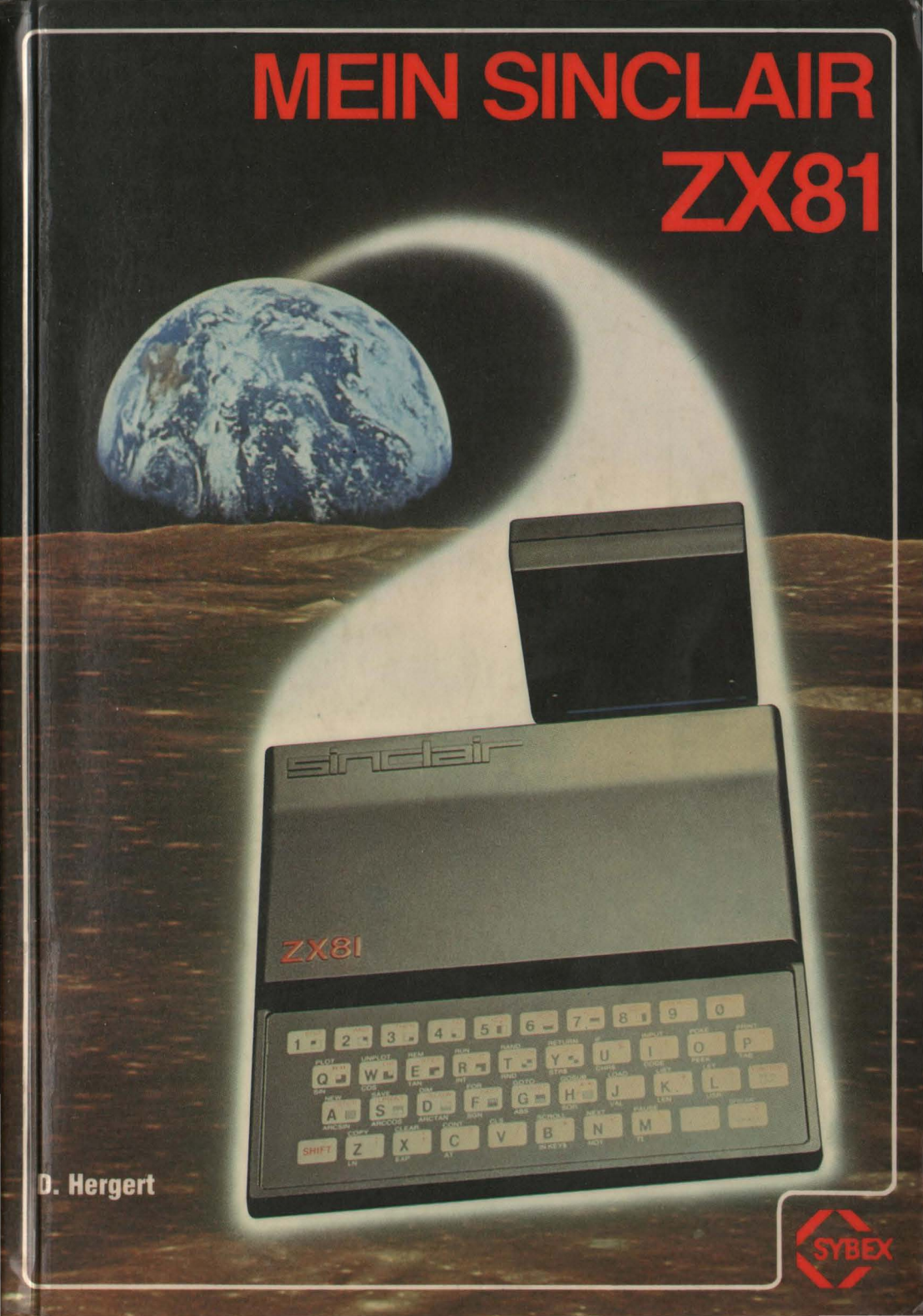
MEIN SINCLAIR ZX81



D. Hergert

SYBEX

Mein Sinclair ZX81

Mein Sinclair ZX81

D. Hergert

SYBEX

25,-

Scha

Stadtbücherei
Dortmund - Scharnhorst

ANMERKUNGEN:

ZX81 ist ein geschütztes Warenzeichen von Sinclair Research, Ltd.
VU-CALC © PSION.
VisiCalc ist ein geschütztes Warenzeichen von VisiCorp, Inc.
SuperCalc ist ein Warenzeichen von Sorcim Corporation.

Originalausgabe in Englisch.
Titel der englischen Ausgabe: Your Timex Sinclair 1000 and ZX81
Original Copyright © 1983 by SYBEX Inc., Berkeley, USA.

Das Zitat aus „The Killion" von Jan Frazier erscheint mit Genehmigung von The New Yorker Magazine, Inc.

Deutsche Übersetzung: Wolfgang Dederichs

Umschlag: Lautenschläger, Schulz, Partner
Satz: tgr – typo-grafik-repro gmbh., remscheid
Gesamtherstellung: Druckerei Hub. Hoch, Düsseldorf

Der Verlag hat alle Sorgfalt walten lassen, um vollständige und akkurate Informationen zu publizieren. SYBEX-Verlag GmbH, Düsseldorf, übernimmt keine Verantwortung für die Nutzung dieser Informationen, auch nicht für die Verletzung von Patent- und anderen Rechten Dritter, die daraus resultieren. Hersteller behalten das Recht, Schaltpläne und technische Charakteristika ohne Bekanntgabe an die Öffentlichkeit zu ändern. Für genaue technische Daten auf dem neuesten Stand wird der Leser an die Hersteller verwiesen.

ISBN 3-88745-021-3
1. Auflage 1983
2. Auflage 1983

Alle deutschsprachigen Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder einem anderen Verfahren) ohne schriftliche Genehmigung des Verlages reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Printed in Germany
Copyright © 1983 by SYBEX-Verlag GmbH., Düsseldorf

GELÖSCHT
STADT- UND
LANDESBIBLIOTHEK
DORTMUND

Stadtbücherei
83 - 22472
Dortmund

The 3

Vorwort zur Übersetzung

In der Grundausstattung wird der ZX81 mit 1K RAM Speicherkapazität ausgeliefert. Diese Kapazität reicht für's erste völlig aus. Anhand der ersten Kapitel wird der Leser behutsam in die Benutzung des Geräts eingeführt. Es ist schon erstaunlich, wieviel man bereits mit der Grundausstattung bewerkstelligen kann. Dies reicht von der einfachen Rechteck-Grafik bis zur Entwicklung eines Super-Taschenrechners. Lediglich bei zwei Programmen mußte die Speichererweiterung hinzugezogen werden. Hierbei handelt es sich um das Zeichen-Programm und um die Balkengrafik. Diese beiden Programme sind jedoch so modular aufgebaut, daß auch mit der Grundversion des Computers die wichtigsten Programmbausteine nachvollzogen werden können. Für eine wirklich sinnvolle und auch vielseitig einsetzbare Nutzung des ZX81 empfiehlt sich auf jeden Fall die Speichererweiterung mit dem RAM-PACK.

Das Testgerät lief auf Anhieb. Lediglich beim Laden vom Kassetten-Recorder gab es einige Schwierigkeiten. Dies erfordert vom Anwender etwas Geduld. Generell erscheinen beim Laden flatternde Streifen auf dem Bildschirm. Diese Streifen sagen noch nichts aus über Erfolg oder Mißerfolg des Programmeinlesens. Mit Hilfe des Lautstärkereglers am Kassettenrecorder muß man die zwei Einstellungen „zu laut" und „zu leise" ausfindig machen. Genau dazwischen liegt dann die günstigste Einstellung für das Laden von Programmen.

Ein abschließendes Wort des Dankes gilt Herrn Sterner, Bochum, der wesentlich an der Übersetzung dieses Buches mitgewirkt hat.

W. D.

Hattingen, den 17. 1. 1983

Inhalt

Einführung:
Eintritt ins Computer-Zeitalter 9

1 Der erste Eindruck 13

2 Der erste Akt:
Die Programmeingabe 29

3 Jetzt wird's spannend:
Ein kurzer und anschaulicher BASIC-Kurs 63

4 Wir können nicht nur bis drei zählen 109

5 Wörter, nichts als Wörter
Strings und Stringfunktionen 143

Anhang A: Der BASIC-Wortschatz 159

Anhang B: Die Fehlermeldungen des ZX81 169

Literatur zum ZX81 170

Stichwortverzeichnis 171

Einführung: Eintritt ins Computer-Zeitalter

Soeben haben Sie Ihren neuen Computer bekommen: den Sinclair ZX81. Dies ist ein bedeutungsvoller Augenblick, denn endlich haben Sie den Schritt ins Computer-Zeitalter getan. In Zukunft werden Sie nicht mehr still und unwissend dasitzen, wenn Ihre Freunde über ihre Personal-Computer reden. Jetzt gehören Sie auch dazu. Sie denken erfreut an die bescheidene Investition, die Sie gemacht haben, im Vergleich zu den vielen tausend Mark, die andere für ihre Computer ausgegeben haben.

Sie schließen Ihren Computer an, verbinden ihn mit dem Fernsehgerät und stellen mit großer Genugtuung fest, daß der Computer wie erwartet reagiert.

Von diesem Zeitpunkt an kann die Geschichte Ihrer folgenden Computer-Erfahrung sich in zwei verschiedenen Richtungen weiterentwikkeln. Die erste Version dieser Geschichte ist völlig unbefriedigend und endet überaus enttäuschend. Die zweite Version jedoch, die Sie durchaus erwarten dürfen, ist die, zu der Ihnen dieses Buch verhelfen will. Schauen wir uns die beiden Versionen einmal an.

In der ersten Version fangen Sie an, auf der Tastatur herumzutippen und sehen sich an, was auf dem Bildschirm passiert. Zunächst wundern Sie sich über die Macht, die Sie anscheinend über den Computer ausüben. Aber allmählich stellen Sie fest, daß Sie gar nicht verstehen, was sich da eigentlich abspielt. Na schön, immer wenn Sie eine Taste drücken, tut der Computer irgendetwas, aber was? Anscheinend ist etwas mehr Aufwand notwendig, als Sie vorher gedacht hatten. Wenn schon, denken Sie, im Handbuch wird's schon erklärt sein. Sie schlagen also das Handbuch auf, das mit Ihrem Computer geliefert wurde, und fangen an zu lesen. Nach ein, zwei Absätzen fühlen Sie sich etwas unsicher. Nach mehreren Seiten kommen Sie sich vollständig verloren vor. Irgendwie scheint dieses Buch an Leute adressiert zu sein, die schon mehr wissen als Sie. Sie spüren, daß es sicherlich nützliche Informationen enthält für jemanden mit etwas Erfahrung – was tatsächlich der Fall ist – aber wie sollen Sie anfangen? Sie wenden sich wieder dem Computer zu, drücken hoffnungsvoll noch auf einigen weiteren Tasten herum und denken, daß Ihnen vielleicht gleich

die Erleuchtung kommt. Aber dieser Augenblick tritt leider allzu selten ein. Irgendwie ist Ihre ganze ursprüngliche Begeisterung dahin, und stattdessen stellt sich ein unerwartetes Gefühl von Verwirrung und Enttäuschung ein. Sie schalten den Computer wieder aus, packen ihn wieder ein und gehen den Rasen mähen . . . Zwei oder drei Monate später stoßen Sie wieder auf den Computer, der immer noch in seinem Karton steckt – eine dünne Staubschicht hat sich inzwischen auf die Tastatur gelegt. Sie haben sich daran gewöhnt, den Leuten zu sagen, daß Sie einen Computer besitzen, aber – nun ja! – Sie müssen sich immer etwas einfallen lassen, um das Thema zu wechseln, wenn Ihre Freunde anfangen, detaillierte Fragen zu stellen . . .

In der zweiten Version der Geschichte Ihrer Computer-Erfahrungen nähern Sie sich dem Computer systematischer. Sie merken gleich, daß es eine Menge über Ihren neuen Computer zu lernen gibt, und je mehr Sie lernen, desto größer wird Ihre Begeisterung. Sie beginnen damit, daß Sie sich mit der Tastatur des Sinclair-Computers vertraut machen. Sie lernen, wie man Kommandos eingibt, und Sie finden heraus, was die einzelnen Kommandos für den Computer bedeuten. Sie tippen ein komplettes BASIC-Programm ein – die Computersprache, die in den ZX81 eingebaut ist – und sind selbst beeindruckt von den unterhaltsamen Ergebnissen. Die Erfahrungen werden immer faszinierender, je öfter Sie an Ihrem Computer sitzen: Sie lernen die Grafik-Möglichkeiten kennen sowie Berechnungen, Verarbeitung von Zeichenketten . . . und stellen auf einmal fest, daß Sie ein gar nicht so schlechter BASIC-Programmierer sind und nicht nur sich selbst, sondern auch Ihre Familie und Ihre Freunde mit all den Dingen beeindrucken, die Sie mit Ihrem neuen Computer machen können. Nach einigen Wochen ist Ihnen bewußt, daß Sie nun mehr sind als ein Computer-Besitzer – was viel wichtiger ist, Sie sind ein Computer-Benutzer.

Dieses Buch – Mein Sinclair ZX81 – wurde deshalb geschrieben, damit Sie die zweite der beiden soeben beschriebenen Möglichkeiten erleben werden. Wenn Computer für Sie völlig neu sind, wenn Sie vollkommen verwirrt sind von dem ganzen neuen Computer-Vokabular, das andere anscheinend selbstverständlich verwenden, dann ist dies ein Buch für Sie, denn es beginnt ganz am Anfang der Geschichte und führt Sie sorgfältig und schrittweise den Weg entlang, den Sie gehen müssen, um ein zufriedener Computer-Benutzer zu werden.

Kapitel 1 – Der erste Eindruck – beschreibt die Hardware – die ganze Ausrüstung, die Sie brauchen, damit Ihr Computer arbeiten kann. Es vermittelt Ihnen außerdem sorgfältige Anweisungen, die Ihnen Schritt für Schritt zeigen, wie Sie die einzelnen Komponenten miteinander ver-

binden müssen. Abschließend wird besprochen, welche Rolle Sie übernehmen müssen, damit Ihr Computer für Sie arbeiten kann, und es wird über die Fähigkeiten gesprochen, die Sie entwickeln werden, wenn Sie Ihren Computer benutzen.

Kapitel 2 – Der erste Akt: Die Programmeingabe – ist eine Einführung in die Software, die Anweisungen und Programme, die Sie erstellen, damit Ihr Computer bestimmte Aufgaben erfüllen kann. Sie beginnen damit, daß Sie die ZX81-Tastatur genau kennenlernen, um mit ihrem Computer kommunizieren zu können. Sie werden ein vollständiges Programm eingeben, es laufen lassen, und dabei Ihre ersten Eindrücke von der Leistungsfähigkeit der Programmiersprache BASIC erhalten. Außerdem lernen Sie, wie man ein Programm auf einer Tonband-Kassette dauerhaft speichert, so daß Sie das Programm immer wieder benutzen können. Das Kapitel endet mit Hinweisen auf Software, die Sie käuflich erwerben können, sowie einer Diskussion über das Für und Wider des Kaufens gegenüber dem Schreiben von eigenen Programmen.

Kapitel 3 – Jetzt wird's spannend – bietet Ihnen einen kurzen und leichten Einführungskurs in BASIC. Sie lernen, wie man dem Computer beibringt, bestimmte Aufgaben fortwährend zu wiederholen, wie Entscheidungen aufgrund gegebener Information möglich sind und wie man die Ergebnisse auf dem Fernseh-Bildschirm sichtbar macht, um sie verwerten zu können. Dazu lernen Sie die beiden Grafik-Möglichkeiten des ZX81 kennen sowie – einfach zu Ihrer Unterhaltung – ein vollständiges Programm, mit dem Sie Bilder auf dem Bildschirm zeichnen können.

Kapitel 4 – Wir können nicht nur bis drei zählen – bringt alles über Zahlen und Berechnungen mit Ihrem Computer. Sie finden heraus, wie Sie große numerische Datenmengen in Ihrem Computer bequem und effizient speichern können. Sie werden sehen, daß Sie Ihren Computer als einen „Super-Rechner" benutzen können. Das Hauptprogramm am Schluß dieses Kapitels zeigt Ihnen, wie man Balkendiagramme aus beliebigen numerischen Daten erstellen kann.

Kapitel 5 – Wörter, Wörter und noch mehr Wörter – beschreibt die BASIC-Funktionen, die für die Verarbeitung von Zeichenketten entwickelt wurden, d. h. für nichtnumerische Informationen. In diesem Kapitel wird ein amüsantes Programm vorgestellt, das Ihnen in schwierigen Augenblicken des Lebens eine Hilfe sein kann, wenn Sie nicht entscheiden können, was Sie tun sollen.

In Anhang A wird das ganze BASIC-Vokabular beschrieben, d. h. sämtliche Anweisungs-Begriffe, die sich auf der Tastatur befinden. Sie können diesen Anhang zum Nachschlagen verwenden, wenn Sie eigene Programme schreiben.

In Anhang B sind die Fehler-Meldungen des ZX81 zusammengestellt.

Dieses Buch ist so aufgebaut, daß die Informationen aus einem Kapitel in dem jeweils folgenden benötigt werden. Jedes Kapitel baut auf den vorhergehenden Kapiteln auf. Um das Buch erfolgreich zu benutzen, sollten Sie es mit dem Computer an der Hand durcharbeiten, indem Sie jede neue Anweisung ausprobieren und alle angeführten Programme eingeben und laufen lassen. Wenn Sie auf jeden Schritt soviel Zeit verwenden, daß Sie ihn beherrschen, bevor Sie zum nächsten übergehen, werden Sie am Ende dieses Buches über ein fundiertes Verständnis der Fähigkeiten Ihres Computers verfügen und die besten Wege kennen, um diese Fähigkeiten zu nutzen.

Das wichtigste, was Sie mit dem Kauf Ihres ZX81 erworben haben, ist die große Begeisterung eines neuen Computer-Besitzers. Verlieren Sie diese Begeisterung nicht. Dieses Buch soll Ihnen zeigen, was Sie wissen müssen, um das Beste aus Ihrem neuen Computer herauszuholen.

Kapitel 1

Der erste Eindruck

Einführung

Wie Sie vermutlich schon wissen, ist der Sinclair ZX81-Computer in seinem kleinen schwarzen Gehäuse eine erstaunliche und leistungsfähige Maschine, die aber für sich allein von nur geringem Nutzen ist. Um ihn wirkungsvoll einzusetzen, muß man noch weitere Geräte anschließen. Einige davon sind ganz gewöhnliche Geräte, die sich in jedem Haushalt finden: ein Fernsehapparat und ein Kassettenrecorder. Es gibt andere Geräte, die ausschließlich für die Verwendung mit dem ZX81-Computer entwickelt wurden: z. B. der Drucker und der Speicher-Erweiterungsbaustein. Die meisten dieser Zusatzgeräte dienen zwei grundsätzlichen Aufgaben: sie senden Informationen zum oder empfangen Informationen vom Computer. Diese beiden Aufgaben werden als *Eingabe* bzw. *Ausgabe* bezeichnet.

In diesem ersten Kapitel wollen wir einen kurzen Blick auf den Computer und einige weitere – notwendige oder mögliche – Zusatzgeräte werfen. Während wir herausfinden, wozu die einzelnen Komponenten dienen, werden wir lernen, wie man sie richtig an den Computer anschließt. Wir werden auch die verschiedenen Rollen besprechen, die Sie als Computer-Benutzer spielen können. Am Ende dieses Kapitels werden Sie in der Lage sein, mit der Arbeit an Ihrem Computer zu beginnen.

In der Hauptrolle: der Sinclair ZX81

Betrachten Sie einmal das Computer-Gehäuse. Es verdient schon einige Aufmerksamkeit, bevor Sie Ihre Arbeit mit dem Computer aufnehmen. Die Abbildungen 1.1 bis 1.3 zeigen den Computer aus drei verschiedenen Blickwinkeln; wir wollen sie der Reihe nach durchgehen.

