sie müssen ja nicht unbedingt darauf warten, bis sich jemand findet, um mit Ihnen zu spielen.

Das Rate-Spiel aus dem letzten Kapitel macht eigentlich alleine spielend gar keinen richtigen Spaß. Jedenfalls nicht für lange. Denn die Geheimnummer kennen Sie ja doch. Also könnten Sie ja gleich beim ersten raten die richtige Lösung nennen und das Spiel wäre sofort wieder zu Ende. Den kleinen Schönheits- fehler könnten wir aber beheben. Wir müßten nur eine kleine Änderung des Programms Kapitel 9 vornehmen.

> Und wieder ist es Zeit, ein neues BASIC-Wort zu lernen - RND. Dieses Kommando bedeutet: "Denke dir eine Zahl aus, sag' mir aber nicht, welche."

Wir wissen schon, was Sie denken. Und wenn dieser neue Trick den Computer zum Denken bringen kann, dann muß es Zauberei sein! Na ja, das ist es zwar nicht ganz, aber es kommt dem ziemlich nahe. Das RND-Kommando bittet den Computer eine Zahl zufällig aus der Luft zu greifen (wie ein Zauberer, der einen Hasen aus einem Zylinderhut zieht).

Wenn es etwas sehr zahlreich gibt, dann sind es Zahlen. Zahlen gibt es wie Sand am Meer, mehr noch! Es gibt Millionen, Milliarden, Trillionen! Was wäre also, wenn Ihr Computer, irgendeine Zahl heraussuchte? Sie könnten Monate damit verbringen, sie herauszufinden.

Fair ist fair, also engen wir den Umfang der Zahlen, die zur Auswahl stehen, etwas ein.

Unmittelbar nach dem RND-Kommando müssen Sie eine Zahl in Klammern einsetzen. Zur Veranschaulichung nehmen wir einfach die Zahl 50. Das RND-Kommando sieht dann so aus:

```
RND (50)
```

Wissen Sie, was dieses Kommando Ihrem Computer befiehlt? Es sagt: "Wähle irgendeine Zahl aus, sie muß nur zwischen 1 und 50 sein!".

In anderen Worten, Ihr Computer hat jetzt 50 Zahlen in seinem Hut, wobei er davon eine ziehen wird, die dann als seine Geheimnummer dienen soll.

Aha, das macht die Sache etwas leichter. Jetzt wissen Sie, daß die Zahl die Sie erraten sollen, zwischen 1 und 50 liegt. Natürlich bleibt es Ihnen überlassen, welche Zahl Sie in die Klammer setzen. Wenn Sie es etwas leichter haben wollen, könnten Sie eine kleinere Zahl nehmen, 10 zum Beispiel. Wenn das Ratespiel eine größere Herausforderung darstellen soll, dann nehmen Sie eine größere, beispielsweise 500.

Jetzt wieder zurück zu unserem Programm. Wenn es noch im Speicher des Computers ist, umso besser. Wir müssen nur die Zeile 10 ändern und zwar so:

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

Die neue Zeile sagt Ihrem Computer:

> "Als erstes markiere einen Speicherplatz mit A. Dann wähle eine Zahl zwischen 1 und 50 aus und speichere sie dort ab."
>
> Falls Sie Ihren Computer zwischenzeitlich ausgeschaltet hatten, dann blättern Sie einfach zurück zum letzten Kapitel und tippen das ganze noch einmal ein. Vergessen Sie aber nicht, die neue Zeile 10 einzugeben.

Und jetzt haben Sie's! Ein Rate-Spiel, das Sie mit Ihrem Computer spielen können, wenn niemand sonst da ist. Und Sie können es sooft spielen wie Sie mögen, es erfordert nur ein RUN und Ihr elektronischer Freund denkt sich von neuem eine Geheimnummer aus, die Sie dann erraten können.

Einiges über 'Funktionen'

Das Wort RND kann die ganze Sache schon recht viel interessanter machen. Nachdem wir es Ihnen schon vorgestellt haben, sagen wir auch, was es ist. RND ist eine 'Funktion'.

Was wäre denn Ihre Antwort auf die Frage, was eine Funktion ist? Wenn Sie nicht gerade mathematik-orientiert sind, dann könnten Sie sagen, es habe vielleicht etwas mit Radios oder Sender zu tun, sie hingen doch auch ir-

gendwie mit Funk zusammen! Aber wenn Sie doch etwas Mathematik beherrschen, würden Sie wahrscheinlich antworten, sie habe etwas mit 'Tätigkeit', Wirksamkeit oder 'Abhängigkeit' zu tun.

Es gibt verschiedene Funktionen. Und sie erfüllen verschiedene Zwecke. Wenn wir eine Zahl in Funktion einsetzen, wird die Vorschrift der betreffenden Funktion die ihr eigene Rechenoperation mit der Zahl durchführen und dabei kommtdann (bis auf gewisse Ausnahmen) eine andere Zahl bzw. Wert heraus.

RND ist nur eine dieser verschiedenen Funktionen, die Sie mit Ihrem Computer anwenden können. Es ist so, daß viele dieser Funktionen sehr nützlich sein können, insbesondere für Leute, die höhere Mathematik betreiben.

Aber natürlich wird das nicht gerade jeder wollen. Sind Sie seit jeher ein leidenschaftlicher Mathematik-Hasser? Wenn ja, dann blättern Sie einfach ganz schnell um. Aber bitte, noch nicht. Hören Sie noch einen kleinen Augenblick zu.

Wir werden Ihnen jetzt von einer anderen Funktion erzählen. Denn irgendwann wird diese Funktion jedem nützlich sein (ja, auch Ihnen). Sie wird SQR genannt und sie wird verwendet, um die Quadratwurzel zu ziehen.

Nehmen wir eine Zahl, sagen wir 16. Die Quadratwurzel wird mit SQR so gezogen:

```
PRINT SQR(16)
```

Wenn wir diese Zeile eintippen, zeigt Ihr Computer auf der nächstunteren Zeile das Ergebnis 4 an.

Nun zeigen wir Ihnen in der folgenden Liste, über welche Funktionen Sie verfügen können. Wenn Sie Mathematik verstehen, dann lesen Sie's durch. Wenn nicht, dann machen Sie sich inzwischen eine Tasse Kaffee!

Eine Liste numerischer Funktionen

Funktion	Operation
ABS (X)	Bildet den absoluten (positiven) Wert von X
SGN (X)	-1, wenn X negativ +1, wenn X positiv 0, wenn X = 0
SQR (X)	
LOG (X)	
EXP (X)	Bildet den Wert e hoch X, wobei e = 2,71828
INT (X)	Ergibt den ganzzahligen Teil von X
RND (X)	X darf dabei nicht negativ sein.

SIN	(X)	der Wert der trigonometrischen Funktionen wird
COS	(X)	wird in rad angegeben, wobei X im Wertebereich von
TAN	(X)	-9999999 bis 9999999 liegen darf.
ATN	(X)	Gibt ARC TANGENS in rad.

11 Wie bricht man Tätigkeiten ab?

Je mehr wir von der Computer-Programmierung entdecken, desto aufregender wird es. Zum Atemholen wollen wir uns in diesem Kapitel wieder auf gewohntem Boden bewegen. Also entspannen Sie sich, holen Sie tief Luft, denn jetzt werden wir etwas mehr über unseren alten Freund "Looping" erfahren.

Aber bevor wir Ihnen sagen, was Sie noch nicht wissen, denken wir noch einmal darüber nach, was wir jetzt schon wissen.

Die erste Schleife, die wir kennenlernten, wurde bei dem nichtendenden Karussell mit GOTO an den Programmanfang, durch die Anwendung der BREAK-Anweisung angewendet, Sie wissen schon, die Art, die bis zum geht nicht mehr Schleifen zieht. Und wenn wir Ihnen nicht gezeigt hätten, wie Sie die Notbremse zu ziehen haben, würde Ihr Computer heute noch Schleifen ziehen Wir haben Ihnen auch das STOP-Statement erklärt das den Computer veranlasst, selbst die Bremsen zu ziehen.

Nun und wie war das mit der Schleife als wir unser Rate-Spiel-Programm schrieben? Ja, genau, die Schleife kam schon zum Ende, zu guter Letzt. Wenn der erste INPUT nicht stimmte, konnten Sie immer wieder neue Eingaben machen, bis Sie die Geheimnummer erraten hatten. Als Sie's dann fanden, war das das Ende der Fahrt.

Da wir aber nicht wußten, wieviele Versuche scheitern würden, bis die richtige Nummer erraten wird, konnten wir auch nicht sagen, wieviele Male die Schleife durchlaufen würde!

Wäre es nicht praktisch, wenn man die Schleifen in ihrer Anzahl begrenzen könnte? So als ob man Ihrem Computer eine Fahrkarte geben würde, die dem Computer sagen würde, wieviele Schleifen er zu ziehen hätte, bis er "aussteigen" müßte. Es ist nun zufällig so, daß man auf zweierlei Art und Weise Schleifen programmieren kann, die eine Anzahl-Begrenzung haben.

Begrenzte Schleifen unter Anwendung von IF-THEN

Ja, wo wären wir ohne die wunderbare Anweisung namens IF-THEN. Hier ist eine weitere Art, wie sie uns helfen kann. Wie Sie wissen, veranlaßt IF-THENIhren Computer eine Entscheidung zu treffen. Warum sollten Sie also nicht Ihrem Computer befehlen, zu entscheiden, wie oft er die Schleife zu ziehen hat?

Wir haben versprochen, in diesem Kapitel auf gewohntem Boden zu bleiben. Also machen wir den Versuch auf etwas, das wir bereits kennen, um zu sehen, wie der neue Trick funktioniert und zwar auf unserem Einmaleins. (Das letzte Programm in Kapitel 7).

Tippen Sie das Programm wieder in den Speicher Ihres Computers ein:

```
5 CLS
10 A=1
20 PRINT A; " X 4 =";
30 PRINT A*4
40 A=A+1
50 GOTO 20
```

Sie wissen genau, was passieren würde, wenn dieses Programm abliefe? Natürlich wissen Sie's! Es wird den Wert der Variablen A jedesmal, wenn es durch die Schleife kommt, um 1 erhöhen.

Jetzt ehmen wir einmal an, daß wir nur bis 12 x 4 gehen wollten. Das ist einfach. Zum Schluß des Programms tippen Sie diese Zeile ein:

```
45 IF A=13 THEN END
```

Wie Sie wissen, fügt sich Zeile 45 zwischen den Zeilen 40 und 50 ein. Jetzt sieht unser Programm so aus:

```
5 CLS
10 A=1
20 PRINT A;" X 4 =";
30 PRINT A*4
40 A=A+1
45 IF A=13 THEN END
50 GOTO 20
```

Wollen Sie es in Aktion sehen? Sie wissen was zu tun ist: Sagen Sie ihm einfach RUN. (Und bereiten Sie sich auf einen Freudensprung vor).

Es funktioniert!

Danke liebes IF-THEN, wir haben soeben unsere erste begrenzte Schleife

programmiert. Und wie haben wir das gemacht? Vielleicht haben Sie es schon erraten.

> Durch die Hinzunahme der Zeile 45 haben wir den Computer dazu veranlaßt, jeden neuen Wert von A dort mit IF zu prüfen. Solange A immer noch kleiner war als 13, hat Ihr Computer einfach noch einmal die Schleife durcheilt.Aber: IF A (als) nicht mehr ungleich 13 war, THEN (dann) hat er entschieden, dieSchleife zu beENDen und damit gleichzeitig das Programm.

Sehen Sie, was dieses Programm bewirkt hat? Es hat Ihrem Computer eine Fahrkarte für das Schleifen-Karussell gegeben und zwar eine, auf der steht: "Dreh' Dich zwölfmal und dann steig' ab."

> Soviel zur Methode 1. Es hat wunderbar geklappt, oder etwa nicht? Wahrscheinlich sind Sie derart beeindruckt, daß Sie nicht abwarten können, die Methode 2 kennenzulernen.

Aber, werden Sie nicht ungeduldig! Bevor wir Ihnen die zweite Methode zeigen, müssen wir Ihnen drei brandneue BASIC Statements beibringen. Und das muß jetzt bis zum nächsten Kapitel warten, denn Sie haben einstweilen schon genug getan.

12 Noch beim "LOOPING": Das FOR-TO-NEXT

Sehen Sie die BASIC-Statements am Anfang dieses Kapitels? In diesem Kapitel werden wir Ihnen den zweiten Weg zeigen, wie Sie Ihrem Computer mit FOR TO NEXT eine Fahrkarte geben können, die ihm vorgibt, eine begrenzte Schleifen-Anzahl zu ziehen.

Als erstes: FOR-TO

Das erste, was die FOR-Anweisung sagt, ist: "Markiere einen Speicherplatz in deinem Speicher mit dem folgenden".

> (Bei dieser Übung nennen wir den Variablenplatz A. Aber Sie können jeden anderen Buchstaben dazu nehmen.).

Kommt Ihnen das bekannt vor? Nun, das sollte es schon, denn die gute alte LET-Anweisung macht nämlich genau das! Aber dort hört auch die Ähnlichkeit auf.

Die LET-Anweisung gibt dem Computer nur eine Zahl oder einen Wert zur *Speicherung* vor, beispielsweise:

```
LET A=7
```

Auf der anderen Seite kann die FOR-Anweisung dem Computer einen ganzen Zahlenbereich (z.B. die Zahlen von 1-12) zugleich vorgeben.

Sie könnten jetzt versuchen, die FOR-Anweisung so zu schreiben:

FOR A = 1 2 3 4 5 6 7 8 9 10 11 12

Tun Sie's bitte nicht! Es wird einfach nicht klappen. Und wenn Sie's trotzdem tun sollten, so wäre es doch recht aufwendig. Besonders, wenn Sie einengroßen Zahlenbereich nehmen würden. (Stellen Sie sich das Schreiben eines FOR-Statements für die Zahlen von 1-100 vor.) Nein, wir haben eine bessere Idee.Wäre es nicht einfacher das Wort TO zu benutzen, um Ihr FOR-Statement so zu schreiben:

FOR A=1 TO 12

Ja, das ist doch wesentlich sinnvoller. "Sinnvoll oder nicht, das wird nicht funktionieren!" werden einige von Ihnen sagen. "Diese ganzen Zahlen werden nicht auf einer Speicher-Stelle Platz finden, oder?"

Sie haben ganz recht, es würde nicht funktionieren. Eine Variable kann niemals mehr als einen Wert haben, zumindest nicht zur gleichen Zeit. Aber geraten Sie noch nicht in Panik! Was FOR A=1 TO 12 wirklich bedeutet, ist dies:

"OK, Computer, hör' gut zu. Ich habe 12 Zahlen, die ich auf Speicherstelle A gespeichert haben will. Du sollst den Wert von A 12 mal ändern,wobei bei jeder Änderung A gleich der nächsten (!) Zahl in der Gruppe zu sein hat."

Z.B.: Der erste Wert in A sei 1. Das nächste Mal ist er 2. Dann gleich 3. Die Werte werden nacheinander auf die Speicherstelle gebracht bis A=12.

(Ist Ihnen das klar? Das ist so, als würde der Computer eine Treppe mit 12 Stufen hinaufgehen und zwar dabei eine Stufe nach der anderen nehmen.)

Als nächstes kommt NEXT

Aha! Jetzt die Preisfrage: Wie bewerkstelligen wir die Veränderung desjeweiligen Wertes in der Variablen A?

Wir sind froh, daß Sie fragen. Sehen Sie, die FOR-TO-Anweisung "eröffnet" eine Schleife. Also benötigen wir am Ende der Schleife so etwas, wie einen "Verschluß", wobei gleichzeitig der Computer nochmal zurück zum Anfang geschicktwird, solange er seinen Job noch nicht vollendet hat.

Bisher hatten wir die GOTO-Anweisung zur Beendigung der Schleife herangezogen. Wie Sie wissen, ist sie eine recht praktische Anweisung. Und wenn wir sie am Ende unserer FOR-TO-Schleife einsetzen würden, würde sie den Computerauf jeden Fall zurück zum Anfang schicken. Da gibt es ein "Aber" und zwar ein ziemlich großes! Die GOTO-Anweisung sagt dem Computer aber nicht, daß er den Wert von A ändern soll.

Offensichtlich benötigen wir eine neue Anweisungsart. Und nun verhält es sich so, daß sich NEXT hier vortrefflich eignet.

NEXT setzen wir zur Beendigung der Schleife mit der Laufvariablen A, folgendermaßen ein:

```
NEXT A
```

Sie sagt Ihrem Computer folgendes:

> "Gehe zurück zu FOR-TO am Anfang der Schleife, wähle den nächsten (NEXT) Wert von A und beginne nochmal."

Ruhe im Studio, Aufnahme!

Nun, jetzt kennen wir FOR-TO-NEXT schon näher. Wollen Sie es in Aktion sehen? Ja, das dachten wir uns. Dazu wollen wir ein neues Programm erstellen. Warum spannen wir den Computer nicht dazu ein, uns wieder das Einmalvier zu rechnen - so, wie wir es im letzten Kapitel gelernt haben? Nur diesmal werden wir die FOR-TO-NEXT-Anweisung einsetzen, statt IF-THEN. Einverstanden? Ab die Post! Tippen Sie dieses Programm ein:

```
5 CLS
10 FOR A=1 TO 12
20 PRINT A; " X 4 =";
30 PRINT A*4
40 NEXT A
50 END
```

Jetzt kommt der große Augenblick! Geben Sie RUN ein und passen Sie auf:

Wenn Sie alles richtig hatten, dann sollte der Computer das Einmalvier bis4x12 gerechnet haben, genau so, als wir dasselbe mit IF-THEN taten. Sind Sie beeindruckt? Wenn ja, dann bleiben Sie auf diesem Sender, denn der nächste Streich folgt sogleich (jedenfalls aber das nächste Kapitel!) Denn über FOR-TO-NEXT gibt es noch einiges zu lernen.

13 Mehr über FOR-TO-NEXT

Als wir das erste Mal FOR-TO-NEXT erwähnten, haben Sie bestimmt nicht

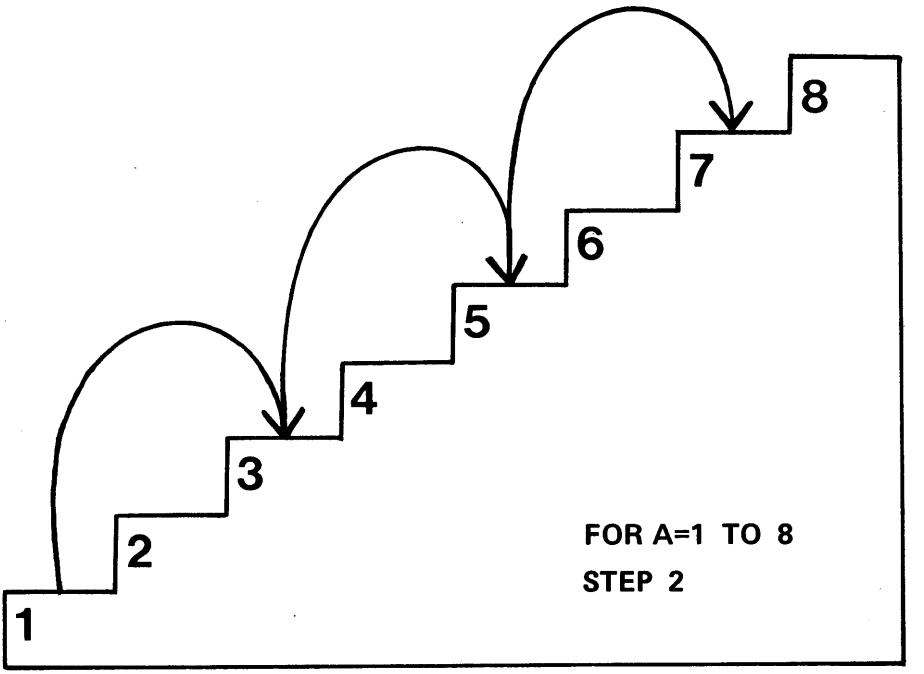
geahnt, wie mächtig diese Anweisung ist. Und nun noch etwas, um Sie in Staunen zu versetzen: Indem wir noch ein BASIC-Wort anhängen, können wir FOR-TO-NEXT dazu bringen, noch tollere Kapriolen zu schlagen.

Nehmen Sie nicht manchmal zwei Stufen, wenn Sie Treppen hochlaufen? Nun, unser nächstes Wort ist STEP und damit wird der Computer in die Lage versetzt, dasselbe zu tun und sogar besser, als es ein Mensch vermag! Mit STEP können Sie Ihren Computer veranlassen, zwei, drei, zehn oder mehr Stufen mit einem Schritt zu nehmen, wenn Sie's wollen.

Hier ist ein Beispiel für die Anwendung von STEP:

```
FOR A=1 TO 12 STEP 2
```

Der Computer macht dann folgendes:



Eine zusätzliche STEP-Eigenschaft ist es, eine FOR-TO-NEXT-Programmschleife auch umgekehrt laufen zu lassen. Dabei wird die höchste Zahl zuerst angesprochen und dann weiter in absteigender Reihenfolge werden die Werte geändert, bis man bei der kleinsten Zahl, der vorgegebenen Untergrenze, ankommt. Erstaunlich! Hier ist ein Beispiel:

```
FOR  A=12 to 1 STEP -1
```

Und das macht Ihr Computer:

Eine Rückwärtsschleife? Das müssen wir uns ansehen! Tippen Sie dieses Programm ein:

```
 5 CLS
10 FOR A=12 TO 1 STEP -1
20 PRINT A; "BUNTE KEGEL,"
30 PRINT "DIE STEHEN SO HERUM,"
40 PRINT A; "BUNTE KEGEL,"
50 PRINT "DIE STEHEN SO HERUM,"
60 PRINT "UND WENN EIN KEGEL WACKELT,"
70 PRINT "DANN FÄLLT ER AUCH GLEICH UM,"
80 PRINT "DANN SIND NOCH "; A-1;"BUNTE KEGEL,"
90 PRINT "DIE STEHEN SO HERUM!"
100 NEXT A
110 END
```

Dieses kleine Lied kennt natürlich nicht jeder. Aber an manchem "bunten" Kegelabend ist es durchaus zu hören. Es beginnt mit 12 bunten Kegeln, wobei Vers um Vers ein Kegel umfällt (je später der Kegelabend, desto "zufälliger" kann das Umfallen sein!) Jeder neue Vers beginnt mit einem Kegel weniger als der vorherige und wenn alle 12 Kegel umgefallen sind, ist das Lied zuende.

Bei genauerem Hinsehen:

Wir wissen bereits, was die FOR-TO-Anweisung in Zeile 10 tun wird. Der erste Wert, der in der Variablen A steht, ist 12, der nächste 11 u.s.w. bis A=1.

Die folgenden 4 Programmzeilen zeigen den momentanen Wert von A an, gefolgt von zwei Verszeilen.

Zeilen 60 und 70 sagen dem Beobachter, daß ein Kegel gleich umfällt. Zeile 80 hat mehrere Aufgaben:

* Sie berechnet die Restanzahl an Kegeln, wenn 1 von A subtrahiert wird und * zeigt dann die Zahl in der Mitte einer Verszeile an.

Zeile 100 schließt die Schleife ab und schickt das Programm zurück zur FOR-TO-Anweisung am Anfang, wo der nächste Wert auf A gespeichert wird.

Wenn der Computer 12-mal die Schleife durchzogen hat (und

dabei alle 12 Verse des Reims gedruckt hat) beendet er mit Zeile 110 das Programm.

Also worauf warten wir? Machen wir einen RUN und schauen, wie eine umgekehrte FOR-TO-NEXT-Schleife abläuft.

Oh, das war etwas enttäuschend! Es funktioniert zwar, aber vielleicht funktioniert es etwas zu gut, was meinen Sie? Wenn all diese Zeilen über den Bildschirm flitzen, ist es schwer, die Verse überhaupt richtig zu lesen. Unglücklicherweise ist das einer der Nachteile von Computern. Sie tun genau das, was wir Ihnen sagen, aber manchmal sind sie so verflixt schnell, daß wir Menschen gar nicht mehr mithalten können.

Damit wir dieses Programm etwas verlangsamen, müssen wir dafür Sorge tragen, daß der Computer gelegentlich Pausen einlegt. Um uns zusätzlich Zeit zu "kaufen", müssen wir dem Computer etwas geben, womit er sich zwischenzeitlich beschäftigt. Während dessen können wir ablesen, was auf dem Bildschirm steht.

Es gibt einen Weg. Betrachten Sie die untenstehende Zeile:

```
FOR T=1 TO 3000:NEXT T
```

Wir wissen, daß eine FOR-TO-Anweisung eine Schleife eröffnet und eine NEXT-Anweisung diese wieder schließt. Also ist die oben gezeigte Schleife leer(!), denn sobald wir sie eröffnen, haben wir sie auch wieder geschlossen. Sie hat keine besondere ausführende Aufgabe. Sie schickt den Computer lediglich "eine Treppe hinauf" und zwar 3000 Stufen. Aber weil Ihr Computer so schnell ist, brauchen wir eine lange Treppe, um ihn lange genug beschäftigt zu halten.

Wo setzen wir diese Schleife ein? Nun, es wäre logisch, sie nach jeder Verszeile einzusetzen. also LISTen Sie Ihr Programm, damit wir die Stelle suchen können.

Dort ist es! Zeile 90 druckt: "DIE STEHEN SO HERUM". Das ist das Ende des Verses. Also fügen wir unsere Leerschleife zwischen den Zeilen 90 und 100 ein und bezeichnen die Programmzeile mit 95. Dadurch wird der Computer etwas innehalten, bevor er seine Schleife zurückzieht, um die nächste Zeile anzuzeigen.

Versuchen Sie RUN, um zu sehen, ob dies hilft.

Ja! Am Ende jeden Verses, während der Computer die 3000 Treppen hinaufspurtet und nirgends ankommt, haben Sie Zeit, den Vers zu lesen.

Solche Schleifen nennt man *Verzögerungs-Schleifen*, denn sie verzögern die Verarbeitung im Computer.

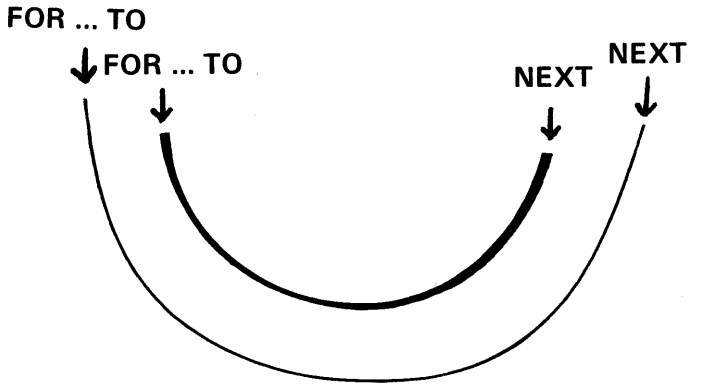
Jetzt etwas über "verschachtelte FOR-TO-NEXT-Schleifen".

Das hört sich aber furchtbar technisch und verwirrend an? Wir haben eine Überraschung für Sie: So etwas haben wir bereits angewendet, und zwar jetzt eben! Unsere "Verzögerungs-Schleife" ist nämlich gleichzeitig eine "verschachtelte Schleife" (Schleife innerhalb einer anderen Schleife).

Mit "verschachtelt" soll ausgedrückt werden, daß eine Sache völlig in einer anderen enthalten ist.

Merken Sie, worauf wir hinauswollen? Eine verschachtelte FOR-TO-THEN-Schleife befindet sich innerhalb einer anderen FOR-TO-NEXT-Schleife.

Sollten wir versuchen, eine zu zeichnen, würde es etwa so aussehen:



(Übrigens: Eine verschachtelte FOR-TO-NEXT-Schleife muß nicht unbedingt leer sein. Sie kann sehr wohl einen Inhalt haben, nur muß die ganze Schleife innerhalb einer anderen liegen.