Der vordere Teil des Computers besteht aus dem Tastaturbereich. Über diese Tastatur werden Sie dem Computer Anweisungen geben oder

Daten eintippen. Deshalb ist die Tastatur das Haupt-Eingabegerät für den Computer. Zunächst wollen Sie vielleicht kaum glauben, daß ein flaches Stück Plastik tatsächlich als Tastatur funktionieren kann. Aber Sie werden bald herausfinden, daß die Tasten-Bereiche auf Berührungen reagieren. Das bedeutet, daß (wenn der Computer eingeschaltet ist) ein leichter Fingerdruck auf eine der Tasten bewirkt, daß von der Tastatur aus eine Nachricht zum Computer geschickt wird.

Die 26 Buchstaben des Alphabets und die zehn Ziffern – in der gleichen Anordnung wie auf einer Schreibmaschine – fallen als erstes auf. Aber außer den Buchstaben und Ziffern stehen auf der Tastatur noch eine ganze Anzahl von Wörtern, Symbolen und winzigen grafischen Blöcken. Die Wörter sind die Kommandos und Anweisungen, die Sie dem Computer übermitteln können, um bestimmte Dinge ausführen zu lassen. Im Laufe der Lektüre dieses Buches und im Zuge unserer Experimente mit dem Computer werden Sie lernen, was all diese Wörter bedeuten.

Abbildung 1.1: Die Tastatur des ZX81-Computers

Der Grund dafür, daß diese Wörter direkt auf der Tastatur stehen, ist einfach. Anstatt die Kommandos zeichenweise einzutippen, brauchen Sie nur eine einzige Taste zu drücken, um ein Kommando oder ein Schlüsselwort zum Computer zu schicken. Das mag Ihnen im Augenblick noch nicht besonders bequem erscheinen: zunächst benötigen Sie vielleicht mehr Zeit, um die Kommandos zu suchen, als für das Ausschreiben der ganzen Wörter. Aber mit einiger Übung werden Sie mit der Tastatur-Anordnung vertraut sein, und Ihre Finger werden die gesuchten Tasten automatisch finden. Natürlich können Sie nicht erwarten, auf dieser Tastatur im „Zehn-Finger-System" zu schreiben, aber Sie werden überrascht sein, wie effektiv Sie schon nach ein paar Tagen mit der Tastatur umgehen können.

Da einige der Tasten mit bis zu fünf verschiedenen Wörtern oder Symbolen belegt sind, braucht man ein „Umschalt-System" für die Tastatur, um zwischen Wörtern und Symbolen abwechseln zu können. Es gibt verschiedene solcher Umschaltmöglichkeiten für den ZX81, und wir werden sie alle in Kapitel 2 betrachten. Ebenso wie die Positionen der Kommandos werden die Umschaltmöglichkeiten Ihnen in kürzester Zeit zu einer Selbstverständlichkeit werden.

Sehen Sie sich nun einmal die linke Seite Ihres Computers an (Abbildung 1.2). Dort finden Sie vier Anschlußmöglichkeiten. Die Funktion der einzelnen Anschlüsse wird aus den Beschriftungen deutlich: der erste dient zur Verbindung mit einem Fernsehgerät (TV), die beiden mittleren sind für einen Kassettenrecorder bestimmt und der letzte, mit „9V DC" beschriftet, dient zur Stromversorgung.

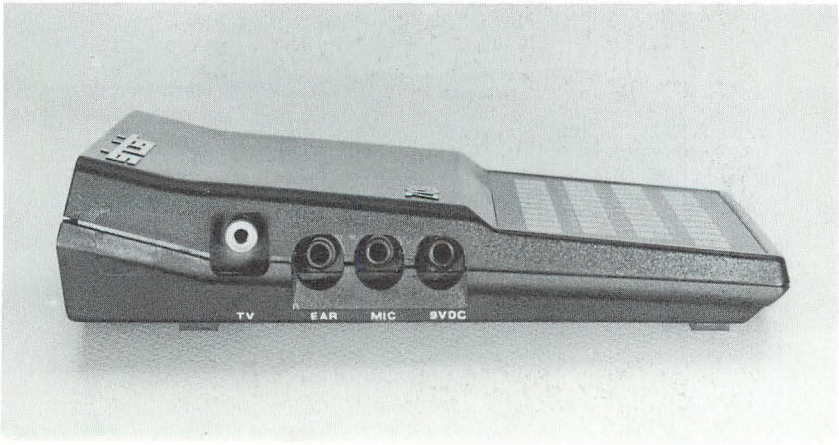


Abbildung 1.2: Die linke Seite des Computers

Alles was Sie benötigen, um die verschiedenen Anschlüsse herzustellen – Stecker, Kabel und andere Hardware – haben Sie zusammen mit Ihrem Computer erhalten. Wir wollen eine kurze Bestandsaufnahme dieser Dinge vornehmen. Eins der Kabel ist etwa einen Meter lang und hat an beiden Enden jeweils einen großen Stecker. Dies ist die Verbindung zum Fernsehgerät. Sie können nun das eine der beiden Enden in den „TV"-Anschluß des Computers stecken. Wir werden nachher sehen, wie das andere Ende ans Fernsehgerät angeschlossen wird. Übrigens ist es nicht gefährlich, eine der Verbindungen zu Ihrem Computer zu berühren, auch dann nicht, wenn der Strom eingeschaltet ist.

Weiterhin sollten Sie ein kürzeres Kabel mit zwei Steckern an beiden Enden vorfinden. Dies ist für den Kassettenrecorder bestimmt. Legen Sie es zunächst beiseite (aber verlieren Sie's nicht); der Anschluß des Kassettenrecorders wird ausführlich in Kapitel 2 beschrieben.

Schließlich finden Sie noch ein drittes Kabel vor mit einem Netzteil und einem Netzstecker für 220 Volt an der einen Seite und einem Stecker an der anderen Seite. Der Netzstecker paßt in eine beliebige Netzsteckdose und der kleine Stecker am anderen Ende in den „9V DC"-Anschluß an der Seite Ihres Computers. Sie können ihn nun einstecken, wenn Sie es ausprobieren möchten. Wenn Sie dies tun, ist der Computer eingeschaltet und bereit zu arbeiten. (Leider werden Sie nicht mitbekommen, was der Computer tut, solange Sie kein Fernsehgerät angeschlossen haben.) Es ist eine gute Idee, den Netzstecker wieder herauszuziehen, wenn Ihr Computer für längere Zeit nicht benutzt wird.

Abbildung 1.3: Die Rückseite des Computers

Auf der Rückseite Ihres Computers (s. Abb. 1.3) befindet sich eine Öffnung im Gehäuse, die etwa 6 cm lang und 1 cm hoch ist. In dieser Öffnung sieht man den Rand einer dünnen Platine, an der der Zusatz-Speicher oder der Drucker angeschlossen werden kann, falls Sie sich entscheiden sollten, eine dieser Erweiterungsmöglichkeiten zu kaufen. Wir werden beide Geräte in diesem Kapitel besprechen. (Falls Sie schon eins oder beide haben, ein Hinweis zur Vorsicht: schließen Sie keine Steckverbindungen an der Rückseite des Computers an und entfernen Sie keine dieser Verbindungen, während der Strom eingeschaltet ist.)

Nachdem wir nun das Äußere des Computers betrachtet haben, bleibt noch die Frage: wie sieht es im Inneren aus – was ist der *Inhalt* dieses schwarzen Kastens? Wieviel müssen Sie über den internen Aufbau des ZX81 wissen, um ein erfolgreicher Computer-Benutzer zu werden? Zweifellos haben Sie auch schon einiges gehört oder gelesen über die Fortschritte in der Technologie der Integrierten Schaltkreise („Chips"), die für die Personal-Computer-Revolution verantwortlich sind. Sie wissen vielleicht auch, daß der Chip, der in Ihrem ZX81 „denkt", ein Z80-Mikroprozessor ist. Doch wieviel müssen Sie über diese Dinge wissen, bevor Sie mit dem Computer Schach spielen oder ihn zur Haushaltsplanung für Ihre Familie benutzen können?

Die Antwort lautet, falls Sie es nicht schon erraten haben: *gar nichts*. Alles was Sie wissen müssen, ist außen sichtbar: die auf der Tastatur aufgedruckten Anweisungen und die Informationen, die auf Ihrem Fernsehbildschirm angezeigt werden. Um es kurz zu machen, kann die innere Architektur Ihres Computers so eigenartig zusammengefaßt werden, wie es Ian Frazier im folgenden Abschnitt seiner Kurzgeschichte „The Killion" getan hat:

> Für diejenigen, die mit Computern nicht vertraut sind, ist es eine Hilfe, wenn sie sich dabei ziemlich große, komplizierte Gebilde vorstellen. Computer bewegen sich in der Größenordnung zwischen einer Kaffeemaschine und einer riesigen Kirmes. Im Inneren des Computers soll man sich dann ein kurzes rotes Kabel vorstellen, an dessen beiden Enden jeweils ein Stecker angeschlossen ist. Außerdem soll es da noch ein blaues Kabel geben, durch das auch wiederum zwei Stecker miteinander verbunden sind, und dann auch noch ein grünes Kabel, ein gelbes Kabel, ein orangefarbiges Kabel, ein rosa Kabel und so weiter . . .*

Sie könnten aber auch an die Geschichte von einem Team speziell trainierter Fliegen glauben . . . Der springende Punkt an dem ganzen Unsinn ist der: Ihr neuer Computer ist so gestaltet, daß Sie sich um seine in-

* Wiedergabe genehmigt; © 1982 Ian Frazier. Original in The New Yorker.

terne Struktur und die Vorgänge in seinem Innenleben gar nicht zu kümmern brauchen. (Bedenken Sie, daß es nicht vorgesehen ist, die schwarze Plastik-Schachtel, in der Ihr Computer steckt, zu öffnen.) Was schon für viele andere Geräte in Ihrem Haus gilt – vom Telefon bis zum Aufzug – so müssen Sie kein technischer Experte sein, um die Maschine vollständig und erfolgreich zu nutzen.

Nachdem wir das festgestellt haben, bleibt noch anzumerken, daß es viele Bücher und Zeitschriften gibt, die Ihnen alle Fragen über Mikroprozessoren und Computer beantworten können, die Sie noch haben sollten. Wenn Sie einmal mit Ihrem Computer arbeiten und immer neue Anwendungen für seinen Gebrauch entdecken, wird vielleicht Ihr Interesse für diese Dinge geweckt, und Sie möchten mehr darüber lesen. Doch ist dies keine Voraussetzung für die Benutzung Ihres Computers. Die Entscheidung, wie weit Sie mit Ihren Forschungen gehen wollen, bleibt Ihnen überlassen.

Zum Schluß noch ein Wort an den Computer-Anfänger: vielleicht sind Sie auch ein wenig besorgt, daß Ihr Computer beschädigt werden könnte. Nun, Sie können beruhigt sein. Sie können mit Ihrem ZX81 kaum etwas falsch machen. Zwar werden Sie gelegentlich einen Hinweis zur Vorsicht finden. (Abgesehen von selbstverständlichen Vorsichtsmaßnahmen wie diesen: lassen Sie Ihren Hund nicht in Ihren Computer beißen; werfen Sie ihn nicht aus dem Fenster im zehnten Stockwerk; usw.) Aber Sie werden an Ihrer Beschäftigung noch mehr Spaß haben, wenn Sie feststellen, daß Sie – durch normalen Gebauch – dem Computer nicht weh tun können. Durch nichts von dem, was Sie eintippen, können Sie irgendetwas am Gerät beschädigen. Sie machen vielleicht gelegentlich Fehler, die Ihnen falsche Ergebnisse oder verlorene Daten bescheren. Aber wenn Sie solche Fehler erkannt haben, werden Sie sie im allgemeinen auch wieder korrigieren können. Fehler sind ein Teil des Lernprozesses; der Computer verzeiht sie immer.

Nachdem wir die Haupteinheit des Computers betrachtet haben, wollen wir nun sehen, wozu die übrigen Geräte dienen.

Die Nebenrollen

Das Fernsehgerät

Sie können jedes Fernsehgerät, das Ihnen zur Verfügung steht, an Ihren ZX81 anschließen. Ein kleines tragbares Gerät wäre am bequemsten, aber jedes andere tut's auch.

Das Fernsehgerät hat, kurz gesagt, folgende Aufgabe: es soll Ihnen alles über die Aktivitäten des Computers mitteilen. Wenn Sie Kommandos auf Ihrer Tastatur eingeben, so sehen Sie diese Kommandos auf dem Bildschirm ausgeschrieben, bevor Sie ihre Ausführung durch den Computer veranlassen. Oft werden Sie dem Computer eine Folge von Kommandos auf einmal eingeben, die der Reihe nach ausgeführt werden sollen. Eine solche Folge von Befehlen nennen wir ein *Programm*. Bevor Sie den Computer veranlassen, die Befehle eines Programms auszuführen, können Sie das Programm selbst auf dem Bildschirm anzeigen lassen, und zwar entweder Zeile für Zeile oder abschnittsweise. Sie können das Programm auf dem Bildschirm so lange stehen lassen, wie Sie wollen, damit Sie es ausgiebig betrachten und gegebenenfalls verbessern können. Diese Anzeige eines Programmes wird als *Liste* bezeichnet. Wenn Sie einen Fehler in Ihrem Programm finden, können Sie den betreffenden Programm-Teil *editieren* (berichtigen). Die einzelnen Schritte dieses Korrekturvorganges können Sie auf dem Bildschirm verfolgen. Schließich, wenn Sie ein Programm *laufen* lassen (d. h. wenn Sie dem Computer sagen, daß er die Befehle des Programms ausführen soll), werden Sie in den meisten Fällen die Ergebnisse auf dem Bildschirm sehen. Die Ergebnisse können aus Zahlen, Texten oder sogar aus Bildern bestehen, da der ZX81 zwei verschiedene Grafik-Möglichkeiten besitzt.

Selbstverständlich hat der Computer immer die Kontrolle über das, was auf dem Bildschirm angezeigt wird. Nachdem Sie sich einmal daran gewöhnt haben, die Informationen zu lesen, die der Computer auf den Bildschirm bringt, werden Sie allmählich sehr detaillierte Angaben darüber machen können, was der Computer gerade tut. Beispielsweise können Sie aus der angezeigten Information erfahren, daß dem Computer der Speicherplatz ausgeht. Wenn Sie einen Fehler gemacht haben, weist Sie der Computer diskret durch verschlüsselte Botschaften darauf hin. In den meisten Fällen werden Sie dann sofort wissen, wo der Fehler steckt und warum dem Computer die Sache nicht gefiel.

Während die Tastatur zu Ihrer Verständigung mit dem Computer dient, ist der Bildschirm das wichtigste und zunächst ausreichende Mittel für den Computer, um Ihnen Informationen zurückzugeben. Im ersten Moment meinen Sie vielleicht, daß es etwas viel ist, was Sie an Informationen auf dem Bildschirm finden, aber Sie werden sich schnell daran gewöhnen, auch die schwierigeren Zeichen und Hinweise auf dem Bildschirm zu verstehen, und Sie werden in der Lage sein, Ihre Aufmerksamkeit auf den Kern der Informationen zu konzentrieren, die der Computer Ihnen gibt.

Wenn Sie ein Programm schreiben, haben Sie außerdem eine Menge Möglichkeiten zu bestimmen, wohin und wie der Computer die Information auf den Bildschirm schreiben soll. Einige der Befehle, die Sie dem Computer geben können, sind speziell dafür vorgesehen zu bestimmen, wie und an welcher Stelle Zahlen, Texte und grafische Darstellungen auf dem Bildschirm erscheinen sollen.

Der Computer wird über einen UHF-Modulator, der bereits im Computer eingebaut ist, mit dem Fernsehgerät verbunden. Der Anschluß erfolgt einfach durch das beiliegende Kabel: während das eine Ende in die „TV"-Buchse des Computers gesteckt wird, wird das andere mit dem Antenneneingang des Fernsehgerätes verbunden.

Schalten Sie nun Ihr Fernsehgerät ein und stimmen Sie es ungefähr auf Kanal 36 (UHF) ab. Den Lautstärkeregler sollten Sie ganz zurückdrehen, um störendes Rauschen zu unterdrücken. Verbinden Sie den Netzstecker mit einer Steckdose und das andere Ende dieses Kabels mit der „9V DC"-Buchse Ihres Computers. Wenn alle Verbindungen richtig hergestellt sind, sollten Sie nun den Buchstaben K in inverser Darstellung (d. h. weiß

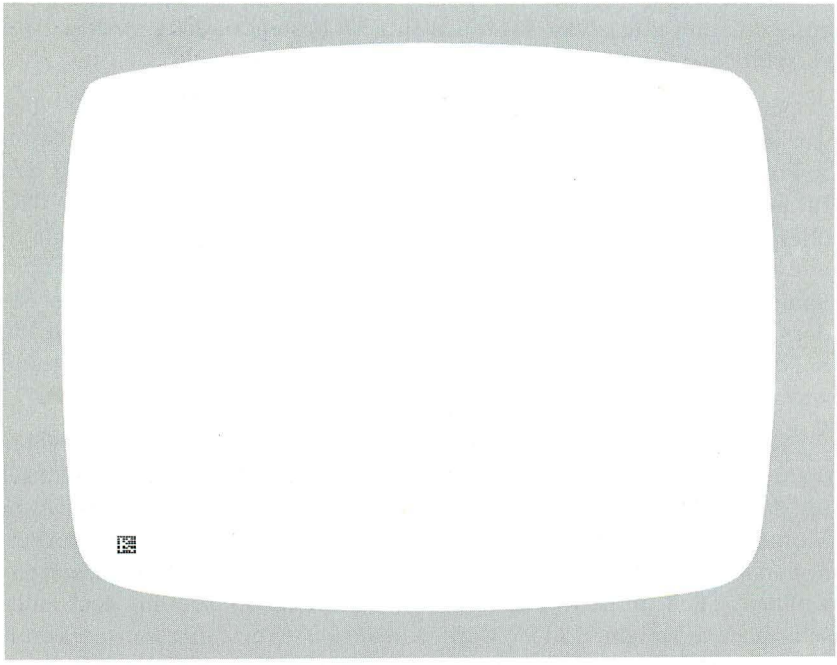


Abbildung 1.4: Nach dem Einschalten erscheint das Zeichen K

StB.-Scha.

auf schwarz) unten links auf dem Bildschirm sehen (vgl. Abbildung 1.4). Dieser Buchstabe steht für „Keyword" (Schlüsselwort) und bedeutet, daß Ihr Computer bereit ist, Kommandos entgegenzunehmen. Die Bildqualität können Sie mit den Reglern für die Helligkeit und den Kontrast am Fernsehgerät verbessern. Falls die Bildqualität nicht zufriedenstellend ist, probieren Sie eine andere Kanaleinstellung.

Sollten Sie gar kein Bild sehen, so überprüfen Sie noch einmal alle Verbindungen:

- Der Netzstecker muß in einer Steckdose in der Wand stecken. Der kleine Stecker am anderen Ende des Kabels muß mit dem „9V DC"-Anschluß an der Seite des Computers verbunden sein.
- Es muß eine Verbindung zwischen dem Antennen-Eingang des Fernsehgerätes und dem „TV"-Anschluß des Computers bestehen.
- An der Sendereinstellung des Fernsehgerätes muß der richtige Kanal – 36 (UHF) – eingestellt sein.

Wenn das Fernsehgerät und der Computer richtig angeschlossen sind, können Sie damit beginnen, Befehle einzugeben und mit den Möglichkeiten des ZX81 zu experimentieren. Tatsächlich brauchen Sie zunächst keine weiteren Geräte anzuschließen. Ein angeschlossener Fernseher reicht aus, um die vielfältigen Möglichkeiten Ihres Computers kennenzulernen.

Mit zunehmender Erfahrung werden Sie wahrscheinlich wünschen, weitere Möglichkeiten in Ihr System einzubauen. Vielleicht wollen Sie eine Speichererweiterung in den Computer einbauen, um längere Programme zu schreiben. Vielleicht möchten Sie Ihre Programme auch in einer dauerhaften Form speichern – oder Sie wünschen sich eine bequeme Möglichkeit, um die Programme anderer Leute auf Ihrem Computer laufen zu lassen. Schließlich möchten Sie vielleicht auch einige Ergebnisse, die Ihr Computer liefert, auf Papier ausdrucken lassen. Die folgenden Abschnitte dieses Kapitels beschreiben die verschiedenen Teile der Ausstattung, die es erlauben, all dies zu realisieren.