Preisfrage: Ist Ihnen an Zeile 95 etwas ungewöhnliches aufgefallen? Nein? Sehen Sie genauer hin:

```
FOR T=1 TO 3000 : NEXT T
```

Auflösung: Zeile 95 enthält 2 BASIC-Statements. Das erste ist FOR-TO,das zweite ist NEXT. Meistens erscheinen beide Statements getrennt in jeweils einzelnen Zeilen. Wenn aber beide in einer Zeile erscheinen sollen, müssen sie durch einen Doppelpunkt getrennt sein!

14 Programme innerhalb anderer Programme

Obwohl der Computer ziemlich clever ist, haben seine Fähigkeiten Grenzen. In der Tat ist die Zahl der Programmzeilen, die er sich merken kann, begrenzt. Denn jede Zeile belegt Platz im Computer-Speicher.

Und, wenn es eine Sache gibt, die ein Computer-Programmierer haßt, dann ist es, Speicherplatz zu verschwenden. (Besonders, wenn die Speicherkapazität für ein sehr, sehr langes Programm benötigt wird.) Jetzt, wo Sie sich zu einem Programmprofi entwickeln, werden Sie wahrscheinlich zustimmen.

Wissen Sie, wie man am besten Speicherplatz verschwendet? Indem man sich unnötigerweise wiederholt. Sie werden gelegentlich beim Eintippen eines Programms in Ihren Computer bemerken, wie sich Wiederholungen einschleichen.

Als Beispiel diene unser "KEGEL"-Programm aus Kapitel 13. Was sagen Sie? Sie dachten, es wäre ein ziemlich tolles Programm? Wir auch. Und das ist es auch wirklich! Aber perfekt ist es noch nicht, wie Sie sehen werden, wenn Sie die Zeilen 20, 30, 40 und 50 betrachten.

> Echt schlimm! 20 und 30 tun genau dasselbe, wie 40 und 50. Das ist jetzt ein gutes Beispiel für die Verschwendung von Programmzeilen.

Dieses Kapitel wird Ihnen zeigen, wie Sie derartige Wiederholungen vermeiden können und zwar durch die Verwendung der BASIC-Kommandos GOSUB und RETURN.

Schließen Sie die Augen und denken Sie angestrengt nach, versuchen Sie sich an Kapitel 3 zu erinnern. Erinnern Sie sich daran, wie wir Ihnen das Programm beschrieben? Genau: Wir sagten, daß ein Programm eine Reihenfolge einzelner Befehle darstellt, die eine bestimmte Aufgabe ausführen.

> Hier ist ein sehr interessanter Gedanke. Die Zeilen 20 und 30 stellen eigentlich auch eine Reihenfolge von Schritten dar, obwohl es sich dabei nur um zwei Zeilen handelt.

Erkennen Sie die besondere Aufgabe dieser beiden Programm-Schritte? Sie sagen dem Computer, daß er zwei Zeilen des "KEGEL"-Liedes auf dem Bildschirm zeigen soll. (Und die Zeilen 40 und 50 sind nichts anderes als eine Wiederholung der Zeilen 20 und 30, weil sie genau gleiche Anweisungen beinhalten.)

Eine Mini-Reihenfolge von Schritten? Das ist doch eine Art "Mini"-Programm!

Eigentlich eine Verschwendung, dieselben Schritte zweimal zu tippen. Denken Sie gerade, was wir denken? Veilleicht könnte man einfach mit einem versteckten Mini-Programm dasselbe erreichen. Ja, das hört sich gut an! Aber für ein Programm innerhalb eines Programms muß es auch eine Bezeichnung geben - was glauben Sie, wie man das nennt? Ein Sub-Programm vielleicht? Kommt nahe heran! Man nennt es englisch "Subroutine", deutsch *Unterprogramm*.

Ein Unterprogramm ist ein Teil des Hauptprogramms, aber es ist auch davon getrennt. Unterprogramme können überall in ein Programm eingebaut werden, je nachdem, was sie bewirken sollen. Wir müssen den Computer dazu veranlassen,

* zum Unterprogramm zu springen und die dort vorzunehmende Funktion auszuüben und dann

* wieder zurück zur nächsten Zeile des Hauptprogramms zu springen.

Wir haben beinahe vergessen, Ihnen etwas zu sagen: Sie können sich wahrscheinlich noch nicht vorstellen, wie man ein Unterprogramm vom Hauptprogramm trennt.

Die einfachste Art, zwei Dinge voneinander getrennt zu halten, ist es, dazwischen ein Hindernis aufzubauen. Akzeptiert? Damit ein Unterprogramm mit dem Hauptprogramm nicht in Konflikt gerät, vergibt man Zeilenziffern an das Unterprogramm, die wesentlich höher sind, als letzte Zeilenziffer des Hauptprogramms. (Die letzte Zeile des Hauptprogramms enthält ja die END-Anweisung.) Versuchen Sie, Ihr Unterprogramm ganz unten in Zeile 3000 zu beginnen. Konflikte kann es dann nicht geben, denn der Computer beENDet das Hauptprogramm, bevor das Unterprogramm erreicht wird.

Somit dient die END-Anweisung des Hauptprogramms als Hindernis, damit das Unterprogramm keinen Schaden anrichten kann.

Das Errichten von Unterprogrammen unter Verwendung von GOSUB und RETURN.

Ihren Computer dazu zubekommen, zu einem Unterprogramm und zurück zuspringen, ist kein Problem, Sie müssen

es nur nur aufrufen. Dabei gibt es nur einen Haken - Sie müssen richtig aufrufen. Und da kommen dann unsere beiden BASIC-Worte zum Zuge - GOSUB und RETURN.

Ja, so ist es! GOSUB ist eine besondere Art des GOTO-Kommandos und veranlaßt Ihren Computer bis zum Anfang des Unterprogramms zu gehen. Und weil es keinen Zweck hat, dem Computer zu sagen, daß er gehen soll, wenn man ihm nicht gleichzeitig sagt wohin, *muß nach GOSUB immer eine Zeilenziffer stehen.*

Um Ihren Computer zu einem Unterprogramm in Zeile 3000 zu schicken, würden Sie eintippen:

GOSUB 3000

Vielleicht wundern Sie sich darüber, daß wir uns mit GOSUB beschäftigen, wenn GOTO dasselbe leisten würde.

In der Programmierwelt hat alles einen Grund. Im vorliegenden Fall wird GOTO nicht funktionieren! GOTO sagt nur: "Springe zu Zeile 3000 - sofort!" Aber GOTO warnt den Computer nicht, daß er zu einem Unterprogramm geschickt wird.

Weil der Computer so kooperativ ist, springt er natürlich sofort. Ohne dabei zu erinnern, von wo aus er gesprungen ist. Das gäbe ein Problem, denn wie soll er wissen, wie er wieder zurückkommt?

Es sieht so aus, als würden wir doch GOSUB benötigen, denn diese Anweisung warnt den Computer. Sie sagt: "Paß auf! Du wirst jetzt zu einem Unterprogramm geschickt, stelle sicher, daß du den Weg wieder hierher zurück findest."

Nun wissen wir, wie man den Computer zu dem kleinen separaten Programm schickt. Und was passiert, wenn das Unterprogramm abgearbeitet ist? Jeder weiß doch, daß er dann zurück zum Hauptprogramm springt.

Na ja, nicht jeder. Sie wissen's, wir wissen's, nur der Computer weiß es nicht. Wie gesagt, der Computer weiß nur, was man ihm sagt. Deswegen müssen Sie Ihr Unterprogramm mit einem RETURN-Befehl abschließen. Es ist so, als würde man sagen:

"Das ist das Ende dieses Unterprogramms. Kehre zum Hauptprogramm zurück, und zwar zu der Zeile, die unmittelbar dem GOSUB-Kommando folgt, das dich hergeschickt hat."

Jetzt, wo wir die Grundsätze des Unterprogramms verstanden haben (und wie man sie anwendet), wollen wir unsere neuen Fähigkeiten ausprobieren und unser "KEGEL-LIED"-Programm verbessern.

Ein Unterprogramm, das die Aufgabe der Zeilen 20 und 30 übernimmt (und der Zeilen 40 und 50) würde so aussehen:

```
3000 PRINT A; "BUNTE KEGEL,"
3010 PRINT "DIE STEHEN SO HERUM,"
3020 RETURN
```

Wenn wir unser Lied-Programm "up to date" machen, dann sieht die neue Version so aus:

```
5 CLS
10 FOR A=10 TO 1 STEP -1
20 GOSUB 3000
30 GOSUB 3000
40 PRINT "UND WENN EIN KEGEL WACKELT,"
50 PRINT "DANN FÄLLT ER AUCH GLEICH UM,"
60 PRINT "DANN SIND NOCH"; A-1 ;"BUNTE KEGEL,"
70 PRINT "DIE STEHEN SO HERUM!"
100 NEXT A
110 END
3000 PRINT A;" BUNTE KEGEL,"
3010 PRINT "DIE STEHEN SO HERUM,"
3020 RETURN
```

Sehen Sie, wie das funktionieren wird? Damit wir sicher sind, daß Sie es sich bildhaft vorstellen können, würden wir Ihnen gerne ein Bild zeigen!

Betrachten Sie die untenstehenden Diagramme (Sie sollen alles klar erscheinen lassen, d.h. ziemlich klar!).

Die Zeilen 20,30,40, und 50 des alten Programms sind durch zwei Zeilen ersetzt worden - durch eine neue Zeile 20 und eine neue Zeile 30. Außerdem sind diese Ersatzzeilen kürzer, darin steht nämlich nur GOSUB 3000.

Wenn Sie Ihrem Computer mit RUN den Startschuß für (Auf die Plätze-fertig-RUN) das neue Programm geben, beginnt er genauso, wie beim alten Programm, bis zur Zeile 20 kommt!

Zeile 20 schickt den Computer zur Subroutine 3000. Ihr Computer flitzt durch die Subroutine und kehrt dann zu Zeile 30 des Hauptprogramms zurück.

Zeile 30 enthält zufällig auch ein GOSUB 3000- Kommando. Alos läuft der Computer wieder zur Subroutine. Diesmal kehrt er zurück zu Zeile 40, bereit mit dem Rest des Programms fortzufahren.

Das war's! Unterprogramme sind wirklich einfach in der Handhabung. Und sie eignen sich gut, wie wir gerade gesehen haben, der Verschwendung von Speicherplatz entgegen zu wirken. (Es ist wirklich so, daß Unterprogramme unbezahlbare Dienste leisten können, wenn bestimmte Schrittfolgen immer wieder anfallen!)

Noch besser:

Wie ist es, wenn man mehrere verschiedene Schrittfolgen in einem Programm vorfindet? Ganz einfach, man bildet Unterprogramme, um auch diese Aufgaben zu übernehmen. Unterprogramme kann man in unbegrenter Anzahl einsetzen. (Sie müssen nur jedes einzelne Unterprogramm mit anderen Zeilenziffern ausstatten. Z.B., das erste Unterprogramm mit 3000, das zweite mit 4000, die nächsten mit 5000!)

Am besten: Erinnern Sie sich an das, was wir über die verşchachtelten FOR-TO-NEXT-Schleifen sagten? Das gleiche trifft auf Unterprogramme zu, das ergibt dann *"verschachtelte Subroutinen"*. Beispiel:

Sie können Ihren Computer veranlassen, vom Hauptprogramm bis zum Unterprogramm 300 zu laufen. Und Unterprogramm 3000 kann ein GOSUB-Kommando enthalten, welches eine Weiterleitung zum Unterprogramm 4000 beinhaltet. Noch ein GOSUB-Kommando im Unterprogramm 4000 führt weiter zum Unterprogramm 5000!

Die gleichen GOSUB-RETURN Regeln kommen zur Anwendung, egal wieviel Unterprogramme verschachtelt sind: Ein RETURN-Kommando schickt den Computer zurück zu der Zeile, die unmittelbar auf den GOSUB-Befehl folgt, der ihn in das Unterprogramm entsandt hatte!

15 Stringvariable

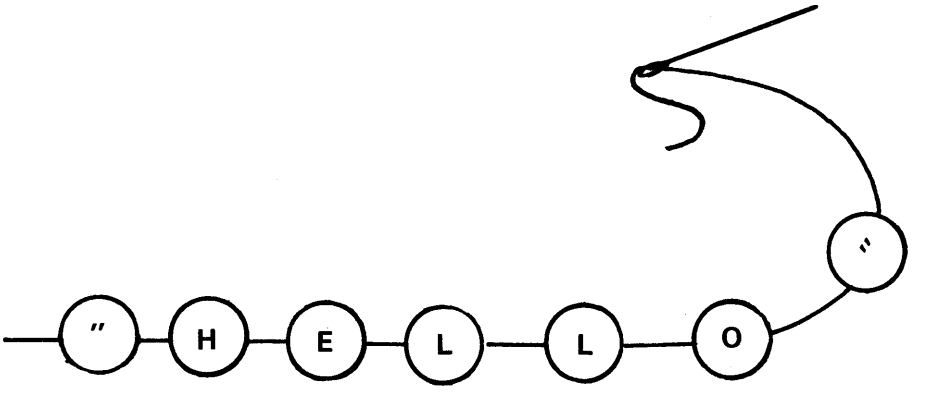
Dieses Kapitel werden sie lieben. Und das, was es für Ihre Programme tun kann, werden Sie auch lieben! Denn jetzt werden wir etwas neues und aufregendes über einen neuen Typ vor Variablen lernen.

Wir wissen bereits, daß die "Abstellplätze" im Speicher des Computer speziell für die Abspeicherung verschiedener Sachen geeignet sind. Zahlen haben wir bereits mit Erfolg abgespeichert. Aber es waäre doch jammerschade, wenn wir sie nur dazu verwenden würden.

Haben Sie schon einmal gefragt, ob man diese Speicherstellen zur Spei-

cherung anderer Dinge benutzen könnt? Z.B. Worte oder sogar Sätze? Das wäre bestimmt sehr praktisch.

Hier ist nun eine gute Nachricht: Ja, man kann! Das ist schon recht erstaunlich, aber es ist trotzdem bestimmt möglich. Dabei gibt es sogar eine eigene Bezeichnung für solche Variablen. Man nenne sie Stringvariable. (String heißt englisch 'Kette'. Wir sagen deutsch 'Textkette'!)



Die Bezeichnung mag einem zunächst etwas eigenartig vorkommen. Aber, wenn Sie genau darüber nachdenken, werden Sie erkennen, daß sie durchaus sinnvoll ist. Wenn man diese Variablen in eine für die Verarbeitung geeignete Form bringt, tut man nichts anderes, als Perlen auf einer Schnur zu einer Kette aufzuziehen. Und das ist ziemlich einfach.

Eine Datenverkettung wird genauso zusammengezogen. Man verkettet dabei nur eine bestimmte Menge von Buchstaben, Leerstellen und anderen Zeichen, wie Perlen einer Kette.

Um eine Variablen-Verkettung zu erstellen, muß man zusätzlich einen Speicherplatz markieren, auf dem man dann die Verkettung speichern kann!

Regeln und Vorschriften

Wie bei den anderen neuen Dingen, die wir gelernt haben, sind auch hier bestimmte Regeln zu beachten. Aber, keine Sorge, sie sind nicht schwer zu erlernen.

Eine Textkette darf nicht länger als 255 Zeichen sein. (incl. Leerstellen). Warum, fragen Sie? Nun, die Speicherplätze haben auch einen begrenzten Umfang. Ein LKW fände auf einem Tiefgaragen-Parkplatz auch nicht genügend Platz!

Der Name, den man der Textkette zuschreibt, muß immer mit einem $-Zeichen enden. (Nein! Das bedeutet nicht, daß sie etwas mit Geld zu tun haben! Das $-Zeichen besagt lediglich, daß auf dieser Speicherstelle eine Textkette und nicht Zahlen gespeichert sind.)

Beispiel: A$, B$, C$ und D$ sind mögliche Namen für Textketten.

Eine Textkette muß beim Eintippen in den Computer immer von Anführungszeichen eingeschlossen sein! Beispiel:

```
A$="EENIE"
B$="MEENIE"
C$="MINIE"
D$="MO"
```

Die neuen Variablen sind genauso zu verarbeiten, wie gewöhnliche Variable,einer Ausnahme:

Man kann nur eine einzige mathematische Operation auf sie anwenden, nämlich die Addition, sprich: Verkettung, Concatenierung.

Will man Textketten addieren, so gibt man dem Computer folgende Anweisung:

```
PRINT A$+B$+C$+D$
```

Und nach RUN sieht das Ergebnis so aus:

EENIEMEENIEMINIEMO

Sehen Sie, wie der Computer verkettet, ohne zwischen den einzelnen Variablen Leerstellen zu lassen? Wenn Sie Leerstellen haben wollen, dann müssen Sie diese innerhalb der Anführungszeichen einfügen

Sie haben sich inzwischen wahrscheinlich alle möglichen Dinge ausgedacht, wie Sie diesen Trick anwenden können. Denn schließlich ist es so, daß bei der leisesten Erwähnung bestimmter Variablen-Namen, Ihr Computer ganze Worte, bzw. Sätze "sprechen" wird.

Wollen Sie's mal probieren? Wir auch. Versuchen wir, den Computer dazu zu bringen, mit einer dritten Person "ein Gespräch" zu führen.

Tippen Sie dieses Programm ein:

```
 5 CLS
10 PRINT "HALLO! ICH BIN IHR Computer."
```

```
 20 PRINT "WIE IST IHR NAME";
 30 INPUT N$
 40 PRINT "HALLO,";N$
 50 PRINT "ES FREUT MICH SEHR, SIE KENNEN ZU LER-
NEN."
 60 PRINT "IN WELCHEM STADTTEIL"
 70 PRINT "WOHNEN SIE";
 80 INPUT S$
 90 PRINT "WOHNEN SIE GERNE IN"
100 PRINT S$;
110 INPUT A$
120 IF A$="NEIN" THEN GOTO 150
130 PRINT "SCHOEN! ZUHAUSE IST'S AM SCHOEN-
STEN!"
140 END
150 PRINT "OH JE! VIELLEICHT SOLLTEN SIE UMZIE-
HEN!"
160 END
```

Wenn Sie das Programm durcharbeiten, sollten Sie sich selbst vorstellen können, wie es funktioniert.

Einige Tips: Zeile 20 bittet Ihren Freund, seinen Namen einzugeben (INPUT). Zeile 30 "hört sich" seine Antwort an und speichert die Information in der Textkette mit Namen N$. Die Zeilen 50, 60 und 70 tun ungefähr dasselbe. Ihr Freund wird nach seinem Wohnort gefragt, die Antwort wird in S$ gespeichert.

Wenn diese Informationen sicher im Computer verstaut sind, verwendet sie der Computer zur Führung eines recht persönlichen "Gesprächs". (Vielleicht täuschen Sie Ihren Freund derart, daß er denkt, daß der Computer wirklich einen eigenen Verstand hat!)

Eine gutgemeinte Warnung

Die Höflichkeit Ihres Computers, Ihre Freunde zu beeindrucken, ist gut und schön. Aber Vorsicht! Wenn der Computer zu charmant wird, könnten Sie Schwierigkeiten bekommen, sie von ihm zu trennen.

16 Variable in READ und DATA

Bevor wir mehr über Computer sagen, möchten wir, daß Sie eine kleine Übung durchführen.

Sind Sie auf alles vorbereitet? Dann befolgen Sie die folgenden Instruktionen genau:

1. Stehen Sie auf
2. Klopfen Sie sich auf die Schulter. (Ja, das ist zwar etwas schwierig, aber es geht schon, wenn Sie dabei etwas Gelenkigkeit entfalten).
3. Werfen Sie sich in die Brust, lächeln Sie und rufen Sie: "Ich hab's geschafft!"

Tut das nicht gut? Und glauben Sie nicht, daß Sie's verdient haben? Seien Sie nicht bescheiden: Lob dem, dem Lob gebührt. Angefangen haben Sie mit wackeligen Schritten, die Welt der Computer zu erforschen, und inzwischen haben Sie sich durch 15 ganze Kapitel dieses Buches durchgearbeitet.

Und jetzt...weiter im Text

Da eine Sache meistens zur nächsten führt, gehen wir in diesem Kapitel wieder zurück zu unserer Tiefgarage. Und was kommt als nächstes auf unsere "Lernliste"? Im letzten Kapitel haben wir über eine neue Sache gsprochen, die wir auf dem Parkplatz abgestellt hatten, die Textkette. Jetzt werden wir über eine neue Art der Speicherung sprechen.

> "Noch eine Art? Aber wir kennen schon zwei verschiedene Arten mit LET und INPUT. Sicherlich gibt es keine zusätzlichen?"

Seien Sie sich dessen nicht so sicher! Es gibt noch ein Ablagesystem, eines, das sich später als sehr praktisch erweisen wird.

Sehen Sie, sowohl LET als auch INPUT sind sehr, sehr nützlich. Aber eines Tages werden Sie eine große Informationsmenge abspeichern wollen. (Insbesondere dann, wenn Ihre Programme länger und komplizierter werden.) Nehmen wir an, Sie wollten 5 Variable erstellen, wobei Sie in jeder etwas haben wollten.

Sie könnten es folgendermaßen tun:

```
LET A=5
LET B=4
LET C=3
LET D=2
LET E=1
```

...aber das würde zuviel Zeit und Platz in Anspruch nehmen!

Also versuchen wir etwas völlig anderes, indem wir nur 2 Zeilen verwenden:

```
DATA 5,4,3,2,1
READ A,B,C,D,E
```

Wahrscheinlich läßt Sie das im Dunkeln. Erlauben Sie uns etwas Licht in das Geheimnis zu bringen.

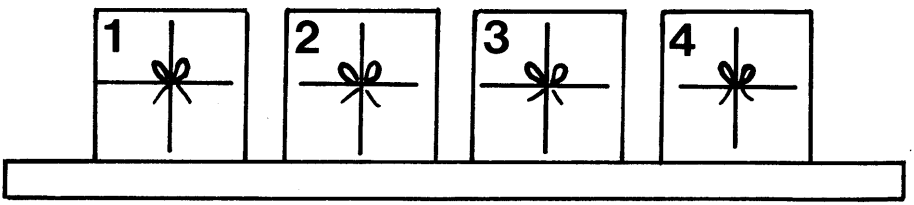
Vor einiger Zeit haben wir eine wichtige Regel bezüglich der Variablen erwähnt. Erinnern Sie sich hieran?:

> Der Computer wird nichts in seinen Speicher übernehmen, bis wir dafür eine Bezeichnung zuordnen, um den Speicherplatz zu markieren.

Bisher haben wir immer *einen Namen für eine Variable angegeben*, bevor wir dem Computer gesagt haben, was er darauf zu speichern hat. Aber dieses Mal, wird es etwas anders sein.

Die erste Zeile in dem oben angegebenen Paar beinhaltet 5 Einträge (in diesem Fall sind es Zahlen). Das Eintippen dieser Zeile ist so, als würden Sie an einem Hotelempfang eine Menge Pakete abgeben und sagen:

> "Hier...passen Sie zwischenzeitlich darauf auf. Später werde ich Ihnen sagen, wo sie abgestellt werden sollen." Diese Pakete müssen irgendwo untergebracht werden. Zum Beispiel in einem Regal.



> Eine Zeile, die mit einer DATA-Anweisung beginnt ist genau wie ein "Paket-Regal". Das ist ein Platz, wo Dinge aufbewahrt werden, bevor sie an ihren endgültigen Platz gelangen. (Übrigens, haben Sie die Kommata in der DATA-Zeile bemerkt? Sie werden dazu verwendet, die einzelnen Einträge auseinander zu halten).

Wenn es an der Zeit ist, die Pakete aus dem Regeal zu entnehmen, beginnt der Computer immer von links und arbeitet sich nach rechts durch. Computer ziehen es vor, in dieser Reihenfolge zu arbeiten, sie sind eigentlich pedantische Leute. (Soll heißen: Maschinen.)

Und nun zur zweiten Zeile, die mit READ beginnt.

Diese Zeile stellt eigentlich nur eine Liste der Variablen-Namen dar. Wir wollen, daß der Computer jeden einzelnen Namen dazu verwendet, einen individuellen Speicherplatz zu markieren. Noch einmal: *Die Kommata sind dazu da, jeden Eintrag von seinem Nachbarn zu trennen.*

Wenn Ihr Computer die READ-Liste liest, denkt er: "Oh, wie schön. Jetzt kann ich daran gehen, die Post zu sortieren." Schauen wir dem Computer über die Schulter, um ihn dabei zu beobachten, wie er mit diesem Ablagesystem arbeitet.

Der Name ganz oben auf der READ-Liste ist A. Also: Ihr Computer verwendet diesen Namen, um einen Abstellplatz damit zu markieren. Dann nimmt er das erste Paket aus dem DATA-Regal und speichert es auf Speicherplatz A.

Richtig, jetzt zu dem zweiten Namen auf der Liste. Das ist B. Und Ihr Computer wird das zweite Paket aus dem Regal auf dem Speicherplatz namens B abspeichern. Haben Sie erraten, was auf Speicherplatz C landet? Ja, das dritte Paket. Und so geht es weiter, bis der letzte Variablen-Name auf der READ-Liste benutzt wurde.

> Sehen Sie, wie effizient dieses System arbeitet? Der Computer wird sich Namen um Namen durch die READ-Liste hindurcharbeiten. Wobei er das nächste Paket vom DATA-Regal in jeden neuen Speicherplatz speichert, den er markiert.

Übrigens, Sie müssen nicht jedes einzelne Paket, das sich im DATA-Regal befindet, verwenden. Nehmen wir z.B. an, daß in Ihrer DATA-Zeile sechs Einträge sind, aber auf Ihrer READ-Liste sich nur fünf Namen befinden. Das macht Ihrem Computer gar nichts aus. Er wird einfach die fünf Pakete in seinen Speicher aufnehmen und die übrig gebliebenen ignorieren.

> Wenn wir diese beiden Zeilen in ein kleines Programm eingefügt hätten, könnten wir dann eine zusätzliche Zeile bringen, die den Computer zum Ausdruck der Werte veranlaßt
>
> PRINT A;B;C;D;E
>
> Und wenn wir dann unser Programm laufen ließen, würde folgendes auf unserem Bildschirm erscheinen:

5 4 3 2 1

Merken Sie was? Wenn Sie Ihrem Computer sagen würden, er solle PRINT E;D;C;B;A dann hätten wir folgendes auf dem Bildschirm gesehen

1 2 3 4 5

Es gibt etwas zu beachten: Sie müssen sicherstellen, daß Sie genügend Daten-Pakete zur Abspeicherung in den dafür vorgesehenen, markierten Speicherplätzen haben. Wollen Sie ein Beispiel sehen? Dann ersetzen Sie die zweite Zeile unseres Beispiels mit folgender

20 READ A,B,C,D,E,F

Versuchen Sie jetzt noch einmal das Programm mit RUN laufen zu lassen. Anfangs wird alles genau richtig laufen. Bis Ihr Computer bei dem letzten Variablen-Namen der Liste angelangt ist. Er wird gehorsam einen Speicherplatz mit F markieren. Aber wenn er versucht noch ein Paket dem DATA-Regal zu entnehmen. Oh je! Das Regal ist leer! Sie haben sechs Abstellplätze genannt, aber es waren nur fünf DATA-Pakete. Es ist keines mehr für den Speicherplatz F übrig.

Wenn so etwas passiert, wird Ihr Computer sagen: "Wie soll ich das denn machen? Siehst Du nicht, daß ich OUT OF DATA bin?"

Die Moral von der Geschicht':

Benützen Sie niemals mehr Namen auf Ihre READ-Zeile, als Einträge auf Ihrer DATA-Zeile. (Benötigen Sie mehr als ein DATA-Regal, um Ihre Pakete zu speichern, dann tun Sie's. Sie können soviele benützen, wie Sie wollen.)

Hier ist eine weitere nützliche Information: Wenn Sie wollen, können Sie auf der READ-Zeile nur einen Variablen-Namen (sagen wir D) einsetzen. Dann können Sie den Computer bis zum Anfang des Programms zurückschleifen lassen. (Wie, indem Sie natürlich eine FOR-TO-NEXT-Schleife verwenden.)