Zusätzlicher Speicher

Der Computer hat zwei verschiedene Arten von Speichern: eine dauerhafte und eine flüchtige. Sie werden vielleicht schon mit den Bezeichnungen dieser beiden Speicherarten vertraut sein: ROM (für Read Only Memory, d. h. nur lesbarer Speicher) und RAM (für Random Access Memory, d. h. Speicher mit wahlfreiem Zugriff). Im ROM, dem dauerhaften Speicher, werden alle die Informationen gespeichert, die die „Persönlichkeit" des ZX81 ausmachen. Diese Informationen bestimmen, welche Befehle Sie Ihrem Computer geben können, und wie der Computer

diese Befehle interpretiert und ausführt. Die im ROM gespeicherte Information geht nie verloren, auch dann nicht, wenn Sie Ihren Computer ausschalten.

In der anderen Speicherart – RAM – wird die Information über die laufenden Aktivitäten des Computers gespeichert. Darin stehen z. B. irgendwelche Befehle oder ein Programm, das Sie in den Computer eingegeben haben. In diesem Speicherbereich wird laufend notiert, welche Bilder der Computer auf den Fernsehbildschirm schickt. Auch die Daten, die Sie Ihrem Computer eingegeben haben, werden darin festgehalten, sowie Zwischen- und Endergebnisse von Berechnungen, die Sie den Computer ausführen ließen. Der RAM-Bereich ist also sehr dynamisch; die darin gespeicherte Information kann sich von einem zum anderen Augenblick ändern. Das wichtigste, was Sie sich über den RAM-Speicher merken müssen, ist dies: wenn Sie den Strom abschalten, ist alle Information, die im RAM gespeichert war, verloren.

Der Speicher des Computers wird in Einheiten gemessen, die *Bytes* genannt werden. Ein Byte ist der Speicherbedarf, den der Computer benötigt, um die Information eines Tastendrucks von der Tastatur zu speichern. Wenn Sie also irgendeinen Befehl oder ein Zeichen von der Tastatur aus eingeben, so braucht der Computer dazu ein Byte Speicherplatz.

Nun, Ihr ZX81 hat etwa 1000 Bytes RAM eingebaut. (Tatsächlich spricht man von 1K Bytes, wobei das K für 1024 steht.) Sie werden sehen, daß dies genug Speicherplatz ist, um kurze, jedoch interessante und nützliche Programme zu schreiben, die der Computer ausführen kann. Es ist sogar ziemlich viel Speicherplatz, solange Sie noch lernen, Ihren Computer zu benutzen.

Ab einem gewissen Punkt werden Sie jedoch wünschen, daß Ihr Computer mehr Speicher besitzt. Vielleicht möchten Sie selbst längere Programme schreiben, oder sehr lange Programme laufen lassen, die andere Leute geschrieben haben. Wenn Sie an diesem Punkt angelangt sind, können Sie ein zusätzliches Speichermodul kaufen, das Sie in den Computer einbauen können. Von Sinclair wird ein 16K RAM-Modul angeboten; andere Firmen produzieren Bausteine mit bis zu 64K. Wieviel Speicher Sie benötigen, hängt letztlich davon ab, was Sie mit Ihrem Computer machen wollen.

Diese Module passen in den offenen Ausschnitt an der Rückseite des Computers. Wenn Sie schon eine Speicher-Erweiterung gekauft haben, können Sie diese direkt anschließen, aber denken Sie daran: erst den Rechner ausschalten, bevor Sie hinten irgendetwas einstecken oder herausziehen! Wenn Sie noch kein zusätzliches Speicher-Modul gekauft haben, so können Sie sich ruhig noch Zeit lassen, bis Sie sich dazu entschlie-

ßen. Sie können sie auch noch später bekommen, und, wie bereits erwähnt, können Sie eine gründliche Einführung in die Fähigkeiten Ihres Computers auch so erwerben – mit dem vorhandenen 1K RAM.

Ein Kassettenrecorder

Erinnern Sie sich, daß wir ein *Programm* definiert haben als eine Folge von Anweisungen, die Sie dem Computer insgesamt übergeben. Wenn Sie ein Programm *starten*, so weisen Sie den Computer an, diese Befehle nacheinander – in einer bestimmten Reihenfolge – auszuführen. Und an folgendes sollten Sie sich noch in bezug auf Programme erinnern: Sie sind im RAM gespeichert, dem flüchtigen Speicherbereich. D. h. wenn Sie an einem Programm arbeiten und dann den Stecker für die Stromversorgung herausziehen – absichtlich oder zufällig – wird das Programm, an dem Sie gerade gearbeitet haben, verloren sein.

Aber Ihr Computer bietet eine gute Lösung für dieses Problem. Sie können Programme speichern, indem Sie sie direkt vom Computer auf Tonbandkassetten überspielen. Wenn Sie dann später ein Programm wieder laufen lassen wollen, können Sie es zurückspielen, und der Computer speichert es wieder im RAM. Um sicherzugehen, erfordert dieser Vorgang recht sorgfältige Verbindungen zwischen dem Kassettenrecorder und dem Computer. Wir werden diese Verbindungen in Kapitel 2 erörtern. Jedoch erfordert das Aufzeichnen von Computer-Programmen keine teuren oder hochentwickelten Aufzeichnungsgeräte. Wenn Sie irgendeinen Tonband-Kassettenrecorder im Haus haben, sind die Aussichten ganz gut, daß er zusammen mit Ihrem Computer benutzt werden kann. Wenn Sie keinen Recorder haben, werden Sie sich eventuell einen zulegen wollen. Ein preiswertes Gerät ist hier gerade richtig und voll ausreichend.

Der Drucker

Ein anderes Zusatzgerät, das Sie vielleicht kaufen wollen, ist der für den ZX81 entwickelte Drucker. Ebenso wie das zusätzliche Speicher-Modul wird der Drucker durch den Ausschnitt an der Rückseite des Computers angeschlossen. Bevor Sie den Anschluß herstellen, sollten Sie den Computer ausschalten. Der Drucker benötigt Spezialpapier-Rollen. Die Dokumente, die er produziert, sind zwar lesbar, aber nur von durchschnittlicher Qualität. Trotzdem wird die Fähigkeit, einen Ausdruck erzeugen zu können, oft von großem Vorteil sein.

Der Computer kennt mehrere Befehle, die es Ihnen erlauben, den Drucker direkt über die Tastatur zu steuern. Sie können z. B. den Com-

puter veranlassen, ein Programm auf dem Drucker aufzulisten oder den gesamten Inhalt des Bildschirms auf das Papier zu übertragen. Sie können auch Programme schreiben, die ihre Ergebnisse auf den Drucker schicken anstatt auf den Bildschirm. Wenn Sie sich entschließen, einen Drucker zu kaufen, werden Sie feststellen, daß es äußerst einfach ist, damit alle möglichen Ergebnisse Ihres Computers zu Papier zu bringen. (Anhang A beschreibt die Befehle, die Sie für die Arbeit mit dem Drucker brauchen: COPY, LLIST und LPRINT.)

Produzent, Regisseur und Drehbuchautor in einer Person

Als nächstes gilt unsere Aufmerksamkeit *Ihnen* und den verschiedenen Rollen, die Sie spielen können, um Ihren Computer für Sie arbeiten zu lassen.

Wir haben gesehen, daß der Computer so konstruiert ist, daß er bestimmte Befehle annehmen und ausführen kann, die Sie ihm übermitteln können, indem Sie die entsprechenden Tasten auf der Tastatur drücken. Wenn Sie die Tastatur betrachten, sehen Sie, daß es dort etwa fünf Dutzend verschiedene Wörter gibt, die der Computer versteht. Sie können sich diese Wörter – zusammen mit ihren jeweiligen Verwendungsregeln – als eine Art Sprache vorstellen: eine *Computersprache*. Diese Computersprache ist viel einfacher als eine menschliche Sprache. Sie hat einen sehr begrenzten Wortschatz und eine einfache, regelmäßige Grammatik. Aus diesem Grund wird es Ihnen viel leichter fallen, sie zu lernen, als es bei einer Fremdsprache wie Französisch oder Italienisch der Fall wäre. Außerdem sind die meisten Wörter dieses Wortschatzes Wörter aus der englischen Umgangssprache. Es wird Ihnen leicht fallen, sie zu behalten, wenn Sie erst einmal ihre Wirkung kennengelernt haben.

Es gibt viele verschiedene Computersprachen. Diejenige, die in den ZX81 – wie auch in die meisten Klein-Computer – eingebaut ist, heißt BASIC (eine Abkürzung für Beginner's All-purpose Symbolic Instruction Code). Sie werden in den nächsten Jahren immer mehr über BASIC hören, denn es ist eine so einfache Sprache, daß sie überall von Menschen erlernt wird, um damit alle möglichen Arten von Problemen mit Computern lösen zu können. Heute lernen die Kinder schon in der Schule, Computer-Programme zu schreiben. Bald wird das Programmieren in BASIC zu Hause und im Büro vielleicht so selbstverständlich sein, wie z. B. das Maschinenschreiben oder die Benutzung von Taschenrechnern.

BASIC wird zunehmend beliebter, da es auch von Leuten benutzt werden kann, die nicht unbedingt Computer-Experten sind. Sie werden jedoch überrascht sein, wie leistungsfähig BASIC ist, wenn Sie es auf ih-

rem Computer benutzen. Diese einfache Sprache hat tatsächlich vieles gemeinsam mit anderen Sprachen, die professionelle Computer-Programmierer auf viel größeren Computern als dem ZX81 benutzen.

Die erste Rolle, die Sie mit Ihrem neuen Computer spielen werden, wird die eines BASIC-Programmierers sein. Wenn Sie an eine Aufgabe denken, die Ihr Computer für Sie erledigen soll, werden Sie feststellen, daß Sie die Aufgabe folgendermaßen organisieren:

1. Sie zerlegen die Aufgabe in eine Reihe von elementaren Schritten, die der Computer nacheinander ausführen kann. Diese Schritte können Berechnungen enthalten, wiederholt vorkommende Aufgaben oder sogar Entscheidungen, die Sie vom Computer aufgrund verfügbarer Daten treffen lassen wollen.
2. Sie bestimmen den effektivsten Weg, um diese Schritte durch eine Reihe von BASIC-Anweisungen auszudrücken. Wenn Ihr Programm lang ist und viele Anweisungen enthält, beginnen Sie wahrscheinlich damit, daß Sie das Programm niederschreiben, anstatt es aus dem Stegreif in den Computer einzutippen.
3. Nachdem Sie ein Programm entwickelt haben, geben Sie es zeilenweise in den Computer ein. Dann machen Sie einen Versuchslauf, um zu sehen, ob er tatsächlich das tut, was Sie sich ausgedacht haben. Wenn das Programm Berechnungen mit numerischen Werten (Zahlen) durchführt, werden Sie es erst einmal mit verschiedenen Zahlenreihen probieren, bis Sie sicher sind, daß exakte Ergebnisse herauskommen.
4. Wenn die Ergebnisse Ihres Programms falsch sind, gehen Sie zurück zu den notierten BASIC-Anweisungen und versuchen Sie herauszufinden, wo der Fehler liegt. Oft werden Sie aus der Art des Fehlers direkt ersehen können, worin das Problem besteht. Bei anderer Gelegenheit können die Fehler schwieriger zu finden sein, und Sie brauchen vielleicht Stunden – oder sogar Tage – bevor Sie das Rätsel Ihres eigenen Programms lösen.

Sie werden Ihr erstes BASIC-Programm in Kapitel 2 sehen. Tatsächlich werden Sie sich bei diesem ersten Programm mehr auf die ZX81-Tastatur konzentrieren als auf die Bedeutung der BASIC-Anweisungen, aber sehr bald werden Sie Ihre eigenen Programme schreiben. Beachten Sie, daß der Vorgang der Programmentwicklung, wie er in den oben angeführten vier Schritten umrissen wurde, auch den Weg zur Fehlersuche und -behandlung zeigt. Es ist nahezu unvermeidlich, daß neue Programme Fehler enthalten. Das ist kein Grund zur Verwirrung. Der ZX81 findet einige Fehler schon bei der Eingabe des Programms. Andere Fehler werden entdeckt, wenn Sie das Programm laufen lassen. Wir werden se-

hen, daß der Computer mehrere Möglichkeiten hat, um Ihnen bei der Fehlersuche und -korrektur in Ihren Programmen zu helfen.

Als Sie Ihren Computer gekauft haben, haben Sie wahrscheinlich festgestellt, daß Sie auch einige Programme kaufen konnten, die von anderen Leuten geschrieben wurden, um für den ZX81 verwendet zu werden. Diese Programme sind auf Kassetten gespeichert und relativ teuer. Doch ist dies eine weitere Möglichkeit, Ihren Computer zu benutzen, indem Sie Programme, die von anderen entwickelt wurden, dazu verwenden, bestimmte Aufgaben zu erfüllen. In Kapitel 2 werden wir einige der Vor- und Nachteile des Programmkaufs (im Gegensatz zu Eigenentwicklungen) besprechen. Im Augenblick stellen Sie sich vielleicht die Frage, warum Sie die Mühe auf sich nehmen sollten, das Programmieren in BASIC zu erlernen, wenn es schon Programme gibt, die Sie einfach kaufen und benutzen können?

Tatsächlich mag es viele Leute geben, die den ZX81-Computer kaufen und niemals lernen, ein BASIC-Programm zu schreiben. Sie werden sich Computerprogramme immer nur bei ihrem Computerhändler nach ihren jeweiligen Bedürfnissen aussuchen. Dies ist ein legitimer Gebrauch des Computers; sie *müssen* keineswegs lernen zu programmieren, um Ihren Computer erfolgreich benutzen zu können. Wenn Sie jedoch niemals versuchen, BASIC zu lernen und Ihre eigenen Programme zu schreiben, verzichten Sie auf eine bedeutende Möglichkeit zu Ihrer Weiterbildung. Wenn Sie lernen, Ihre eigenen Programme zu entwickeln, wird ihr Computer ein wertvolles Werkzeug werden, das Sie stets für Ihre individuellen Computeranwendungen benutzen können. Außerdem gibt Ihnen das Erlernen einer Programmiersprache neue Einsichten in ein Werkzeug, das in unserer Gesellschaft zunehmende Bedeutung gewinnt.

Zusammenfassung

Damit wir uns mit dem Computer verständigen können, benötigen wir Ein- und Ausgabegeräte. Beim ZX81 ist das hauptsächliche Eingabegerät die eingebaute Tastatur. Auf dieser Tastatur sind alle Befehle aufgedruckt, die Sie Ihrem Computer geben können, so daß jedes Wort durch einen einzigen Tastendruck eingegeben werden kann.

Der Fernseh-Bildschirm ist das hauptsächliche Ausgabegerät. Über ihn erhalten Sie eine Menge von Informationen über Ihre Programme, deren Ergebnisse und die derzeitigen Aktionen Ihres Computers. Zusätzlich können Sie einen Drucker mit Ihrem Computer verbinden, wenn Sie dauerhaftere Ausgaben wünschen.

Ein einfacher Tonband-Kassettenrecorder kann ebenfalls mit dem ZX81 verbunden werden, wenn Sie die Programme, die Sie geschrieben haben, speichern wollen oder wenn Sie gekaufte Programme laufen lassen wollen. Außerdem gibt es zusätzliche Speicher-Module, um den flüchtigen Speicher-Bereich (RAM) Ihres Computers zu vergrößern.

Die Wörter, die Sie auf Ihrer Tastatur sehen, stellen das Vokabular der Programmiersprache BASIC dar. Obwohl das Erlernen dieser Sprache nicht unbedingt notwendig ist, um den Computer zu benutzen, werden doch Ihre Computer-Erfahrungen dadurch gründlicher und zufriedenstellender.

Kapitel 2

Der erste Akt: Die Programmeingabe

Einleitung

Wenn Sie den Anweisungen gefolgt sind, die in Kapitel 1 im einzelnen ausgeführt wurden, ist Ihr Computer nun eingeschaltet, das Fernsehgerät ist angeschlossen, und Sie sind so weit, daß Sie mit Ihrer Arbeit beginnen können. Als erstes müssen Sie lernen, mit der Tastatur umzugehen; das ist das Hauptziel dieses Kapitels. Aber zunächst wollen wir uns die Programmiersprache BASIC etwas genauer betrachten. Wir werden sehen, wie ein Programm aussieht, und werden in Grundzügen über den Aufbau eines Programmes sprechen. Dann werden Sie ein kurzes Programm in den Computer eingeben. Dabei werden Sie sehen, wie der Computer Ihnen hilft, Fehler, die Sie vielleicht gemacht haben, zu erkennen und zu verbessern.

Dann werden Sie sehen, wie Sie einen Kassettenrecorder zusammen mit Ihrem Computer verwenden können. Wir werden sorgfältig und ausführlich auf die Verbindungen eingehen, die Sie herstellen müssen, um Programme auf einer Kassette zu speichern bzw. um ein Programm von der Kassette wieder in den Computer zu laden.

Das bringt uns auf ein drittes Thema: Programme, die Sie auf Kassetten kaufen können. Wir werden uns dazu ein Beispiel ansehen und das Kapitel mit einigen Hinweisen beschließen, die Sie als Käufer von fertigen Programmen für Ihren ZX81 berücksichtigen sollten.

Der Computer lernt seine Lektionen

Jeder Computer muß natürlich zunächst einmal lernen, sich freundlich und höflich zu verhalten. Deshalb soll das erste Programm, das wir uns anschauen, dem Computer folgende Dinge beibringen:

1. nach Ihrem Namen fragen;
2. ein Muster auf den Bildschirm zeichnen;
3. eine Begrüßung auf dem Bildschirm ausgeben.

Vielleicht meinen Sie, daß dies etwas lächerliche Aufgaben für ein Computer-Programm sind. Im Augenblick geht es uns aber nicht um Zweckmäßigkeit, sondern darum, Ihnen eine gründliche Einführung in die Benutzung der Tastatur zu geben. Nachdem Sie das Programm eingegeben haben, werden Sie sämtliche Verwendungsmöglichkeiten der Tastatur kennengelernt haben.

Die Programmiersprache BASIC

Das „Begrüßungsprogramm" ist in Abbildung 2.1 wiedergegeben. Sehen Sie es sich erst einmal genau an, bevor Sie es in Ihren Computer eingeben. Sie können eine Menge über die Sprache BASIC lernen, wenn Sie sich ein BASIC-Programm nur ansehen.

Wir haben in Kapitel 1 erfahren, daß BASIC nur einen sehr begrenzten Wortschatz hat, der überwiegend aus gebräuchlichen Wörtern der englischen Sprache besteht. Tatsächlich stehen in dem Begrüßungs-Programm nur ein paar Wörter, die so aussehen, als wären sie Abkürzungen für bestimmte Wörter (CLS, LEN, CHR$); die meisten anderen sind

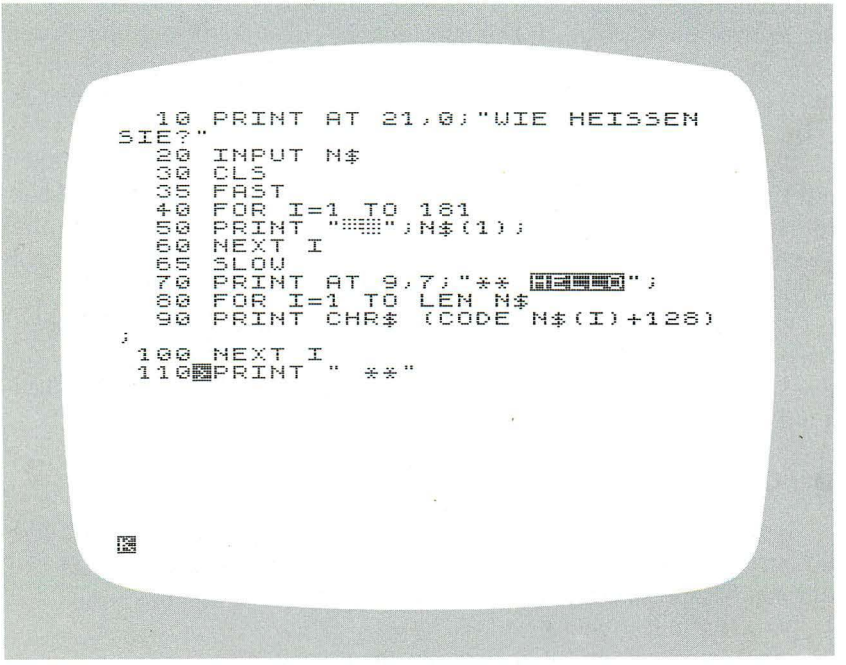


Abbildung 2.1: Das Begrüßungsprogramm

ganz geläufige englische Wörter. Wenn Sie die englischen Wörter kennen, so heißt das nicht, daß Sie unmittelbar verstehen können, was die einzelnen Programmzeilen tun sollen, aber zumindest können Sie feststellen, daß das Programm nicht allzu einschüchternd wirkt.