Jedesmal, wenn Ihr Computer dann zu einer READ-Zeile kommt, wird er von früher, die in D vorhandenen Daten mit den nächsten vom DATA-Regal ersetzen.

Und wie finden Sie heraus, wieviele Male der Computer durchschleifen muß. Einfach! Ihr Computer muß für jeden Eintrag in der DATA-Zeile eine Schleife machen.

Um Ihnen zu beweisen, wie nützlich dieses neue System sein kann, möch-

ten wir, daß Sie als Beispiel das folgende Programm eintippen. Aber zuerst wollen wir Sie unserem imaginären Freund vorstellen:

Herrn Vorsicht

Herr Vorsicht ist Ladeninhaber. Wie die meisten Geschäftsleute, überprüft er, ob ein Kunde zuverlässig ist oder nicht. (d.h. er überprüft den Kunden, ob er seine Rechnungen bezahlt.)

Dieses Programm soll diese Überprüfung für ihn erledigen. Denn es ist nämlich so, daß Herr Vorsicht jedem Kunden eine besondere Kundennummer zuteilt. Und in diesem Programm sind die Kunden aufgeführt, die man als schlecht zahlende Kunden bezeichnet.

Herr Vorsicht kann per INPUT jede Kundennummer, die er überprüft haben will, überprüfen. Und der Computer wird ihm sagen, ob der Kunde auf der "schwarzen Liste " steht oder nicht.

Hört sich clever an, oder nicht? Schauen wir, ob es funktioniert!

```
  5 CLS
 10 DATA 11067, 12549, 22871, 57824, 86256
 20 INPUT "KUNDEN-NUMMER";N
 30 FOR L=1 TO 5
 40 READ D
 50 IF D=N THEN GOTO 90
 60 NEXT L
 70 PRINT "DIESE KUNDEN NUMMER IST OK"
 80 END
 90 PRINT "NEIN! DIESE KUNDEN-NUMMER IST FAUL!"
100 END
```

Etwas verwirrt? Machen Sie sich nichts 'draus. Bei der Analyse werden Sie merken, wie einfach es ist.

Zeile 5:

Sie gibt uns einen schönen, klaren Bildschirm, mit dem wir dann nach RUN alles klar sehen werden.

Zeile 10:

Unser DATA-Regal: Hierin sind 5 Einträge gespeichert. Und merken Sie sich: Jedes "Datenpaket" ist in Wirklichkeit die Nummer eines Kunden der seine Rechnungen nicht bezahlt hat.

Zeile 20:

Eine solche Zeile haben wir bisher noch nicht gehabt. Dies ist wirklich nur eine gewöhnliche INPUT-Anweisung. Aber wir haben ihr nur

etws Gewürz beigemischt, indem wir eine Einfüge-Zeile verwendet haben, die den Anwender auffordert, seine Antwort zu INPUTen.

Zeile 30:

Hier wird die Schleife unseres Dateneinlesens begonnen. Weil wir fünf verschiedene Einträge auf der DATA-Zeile haben, werden wir genauso oft durchschleifen müssen. (Den Namen L haben wir nicht aus einem besonderen Grund zugeordnet, sondern nur weil L für LOOP steht.)

Zeile 40:

Wir haben Ihnen gerade erklärt, welche Funktion diese Zeile hat. Bei der ersten Schleife veranlaßt die READ-Zeile den Computer das erste Paket aus dem DATA-Regal auf einen mit D markierten Speicherplatz zu speichern. Beim zweiten Mal speichert er das zweite Paket u.s.w.

Zeile 50:

Durch die Verwendung unserer altbewährten IF-THEN-Anweisung, wird die Kunden-Nummer zwecks Überprüfung auf N eingespeichert. IF sie gleich ist mit der Kundennummer in D, THEN dann wird Ihr Computer mit GOTO zu Zeile 90 gehen. Zeile 90 druckt eine Nachricht als Information daß sie auf der "schwarzen Liste" ist. Und unmittelbar nach dieser Zeile, folgt Zeile 100, die das Programm beENDet. Aber wenn N nicht gleich D ist, wird der Computer zur nächsten Zeile gehen und das Programm fortsetzen.

Zeile 60:

Hier befindet sich die NEXT-Anweisung, die die Schleife schließt. Sie schickt den Computer zurück zum Anfang der FOR-TO-NEXT-Schleife. Dann ersetzt sie die bereits in D befindliche Zahl mit dem zweiten Paket aus dem DATA-Regal, bereit, sie mit N zu vergleichen. Diese Zeile hält den Computer solange auf der Schleife, bis er zum fünften Mal ge- kreist hat. Erst danach ist er frei, zur Zeile 70 zu gehen.

Zeile 70:

Wenn Ihr Computer soweit gekommen ist, dann sind alle faulen Kundennummern abgeprüft. Diese Zeile wird eine Nachricht drucken, die besagt, daß die Nummer OK ist.

Zeile 80:

Diese spricht für sich selber.

Versuchen Sie es selbst! Beim ersten RUN des Programms denken Sie

sich eine Kunden-Nummer aus, die Sie eingeben(INPUT):

Funktioniert es? Großartig! Jetzt versuchen Sie es nochmal. Aber dieses Mal geben Sie eine Nummer ein, von der Sie wissen, daß Sie auf Herrn Vorsichts "schwarze Liste" steht. (Nur um sicher zu gehen, daß auch dieser Teil des Programms funktioniert.)

17 Zusätzliches über DATA und READ

Wir werden Ihnen jetzt noch hilfreiche Dinge bezüglich DATA und READ erzählen.

Nachdem wir Herrn Vorsichts Überprüfungsprogramm gefahren hatten, was glauben Sie, ist mit den fünf Paketen passiert? Wir wissen, daß sie alle aus dem DATA-Regal entnommen wurden und einzeln in dem Abstellplatz D gespeichert wurden. Aber was in aller Welt ist mit den einzelnen Paketen passiert, nachdem sie aus D entfernt und durch das nächste ersetzt wurden?

Gute Frage. Wir haben dem Computer nicht gesagt, daß er die Pakete wieder in das DATA-Regal zurücktun soll, also befinden sie sich im Moment überhaupt nirgends.

Wenn wir einen anderen Kunden überprüfen wollten, müssen wir die Pakete in ihre ursprünglichen Positionen auf dem Regal zurückspeichern. Dies könnte ein nochmaliges Fahren des ganzen Programms erfordern, aber dies wäre ziemlich aufwendig.

Hier ist eine bessere Idee: Wir könnten ein neues BASIC-Wort erlernen, das die Pakete zurückspeichern würde, ohne daß wir von Anfang an neu beginnen müssen. Das neue Wort wird (logischer Weise) RESTORE genannt. Und zur Anwendung in Herrn Vorsichts Programm müssen wir die folgenden vier Zeilen hinzunehmen.

```
72 RESTORE
74 GOTO 5
92 RESTORE
94 GOTO 5
```

Jetzt sieht unser Programm folgendermaßen aus:

```
 5 CLS
10 DATA 11067, 12549, 22871, 57824, 86256
20 INPUT "KUNDEN-NUMMER";N
```

```
 30 FOR L=1 TO 5
 40 READ D
 50 IF D=N THEN GOTO 90
 60 NEXT L
 70 PRINT "DIESE KUNDEN NUMMER IST OK!"
 72 RESTORE
 74 GOTO 5
 80 END
 90 PRINT "NEIN, DIESE KUNDEN-NUMMER IST FAUL!"
 92 RESTORE
 94 GOTO 5
100 END
```

Raten Sie mal, was es jetzt tut. Wenn der Computer eine RESTORE-Anweisung findet, stellt er alle Pakete zurück in das DATA-Regal und zwar in der ursprünglichen Reihenfolge. Das GOTO-Kommando unmittelbar nach der RESTORE-Zeile sagt einfach "Gehe zurück zu dem Anfang des Programms".

Und wenn der Computer wieder am Anfang steht, können wir eine andere zu überprüfende Zahl INPUTen.

Warum benötigen wir zwei Paar an RESTORE- und GOTO-Zeilen? Wenn Sie das Programm genau betrachten, werden Sie merken, daß es auf zweierlei Art und Weise beendet werden kann. Das ist der Grund dafür, daß wir sowohl einen RESTORE als auch einen GOTO-Befehl in den folgenden Zeilen einfügen mußten.

* Nach Zeile 70 (das Ende des Programms, wenn die zu überprüfende Zahl OK ist) und

* Nach Zeile 90 (das Ende des Programms, wenn die Nummer faul ist.)

Versuchen Sie jetzt Ihr geändertes Programm!

Nebenbei bemerkt: Haben Sie die END-Anweisungen in den Zeilen 80 und 100 Ihres neuen Programms bemerkt? Und haben Sie auch bemerkt, daß Sie eigentlich nicht wirklich notwendig sind, denn das Programm geht vorher in einer Schleife zurück zum Anfang, bevor es dort ankommt? Um die Wahrheit zu sagen: Wir haben sie deswegen nur eingeschlossen, weil es eine gute Gewohnheit ist, jedes Programm zu beENDen. (Sogar wenn manchmal die Anweisung überhaupt nicht erforderlich ist.)

18 Musik in Ihren Ohren!

Musik wird seit den frühesten Tagen der Menschheit auf die verschiedenste Weise erzeugt. Seit einigen Jahrhunderten bedienen wir uns der Notenschrift, um einmal komponierte Musik dauerhaft aufschreiben zu können.

> Unser System der Musikschreibung ist ziemlich effizient. Es erlaubt uns, uns fast jeden Ton auszudenken, den wir wollen und auf Papier zu bringen. Nun, dieses System funktioniert ausgezeichnet für Menschen. Wenn ein Musiker eine Melodie (das ist eine Reihenfolge von Tönen) spielen möchte, die auf Papier aufgeschrieben ist, muß er lediglich folgendes anschauen:
>
> 1. Die Position der Noten, und
> 2. deren Form

und er wird genau wissen, welche Töne er produzieren soll.

Wenn es um das Lesen von Musik geht, hat Ihr Computer leider aus verständlichen Gründen ein Handicap. Um etwas zu sehen, besitzen Menschen zwei Augen. Und der Computer hat natürlich keine. Jedenfalls, auch wenn er unsere Musik doch ansehen könnte, würde er unser bildhaftes Schreibsystem nicht verstehen. (Computer können so nicht denken.)

Da also der Computer die Noten eines Liedes nicht sehen kann, müssen wir einen Weg finden, ihm jede Note einzeln zu beschreiben.

Die einfachste Methode, dies zu tun, ist es, ein codiertes Nummernsystem mit Zahlenpaaren zu verwenden. Das erste, was Sie Ihrem Computer sagen müssen, ist: Die Position der Note auf den Notenzeilen (die Tonhöhe).

Der Computer kann 31 verschiedene Noten (Tonhöhen) spielen, plus einem "Rest". (Das nennt man Pause in der Musik). In der musikalischen Notengebung wird es durch eine Vielzahl von Symbolen dargestellt (verschiedene Formen für die Pausen von verschiedenen Längen).

Jede Note (Tonhöhe) hat eine Code-Nummer für sich. Die komplette Liste der Noten und deren Code-Nummern wird nachfolgend gezeigt.

FREQUENZ (Tonhöhe)

Code	Ton	Code	Ton
Ø	rest	16	C4
1	A2	17	C#4
2	A#2	18	D4
3	B2	19	D#4
4	C3	2Ø	E4
5	C#3	21	F4
6	D3	22	F#4
7	D#3	23	G4
8	E3	24	G#4
9	F	25	A4
1Ø	F#3	26	A#4
11	G3	27	B4
12	G#3	28	C5
13	A3	29	C#5
14	A#3	3Ø	D5
15	B3	31	D#5

Die zweite Zahl im Zahlenpaar ist die Notenlänge, die hier als Zählzeit im Code angegeben ist. Nun, da haben Sie Ihr Instruktionspaar. Zusammen bilden sie eine Beschreibung einer einzelnen Note.

TON-DAUER

Code	Note	Notenlänge
1		$\frac{1}{8}$
2		$\frac{1}{4}$
3		$\frac{3}{8}$
4		$\frac{1}{2}$
5		$\frac{3}{4}$
6		1
7		$1\frac{1}{2}$
8		2
9		3

Gar nicht so schwer, oder? Indem Sie einfach die zwei oben abgebildeten Tabellen verwenden, sollten Sie Ihrem Computer die Musikstücke beschreiben können. Nun, wir empfehlen garantiert nicht Beethovens 5. Symphonie zu spielen, (so kultiviert ist der Computer auch wieder nicht). Bitte bleiben Sie einfach bei kleinen Melodien. Sie machen wirklich genau soviel Spaß! Solche finden sich vorzugsweise in Kindergesangsbüchern.

Zur "Übersetzung" eines Musikstückes in eine dem Computer verständliche Sprache, beachten Sie:

1. Stellen Sie fest, wo sich die einzelne Note auf der Notenlinie befindet und suchen die entsprechende Code-Zahl dieser Position aus der Tonhöhen- Tabelle heraus.
2. Stellen Sie fest, welche Form diese Note hat und suchen Sie die ent- sprechende Code-Zahl aus der Tondauer-Tabelle heraus.

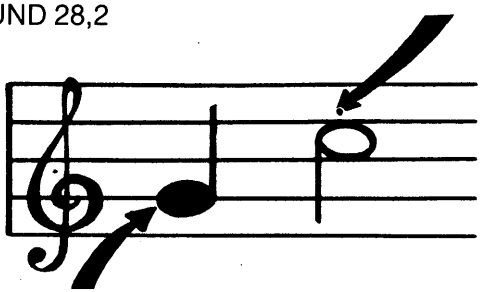
Das Musikstück, das Sie Ihrem Computer beschreiben, wird aus Reihen verschiedener Code-Zahlen-Paare bestehen, statt aus Reihen verschieden geformter Noten, mit verschiedenen Positionen auf der Noten-Tafel.

Wir haben fast etwas sehr wichtiges vergessen: Diese ganzen Nummern-Paare bedeuten dem Computer überhaupt nichts, bis Sie ihm sagen, wozu diese dastehen. Also werden wir vor jedem Nummern-Paar eine Art Hinweisschild aufstellen müssen. Ein Schild, daß Ihrem Computer sagt: "Diese beiden Zahlen stellen eine Musik-Note dar. Bitte spiele sie für mich!"

Um dies zu tun, müssen wir noch ein BASIC-Wort erlernen. Und wie heißt es? Sie haben es erraten: SOUND!

Das SOUND-Kommando geht jedem Code-Zahlen-Paar voraus.
Hier ist ein Beispiel:

Der Code für die Tonhöhe der ersten Note ist 28 und der Code für die Notenform ist 2. Um es zu beschreiben, sagen wir SOUND 28,2



Der Code für die Tonhöhe der zweiten Note ist 23 und der Code für die Notenform ist 6. Also, um es zu beschreiben, sagen wir SOUND 23,6

Die Position des Kaulquappen-Schwanzes der Note ist nicht wichtig. Er kann von der Note herunterhängen oder senkrecht nach oben stehen. Also, was wirklich zählt, ist, ob die Note ausgefüllt oder nicht ausgefüllt ist, ob irgendwelche "Fahnen" am Kaulquappen-Schwanz angebracht sind, oder ob sich unmittelbar ein Pünktchen dahinter befindet.

Von Lauten zu Liedern

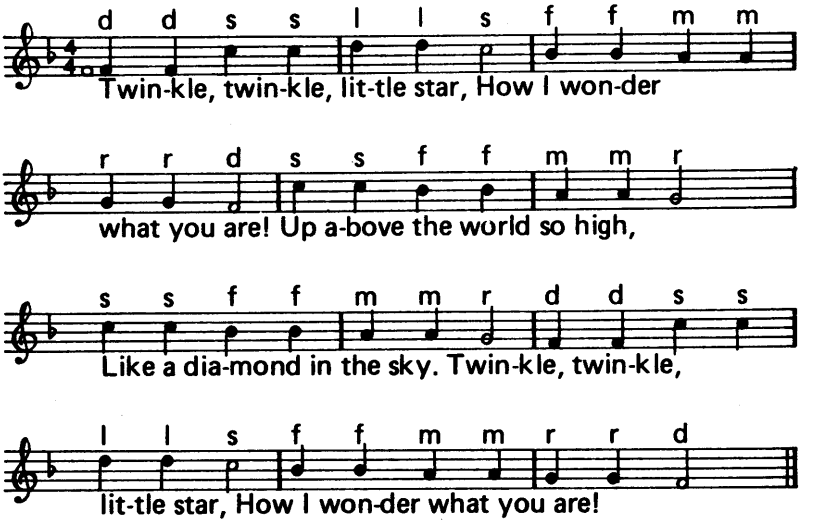
Wie Sie Ihrem Computer das Singen beibringen.

MELODIEN

In dem jetzt folgenden Beispiel wird ein Musikstück zu einem Computerprogramm umgesetzt. Verfolgen Sie anhand der Tabelle die Umsetzung!

TWINKLE, TWINKLE, LITTLE STAR
Nursery Rhyme

Key F



Weil Sie jetzt wissen, wie man dem Computer einzelne Noten beschreibt, kann Sie nichts mehr aufhalten! Um Ihre neu entwickelte Fähigkeit zum Einsatz zu bringen, wollen wir etwas besonders aufregendes probieren. Versuchen wir Ihrem Computer beizubringen, uns "ein Lied zu singen". Wie wär's mit einem schönen und alten Kinderreim? "Twinkle, twinkle little star" wäre ideal.

Was müssen wir tun, bevor Ihr Computer für uns auftreten kann? Wir müssen ihm eine Beschreibung jeder einzelnen Note des Liedes geben. Dazu verwenden wir:

Wenn Sie alle Noten dieses Liedes zusammenzählen, werden Sie feststellen, daß es genau 42 sind. Heißt das, daß wir 42 SOUND-Kommandos in unserem Programm zur Anwendung bringen müssen? Nein, beruhigen Sie sich, es gibt einen bequemen Ausweg. Diese vielen SOUND-Kommandos sind überhaupt nicht notwendig. Statt dessen wenden wir einen kleinen cleveren Trick an, einen, den wir Ihnen schon gezeigt hatten.

> Haben Sie erraten, was wir meinen? Hier ist ein kleiner Hinweis: Es sind zwei BASIC-Statements betroffen (und, sie gehen ein bißchen Hand in Hand). Als wir sie zum ersten Mal erwähnten, sagten wir, sie seien zur Speicherung vieler Informationen bestens geeignet.

Wenn Sie DATA und READ geraten haben, dann haben Sie beim ersten Versuch schon recht gehabt! Diese beiden Statements werden uns ganz sicher eine Menge unnötiger Arbeit ersparen. (Denn wir haben wirklich eine Menge Informationen zu speichern.) Wollen Sie sehen, wie? Dann geben Sie das unten stehende Programm ganz sorgfältig ein.

```
5 CLS
10 DATA 21,4,21,4,28,4,28,4,30,4,30,4,28,6,26,4,26,4,25,4
20 DATA 25,4,23,4,23,4,21,6,28,4,26,4,26,4,25,4,25,4,25,4,23,6
30 DATA 28,4,28,4,26,4,26,4,25,4,25,4,23,6,21,4,21,4,28,4,28,4
40 DATA 30,4,30,4,28,6,26,4,26,4,25,4,25,4,23,4,23,4,21,6
50 FOR L=1 TO 42
60 READ P,D:SOUND P,D:NEXT L:END
```

Programm-Analyse

Zeilen 10, 20, 30 und 40: Wie Sie aus den DATA-Anweisungen gleich zu Anfang ersehen können, benutzen wir diese 4 Zeilen als "Speicher-Regale". (Wir benötigen 4, weil wir soviele DATA speichern müssen.) Und was genau speichern wir? Codezahlen-Paare! Jedes Paar ist eine Beschreibung einer Note unseres Liedes. (Die erste Zahl unseres Paares beschreibt die Tonhöhe. Die zweite Zahl beschreibt die Tondauer.) Zählen Sie sie doch aus: Sie werden feststellen, daß sich 42 Paare auf unserem DATA-Regal befinden. Beispiel:

10 DATA 21,4 , 21,4 , 28,4 , 28,4 , 30,4 usw..

Wir wollen, daß der Computer diese Zahlen (je zwei) liest (READ) und jede einzelne auf einen Variablen-Speicherplatz abstellt. Sobald unsere beiden Code-Zahlen sicher gespeichert sind, gehen wir mit nur einem SOUND-Kommando vor, gefolgt von dem Zusatz des Variablen-Paares. Der Computer wird den Ton spielen, der durch die jeweiligen zwei Zahlen der Speicher-Plätze beschrieben ist.

Zeile 50: Unsere "Zahl"-Schleife. Der Computer muß 42-mal seine Schleife ziehen, weil es 42 Code-Zahlen-Paare sind. (Bei jeder Schleife wird ein Zahlen-Paar durch das nächste aus dem DATA-Regal ersetzt.)

Zeile 60: Diese ist eine der schlauen Zeilen, in denen mehrere Befehle enthalten sind, Merken Sie sich, daß ist OK, solange ein Befehl vom anderen durch einen Doppelpunkt (:) getrennt ist.

Erstes Kommando bis zum Doppelpunkt: Dies ist unsere READ-Anweisung. Sie speichert die Code-Zahlen, Paar um Paar, auf zwei Speicherplätzen. Die erste ist mit P markiert (hier kommt der Tonhöhen-Code hin); und die zweite ist mit D markiert (hier befindet sich der Code für die Tondauer.).

Zweites Kommando ab SOUND: weist den Computer an, die durch P und D beschriebene Note zu spielen.

Drittes Kommando ab NEXT: schickt Ihren Computer auf eine Schleife, bei der die nächsten Code-Zahlen in P und D gespeichert werden.

Viertes und letztes Kommando: beENDet das Lied, nachdem die Schleife zum 42. Mal durchgezogen wurde.

Haben Sie das Programm schon in Ihren Computer eingetippt? Nun, es wäre schon eine gute Idee, das Programm aufzuLISTen, nur um sicher zu sein, daß Sie jede einzelne Zahl richtig haben. (Denn, wenn Sie's nicht haben, könnte es sein, daß Ihr Computer etwas "falsch" singt.)

Und jetzt ist der große Augenblick gekommen, Zeit, das Programm abzufahren (RUN). Ist das nicht wundervoll? Wenn Sie glauben, daß stehende Ovationen angebracht sind, dann bitte! Und wenn Sie eine Zugabe wollen, dann lassen Sie das Programm einfach nochmal laufen.

Und jetzt eine besonders interessante (wenn auch völlig unbrauchbare) Information: Ihr Computer hat Ihnen ein Lied gespielt, das von einem sehr, sehr berühmten Komponisten geschrieben wurde.

"Twinkle, twinkle little Star" wurde von keinem geringeren als Wolfgang Amadeus Mozart komponiert.

19 Spaß mit Graphik

Wird Ihr kleiner Computer je aufhören, Sie mit seinen Fähigkeiten zu überraschen? In diesem Kapitel werden wir die versteckte künstlerische Fähigkeit Ihres Computers aufdecken. Wir werden jetzt lernen, auf dem Bildschirm Ihres Computer zu zeichnen. Aber, erwarten Sie nicht, eine Mona Lisa produzieren zu können!

Die Kunst, Bilder oder Muster mit dem Computer zu erstellen, nennt man Computer-Graphik. Darin ein Meister zu werden, erfordert Zeit und Geduld. Aber es macht unwahrscheinlich viel Spaß!

Ihr Computer hat zwei verschiedene Modi (Betriebsarten), in denen er Graphiken produzieren kann. Der erste Modus ist einfach Modus (0). In diesem befindet er sich sofort und automatisch beim Einschalten.

Mit Modus (0) werden die Zeichen der Tastatur auf den Bildschirm geschrieben.

Und was genau sind Tastatur-Zeichen? Ganz einfach, das sind alle Zeichen (und Kommandos), die sich auf der Tastatur des Computers befinden. Einschließlich dieser komischen Dinger namens "Graphik-Zeichen".

Diese verschiedenen Graphik-Zeichen sind mit Fliesen vergleichbar (die Art, die man in Badezimmern findet). Jede sieht anders aus, und es sind 15 Stück. Dadurch, daß man sie auf bestimmte Art und Weise auf dem Bildschirm anordnet, kann man ein Mosaik schaffen, (so ziemlich).

Unglücklicherweise sind die Fliesen (oder, wenn Sie wollen Bildbausteine) recht groß und klobig. Das heißt, daß die Bilder, die man mit ihnen zusammenstellt, auch grob gerastert sein werden.

Wie wir sagten, "Computer-Graphik" ist eine eigenartige Bezeichnung. Wenn Sie das Wort Graphik betrachten, dann werden Sie vielleicht darin ein anderes Wort entdecken. Eines, das die ganze Idee recht gut beschreibt. Es heißt "Graph" (Diagramm). Und es ist eine sehr nützliche Beschreibung, denn der Bildschirm ist nichts anderes, als ein großer "Graph".

Im Modus (0) ist der Graph 64 Blocks breit und 32 hoch.

Ein Tastatur-Zeichen benötigt den Platz von vier Blocks. Und deswegen können wir im Text-Modus nur recht grobe Zeichnungen herstellen. Wäre es nicht wunderbar, wenn wir den Bildschirm in viel kleinere Kästchen aufteilen könnten? Dadurch könnten wir Zeichnungen produzieren, die nicht so plump sind. Dazu braucht man nur den Computer auf den *zweiten Modus (Modus 1)* umzuschalten.

Leichter getan, als gesagt! Tippen Sie einfach MODE (1) ein und drücken Sie RETURN.
In MODE (1) ist der Graph 128 Blocks breit und 64 hoch und das ist schon wesentlich feiner.

In MODE (1) ist die Bearbeitung etwas anders. Es fängt damit an, daß man keine Textzeichen mit der Tastatur eintippen kann. Wozu ist dieser Modus dann zu gebrauchen?

Wir werden jetzt ein BASIC-Kommando verwenden, um etwas auf den Bildschirm zu bringen. Es heißt SET.

SET sagt Ihrem Computer, daß er einen dieser Blocks mit Farbe ausfüllen soll.

Jeder Block hat einen ihm eigene Position auf dem Bildschirm. Also, bevor wir dem Computer genau erklären können, welchen Punkt er beSETzen soll, müssen wir ihn mit "Koordinaten" belegen.

"Koordinaten? Hilfe! Das wird etwas zu technisch!"

Lassen Sie sich bitte durch diesen Begriff nicht abschrecken. Denn wahrscheinlich kennen Sie Koordinaten besser als Sie glauben, Überlegen Sie mal kurz, wie man mit einem Stadtplan umgeht. Man schaut nur im Inhaltsverzeichnis nach dem Straßen-Namen. Neben dem Straßen-Namen finden sich dann meistens ein Buchstabe und eine Ziffer.

Die meisten Stadtpläne sind genauso aufgeteilt, wie der Bildschirm Ihres Computers, wobei Linien jeweils horizontal und vertikal laufen. Die Buchstaben bestimmen dabei die Horizontale und die Ziffern die Vertikale. Um

das Ziel zu erreichen, müssen Sie nur den beiden Linien bis zu ihrem Zusammentreffen folgen. Dort finden Sie dann die gesuchte Straße. Der Buchstabe und die Ziffer in dem Straßenverzeichnis sind nichts anderes als Koordinaten.

Und mit SET einen Punkt auf Ihrem Bildschirm zu definieren, ist genauso einfach. Sie müssen lediglich SET eintippen und dann Ihrem Computer einen Satz (2) Koordinaten eingeben. Die erste bezeichnet die vertikale Linie des Graphens (und kann zwischen 0 und 127 liegen). Die zweite ist natürlich die Bezeichnung der horizontalen Graphen-Linie (sie liegt zwischen 0 und 63).

Wie bewerkstelligt man nun das Löschen eines Blocks, den man bereits mit SET coloriert hat? Kein Problem! Man kann dazu ein BASIC-Kommando verwenden, welches das Gegenteil von SET bewirkt, nämlich RESET. Dieses Kommando muß mit denselben Koordinaten kombiniert sein, mit denen vorher das SET-Kommando versehen war.