Was können wir nun an diesem Programm über die Eigenschaften von BASIC herausfinden? Zunächst einmal, daß ein BASIC-Programm in Zeilen gegliedert ist, die jeweils numeriert sind. Dieses Programm hat 13 Zeilen. Die Länge eines Programms kann sich zwischen einer einzigen Zeile und vielen hundert Zeilen bewegen, je nachdem, welche Aufgabe das Programm erfüllen soll.

Die Zeilen in unserem Programm sind von 10 bis 110 durchnumeriert, wobei die Abstände jedoch nicht gleichmäßig sind. Die meisten Zeilennummern sind in Zehner-Schritten, einige aber auch in Fünfer-Schritten geschrieben. Wenn Sie ein BASIC-Programm schreiben, können Sie die Zeilen beliebig durchnumerieren. Für den Computer ist lediglich die Reihenfolge der Zeilennummern wichtig, nicht aber die Abstände zwischen ihnen. Dieses Programm könnte genausogut von 1 bis 13 oder von 100 bis 1300 durchnumeriert sein, der Computer würde es ebenso akzeptieren. Der Vorteil der Verwendung von gleichmäßigen Zeilenabständen, z. B. einer Schrittweite von 10, liegt darin, daß Sie dadurch die Möglichkeit haben, auch nachträglich noch Zeilen in ein Programm einzufügen. Dies ist leicht möglich, wenn Sie bei der Numerierung entsprechende Zwischenräume benutzen. In unserem Programm wurden z. B. die Zeilen 35 und 65 eingefügt, nachdem das übrige Programm schon geschrieben war.

Zwei der numerierten Zeilen in diesem Programm – 10 und 90 – sind länger als eine Bildschirmzeile. Das ist vollkommen in Ordnung. Der Computer baut die Informationen auf dem Bildschirm so auf, daß jede Bildschirmzeile bis zu 32 Zeichen enthalten kann; dagegen kann eine Zeile eines BASIC-Programms viel länger sein. Wenn eine Programmzeile die Länge einer Bildschirmzeile überschreitet, wird sie einfach auf der nächsten Zeile fortgesetzt. Dadurch ist es möglich, daß eine Zeile mitten im Wort getrennt wird, oder – wie z. B. in Zeile 90 – nur noch das letzte Zeichen der Programmzeile, ein Semikolon, automatisch in die nächste Bildschirmzeile rückt. Dies wirkt sich aber in keiner Weise auf die Ausführung der Zeile durch den Computer aus.

Das erste Wort in jeder Zeile eines BASIC-Programms für den ZX81 ist immer ein Schlüsselwort. Das Schlüsselwort teilt dem Computer mit, welcher Art die nachfolgende Anweisung ist. Jedes Schlüsselwort muß sich nach seiner eigenen, besonderen „Grammatik" richten; d. h. der Computer weiß, welche Art von Anweisungsschema nach jedem Schlüsselwort zu erwarten ist. Es gibt 26 Schlüsselwörter auf Ihrer Tastatur, die

Abbildung 2.2: Die Schlüsselwörter

jeweils über einer Buchstaben-Taste stehen. Abbildung 2.2 zeigt die Anordnung der Schlüsselwörter.

Wenn Sie eine numerierte Zeile eines BASIC-Programms in Ihren Computer eingeben, legt der Computer diese Zeile in seinem Speicher ab und zeigt Sie Ihnen auf dem Bildschirm an. Er beginnt mit der Ausführung des Programms erst dann, wenn Sie ihn dazu veranlassen. Vorher überwacht er Ihr Programm lediglich und wartet darauf, daß Sie ihm mitteilen, was er als nächstes tun soll. Wir werden sehen, wie dies funktioniert, nachdem wir die Eingabe des Programms beschrieben haben.

Die Tastatur – Schlüsselwörter und Betriebsarten

Im Augenblick ist Ihr Bildschirm leer, abgesehen von dem invers dargestellten K in der unteren linken Ecke. Sie erinnern sich, daß das [K] für *Schlüsselwort* (engl.: keyword) steht. Durch diese *Hinweis-Zeichen* (in Ihrem Benutzer-Handbuch als *Cursor* bezeichnet) teilt Ihnen der Computer mit, daß er bereit ist, ein beliebiges Schlüsselwort von der Tastatur anzunehmen. Wenn Sie anfangen, das Programm einzugeben, werden Sie noch andere Hinweis-Zeichen sehen – und zwar [L], [G], [F] und [S], alle in inverser Darstellung. Aus diesen Hinweisen erfahren Sie, in welcher „Betriebsart" die Tastatur ist, d. h. welche Art von Wörtern oder Symbolen Sie dem Computer als nächstes eingeben können.

Wir werden jede Zeile des Programms besprechen, einen Tastendruck nach dem anderen. Sie sollten beim Eingeben jeder Zeile jeweils die dazugehörige Beschreibung lesen. (Zusätzlich werden Sie eine kurze, allgemeine Erklärung dessen lesen, was jede Zeile *tut*. Im Augenblick jedoch sollten Sie sich auf die Tastatur konzentrieren, nicht auf die Bedeutung der Zeilen. Später, wenn Sie die Tastatur genau kennen, werden Sie viel Zeit haben, sich näher mit BASIC vertraut zu machen.)

Das Programm wird zeilenweise eingegeben

Die erste Zeile des Programms enthält eine PRINT-Anweisung. PRINT weist den Computer an, eine bestimmte Information auf dem Bildschirm anzuzeigen. In der ersten Zeile steht nach PRINT das Wort AT, welches eine von mehreren Möglichkeiten ist, genau anzugeben, *wo* Sie die Information auf dem Bildschirm sehen möchten. Die „Information" ist in diesem Fall eine Frage – WIE HEISSEN SIE? – die zwischen Anführungszeichen am Ende der PRINT-Anweisung steht.

Wir wollen diese Zeile nun eingeben. Wenn Sie schon etwas anderes in Ihren Computer eingetippt haben, werden Sie bemerkt haben, daß für die Tasten kein besonders starker Druck erforderlich ist. Ein leichtes An-

tippen genügt. Während des Tippens müssen Sie allerdings den Bildschirm beobachten, um sich zu vergewissern, daß Sie die Tasten auch richtig gedrückt haben.

Wie wir gesehen haben, müssen Sie zu Beginn jeder Zeile Ihres BASIC-Programms zuerst eine Zeilennummer eintippen. Für die erste Zeile tippen Sie die Ziffern 1 und 0. Nach jeder Eingabe eines Zeichens sehen Sie, daß der Cursor mit dem Buchstaben K um eine Stelle nach rechts rückt und die Ziffern links davon erscheinen.

Als nächstes suchen Sie den PRINT-Befehl. Er steht über der Taste mit dem Buchstaben P (ziemlich weit rechts auf der Tastatur, in der zweiten Reihe von oben). Berühren Sie die Taste und schauen Sie sich an, was auf dem Bildschirm geschieht. Das ganze Wort PRINT erscheint auf einmal in der Zeile. Außerdem ändert sich das Cursor-Zeichen in ein [L]. Das weist Sie darauf hin, daß Sie jetzt die Betriebsart zur Eingabe eines Schlüsselwortes verlassen haben und nun in der Betriebsart zur Eingabe von *Buchstaben* (letter) sind. Die nächste Taste, die Sie drücken, wird als Buchstabe oder Ziffer registriert, nicht als Schlüsselwort.

Um ein beliebiges Schlüsselwort einzugeben, müssen Sie also nichts weiter tun, als eine Taste zu drücken, auf der das Schlüsselwort steht. Tatsächlich *können* Sie ein Schlüsselwort nicht in einzelnen Buchstaben eingeben. Wenn Sie die Buchstaben P-R-I-N-T einzeln eintippen würden, würde der Computer dies *nicht* als das Schlüsselwort PRINT erkennen. Immer wenn ein Schlüsselwort der BASIC-Grammatik entsprechend richtig eingegeben wird, zeigt der Computer dies durch den Cursor mit dem Buchstaben K an. Wenn Sie das inverse Zeichen K nicht sehen, wissen Sie, daß Sie kein Schlüsselwort eingeben können.

Das nächste Wort in der ersten Zeile ist AT. AT ist eine von 25 *Funktionen* auf der Tastatur. Die Funktionen stehen unterhalb der Buchstaben-Tasten (mit Ausnahme der V-Taste). Sie sind in Abbildung 2.3 zu sehen. Einige dieser Funktionen sind Funktionen im mathematischen Sinn, wie Sinus (SIN), Cosinus (COS) und der natürliche Logarithmus (LN). Wenn Sie vorhaben, Ihren Computer für wissenschaftliche oder mathematische Anwendungen zu benutzen, kennen Sie diese Funktionen sicherlich schon. Andere Funktionen auf Ihrer Tastatur sind ganz und gar nicht mathematisch. Sie können sie sich als Programmier-Funktionen vorstellen, denn nachdem Sie sie einmal kennengelernt haben, werden viele Programmieraufgaben leichter werden. AT z. B. ist eine „Funktion", die es Ihnen recht bequem macht, zu bestimmen, an welcher Stelle auf dem Bildschirm eine Information erscheinen soll.

Alle diese Funktionen – mathematische und andere – haben eins gemeinsam. Um eine Funktion einzugeben, müssen Sie die Tastatur in die

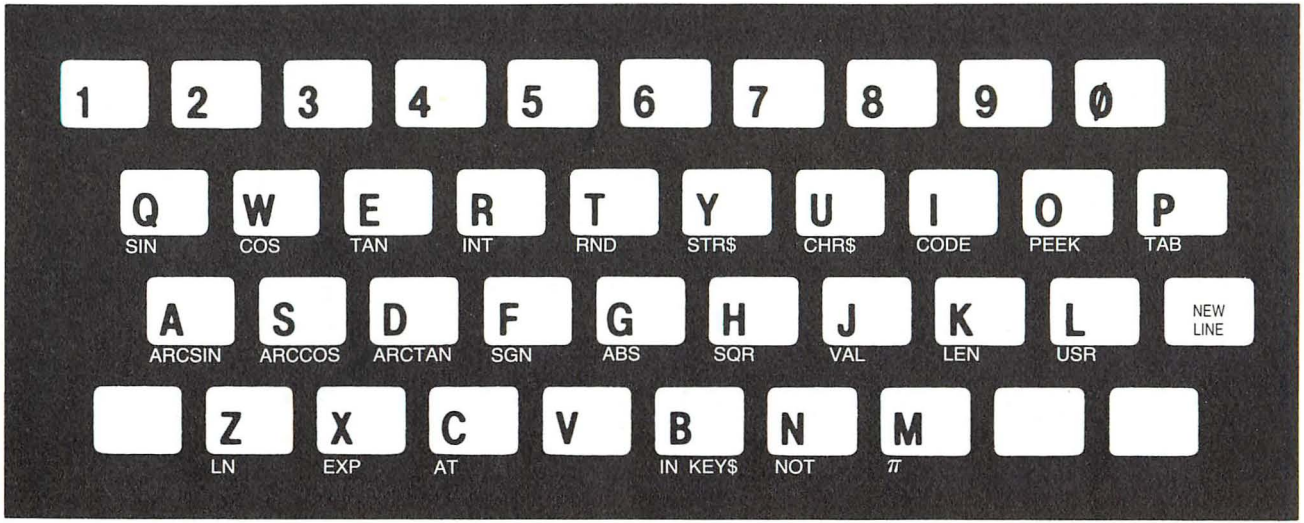


Abbildung 2.3: Die Funktionen

Funktions-Betriebsart umschalten (engl.: shift). Dazu müssen Sie zwei Tasten gleichzeitig drücken:

- die SHIFT-Taste in der unteren linken Ecke der Tastatur und
- die FUNCTION-Eingabe-Taste auf der rechten Seite der Tastatur, direkt unter der P-Taste.

Drücken Sie zunächst mit einem Finger die SHIFT-Taste, dann die FUNCTION-Taste, und beobachten Sie, was auf dem Bildschirm passiert. Das Cursor-Zeichen verändert sich von einem [L] zu einem [F]. Sie sind nun in der Betriebsart zur Eingabe von Funktionen. Sie können eine beliebige Buchstabentaste drücken, unter der eine Funktion steht, und die Funktion erscheint auf dem Bildschirm innerhalb Ihrer BASIC-Zeile.

Suchen Sie nun die AT-Funktion auf Ihrer Tastatur. Sie befindet sich unterhalb der C-Taste. Drücken Sie diese Taste, und in Ihrer Zeile erscheint das Wort AT. Ihre Zeile sieht nun so aus:

```
10 PRINT AT
```

Beachten Sie, daß das Cursor-Zeichen nun wieder ein [L] ist.

Genauso wie die Schlüsselwörter werden auch die Funktionen durch einen einzigen Tastendruck eingegeben. Die Eingabe der einzelnen Buchstaben A-T in der Betriebsart [L] hätte eine andere Bedeutung als die Eingabe der AT-Funktion in der Funktions-Betriebsart.

Als nächstes müssen wir in unserer Zeile zwei Zahlen eingeben. Wie Sie vielleicht schon vermutet haben, stellen diese Zahlen die „Adresse" auf dem Bildschirm dar, an der der Computer die Information ausgeben soll. (Die Zahlen 21,0 bedeuten Zeile 21, Spalte 0; wir werden genau sehen, wie dies funktioniert, wenn wir die AT-Funktion in Kapitel 3 näher betrachten.)

Bevor wir jedoch weitertippen, sollten wir erfahren, was man tut, wenn man sich vertippt hat. Solche Fehler sind unvermeidbar. Vielleicht haben Sie schon einen gemacht und wissen nicht recht, wie Sie weitermachen sollen.

Der Computer ermöglicht Ihnen auf sehr einfache Weise, jede beliebige Eingabe wieder zu „löschen". Dazu müssen Sie wieder zwei Tasten gleichzeitig drücken:

- die SHIFT-Taste und
- die 0-(Null-)Taste.

Beachten Sie bitte zwei Dinge in bezug auf die 0-Taste (oben rechts auf der Tastatur). Erstens: die Null hat einen Querstrich, damit man sie von dem Buchstaben O unterscheiden kann. (Dieser Querstrich erscheint auch auf der Bildschirmdarstellung). Zweitens: über der Null-Taste steht

– in roter Schrift – das Wort RUBOUT (Löschen). Durch gleichzeitiges Drücken der SHIFT- und der RUBOUT-Taste können Sie alle Tipp-Fehler „auslöschen".

Versuchen Sie es nun einmal. Tippen Sie einen Buchstaben, der nicht in Ihre Zeile gehört. Drücken Sie dann gleichzeitig die SHIFT- und die RUBOUT-Taste. Beobachten Sie, was auf dem Bildschirm passiert. Der [L]-Cursor geht zurück über den unerwünschten Buchstaben. Der Buchstabe ist verschwunden, und Sie können weitertippen.

Tippen Sie nun die beiden Zahlen ein, zunächst die 21, dann ein Komma (,), dann die 0 und ein Semikolon (;). Diese Zeichensetzung – auch wenn sie Ihnen im Augenblick noch unverständlich erscheinen mag – ist für die Bedeutung der Zeile wesentlich und muß deshalb genau beachtet werden. Das Komma befindet sich in der unteren rechten Ecke der Tastatur, auf der gleichen Taste wie der Punkt (.). Das Semikolon liegt auf der X-Taste. Sowohl das Komma als auch das Semikolon sind auf der Tastatur rot aufgedruckt.

Tatsächlich hat jede Taste auf der Tastatur ein Zeichen oder Wort, das in rot geschrieben ist. Wir werden diese als Umschalt- oder *„SHIFT"-Funktionen* bezeichnen. Um sie einzugeben, müssen Sie zunächst die SHIFT-Taste drücken und dann – gleichzeitig – die gewünschte Taste. Sämtliche *Umschalt-Funktionen* sind in Abbildung 2.4 angegeben.

Versuchen Sie, ein Komma einzutippen. Wenn Sie zuerst einen Punkt erhalten, so heißt das, daß Sie die SHIFT-Taste nicht fest genug gedrückt haben (oder nicht genau an der richtigen Stelle). Löschen Sie den Punkt wieder (SHIFT-RUBOUT) und versuchen Sie's noch einmal. Tippen Sie dann noch die Null und das Semikolon ein, um die „Adresse" zu vervollständigen. Die Zeile auf Ihrem Bildschirm sollte nun wie folgt aussehen:

```
10 PRINT AT 21,0;
```

Der Rest der Zeile besteht aus der aktuellen Mitteilung, die auf dem Bildschirm ausgegeben wird, wenn der Computer diese PRINT-Anweisung ausführt. Der Text steht zwischen Anführungszeichen. Das Anführungszeichen steht auf der P-Taste; es ist ein Umschalt-(SHIFT-)Zeichen. Drücken Sie SHIFT-P; das Anführungszeichen sollte auf dem Bildschirm erscheinen. Nun tippen Sie den Text ein: WIE HEISSEN SIE, einen Buchstaben nach dem anderen. (Die Leertaste ist in der unteren rechten Ecke der Tastatur.) Wenn Sie das I von SIE eintippen, bemerken Sie, daß die Bildschirmzeile zu Ende ist. Wenn Sie jetzt das E eintippen, erscheint dies auf dem Bildschirm automatisch in der nächsten Zeile. Die Frage endet mit einem Fragezeichen, das als Umschaltzeichen auf der C-Taste liegt.

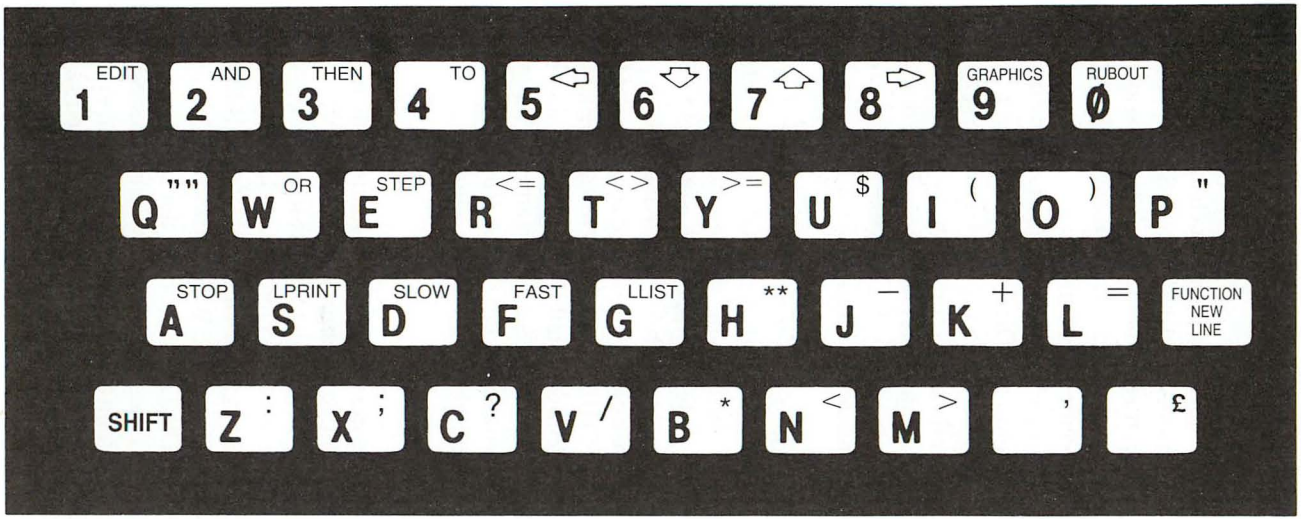


Abbildung 2.4: Die Umschalt-Funktionen

Bevor Sie das Anführungszeichen am Schluß eingeben, wollen wir eine andere besondere Eigenschaft Ihres Computers ausprobieren. Normalerweise würden Sie die Zeile fertig schreiben und dann die NEW LINE-Taste drücken, um dem Computer mitzuteilen, daß die Zeile vollständig ist. Jetzt aber – um zu sehen, was passiert, wenn Sie einen Fehler machen – drücken Sie NEW LINE, bevor Sie das letzte Anführungszeichen eintippen.

Automatische Fehlersuche

Was Sie auf dem Bildschirm sehen, nachdem Sie die NEW LINE-Taste gedrückt haben, ist in Abbildung 2.5 wiedergegeben. Ein neues Cursor-Zeichen ist neben dem L-Cursor aufgetaucht. Der [S]-Cursor steht für *Syntax-Fehler*. Das Wort *Syntax* bezieht sich auf die Regeln der speziellen „Grammatik", die Sie immer beachten müssen, wenn Sie ein BASIC-Programm schreiben. Wenn Sie eine Zeile eingeben, die gegen eine dieser Grammatik-Regeln verstößt, stellt der Computer dies sofort fest und wei-

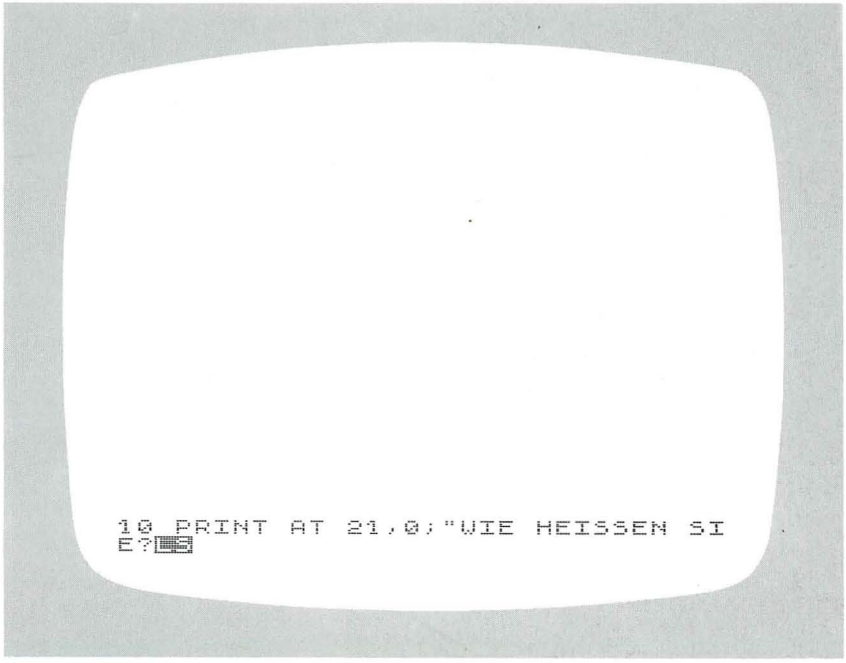


Abbildung 2.5: Der Computer antwortet auf einen Syntax-Fehler

gert sich, die Zeile als Teil des Programms anzunehmen. Der S-Cursor ist die vornehme Art des Computers, Ihnen mitzuteilen, daß Sie etwas falsch gemacht haben. Sie müssen sich die fragliche Zeile noch einmal anschauen.

Da Sie diesen besonderen Fehler absichtlich gemacht haben, wissen Sie natürlich genau, was falsch ist. Das abschließende Anführungszeichen fehlt. Wenn Sie jetzt das Anführungszeichen eintippen (SHIFT-P), verschwindet der S-Cursor wieder und Ihre Zeile ist vollständig. Drücken Sie wieder die NEW-LINE-Taste und beobachten Sie, was passiert. Abbildung 2.6 zeigt die Ergebnisse. Die fehlerfreie Programmzeile bewegt sich zum oberen Teil des Bildschirms. D.h., daß der Computer sie in seinem Speicher als erste (und im Augenblick auch einzige) Programmzeile abgelegt hat. Der [K]-Cursor ist wieder in der unteren linken Ecke des Bildschirms aufgetaucht, um Ihnen anzuzeigen, daß der Computer nun die nächsten Anweisungen erwartet.

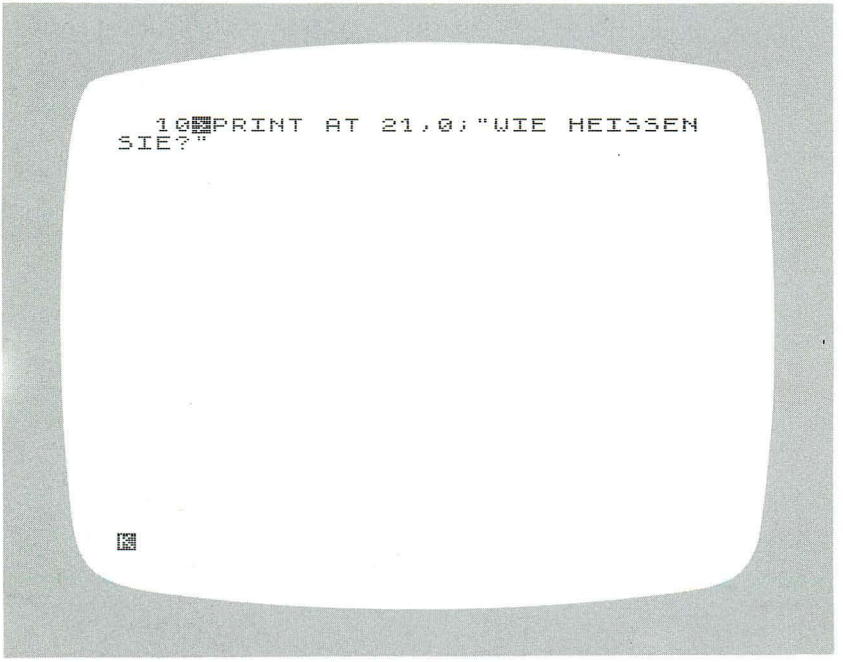


Abbildung 2.6: Korrekte Eingabe einer Programmzeile

Die zweite Zeile: Die Benutzung des Editors

Lassen Sie uns nun die zweite Zeile des Programms eintippen, die Zeile 20:

```
20 INPUT N$
```

Das Schlüsselwort INPUT weist den Computer an, die Programmausführung zu unterbrechen und auf eine Eingabe über die Tastatur zu warten. Durch den INPUT-Befehl können Sie eine Art von *Dialog* zwischen dem Computer und dem Computer-Benutzer herstellen. Wie dies im einzelnen geht, erfahren Sie etwas später, wenn Sie dieses Programm vollständig eingegeben haben und es laufen lassen.

Die Zeichen N$ nach dem Schlüsselwort INPUT stellen einen sogenannten *Variablen-Namen* dar. Lassen Sie sich durch diesen Begriff nicht einschüchtern. Eine Variable ist einfach ein Platz, den der Computer in seinem Speicher für einen bestimmten Typ von Information reserviert. Sie können viele Variablen für ein Programm bestimmen, nur müssen Sie jeder einen Namen geben. Der Computer kann dann über diesen Namen auf die Variablen Bezug nehmen. Die Information, die eine Variable enthält, kann während eines Programmablaufs häufig wechseln – daher der Name Variable –, der *Name* aber ändert sich niemals.

Wenn der Computer die Anweisung INPUT N$ ausführt, wartet er zunächst einmal auf eine Information, die über die Tasten eingegeben wird. Wenn die Information eingegeben wurde, speichert der Computer sie in einer Variablen, der er den Namen N$ gibt. Von nun an weiß der Computer, daß immer, wenn Sie den Namen N$ benutzen, Sie sich für die Information interessieren, die in dieser Variablen gespeichert ist.

Sie können zwei Variablentypen in einem Programm definieren – Variablen, die numerische Werte (Zahlenwerte) speichern, und Variablen zur Speicherung von Texten (d. h. nichtnumerische Daten). Im Computer-Sprachgebrauch bezeichnen wir eine nichtnumerische Information als *String* (Zeichenkette). Bei der Definition einer String-Variablen müssen Sie im *Variablen-Namen* angeben, daß der Computer Speicherplatz für einen String bereitstellen soll. Dies geschieht durch das Anhängen eines Dollar-Zeichens ($) an das Ende des Variablen-Namens.

Also noch einmal: INPUT N$ teilt dem Computer mit, daß er einen String von der Tastatur annehmen und ihn unter dem Namen N$ in einer Stelle des Speichers unterbringen soll. Wenn Ihnen der Variablen-Begriff immer noch ein wenig unklar erscheint, so brauchen Sie sich im Moment darüber keine allzu großen Sorgen zu machen. Wir werden später noch einmal darauf zurückkommen. Denken Sie daran, daß wir im Augenblick

in erster Linie den Umgang mit der Tastatur erlernen wollen. Wir haben jetzt nur deshalb über Variablen-Namen gesprochen, damit Sie eine weitere Möglichkeit, die Ihr Computer zur Fehlerüberprüfung besitzt, besser verstehen können – den Editor.

Wenn Sie die Zeile 20 in Ihren Computer eingeben, tippen Sie sie zunächst einmal wie folgt ein – indem Sie das „$"-Zeichen absichtlich weglassen:

```
20 INPUT N
```

Grammatikalisch ist diese Anweisung völlig in Ordnung. Sie besagt: „Übernimm von der Tastatur eine *numerische* Information und lege diese im Speicher unter dem Namen N ab." Eine *numerische* Information deshalb, weil das Zeichen „$", das eine Stringvariable kennzeichnet, am Ende des Variablennamens fehlt. Geben Sie die Zeile vorläufig so ein. Das Schlüsselwort INPUT liegt übrigens über der I-Taste. Tippen Sie die Zeile ein und drücken Sie die NEW-LINE-Taste. Die Zeile springt in die obere Bildschirmhälfte, direkt unter die Zeile 10.

Während der Programmeingabe werden Sie oft solche Fehler machen. Sie haben eine völlig richtige Zeile eingegeben, nur eben nicht die, die Sie eingeben wollten. Da die Zeile grammatikalisch in Ordnung ist, hat der Computer keine Möglichkeit festzustellen, daß da irgend etwas falsch ist, so daß er also die Zeile ohne weiteres annimmt. Nachdem Sie die Zeile eingegeben haben und sich das Programm auf dem Bildschirm noch einmal ansehen, werden Sie wahrscheinlich den Fehler entdecken und ändern wollen. Der ZX81-Computer bietet eine äußerst einfache Möglichkeit, solche Fehler zu korrigieren. Dies geschieht mit dem sogenannten *Editor*.

Da es sich bei der Zeile, die Sie editieren (= bearbeiten, korrigieren) möchten, um die zuletzt eingegebene handelt, können Sie den Editor direkt benutzen. Suchen Sie das Wort EDIT auf Ihrer Tastatur, es steht auf der l-Taste. EDIT ist rot aufgedruckt, Sie wissen also, daß Sie umschalten müssen. Drücken Sie die SHIFT-Taste und gleichzeitig die EDIT-Taste. Abbildung 2.7 zeigt, wie Ihr Bildschirm dann aussieht.

Sie sehen, daß eine Kopie der Zeile wieder am unteren Bildschirmrand erscheint, in dem Bereich also, in dem Sie alle neuen Zeilen eintippen können. Der [K]-Cursor steht innerhalb der Zeile direkt nach der Zeilennummer.

Sie können nun die Tasten mit den Pfeilen nach links und rechts benutzen (SHIFT-5 bzw. SHIFT-8), um den Cursor in der Zeile vorwärts und rückwärts zu bewegen. Zum Löschen setzen Sie den Cursor direkt rechts

neben das Zeichen oder Wort, das Sie löschen möchten, und drücken dann SHIFT-RUBOUT, genauso wie bei einer neuen Zeile.

In diesem Fall möchten Sie ein Zeichen in die Zeile einfügen, nämlich das $-Zeichen am Ende des Variablennamens. Drücken Sie zweimal SHIFT-8 (d. h. die SHIFT-Taste und die 8-Taste gleichzeitig), um den Cursor an die richtige Stelle zu bringen, direkt hinter das N. Das $-Zeichen ist ebenfalls ein Umschaltzeichen und liegt auf der U-Taste. Drükken Sie SHIFT-U, um das Zeichen einzugeben.

Dann drücken Sie NEW LINE, um die geänderte Version der Zeile einzugeben. Sie sehen, daß die korrigierte Zeile oben auf dem Bildschirm erscheint. Außerdem erscheint der Cursor mit dem [K] wieder in der unteren linken Ecke: der Computer wartet auf Ihr nächstes Kommando.

Bis jetzt haben Sie zwar nur zwei Programmzeilen eingegeben, dabei jedoch schon eine Menge über Ihre Tastatur und Ihren Computer gelernt. Lassen Sie uns kurz zusammenfassen, was Sie gelernt haben:

- Wenn Sie anfangen, eine Zeile in den Computer einzugeben, erscheint sie am unteren Rand des Bildschirms. Wenn Sie jetzt einen Tippfehler

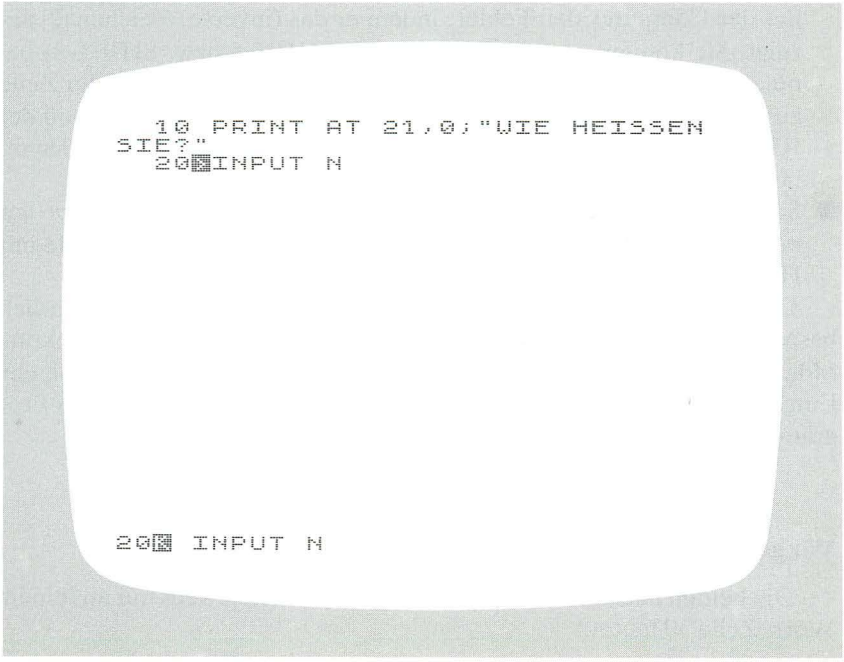


Abbildung 2.7: Editieren der zuletzt eingegebenen Zeile

machen, können Sie den vorigen Tastendruck auslöschen, indem Sie gleichzeitig die SHIFT- und die RUBOUT-Taste drücken.

- Das erste Wort jeder Zeile ist ein Schlüsselwort, das Sie mit einem einzigen Tastendruck eingeben können. Danach verändert sich das Cursor-Zeichen in ein invers dargestelltes [L], das die Betriebsart zur Eingabe von Buchstaben (*letter*) anzeigt. Wenn Sie eine Funktion eingeben wollen, müssen Sie zuerst in die *Funktions-Betriebsart* umschalten, indem Sie SHIFT-FUNCTION drücken. Auch sämtliche Funktionen werden mit einem einzigen Tastendruck eingegeben.
- Alle Symbole und Wörter, die auf der Tastatur rot gedruckt sind, sind Umschalt-Funktionen. Um sie einzugeben, müssen Sie also zunächst die SHIFT-Taste und dann gleichzeitig die entsprechende Taste drükken.
- Wenn Sie eine Zeile vollständig eingegeben haben, drücken Sie die Taste NEW LINE, damit der Computer weiß, daß er diese Zeile jetzt in den Programm-Speicher aufnehmen soll. Sofern die Zeile keine Syntax-Fehler enthält, wandert sie nun in den oberen Bildschirm-Bereich, in dem gegebenenfalls schon andere Programm-Zeilen stehen.
- Wenn Sie versuchen, eine Zeile mit Syntax-Fehlern einzugeben, meldet der Computer den Fehler, indem er das (inverse) Zeichen [S] anzeigt. Sie können nun die Pfeil-Tasten (SHIFT-5 bzw. SHIFT-8) benutzen, um den Cursor nach links oder nach rechts innerhalb der Zeile zu bewegen. Nun können Sie Ihre Fehler wieder auslöschen – mit der RUBOUT-Taste (SHIFT-0) – oder auch zusätzliche Wörter und Symbole in die Zeile einfügen.
- Sollten Sie einen Fehler erst entdecken, nachdem Sie die Eingabe der betreffenden Zeile abgeschlossen haben, so können Sie die Zeile mit Hilfe der EDIT-Funktion (SHIFT-1) korrigieren.

Die restlichen Programm-Zeilen werden wir nicht mehr so ausführlich besprechen, doch sollten Sie ausgiebigen Gebrauch von den Korrektur-Möglichkeiten machen, die Ihnen der Computer bietet. Wenn Sie mit der Eingabe des Programmes fertig sind, sollte Ihr Bildschirm mit dem Programm-Ausdruck der Abbildung 2.1 übereinstimmen.

Wir geben den Rest des Programms ein

Die beiden nächsten Zeilen, 30 und 35, bestehen jeweils nur aus einem Wort. Zeile 30 lautet:

```
30 CLS
```

Das Schlüsselwort CLS, das über der V-Taste liegt, bewirkt das Löschen des Bildschirms. Die Anweisung in Zeile 35 schaltet den Computer in die *schnelle Betriebsart* (die wir in den Kapiteln 3 und 4 besprechen werden):

```
35 FAST
```

Das Wort FAST ist eine Umschalt-Funktion, die über der F-Taste liegt.

Die nächsten drei Zeilen, 40 bis 60, gehören zusammen und lassen den Computer eine bestimmte Aufgabe wiederholt ausführen. Man verwendet dafür den anschaulichen Begriff *Schleife*. Die Schlüsselwörter, aus denen eine derartige Schleife gebildet wird, heißen FOR (Zeile 40) und NEXT (Zeile 60). In unserem Beispiel besteht die zu wiederholende Aufgabe zwar nur aus einer einzigen Zeile (Zeile 50), aber es ist durchaus möglich, sehr viel mehr Zeilen in eine Schleife einzubauen. Das Ende einer Schleife wird durch die Zeile mit dem Schlüsselwort NEXT angezeigt. Die FOR-Zeile gibt nicht nur an, wo die Schleife beginnt, sondern enthält auch noch die Angabe, wie oft der betreffende Zeilenbereich durchlaufen werden soll. Auf die syntaktischen Einzelheiten der FOR/NEXT-Schleife wird in Kapitel 3 eingegangen.

Wenn Sie die Ergebnisse dieses Programms sehen, werden Sie sicherlich leicht erraten können, worin die wiederholte Aufgabe besteht. Geben Sie Zeile 40 ein:

```
40 FOR I = 1 TO 181
```

Das Schlüsselwort FOR liegt über der F-Taste. Das Gleichheitszeichen (=) und das Wort TO sind Umschalt-Funktionen, die auf der L- bzw. 4-Taste liegen.

In Zeile 50 steht wieder ein neues Schlüsselwort, wie Sie in der Abbildung 2.1 sehen können. Diese Zeile enthält eine PRINT-Anweisung. Nach dem Schlüsselwort – PRINT – stehen zwei *grafische Zeichen*, die in Anführungszeichen eingeschlossen sind. Der ZX81-Computer besitzt 20 grafische Zeichen, die in beliebiger Kombination an beliebigen Bildschirm-Positionen angezeigt werden können. Auf der Tastatur finden Sie diese Zeichen auf den ersten acht Ziffern-Tasten sowie auf zwölf Buchstaben-Tasten (vgl. Abbildung 2.8).