Es mag Stunden dauern, bis man mit SET ein größeres Feld eincoloriert hat, insofern man dabei Block um block vorgehen muß. Eigentlich nicht. Es gibt da einen kleinen Trick, der einem diesen Aufwand erspart. Dieser Trick ist nicht neu. Dergleichen haben wir bereits angewendet, als wir dem Computer das "Singen" beigebracht haben.

Wir verwenden dabei nur einen SET-Befehl, gefolgt von zwei Variablen. Im ersten Speicherplatz (nennen wir sie V) speichern wir die vertikale Koordinate. Und auf den zweiten Speicherplatz, (H genannt) setzen wir die horizontale Koordinate. Wollen Sie sehen, wie es funktioniert? Dann tippen Sie dieses Programm ein:

```
5 CLS
10 MODE (1)
20 FOR V=0 TO 127
30 FOR H=0 TO 63
40 SET V,H
50 NEXT H
60 NEXT V
```

Progamm-Analyse

Zeile 10: Schaltet in den richtigen Modus (das ist Modus (1).) Zeile 20: Ist der Anfang einer FOR-TO-Schleife. Bei jeder neuen Schleife wird der Wert von V um 1 erhöht.
Zeile 30: Wir eröffnen eine neue FOR-RO-Schleife. Wie bei der ersten Schleife, wird sie den Wert um 1 erhöhen (diesmal H).

Zeile 40: Und hier ist unser SET-Befehl. Jeder Punkt, der durch die Koordinaten definiert ist, die sich in den Speicher-Plätzen befinden, wird coloriert.

Zeile 50: Sagt NEXT H und beendet damit natürlich eine unserer FOR-TO-Schleifen. Merken Sie sich: Die H-Schleife ist "verschachtelt" innerhalb der V-Schleife, also müssen wir die H-Schleife zuerst beenden. Wenn der Computer von dieser Zeile kommt, flitzt er zurück zu Zeile 30 - dem Anfang dieser FOR-TO-Schleife und erhöht H um 1.

Der Computer fährt in der Schleife fort, bis der Wert 63 in H gespeichert ist. Dann fährt er fort:

Zeile 60: Die sagt NEXT V. Hier wird das Ende unserer "äußeren" Schleife markiert. Also wird der Computer zurück zum Anfang dieser Schleife gehen (das ist Zeile 20) und den Wert von V um 1 erhöhen.

Das ist alles! Sind Sie bereit es auszuprobieren? Dann tippen Sie RUN ein.

> Der Vz-200 hat begonnen, den ganzen Bildschirm zu "bemalen". Etwas Geduld, bitte. Es wird etwas dauern, bis er fertig ist.

Wollen Sie das "Gemälde" wieder loswerden? Das ist einfach genug: Wenn Ihr Computer fertig ist, dann ändern Sie Zeile 40 in:

> RESET V,H

Sie sollten einmal sehen, wie der Computer den "Lack abwischt", wenn Sie dieses geänderte Programm mit RUN laufen lassen.

Wir machen Regenbögen! (Er wird nicht umsonst *"Colour-Computer"* genannt.)

An Ihrem Computer gibt es etwas, das Ihnen von Anfang an wahrscheinlich ein Rätsel war. Und dieses Etwas ist die Reihe von "Farbmarkierern" auf der Tastatur. (Sehen Sie diese? Sie befinden sich oberhalb der ersten acht Tasten der obersten Reihe.)

> Der Markierer oberhalb der Taste 1 ist grün. Oberhalb der 2 ist gelb und über der 3 ist blau. Dann folgen rot, hellbraun, hellblau, magentarot bis zur Taste 8 mit orange. Sie sehen schon schön aus. Aber zur Dekoration allein sind sie nicht da.

Die Zahl auf jeder Taste stellt den Code für die betreffende Farbe dar. Diese Codes, zusammen mit einer ganz neuen BASIC-Anweisung (haben Sie bitte noch etwas Geduld, wir werden gleich mehr darüber hören), verwenden wir, um auf dem Bildschirm Farben zu produzieren.

Also, inzwischen haben Sie sicherlich erkannt, wie logisch die Namen der meisten BASIC-Anweisungen sind. Eigentlich müßten wir Ihnen gar nicht sagen, wie unser neuer BASIC-Befehl heißt. Er heißt, wer hätte das gedacht, COLOR!

* Die erste Zahl beschreibt die Vordergrund-Farbe und
* die zweite Zahl die Hintergrund-Farbe.

Erinnern Sie sich an die zwei verschiedenen Modi, von denen wir Ihnen erzählten? Nun, wenn es um Farbe geht, hat jeder Modus etwas andere Regeln.

Modus (0)

Sie haben in diesem Modus die Wahl zwischen zwei verschiedenen Hintergrund-Farben. Die erste ist grün. Wenn sie benutzt wird, hat sie eine besondere Code-Zahl, nämlich 0. (Wir sind alle bereits daran gewöhnt, einen grünen Bildschirm zu sehen, denn der Computer wählt beim Einschalten automatisch diese Farbe.)

Beispiel: Wenn der Computer einen grünen Hintergrund und einen blauen Vordergrund auf den Bildschirm bringen soll, dann würden Sie folgende Anweisung eingeben:

```
COLOR 3 , 0
```

In diesem Fall kommt 3, der Code für blau, als erstes. Der (Hintergrund)-Code für grün ist 0 und ist somit die zweite Zahl des Paares. (Übrigens, haben Sie bemerkt, daß wir mit einem Komma die einzelnen Code-Zahlen voneinander getrennt halten?)

Dazu gibt es jedoch eine Alternative. Wir wählen als zweites orange als Hintergrund-Farbe, die Code-Zahl dafür ist 1. Etwas Abwechslung kann nicht schaden. Versuchen wir's mit unserer neuen COLOR-Anweisung.

Eigenartigerweise: Weil wir uns im Moment nur mit der Hintergrund-Farbe beschäftigen, können wir die erste (Vordergrund-)Code-Zahl auslassen und lediglich sagen:

```
COLOR, 1 RETURN
```

Natürlich ist die Stelle, an der der Vordergrund-Farben-Code hinkommt leer. Aber wir müssen das Komma, das sonst danach angeführt wäre, trotzdem eintippen. Dies sagt dem Computer, daß er nur einen Code für die Hintergrund-Farbe erhält.

Sind Sie sicher, daß Sie jetzt bereit sind? Obwohl es einfach ist, ist dies wahrscheinlich das spektakulärste, was wir bis jetzt

gemacht haben. Nun, wenn Sie glauben die Aufregung aushalten zu können, dann tippen Sie COLOR,1.

Haben Sie jemals etwas derartig Oranges gesehen? Wir hatten Sie gewarnt, daß es ein ziemlicher Unterschied sein würde. Und wieder zu grün zurück zuwechseln, ist genauso leicht. Tippen Sie einfach: COLOR,0 und da haben Sie's.

Lassen Sie uns noch gescheiter werden und etwas Vordergrund-Farbe ins Bild bringen. Wie wäre es mit einem roten Vordergrund und einem grünen Hintergrund?

Abgemacht? Tippen Sie einfach:

COLOR 4,0

"Das Grün sehe ich, aber wo ist das Rot? Es hat nicht funktioniert!" Würden wir Ihnen eine "Blindgänger"-Anweisung geben? Natürlich nicht! Es hat tatsächlich funktioniert. Es gibt einen guten Grund dafür, warum Sie die Vordergrund-Farbe nicht sehen können. Und der Grund ist, daß sich auf Ihrem Schirm nichts als Text befindet.

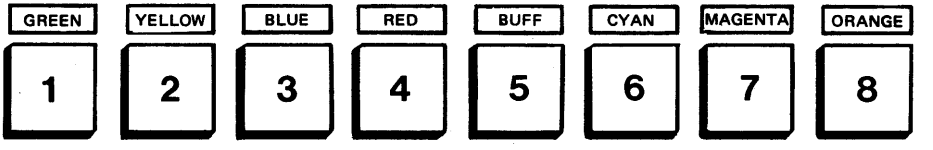
Die einzigen Tastatur-Zeichen, die andersfarbig darstellbar sind, sind die GRAPHIK-ZEICHEN.

Versuchen Sie einige Graphik-Zeichen. Sie werden sehen, daß sie so rot wie nur möglich sind.

Modus 1

Die Farbauswahl in diesem Modus ist etwas begrenzt. (Leider, aber man kann nicht alles haben!)

Ihre Hintergrund-Farbe kann entweder grün (dessen Code immer noch 0 ist) oder hellbraun sein (Code-Zahl 1).



Grün, Gelb,Blau,Rot: 1 2 3 4.
Wenn Sie grün als Hintergrund nehmen, können Sie nur diese Farbpalette benutzen.

Hellbraun, Hellblau,Magenta,Orange: 5 6 7 8.
Und wenn Sie hellbraun als Hintergrund wählen, müssen Sie sich an diese Farbpalette halten.

Nun wollen wir eine Sache, die offensichtlich ist, nicht besonders hervorheben. Aber für alle Fälle:

Damit die COLOR-Anweisung funktioniert, benötigenSie ein Farbfernsehgerät.

20 Letzte Hinweise

Und jetzt, während die Sonne sich langsam im Westen neigt, bringen wir unser kleines Buch zum Abschluß. Haben Sie Ihre ersten Eskapaden in die Welt der Computer genossen? Das nehmen wir an, denn es hat uns Spaß gemacht, Ihnen den Weg zu zeigen.

Und es war sicherlich ein langer, langer Weg. Seit dem Augenblick, als Sie dieses Buch das erste Mal öffneten, sind wir Herausforderungen entgegengetreten und haben sie ohne viel Mühe bezwungen. Den guten alten Zeiten zuliebe, wollen wir uns einigen der bezwungen Hürden nochmals zuwenden:

* "Was? Sie erwarten, daß ich mit diesem Computer umgehe? Ich weiß ja überhaupt nicht, was das verflixte Ding ist!"

Dies war einigen von Ihnen wahrscheinlich die größte Sorge: Die Computer/Mensch Sprach-Barriere. Ist Ihnen klar geworden, daß Sie jetzt zweisprachig sind? Sie können sich jetzt - fast fließend - in BASIC verstän- digen. (Wirklich! Denken Sie nur an all die Wörter, die Sie gelernt haben.)

* Die furchtbare Sache namens "Computer-Programmierung"

Erinnern Sie sich, als wir dies zum ersten Mal erwähnten? Wahrscheinlich hat es die meisten von Ihnen ziemlich erschreckt. Und doch, hier sind Sie, bereit, Ihre eigenen Programme zu schreiben!

Die Hindernisse waren ziemlich schwierig und dabei waren das nur ein paar! (Nachdem Sie soviel Übung im Hürden-Nehmen haben, sind Sie wahrscheinlich olympiareif.)

Die Welt der Computer ist ein sehr umfangreiches Gebiet. Viel zu umfangreich, um kurz (oder in diesem kleinen Buch) abgehandelt zu werden. Was wir hoffen konnten war, Ihnen Grundwissen über Computer im allgemeinen und den Computer im besonderen zu vermitteln. Wir wollen, daß Sie für Ihre weitere Entwicklung als Programmierer gut ausgestattet sind. Und wenn das durch dieses Handbuch ermöglicht wurde, könnten wir zufriedener sein.

Sie wissen, daß eine gute, solide Freundschaft nichts, außer Zeit kostet. Also seien Sie bereit, den Aufwand zu akzeptieren, der notwendig ist, um Ihren Computer näher kennen zu lernen. Nehmen Sie sich Zeit für Experimente, Zeit zum Üben und Zeit zum Spielen. (Wir versprechen Ihnen: Der Aufwand lohnt sich!)

Und nun ein letzter Rat. Falls Sie jemals Schwierigkeiten haben sollten, zögern Sie nicht, zurück zu den BASICs zu gehen. Geht etwas schief? Irgendetwas? Kommen Sie einfach zurück zu diesem kleinen Handbuch. Es ist immerzu da, es noch einmal von Anfang an zu erklären.

Und: Scheuen Sie nicht zu experimentieren! Versuch und Fehler sind die besten Lehrmethoden und indem Sie Ihre Ideen mit dem Computer ausprobieren, können Sie ihm unmöglich Schaden zufügen.

Also: Auf Wiedersehen einstweilen und viel Glück! Ein vertiefendes Buch für Ihren Computer wird im Sommer 1984 erscheinen.

Wie Sie Ihren Computer gut behandeln

Das ist nicht schwer - wirklich nicht. Lesen Sie einfach die unten stehenden Anmerkungen. Es sind gute Ratschläge und, wenn Sie diese befolgen, wird Ihr Computer dafür dankbar sein.

Vorsichtig damit umgehen!

Fallen gelassen zu werden, bedeutet für den Computer eine traumatische Erfahrung zu machen. Versuchen Sie bitte deswegen nicht, ihn zu stoßen, oder mit ihm eine Tür aufzustoßen. Wenn Ihnen einer das antäte, würden Sie genauso schlecht arbeiten, wie Ihr Computer in einem solchen Fall.

Hitze stört Ihren Computer auch. Deswegen sollten Sie ihn nicht auf einer Fensterbank oder auf der Terrasse oder sonst wo, wo er starker Sonnen-

bestrahlung ausgesetzt ist, stehen lassen. Nein, einen Sonnenbrand bekäme er nicht, aber krank würde er doch werden.

Nirgends ist es so schön, wie zu Hause.

Für Ihren Computer bedeutet zu Hause der Platz (ein sicherer natürlich), wo Sie ihn hinstellen, wenn Sie mit ihm fertig sind. Wenn er die ganze Zeit über an das Netz angschlossen ist, würde er durch die darauf folgende Überhitzung gestört werden.

Etwas Etikette, bitte!

Anders als bei Menschen, wird es Ihrem Computer nichts ausmachen, wenn Sie vor ihm essen. Aber bitte erinnern Sie sich stets Ihrer Tischmanieren. Denn Ihr Computer wird ganz bestimmt beleidigt sein, wenn Sie Orangen-Saft oder Brotkrümel auf die Tastatur verschütten. Es ist wirklich eine vorteilhafte Einstellung, dem Computer mit Lebensmitteln fern zu bleiben. (Nur um sicher zu sein.)