Um diese Zeichen eingeben zu können, müssen Sie zuerst in die *Grafik-Betriebsart* umschalten. Tippen Sie zunächst die Zeilennummer, das Schlüsselwort und das erste Anführungszeichen ein:

```
50 PRINT"
```

Der [L]-Cursor zeigt an, daß Sie gerade in der Betriebsart zur Eingabe von Buchstaben sind. Zur Umschaltung in die Grafik-Betriebsart drükken Sie die SHIFT-Taste und gleichzeitig die 9-Taste. (Wie Sie sehen, steht auf der 9-Taste in rot das Wort GRAPHICS.) Dabei wird der Cursor zu einem invers dargestellten [G], das Ihnen anzeigt, daß Sie sich in der Grafik-Betriebsart befinden.

Alle Grafik-Symbole auf der Tastatur sind Umschalt-Zeichen. Die beiden, die Sie für diese Zeile benötigen, liegen auf der S-Taste und der H-Taste. Drücken Sie zunächst SHIFT-S. Das Zeichen erscheint in der Zeile, und der Computer bleibt weiterhin in der Grafik-Betriebsart. Nun drücken Sie SHIFT-H für das zweite Zeichen. Abbildung 2.9 zeigt, wie Ihr Bildschirm im Augenblick aussehen sollte.

Für den Rest der Zeile müssen Sie die Grafik-Betriebsart verlassen und wieder zurück in die Buchstaben-Betriebsart. Dazu drücken Sie einfach noch einmal SHIFT-9. Denken Sie daran, daß das Wort GRAPHICS auf der 9-Taste zum Ein- und Ausschalten der Grafik-Betriebsart dient.

Wenn Sie wieder den [L]-Cursor sehen, geben Sie den Rest der Zeile ein:

```
";N$(1);
```

Halten Sie die Zeichensetzung genau ein. Die 1 in der Klammer nach N$ bezieht sich auf das *erste* Zeichen der Zeichenkette, die in der Variablen N$ gespeichert ist. Die beiden Klammern sind Umschaltzeichen, die auf der I- bzw. der O-Taste liegen.

Mit Zeile 60 wird die Schleife beendet:

```
60 NEXT I
```

Das Schlüsselwort NEXT liegt über der N-Taste. Zeile 65 schaltet den Computer wieder zurück in die *langsame Betriebsart*, die wir in den Kapiteln 3 und 4 besprechen werden:

```
65 SLOW
```

Das Wort SLOW ist eine Umschaltfunktion, die auf der D-Taste liegt.

In Zeile 70 wird wieder von den besonderen Grafikmöglichkeiten des ZX81 Gebrauch gemacht. In der Grafik-Betriebsart können Sie zusätzlich zu den 20 speziellen Grafik-Zeichen invers dargestellte Zeichen eingeben. Um zu sehen, wie dies funktioniert, tippen Sie zunächst den ersten Teil der Zeile ein:

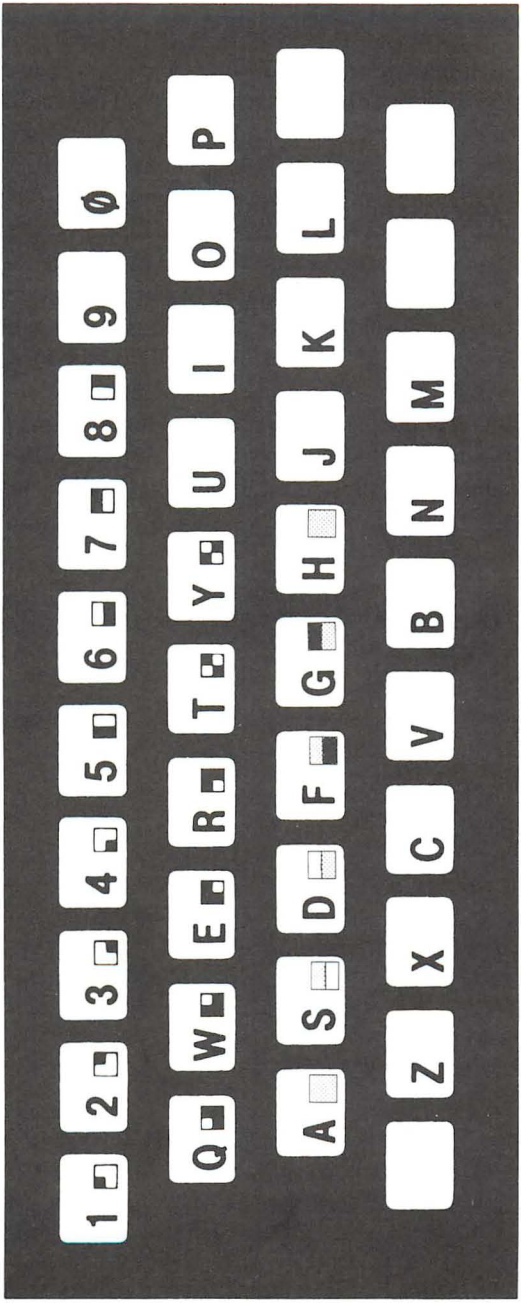


Abbildung 2.8: Die Grafik-Zeichen

```
70 PRINT AT 9,7;" **
```

(Nach dem ersten Anführungszeichen stehen ein Leerzeichen und dann zwei Sternchen. Das Sternchen ist ein Umschaltzeichen und befindet sich auf der B-Taste.) Schalten Sie nun in die Grafik-Betriebsart (SHIFT-9) und tippen Sie das Wort HALLO ein, ohne die SHIFT-Taste zu drücken. Sie sehen, daß das Wort weiß auf schwarz, also invers, erscheint. Drücken Sie jetzt die Leertaste; ein inverses Leerzeichen ist einfach ein schwarzes Quadrat. Das Leerzeichen beendet die Mitteilung. Gehen Sie also aus der Grafik-Betriebsart heraus und tippen die beiden letzten Zeichen der Zeile ein – das abschließende Anführungszeichen und das Semikolon. Drükken Sie NEW LINE, um die Zeileneingabe abzuschließen.

Die drei nächsten Programmzeilen – 80 bis 100 – bilden eine weitere FOR/NEXT-Schleife. Inzwischen dürfte es Ihnen keine Schwierigkeiten mehr bereiten, diese Zeilen einzugeben. Darin kommen drei neue Programmierfunktionen vor, so daß Sie also dreimal in die Funktions-Be-

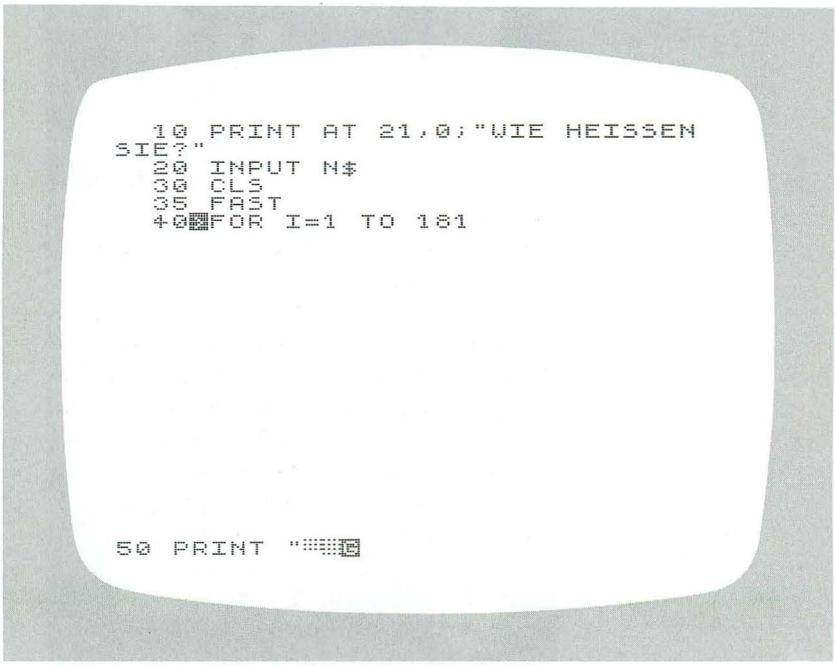


Abbildung 2.9: Eingabe der Grafik-Zeichen

triebsart umschalten müssen. In Zeile 80 wird die LEN-Funktion benutzt, die unterhalb der K-Taste liegt:

```
80 FOR I = 1 TO LEN N$
```

Die LEN-Funktion gibt die *Länge* eines Strings in Zeichen an. In Zeile 90 steht die CHR$-Funktion (unterhalb der U-Taste) und die CODE-Funktion (unterhalb der I-Taste):

```
90 PRINT CHR$ (CODE N$(I)+128);
```

Dies ist wohl die komplizierteste Anweisung im ganzen Programm. Kurz gesagt, formt sie die Zeichen des Strings N$ in die entsprechenden inversen Zeichen um. Wir werden die CHR$- und CODE-Funktion in Kapitel 5 näher betrachten. Zeile 90 enthält noch ein neues Element: das Plus-Zeichen (+). Es ist ein Umschalt-Zeichen und liegt auf der K-Taste.

Zeile 100 schließt die FOR/NEXT-Schleife ab:

```
100 NEXT I
```

In Zeile 110 steht eine weitere PRINT-Anweisung:

```
110 PRINT " **"
```

Beachten Sie, daß zwischen den Anführungszeichen drei Zeichen stehen: zuerst ein Leerzeichen, dann zwei Sternchen.

Sie haben nun das ganze Programm in den Computer eingegeben und können gleich sehen, was passiert, wenn der Computer die einzelnen Anweisungen ausführt. Bevor Sie jedoch das Programm laufen lassen, beachten Sie zwei Dinge. Vergleichen Sie zuerst noch einmal Ihren Bildschirm sorgfältig mit dem Programmausdruck in Abbildung 2.1. Vergewissern Sie sich, daß beide übereinstimmen. Sollten Sie Fehler finden, können Sie sie leicht berichtigen. In dem Abschnitt „Korrigieren von Programmzeilen" in diesem Kapitel können Sie nachlesen, wie nachträgliche Korrekturen gemacht werden. (Sie wissen ja schon, wie man den Editor benutzt, und brauchen nur noch zu lernen, wie der Zugriff auf eine beliebige Zeile erfolgt.)

Außerdem sollten Sie sich noch einmal die verschiedenen Betriebsarten der Tastatur klarmachen, bevor Sie das Programm laufen lassen. Sehen Sie sich dazu Abbildung 2.10 an, in der die verschiedenen Cursor-Zeichen zusammengefaßt sind.

Es geht los!

Das Schlüsselwort RUN – über der R-Taste – bewirkt, daß der Computer mit der Ausführung der Anweisungen beginnt, die in seinem Programm-Speicher stehen. Wenn Sie nun das Programm laufen lassen, werden Sie wahrscheinlich zumindest einen Teil der Anweisungen wiedererkennen, die Sie mit so viel Aufwand in den Computer eingetippt haben. Da Sie BASIC noch nicht kennengelernt haben, werden Ihre Eindrücke im Augenblick eher bruchstückhaft als systematisch sein. Aber trotzdem sind Sie vielleicht überrascht, daß Sie die Bedeutung der meisten Programmzeilen doch ziemlich genau erraten können, nachdem Sie einmal gesehen haben, was sie bewirken.

Ihre Unterhaltung mit dem Computer

Geben Sie das Schlüsselwort RUN direkt ein, ohne eine Zeilennummer davorzusetzen. Drücken Sie die Taste NEW LINE und sehen Sie zu, was passiert.

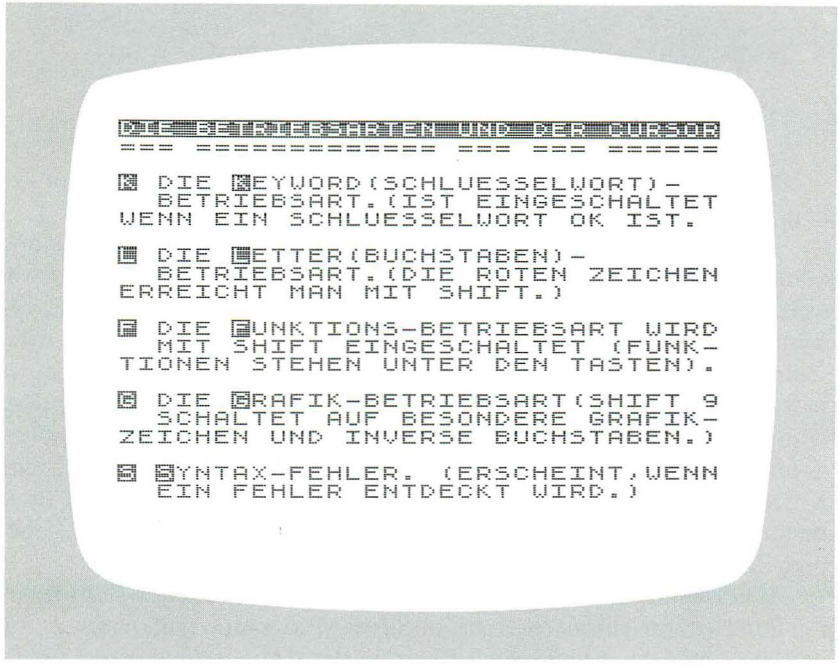


Abbildung 2.10: Die Betriebsarten und der Kursor

Zum ersten Mal sind die eingegebenen Programmzeilen vom Bildschirm verschwunden. Selbstverständlich stehen die Zeilen immer noch im Speicher des Computers, aber jetzt wird der Bildschirm für die *Ergebnisse* des Programms benötigt.

Das Ganze spielt sich in zwei verschiedenen Schritten ab, wie Sie gleich sehen werden. Im ersten Schritt erscheint die Frage:

```
WIE HEISSEN SIE?
```

auf dem Bildschirm. Der [L]-Cursor steht zwischen Anführungszeichen unter der Frage. Damit wissen Sie, daß der Computer erwartet, daß Sie eine Antwort über die Tastatur eintippen. Geben Sie also Ihren Namen ein. (Abbildung 2.11 zeigt, wie der Bildschirm aussieht, wenn Sie z. B. HILDA heißen). Wenn Sie während der Eingabe einen Fehler machen, können Sie die RUBOUT-Taste (SHIFT-0) benutzen, um den Fehler zu löschen, genauso wie bei der Eingabe eines Programms.

Abbildung 2.11: Frage und Antwort auf dem Bildschirm

Wenn Sie Ihren Namen eingetippt haben, drücken Sie die Taste NEW LINE. Der Bildschirm wird für einen Augenblick gelöscht, und dann . . . nun, Sie werden sehen. (Abbildung 2.12 zeigt die Begrüßung für Hilda.) Sie sehen, wie geschickt der Computer den Anfangsbuchstaben Ihres Namens für die Gestaltung des Hintergrundes benutzt hat.

Bevor Sie weitermachen, sollten Sie sich noch einmal über die „Täuschung" klar werden, die durch Ihr Programm hervorgerufen wird. Es sieht aus, als ob *der Computer* Ihnen eine ganz gewöhnliche Frage gestellt hätte, Ihre Antwort verstanden und daraufhin eine zwar etwas ungewöhnliche, aber korrekte und anscheinend intelligente Antwort gegeben hätte. Natürlich wissen Sie jetzt, daß dieser „Dialog" nicht ganz „echt" ist, weil nämlich jedes Detail vollständig durch das von Ihnen eingegebene Programm bestimmt ist. Der Computer stört sich nicht im geringsten daran, wie Sie heißen und hat keinerlei Bedürfnis, freundlich zu sein. Aber Sie können den Eindruck von Freundlichkeit erwecken – oder irgend einer anderen Verhaltensweise –, indem Sie entsprechende Programme schreiben. Ja sogar die Tatsache, daß da irgendwo im Hinter-

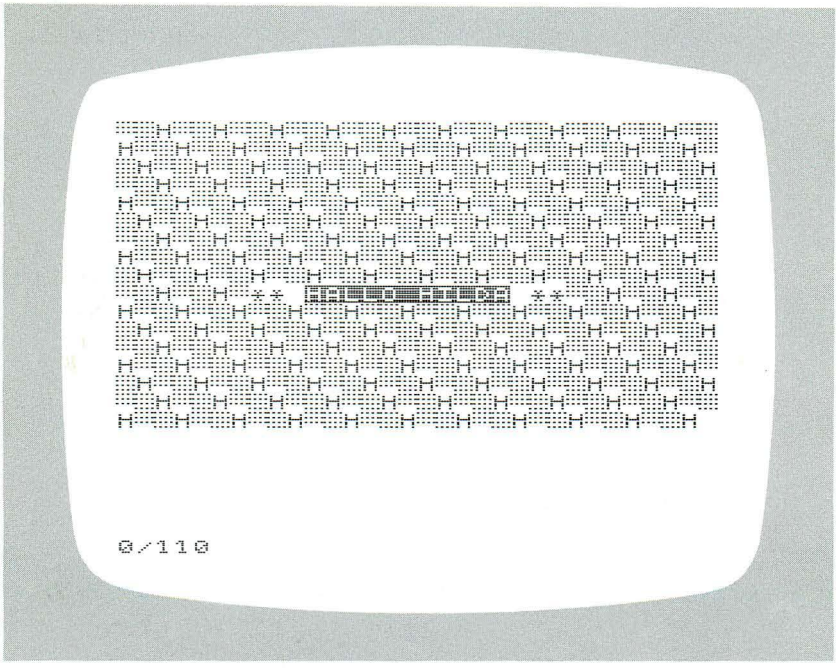


Abbildung 2.12: Der Computer begrüßt Sie

grund ein Programm existiert, ist für eine Person, die die auf dem Bildschirm angezeigte Frage beantwortet, gar nicht ersichtlich. (Lassen Sie doch einmal versuchsweise das Programm für eine andere Person laufen und beobachten Sie, wie sie reagiert.) Es ist eine wesentliche Eigenschaft des Programmierens von Computern überhaupt, durch korrektes Antworten auf Eingaben eines Benutzers den Anschein von Intelligenz zu erwecken. Der Computer kann nichts tun, ohne von Ihnen detaillierte Anweisungen zu erhalten, aber ironischerweise wird die Intelligenz, die in Ihren Programmen steckt, dem Computer zugeschrieben. Es denkt überhaupt niemand daran, daß ein Programm dahintersteckt.

Wenden wir uns wieder dem Bildschirm zu. In der unteren linken Ekke ist noch eine kurze Nachricht aufgetaucht, die der Computer nach der Ausführung des Programms ausgegeben hat. Die Zahlen stellen eine Art Bericht über den Verlauf des Programms dar:

```
0/110
```

In den nächsten Kapiteln werden wir uns den Inhalt dieser Berichte näher ansehen. In unserem Fall bedeutet die Null, daß keine Fehler im Programm aufgetaucht sind. Die Zahl 110 gibt die Nummer der zuletzt ausgeführten Zeile an.

Korrigieren von Programmzeilen

Nachdem Sie die liebenswürdige Begrüßung des Computers ausgiebig bewundert haben, drücken Sie jetzt die Taste NEW LINE. Die Begrüßung verschwindet, und Ihre Programmzeilen kommen wieder auf den Bildschirm zurück. Sie können das Programm so oft laufen lassen, wie Sie wollen, indem Sie das Schlüsselwort RUN eingeben, dem Computer wird die Sache niemals langweilig.

Sie können das Programm auch verändern, wenn Sie wollen. Das können Sie auf verschiedene Weise tun, es hängt von der Art der gewünschten Änderung ab:

- Sie können eine neue Zeile hinzufügen, indem Sie sie einfach in den Computer tippen. Dabei dürfen Sie keine Zeilennummer verwenden, die im Programm schon vorgekommen ist.
- Sie können eine Zeile löschen, indem Sie die Zeilennummer und NEW LINE eingeben. (Andererseits müssen Sie natürlich aufpassen, daß Sie auf diese Weise nicht unbeabsichtigt eine Zeile löschen.)
- Sie können eine ganze Zeile vollständig neu schreiben, indem Sie eine

neue Anweisung mit der alten Zeilennummer eingeben. Die alte Zeile verschwindet und wird durch die neue ersetzt.

- Wenn Sie aber nur kleine Änderungen in einer Zeile durchführen wollen, können Sie die Zeile *editieren*. Das wollen wir jetzt versuchen.