Suchen Sie sich eine sichere Stelle

Sie würden ganz gewiß Ihren Computer nicht mitten auf einer Autobahn aufstellen wollen. Unglücklicherweise, gibt es im Haushalt ebenso gefährliche Aufstellplätze. Also sollten Sie einen ruhigen Platz, wo Sie niemandem im Wege sind, suchen, wenn Sie sich zu einer längeren Sitzung mit Ihrem Computer zusammen finden. (Glauben Sie uns, es gibt nichts frustrierenderes, als wenn jemand über das Netzkabel stolpert!)

Spielen Sie nicht Arzt!

Würden Sie sich von einem Ihrer Freunde operieren lassen, wenn Sie krank wären? Nein - das haben wir auch nicht angenommen. Wenn wir Menschen krank werden, gehen wir zum Arzt...und wenn Computer krank sind, müssen sie zum Computer-"Arzt"! Wenn Ihr Computer also nicht richtig arbeitet (und Sie sicher sind, daß es nicht an Ihnen liegt), dann bringen Sie ihn zum Händler. Durch einen unserer "Computer-Ärzte" wird der Computer bald wieder einsatzbereit sein. Bitte, versuchen Sie nicht, selbst "Operationen" vorzunehmen!

Anhang A

Die BASIC-Kommandos, die wir gelernt haben

Kapitel 4

PRINT (mit Anwendungen) GOTO END LIST RUN BREAK CONT CLS NEW

Kapitel 5

PRINT (ohne Anwendungen)

Kapitel 6

LET

Kapitel 7

INPUT

Kapitel 8

IF-THEN ELSE usw.

Kapitel 10

RND SQR

Kapitel 12

FOR-TO NEXT

Kapitel 13

STEP Kapitel 14

GOSUB RETURN

Kapitel 16

DATA READ

Kapitel 17

RESTORE

Kapitel 18

SOUND

Anhang B

Reservierte Wörter

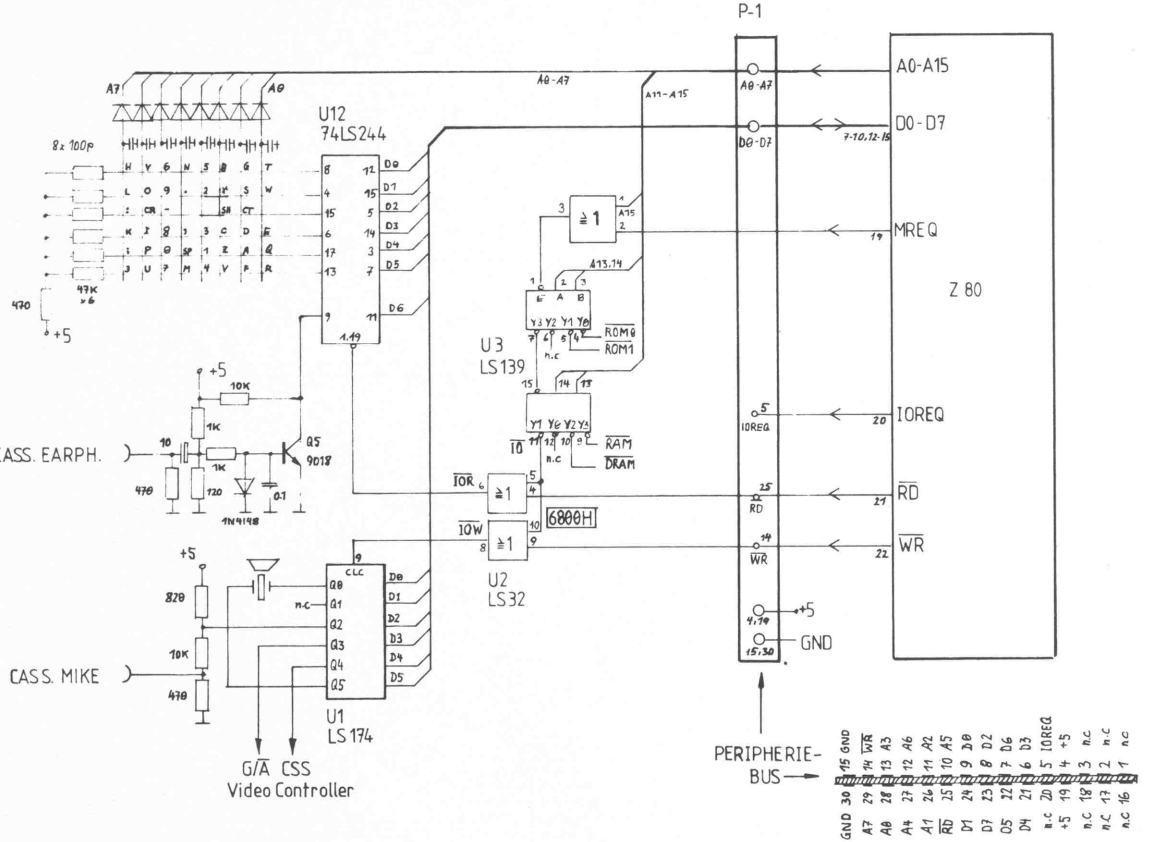
Bei der Markierung der Variablen-Abstellplätze, dürfen die folgenden Wörter nicht zur Anwendung kommen! Der Computer hat sie für besondere Zwecke reserviert...wenn Sie trotzdem versuchen, diese anzuwenden, werden Sie viel Verwirrung entfachen!

Unbedingt merken: Sie dürfen nicht verwenden:

** Wörter, die auf der reservierten Liste sind, oder*
** solche, deren erste zwei Zeichen die gleichen sind, wie die ersten zwei Zeichen der reservierten Wörter!*

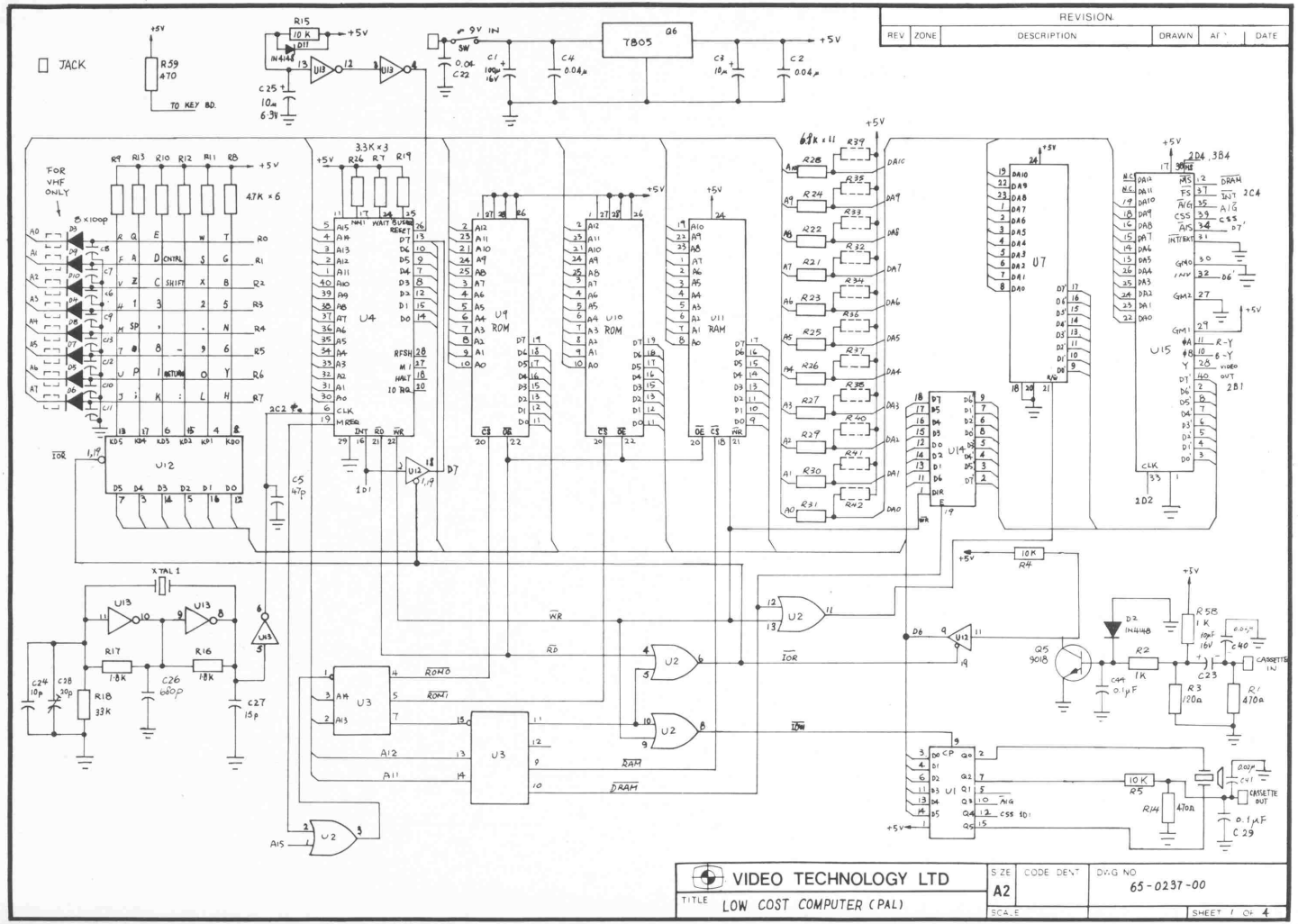
(Warum? Weil der Computer nur die ersten zwei Zeichen des Variablen-Namens liest. Ihrem Computer bedeuten die Namen, DATA und DAVID dasselbe!)

ABS AND ASC ATN
CHR$ CLOAD CLS COLOR CONT COPY COS CRUN CSAVE
DATA DIM
ELSE END EXP
FOR
GOSUB GOTO
IF INKEY$ INP INPUT INT
LEFT$ LEN LET LIST LOG LLIST LPRINT
MODE MID$
NEW NEXT NOT
OR OUT
PEEK POKE POINT PRINT
READ REM RESET RESTORE RETURN RND RUN
SET SGN SOUND SIN SQR STEP STOP STR$
TAB TAN TO THEN
USING USR

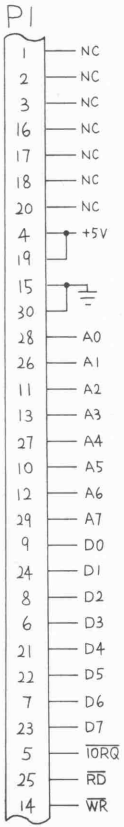


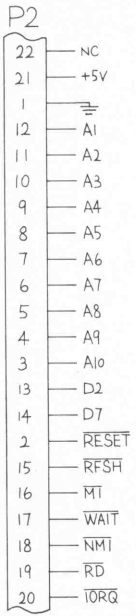
GND	A7	A8	A4	A1	RD	D1	D7	D5	D4	n.c	+5	n.c	n.c	n.c	
30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	M1	WR	A3	A6	A2	A5	D0	D2	D6	D3	IOREQ	+5	n.c	n.c	n.c

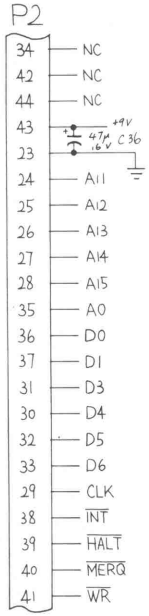
LASER 110/210: I/O


REVISION
REV
ZONE
DESCRIPTION
DRAWN
DATE
JACK
R59 470
TO KEY BD.
R15 10K
D11 IN4148
C25 10µ 6.3V
+5V
9V IN
SW
C1 1000µ 16V
C22 0.01
C4 0.04µ
7805
Q6
C3 10µ
C2 0.04µ
FOR VHF ONLY
47K x 6
8x100p
R9 R13 R10 R12 R11 R8
R0 R1 R2 R3 R4 R5 R6 R7
3.3K x 3
R26 R7 R19
U4
RFSH
M1
HALT
10 INT
WAIT
BUSRQ
RESET
NMI
CLK
MREQ
INT RD WR
U9 ROM
U10 ROM
U11 RAM
68K x 11
R39 R38 R35 R24 R33 R22 R32 R21 R34 R23 R36 R25 R37 R26 R35 R27 R40 R29 R41 R30 R42 R31
DA10 DA9 DA8 DA7 DA6 DA5 DA4 DA3 DA2 DA1 DA0
U7
U15
2D4, 3B4
DRAM
INT
A/G
CSS
A/S
INT/EXT
GM0
INV
GM2
GM1
A
B
Y
VIDEO OUT
2B1
2D2
U12
KD5 KD4 KD3 KD2 KD1 KD0
IOR
C5 47p
XTAL 1
U13
R17 1.8K
R16 1.8K
C24 10p
C28 30p
C26 680p
C27 15p
R18 33K
U3
ROM0
ROM1
A12
A11
A15
RAM
DRAM
U2
WR
RD
IOR
IOW
U13
D7
D6
U14
R4 10K
Q5 9018
D2 IN4148
R58 1K
C40
C23
R2 1K
C41 0.1µF
R3 120Ω
R1 470Ω
CASSETTE IN
U1
R5 10K
RH 470Ω
C41 0.01µF
CASSETTE OUT
C29 0.1µF
VIDEO TECHNOLOGY LTD
TITLE
LOW COST COMPUTER (PAL)
SIZE
A2
CODE DEVT
DWG NO
65-0237-00
SCALE
SHEET 1 OF 4
06

REVISION					
REV	ZONE	DESCRIPTION	DRAWN	APPD	DATE







UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN MM INCHES AND TOLERANCES ARE
XXX XX ANGULAR
± ± ±
ALL MACHINED SURFACE √
REMOVE BURRS
BREAK ALL SHARP CORNERS
DO NOT SCALE DRAWING

	SIGNATURE	DATE
DWN BY		
CHECK BY		
ENGR		
ENGRG AUTH		

VIDEO TECHNOLOGY LTD

TITLE: LOW COST COMPUTER

NEXT ASSY		MATERIAL	SIZE	CODE IDENT	DWG NO
USED ON		FINISH	A2		65-0237-00
FIRST APPLICATION			SCALE		SHEET 4 OF 4

Notizen

Notizen

Notizen

Notizen

Notizen

Einführung in das Programmieren mit Schaltbildern