Sie werden wahrscheinlich schon den kleinen invers dargestellten Pfeil bemerkt haben, der auf die zuletzt eingegebene Programmzeile zeigt. Wenn Sie die Zeilen des Programms der Reihe nach eingegeben haben, zeigt dieser Pfeil jetzt auf Zeile 110. (Sie sehen den Pfeil in Abbildung 2.1.) Dieser Pfeil weist auf die jeweils *aktuelle Zeile*, d. h. die Zeile, die editiert wird, wenn Sie die EDIT-Taste drücken. Sie können den Pfeil in Ihrem Programm auf- und abbewegen, indem sie die entsprechenden Pfeiltasten drücken (SHIFT-6 und SHIFT-7).

Nehmen wir an, daß Sie das Begrüßungsprogramm ändern wollen. Sie wollen das Wort HALLO durch HELLO ersetzen. Dazu müssen Sie die Zeile 70 ändern.

Wenn der Pfeil auf die Zeile 110 zeigt, müssen Sie ihn um vier Zeilen nach oben bewegen, um die Zeile 70 zu korrigieren. Hier die Schritte im einzelnen:

1. Drücken Sie die SHIFT-Taste und gleichzeitig viermal die 7-Taste. Bei jedem Tastendruck bewegt sich der Pfeil um eine Zeile nach oben.
2. Nun zeigt der Pfeil auf die Zeile 70. Drücken Sie die EDIT-Taste (SHIFT-1), und die Zeile erscheint zum Editieren am unteren Bildschirmrand.
3. Mit Hilfe der Pfeiltaste (→, SHIFT-8) bewegen Sie den L-Cursor rechts neben den Buchstaben A in HALLO. Drücken Sie die RUBOUT-Taste einmal, um den Buchstaben A zu löschen.
4. Schalten Sie um in die Grafik-Betriebsart (SHIFT-9) und geben Sie den neuen Buchstaben ein: E.
5. Dann gehen Sie wieder aus der Grafik-Betriebsart heraus (SHIFT-9) und drücken die NEW LINE-Taste. Die neue Version der Zeile 70 steht nun in der Programmliste.

Lassen Sie das Programm laufen, um sich zu vergewissern, daß es noch richtig funktioniert. Es sollte alles wie vorher laufen, abgesehen davon, daß das Wort HALLO durch HELLO ersetzt ist.

Hier noch eine Anmerkung, wie Sie den Hinweispfeil schneller durch das Programm bewegen können, was bei umfangreichen Programmen be-

sonders nützlich ist: Sie verwenden dazu das Schlüsselwort LIST. Angenommen, Sie möchten den Pfeil zur Zeile 40 schicken. Dazu geben Sie die Anweisung

```
LIST 40
```

Ihr Programm wird ab Zeile 40 auf dem Bildschirm aufgelistet. Tippen Sie nun die EDIT-Taste, und die Zeile 40 erscheint zum Editieren am unteren Bildschirmrand.

Auf zur nächsten Vorstellung! . . .

Im Kapitel 1 haben wir darüber gesprochen, daß es sinnvoll ist, einen Kassetten-Recorder für die langfristige Speicherung von Programmen zur Verfügung zu haben. Sie wissen, daß die Programme, die in dem flüchtigen Speicher des Computers stehen, gelöscht werden, wenn Sie den Strom ausschalten.

In diesem Abschnitt lernen Sie, wie man einen Kassetten-Recorder in Verbindung mit dem Computer benutzt. Wenn Sie einen Recorder haben, können Sie gleich versuchen, das Begrüßungsprogramm zu speichern.

Speichern Sie Ihr Programm auf Band

Nehmen Sie das kurze Kabel, das an beiden Enden ein Steckerpaar hat. Diese Stecker dienen zur Verbindung des Kopfhörer- und Mikrofon-Anschlusses Ihres Recorders mit dem Computer. (Die Anschlüsse am Computer sind mit EAR und MIC bezeichnet. Eventuell benötigen Sie ein Adapter-Kabel.) Die Stecker sind verschiedenfarbig, so daß Sie leicht die richtigen Verbindungen herstellen können.

Um ein Programm vom Computer auf eine Kassette zu überspielen, verbinden Sie den Mikrofon-Eingang des Recorders mit dem mit MIC bezeichneten Anschluß an der linken Seite des Computers. Überprüfen Sie die Verbindung noch einmal.

Jetzt müssen Sie den Lautstärke- und den Klangregler des Recorders richtig einstellen. Wählen Sie zunächst etwa eine mittlere Position für den Lautstärkeregler. Den Klangregler stellen Sie auf die höchste Stellung für die Höhen- und/oder die niedrigste Baß-Einstellung. Wenn Ihr erster Versuch, ein Programm zu speichern, nicht gelingt, drehen Sie den Lautstärkeregler weiter auf.

Legen Sie eine leere Kassette in den Recorder und spulen Sie sie gegebenenfalls an den Anfang. Wenn Ihr Recorder ein Zählwerk hat, setzen

Sie es auf Null, um den Anfang und das Ende Ihrer Aufzeichnung leicht wiederfinden zu können.

Das Schlüsselwort, mit dem Sie ein Programm auf eine Kassette aufnehmen können, heißt SAVE und liegt über der S-Taste. Das SAVE-Kommando benötigt einen Programmnamen, der zwischen Anführungszeichen gesetzt wird. Sie sollten den Namen so wählen, daß Sie später noch wissen, was das Programm tut. Wir wollen dieses Programm BEGRUESSUNG nennen. Geben Sie das Kommando ein, aber schließen Sie die Eingabe noch nicht mit NEW LINE ab:

```
SAVE "BEGRUESSUNG"
```

Drücken Sie nun die Aufnahmetaste an Ihrem Recorder. Lassen Sie ihn ein paar Sekunden laufen und drücken dann die NEW LINE-Taste auf dem Computer. Ihr Bildschirm wird für einige Augenblicke gelöscht. Wenn auf Ihrem Bildschirm schwarze Querstreifen (ca. 2–3 cm breit) erscheinen, die sich schnell auf- und abbewegen, wird Ihr Programm gerade auf den Kassetten-Recorder übertragen. Die Übertragungszeit hängt von der Länge des Programms ab. Das Begrüßungsprogramm braucht nur wenige Sekunden zur Aufzeichnung, sehr lange Programme brauchen einige Minuten.

Wenn die Aufnahme beendet ist, ist der Bildschirm wieder leer und zeigt in der unteren linken Ecke nur die folgende Information an:

```
0/0
```

Sie können sich sogar die Aufzeichnungen anhören, um sicher zu sein, daß die Übertragung funktioniert hat. Die Aufzeichnung eines Programms erzeugt ein hohes, unangenehm kreischendes Geräusch, das an ein Alarmsignal erinnert. Sie müssen sich das auch nicht sehr lange anhören. Wenn Sie überhaupt etwas hören, ist die Aufnahme wahrscheinlich gelungen. (Wenn Sie nichts hören, versuchen Sie es noch einmal.)

Von wichtigen Programmen sollten Sie immer zwei Kopien auf zwei verschiedenen Kassetten machen. Legen Sie eine davon an einen sicheren Platz. Dies ist Ihre „Reserve"-Kopie für den Fall, daß die andere zerstört wird. Schreiben Sie auf jede Kassette den Namen des Programms und, wenn möglich, die Anfangs- und Endnummern des Zählwerks.

Um ein Programm von der Kassette in den Computer zu laden, müssen Sie den Ausgang Ihres Recorders und den mit EAR bezeichneten Anschluß des Computers miteinander verbinden.

Mit Hilfe des LOAD-Kommandos werden die Programme von der

Kassette in den Computer geladen. Um das Begrüßungsprogramm zu laden, geben Sie die folgende Anweisung:

```
LOAD "BEGRUESSUNG"
```

Wenn Sie den Namen eines Programmes nicht wissen, können Sie dieses Kommando auch ohne den Namen verwenden. Die beiden Anführungszeichen müssen Sie aber in jedem Fall eingeben:

```
LOAD ""
```

Vergewissern Sie sich, daß die Kassette auf die richtige Position zurückgespult ist, und drücken Sie an Ihrem Recorder die Wiedergabetaste. Zunächst sehen Sie ein diagonales Muster von dünnen (ca. 5 mm breiten), ziemlich beweglichen schwarzen Linien auf dem Bildschirm (siehe Abb. 2.13). Während des Überspielens sehen Sie dann breite schwarze und weiße Streifen (siehe Abb. 2.14). Wenn das Programm vollständig

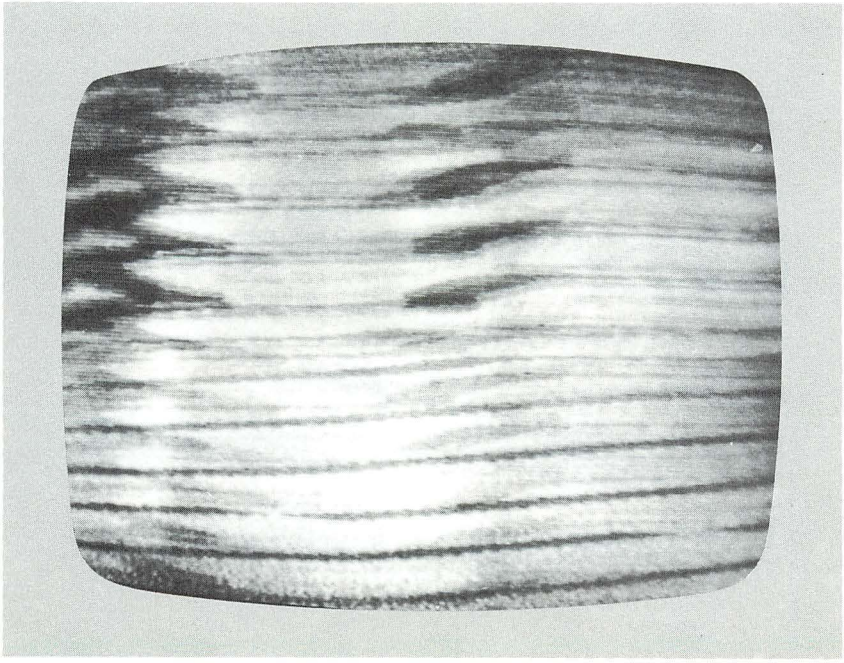


Abbildung 2.13: Der Vorspann bei Lade-Beginn

überspielt ist, wird der Bildschirm wieder gelöscht, und Sie sehen in der Bildschirmecke wieder die Mitteilung

```
0/0
```

Damit wird angezeigt, daß der Ladevorgang erfolgreich beendet wurde. Drücken Sie die Taste NEW LINE, und das Programm wird auf dem Bildschirm aufgelistet.

Wenn das Streifenmuster nicht verschwindet, ist irgendetwas falsch gelaufen. Schalten Sie den Kassettenrecorder aus und drücken Sie die SPACE-Taste auf dem Computer. (Über dieser Taste steht das Wort BREAK. Sie können BREAK immer benutzen, um einen Programmlauf oder das Speichern bzw. Laden eines Programmes abzubrechen.)

Die häufigsten Fehlerursachen beim Speichern oder Laden eines Programmes sind:

1. Fehlerhafte Verbindungen. (Überprüfen Sie, ob alle Stecker fest sitzen.)

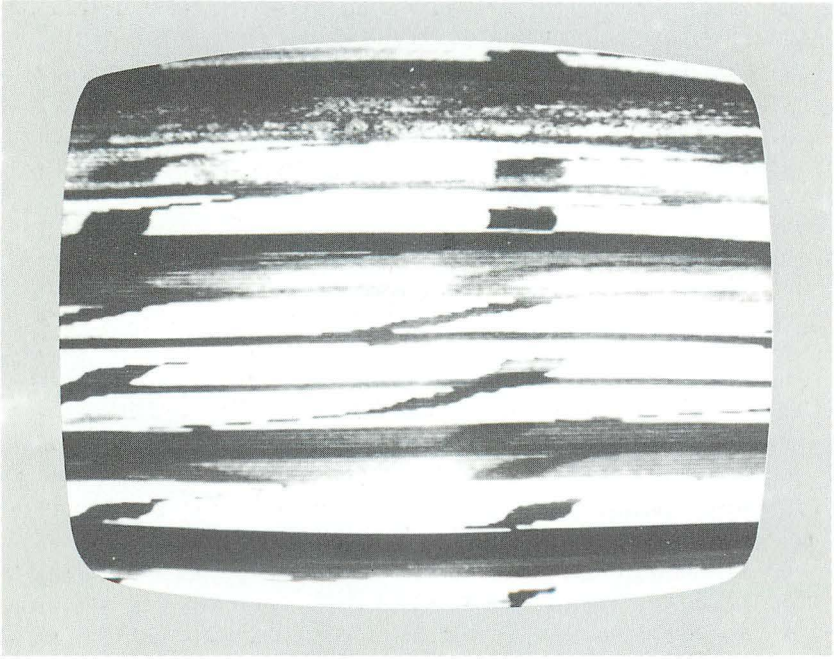


Abbildung 2.14: Momentaufnahme vom Ladevorgang

2. Falsche Einstellung der Lautstärke- und Klangregler. (Sie müssen durch Versuche die beste Einstellung für Ihren Recorder herausfinden.)

Der Kauf von Software

Sie müssen schon etwas Geduld und Ausdauer aufbringen, bis Sie das Laden und Speichern mit Hilfe des Kassetten-Recorders richtig beherrschen. Zur Belohnung können Sie dann aber nicht nur Ihre eigenen Programme speichern, sondern auch von anderen erstellte Programme benutzen. Sie können dann auch einige der komplizierten Programme erwerben, die Ihnen ganz neue Möglichkeiten Ihres Computers erschließen.

Die Zahl der für den ZX81 auf Kassette verfügbaren Programme wächst beständig. (Die Computer-Programme bezeichnet man übrigens als *Software*, im Gegensatz zur *Hardware*, womit der Computer selbst und die zugehörigen Geräte gemeint sind.) Insbesondere in solchen Bereichen wie Geschäft, Haushalt, Bildung und Spiel können Sie unter einem großen Programmangebot auswählen. Für viele dieser Programme ist das zusätzliche 16K Speichermodul erforderlich, jedoch können einige auch ohne diese Erweiterung benutzt werden.

Damit Sie eine Vorstellung davon haben, welche Arten von Programmen Sie kaufen können, wollen wir uns im nächsten Abschnitt dieses Kapitels ein Beispiel ansehen. Wir wollen ein Programm mit dem Namen VU-CALC besprechen, das zu den meistgekauften Programmen für den ZX81 gehört.

VU-CALC, ein Kalkulationsprogramm

Vielleicht haben Sie schon von sogenannten Kalkulationsprogrammen gehört. Diese Programme erlauben die Verwaltung großer Zahlenmengen und können schnelle und effiziente Berechnungen durchführen. Solche Kalkulationsprogramme gibt es inzwischen für die meisten Kleincomputer. Zwei Beispiele für solche Programme sind VisiCalc und SuperCalc, die sich bei den Benutzern von Kleincomputern außerordentlicher Beliebtheit erfreuen. Diese Programme vereinfachen und beschleunigen alle möglichen Berechnungen mit großen Zahlenmengen.

Das für den ZX81 (mit Zusatzspeicher) entwickelte VU-CALC ist der „kleine Bruder" dieser Programme. Obwohl es weder so leistungsfähig noch so vielseitig ist wie die entsprechenden Programme für größere Computer, ist es doch eine wertvolle Hilfe für jeden, der etwas aufwendigere Berechnungen durchführen muß.

Abbildung 2.15 zeigt ein einfaches Beispiel für ein VU-CALC-Kalkulations-Programm. Man kann sich VU-CALC als ein (zunächst leeres) Arbeitsblatt vorstellen, das aus 36 Spalten und 26 Zeilen besteht, in die Sie Wörter, Zahlen oder Formeln eingeben können. Jede Position auf dem Arbeitsblatt wird durch eine „Adresse" angesprochen, die aus zwei Koordinaten besteht: aus einem Buchstaben (der die Zeile angibt) und aus einer Zahl (welche die Spalte angibt). So steht z. B. in der Position F01 auf dem Arbeitsblatt in Abbildung 2.15 das Wort NETTO. In F02 und F03 stehen die NETTO-Beträge für die Jahre 1981 bzw. 1982.

Nachdem Sie einige Daten auf Ihrem Arbeitsblatt eingetragen haben, können Sie neue Daten erzeugen, indem Sie Formeln eintragen, die sich auf die bereits vorhandenen Daten beziehen. So wurde z. B. in den Positionen G02 und G03 die Einkommenssteuer für die NETTO-Beträge aus F02 und F03 berechnet. In der unteren linken Bildschirmecke können Sie die Formel sehen, mit der der Wert in Position G03 berechnet wurde:

```
F03 * 0.35
```

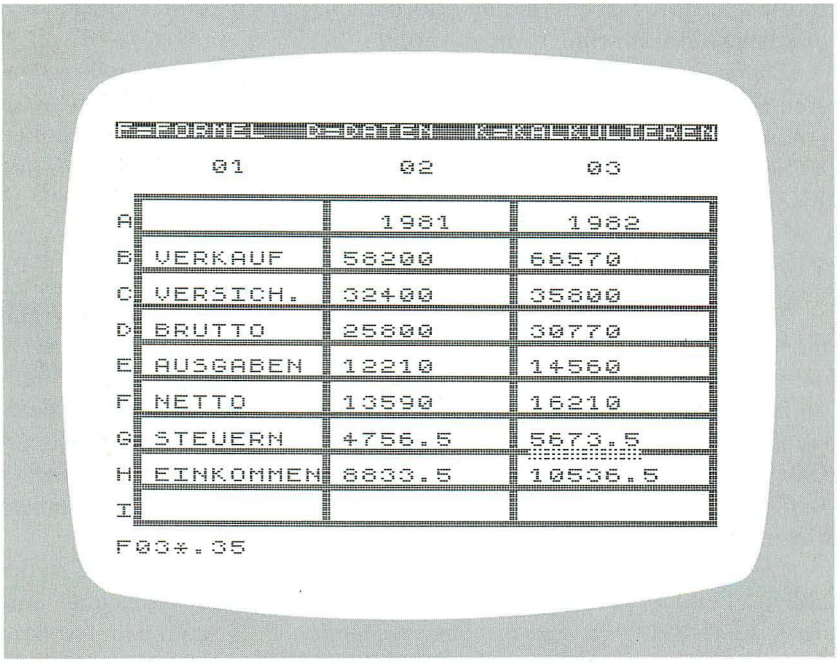


Abbildung 2.15: Ein VU-CALC-Beispiel

(Das Sternchen in der Formel dient als Multiplikationszeichen. Die Formel wird deshalb auf dem Bildschirm angezeigt, weil der *Cursor* auf dem Arbeitsblatt im Moment in der Position G03 steht.) Eine solche Formel brauchen Sie nur einmal zu schreiben, auch wenn Sie sie mehrmals verwenden wollen. Das wichtigste an der ganzen Sache ist, daß solche Formeln dynamisch sind: sobald sich irgendwelche Daten auf dem Arbeitsblatt ändern, werden alle davon abhängigen Berechnungen neu durchgeführt.

Auf dem Bildschirm sehen Sie zwar jeweils nur drei Spalten und neun Zeilen Ihres Arbeitsblattes, jedoch können Sie einen beliebigen Teil betrachten, indem Sie den Bildschirm-Ausschnitt mit Hilfe der vier Pfeiltasten verschieben. D. h. der Bildschirm stellt also gewissermaßen ein „Fenster" über dem gesamten Arbeitsblatt dar.

Sie können die Daten Ihres Arbeitsblattes (zusammen mit dem VU-CALC-Programm) auch auf Kassetten speichern, um immer wieder darauf zurückgreifen zu können.

Hinweise zum Kauf von Software für den ZX81

Leider sind die auf Kassetten erhältlichen Programme für Ihren Computer ziemlich teuer. Wenn Sie ein halbes Dutzend Programme kaufen, geben Sie für die Software etwa ebensoviel aus, wie Sie für den ganzen Computer bezahlt haben. Deshalb sollten Sie sorgfältig auswählen und sich vor dem Kauf gut informieren. Am besten ist es, wenn Sie ein Programm zunächst bei einem Händler ausprobieren können, bevor Sie es kaufen. (Viele Händler haben Demonstrations-Versionen der von ihnen angebotenen Programme.)

Zu jedem Programm gibt es eine „Dokumentation", eine Programm-Beschreibung. Nehmen Sie sich die Zeit, und sehen Sie sich die Dokumentation eines Programmes an, das Sie kaufen wollen. In dieser Beschreibung sollte alles, was Sie für den erfolgreichen Gebrauch des Programmes wissen müssen, klar dargestellt sein. Wenn die Dokumentation schlecht ist, kann der Zeitaufwand, den Sie zum Verstehen des Programmes benötigen, die Vorteile, die Sie durch das Programm haben können, leicht übersteigen.

Hier sind einige Fragen, die Sie sich stellen sollten, bevor Sie ein neues Programm erstehen:

- Sofern es sich um ein Geschäfts- oder Haushalts-Programm handelt: Kann man es in möglichst vielen verschiedenen Situationen verwen-

den? Kann es Ihre speziellen Berechnungsprobleme lösen? (Je besser Sie BASIC beherrschen, desto eher können Sie viele Programme für Ihren persönlichen Bedarf selber schreiben.)

- Sofern es sich um ein Spiel handelt: Ist es so interessant, daß Sie es überhaupt mehr als einmal spielen werden?
- Wenn Sie Fragen in bezug auf das Programm haben: Werden sie durch die Dokumentation beantwortet oder kann Ihnen Ihr Händler helfen? Wenn Sie das Programm im Laden ausprobieren, achten Sie darauf, ob es so läuft, wie es in der Dokumentation beschrieben ist.
- Wie verhält sich das Programm, wenn Sie einen Fehler begehen? Bietet es einfache Möglichkeiten, solche Fehler zu korrigieren? Machen Sie beim Testen absichtliche Fehler und notieren Sie die Ergebnisse.

Solche Fragen können Ihnen helfen zu entscheiden, ob Sie Ihre Zeit und Ihr Geld für ein bestimmtes Programm hergeben sollten.

Zusammenfassung

Die Benutzung der Tastatur läßt sich leicht erlernen, wenn Sie erst einmal die verschiedenen Umschalt-Möglichkeiten beherrschen. Da jede Zeile eines BASIC-Programmes mit einem Schlüsselwort beginnen muß, fängt der Computer immer mit dem [K]-Cursor an. Nach dem Schlüsselwort können Buchstaben, Zahlen oder Umschaltzeichen folgen (die [L]-Betriebsart) oder auch grafische und invers dargestellte Zeichen (die [G]-Betriebsart) sowie Funktionen (die [F]-Betriebsart).

Der Computer kann Ihnen auf verschiedene Weise bei der Fehlersuche und Korrektur behilflich sein. Sobald Sie eine Zeile eingeben wollen, die einen Syntax-Fehler enthält, weist der Computer diese Zeile zurück und markiert die fehlerhafte Stelle mit dem inversen S-Zeichen. Um den Fehler zu korrigieren, können Sie die Pfeiltasten (← und →) benutzen, um zu der entsprechenden Stelle in der Zeile zu gelangen, wo Sie dann Zeichen einfügen oder löschen können. Andererseits können Sie auch schon eingegebene – also syntaktisch korrekte – Zeilen später noch mit Hilfe der EDIT-Funktion ändern.

Um die Verbindung des Kassetten-Recorders mit dem Computer herzustellen, müssen Sie vielleicht zunächst etwas herumprobieren. Wenn Sie jedoch die richtigen Einstellungen am Recorder einmal herausgefunden haben, wird Ihr Computer Ihnen noch weitaus mehr nützen können. Zum einen werden Sie in der Lage sein, die Vor- und Nachteile von gekauften Programmen abzuwägen, zum anderen werden Sie allmählich eine Bibliothek von selbst geschriebenen Programmen aufbauen können.

Kapitel 3

Jetzt wird's spannend: Ein kurzer und anschaulicher BASIC-Kurs

Einführung

In diesem Kapitel werden Sie die Grundzüge von BASIC kennenlernen. Dabei können Sie sich gleichzeitig ein wenig mit den grafischen Möglichkeiten Ihres Computers vergnügen. Die Programmierkenntnisse, die Sie in diesem Kapitel erwerben, lassen sich natürlich nicht nur auf Grafik-Programme anwenden. Sie werden die gleichen Kommandos und Techniken auch später benutzen, wenn Sie ganz andere Programme schreiben; jedoch stellen die grafischen Programmier-Übungen zu diesem Kapitel eine besonders unterhaltsame Art dar, BASIC zu lernen. Jederzeit können Sie sich so – ganz wörtlich – ein Bild machen von dem, was der Computer auf Ihre Anweisungen hin von sich gibt.

Der ZX81 hat zwei verschiedene Systeme, um auf dem Bildschirm Grafiken zu erzeugen. Ein Beispiel haben Sie in Kapitel 2 gesehen. Dabei wurden die besonderen Grafik-Zeichen und invers dargestellte Buchstaben verwendet. Wir beginnen dieses Kapitel damit, daß wir beide Systeme im einzelnen untersuchen. Dann absolvieren wir zügig einen kurzen BASIC-Kurs, in dem wir die am häufigsten benötigten Kommandos behandeln. Jedes Kommando wird anhand von zahlreichen kleinen Programmen erläutert, und um noch einmal zusammenzufassen, was wir gelernt haben, werden wir ein kurzes grafisches Programm durcharbeiten, das Rechtecke auf den Bildschirm zeichnet.

Das letzte Programm in diesem Kapitel ist verhältnismäßig lang; es benötigt etwa 2K Speicherplatz. Dieses Programm läuft nur, wenn Ihr ZX81 mit einem 16K RAM-PACK ausgerüstet ist. Mit diesem Programm können Sie viele verschiedene Bilder und Muster auf den Bildschirm zeichnen. Die Bilder werden umso besser, je besser Sie mit dem Programm

umgehen können. Das Programm selbst verdient ein sorgfältiges Studium, weil es verschiedene wichtige Techniken veranschaulicht, mit deren Hilfe Sie klarere und leistungsfähigere Programme schreiben können.

Direkt ausführbare Kommandos

In Kapitel 2 haben wir drei Eigenschaften eines BASIC-Programmes erörtert:

- die Zeilen sind numeriert;
- jede Zeile beginnt mit einem Schlüsselwort;
- der Computer speichert die Zeilen nur, ohne irgendeine Anweisung auszuführen, bis Sie RUN eintippen.

Viele Schlüsselwörter (jedoch nicht alle) können auch direkt benutzt werden; d. h. Sie können eine Anweisung ohne Zeilennummer eintippen, und der Computer führt diese Anweisung sofort aus, ohne auf den Befehl RUN zu warten. Gelegentlich können Ihnen die direkten Kommandos dazu dienen herauszufinden, wie der Computer auf die eine oder andere Anweisung reagiert.

Geben Sie z. B. das folgende Kommando über die Tastatur ein:

```
PRINT "ZX81"
```

Das Ergebnis ist, wie Sie sehen, daß der Computer sofort ZX81 oben auf den Bildschirm schreibt. Wenn Sie jetzt wieder NEW LINE drücken, werden Sie feststellen, daß der Computer diese PRINT-Anweisung nicht gespeichert hat. Direkte Kommandos werden nur einmal ausgeführt und sind dann verloren.

Sie haben schon verschiedene andere Beispiele für direkte Kommandos gesehen. Sie wissen, daß SAVE und LOAD dazu dienen, Programme auf Kassetten zu speichern bzw. wieder zu laden. Durch LIST wird der Computer angewiesen, die Programmzeilen auf dem Bildschirm anzuzeigen. RUN bewirkt, daß der Computer damit beginnt, die Anweisungen eines Programmes auszuführen. Ein anderes Kommando, das Sie bei Gelegenheit benutzen können, heißt NEW; damit wird das zur Zeit gespeicherte Programm gelöscht, damit Sie ein neues Programm eingeben können. Alle diese Anweisungen werden im allgemeinen direkt gegeben – obwohl sie auch in Programmzeilen benutzt werden können.

Andere Kommandos, so wie PRINT, werden normalerweise in Programmen benutzt, können aber auch direkt ausgeführt werden. Wir wer-

den die PRINT-Anweisung im nächsten Abschnitt als direkte Anweisung benutzen, um schnell einige Experimente durchzuführen.

Zwei Grafik-Systeme

Die Anweisung PRINT AT

Wenn Sie den PRINT-Befehl verwenden, um Informationen auf dem Bildschirm anzuzeigen, benötigen alle darstellbaren Zeichen gleich viel Platz. Denken Sie sich den Bildschirm als ein Gitternetz, das in Zeilen und Spalten aufgeteilt ist, wobei Sie in jedes Kästchen ein Zeichen setzen können. Die gesamte Bildschirmfläche besteht aus 22 Zeilen und 32 Spalten. (Der untere Teil des Bildschirms – unterhalb der 22. Zeile – wird für die Eingabe neuer Zeilen verwendet und ist über die PRINT-Anweisung nicht zugänglich.)

Mit Hilfe der Anweisung PRINT AT können Sie genau bestimmen, an welcher Stelle des Bildschirms eine Information erscheinen soll. Wie Sie in Kapitel 2 gesehen haben, folgen auf PRINT AT immer zwei Zahlen.

Abbildung 3.1: Der Gebrauch der PRINT AT-Anweisung

Diese Zahlen stellen eine „Adresse" auf dem Bildschirm dar. Aus Abbildung 3.1 können Sie ersehen, wie diese Adressen festgelegt werden. Die erste Zahl, die nach PRINT AT steht, ist die Zeile, und die zweite Zahl ist die Spalte der Adresse. Allgemein sieht eine PRINT AT-Anweisung also so aus:

```
PRINT AT Zeile, Spalte; "irgendeine Information"
```

Merken Sie sich noch einmal, daß die beiden Zahlen – die zur Bezeichnung der Zeile und der Spalte dienen – durch ein Komma getrennt sind, wogegen die Adresse von der Information durch ein Semikolon getrennt sein muß.

Sehen Sie sich jetzt wieder die Abbildung 3.1 an. Die Adressen beginnen in der oberen linken Ecke des Bildschirms. Sie können sich diese Stelle als den Nullpunkt (oder Ursprung) vorstellen, wenn Sie wollen. Seine Adresse heißt 0,0. Achten Sie auf die Adressen der drei anderen Bildschirmecken. Die Zeilennummern werden größer, wenn Sie den Bildschirm heruntergehen, die Spaltenzahlen werden größer, wenn Sie sich nach rechts bewegen.

Um sich zu vergewissern, daß Sie dieses Adressierungs-System verstehen, sollten Sie sich die folgenden Anweisungen genau anschauen. Versuchen Sie dann vorherzusagen, an welcher Stelle des Bildschirms die einzelnen Informationen erscheinen werden:

```
PRINT AT 19,5; "X"
PRINT AT 7,7; "WO STEHT DIES?"
PRINT AT 15,25; "HALLO"
```

Geben Sie die Anweisungen nun direkt ein – einzeln nacheinander – und überprüfen Sie, ob Ihre Vorhersagen richtig waren. Wenn die Information aus mehreren Zeichen besteht, erscheint das erste Zeichen an der angegebenen Adresse; die übrigen folgen danach.

Sie können mit einer einzigen PRINT-Anweisung mehrere Informationen an verschiedenen Bildschirmstellen ausgeben. Versuchen Sie es mal mit der folgenden Anweisung:

```
PRINT AT 3,3; "HIER";AT 20,25; "DORT"
```

Zusammenfassend gesagt kann PRINT AT dazu benutzt werden, beliebige Zeichen, Wörter, Zahlen oder besondere Grafik-Zeichen an beliebiger Stelle des Bildschirmrasters von 22 mal 32 Positionen auszugeben.

PLOT und UNPLOT

Ihr Computer kennt noch eine weitere Methode, um grafische Darstellungen auf den Bildschirm zu schicken. Dazu dienen die Schlüsselwörter PLOT und UNPLOT. Diese Kommandos benutzen eine andere Bildschirmaufteilung und erfordern demnach ein anderes Adressierungs-System.

In Abbildung 3.2 ist die Benutzung von PLOT und UNPLOT dargestellt. Der PLOT-Befehl erzeugt an einer bestimmten Adresse des Bildschirms ein kleines schwarzes Quadrat, das als Pixel (für „picture element", d. h. Bildelement) bezeichnet wird. Mit jedem PLOT-Befehl wird ein solches Bildelement auf den Bildschirm gesetzt. Die vier Bildschirmecken in der Abbildung 3.2 enthalten jeweils ein Pixel. Insgesamt können auf dem Bildschirm 44 Zeilen und 64 Spalten mit Pixel besetzt werden. Erinnern Sie sich an das Gitter-Adressen-System bei der PRINT AT-Anweisung. Wenn Sie jedes Quadrat dieses Gitters in vier kleinere Quadrate aufteilen, erhalten Sie die Größe eines Pixels.

Abbildung 3.2: Der Gebrauch der Anweisungen PLOT und UNPLOT

Der Ursprung (oder Nullpunkt) der Pixel-Adressen ist in der unteren linken Bildschirmecke anstatt in der oberen. Diese Position hat die Adresse 0,0. Die Zeilenadressen werden größer (bis 43), wenn Sie den Bildschirm hinaufgehen, die Spaltenadressen werden größer (bis 63), wenn Sie sich nach rechts bewegen. Die allgemeine Form der PLOT-Anweisung lautet:

```
PLOT Spalte, Zeile
```

Damit Sie sich an die Pixel und das Adressierungs-System für den PLOT-Befehl gewöhnen können, geben Sie nun die folgenden Zeilen als direkte Anweisungen nacheinander ein, und achten Sie darauf, wohin die Pixels jeweils gesetzt werden:

```
PLOT 5,5
PLOT 55,3
PLOT 40,40
```

Die Anweisung UNPLOT „löscht" ein Pixel auf dem Bildschirm. Sie hat die gleiche Form wie die PLOT-Anweisung:

```
UNPLOT Spalte, Zeile
```

In den Programmen dieses Kapitels werden Sie eine ganze Reihe von PRINT AT, PLOT und UNPLOT-Anweisungen sehen. Deshalb sollten Sie sich klarmachen, wie diese Anweisungen funktionieren, bevor Sie weiterlesen. Wenn Ihnen der Unterschied zwischen den beiden grafischen Darstellungsformen noch nicht ganz klar ist, schauen Sie sich noch einmal die Abbildungen 3.1 und 3.2 an und experimentieren Sie noch etwas mit den direkten Anweisungen.

Ein kurzer BASIC-Kurs

Dieser Abschnitt bietet Ihnen eine Einführung in die meisten Grundbegriffe der Programmierung in BASIC für Ihren ZX81: die Definition von Variablen, das Eingeben von Informationen, wiederholt auszuführende Anweisungen, Verzweigungen in der Programmausführung, das Treffen von Entscheidungen. Sie werden viele kurze Beispielprogramme vorfinden, die jede Einzelheit veranschaulichen. Es ist wichtig, daß Sie während des Lesens jedes dieser Programme in Ihren Computer eingeben

und es ausprobieren. Wenn Sie nicht verstehen, wie die Ergebnisse eines Programmes zustande kommen, gehen Sie zurück und sehen Sie sich die dazugehörige Erklärung noch einmal an.

Noch einmal: Variablen

Wir haben schon festgestellt, daß eine Variable einfach ein Platz im Speicher des Computers ist, der für eine bestimmte Art von Information reserviert ist. In einem Programm können Sie viele verschiedene Variablen definieren. Damit Sie auf die Information leicht zugreifen können, hat jede Variable einen Namen. Die in einer Variablen gespeicherte Information kann sich ändern, der Name jedoch bleibt unverändert. Wir haben auch bereits festgestellt, daß es zwei Typen von Variablen gibt – numerische und nichtnumerische oder String-Variablen.

Variablen werden mit der LET-Anweisung definiert oder verändert. Die allgemeine Form der LET-Anweisung ist:

```
LET NAME = WERT
```

Diese Anweisung besagt: „Speichere einen WERT in einer Variablen, die NAME heißt." Wenn der Computer eine LET-Anweisung ausführt, berechnet er zunächst den Ausdruck, der auf der rechten Seite des Gleichheitszeichens steht, und speichert dann diese Information in der Variablen, die auf der linken Seite des Gleichheitszeichens steht.

Lassen Sie uns einige Beispiele betrachten. Die LET-Anweisung kann als direkte Anweisung benutzt werden, so daß Sie die folgenden Anweisungen ohne Zeilennummern eingeben können. Angenommen, Sie wollen eine Variable einführen, die EINKOMMEN heißt, in der Sie den Betrag Ihres diesjährigen Einkommens speichern wollen. Die Anweisung dazu könnte etwa so aussehen:

```
LET EINKOMMEN = 15000
```

Geben Sie diese Anweisung ein und achten Sie auf die Ergebnisse. Sie sehen lediglich die Information 0/0 in der unteren linken Ecke des Bildschirms – d. h. der Computer hat die Anweisung ausgeführt.

Nun vergewissern Sie sich, daß EINKOMMEN definiert wurde, indem Sie eingeben:

```
PRINT EINKOMMEN
```

Sie werden sehen, daß der Wert der Variablen EINKOMMEN – 15000 – oben auf dem Bildschirm ausgegeben wird. Wenn Sie also einen Variablen-Namen in einer PRINT-Anweisung angeben, zeigt der Computer den Wert dieser Variablen auf dem Bildschirm an.

Es steht Ihnen frei, beliebige Namen für eine Variable zu wählen. Wenn möglich, sollten Sie jedoch einen Namen so wählen, daß er einen Hinweis auf den gespeicherten Wert darstellt. Bei numerischen Variablen können Sie auch selbst entscheiden, wie lang der Variablenname sein soll. Viele Leute bevorzugen lange Variablennamen, weil ein Programm dadurch besser lesbar wird. Sie können sofort sehen, wofür die Variable steht. Die Nachteile von langen Variablennamen bestehen einerseits darin, daß es aufwendiger ist, sie einzutippen, und zum anderen darin, daß sie mehr Speicherplatz im Computer belegen. Vielleicht verwenden Sie deshalb auch kürzere Variablennamen, z. B. Abkürzungen wie EIN oder einfach E.

Nachdem Sie einmal eine Variable mit einem bestimmten Namen definiert haben, erkennt der Computer diesen Namen nur dann wieder, wenn Sie ihn ganz genau so wie beim ersten Mal schreiben. Wenn Sie sich verschreiben oder versuchen, den Namen abzukürzen, weiß der Computer nicht, was Sie meinen. Versuchen Sie z. B. die folgenden PRINT-Anweisungen einzugeben (nachdem Sie zuvor die Variable EINKOMMEN definiert haben:

```
PRINT EIMKOMMEN
PRINT EIN
PRINT E
```

Mit keinem dieser Namen kommt man an die Information in der Variablen EINKOMMEN heran. Nach jedem dieser Kommandos sehen Sie die Meldung:

```
2/0
```

in der unteren linken Bildschirmecke. Die 2 ist der Code für eine Fehlermeldung des Computers; sie bedeutet einfach, daß Sie einen nicht definierten Variablennamen benutzt haben. (Sie finden die Codes für sämtliche Fehlermeldungen im Anhang B.)

Variablen in einem Programm dienen dazu, Informationen zu speichern, die Sie noch weiter benötigen. Außer numerischen Informationen – also Zahlen – können Sie auch beliebige Zeichen, Wörter oder Texte im Computer speichern. Wie bereits gesagt, heißen die nichtnumerischen

Daten Strings (oder Zeichenketten). Dementsprechend werden die Variablen, in denen solche Informationen gespeichert sind, als String-Variablen bezeichnet. Der Name einer String-Variablen muß (Sie erinnern sich!) mit dem Dollarzeichen „$" enden. Im Unterschied zu numerischen Variablen darf der Name einer String-Variablen nur aus zwei Zeichen bestehen: einem Buchstaben und dem $-Zeichen.

Hier sind einige Beispiele für LET-Anweisungen, durch die String-Variablen definiert werden:

```
LET S$ = "HAUPTSTR. 123"
LET P$ = "5000"
LET O$ = "KOELN"
```

Geben Sie diese drei LET-Anweisungen und anschließend das folgende Kommando ein:

```
PRINT P$; O$; S$
```

Sie erhalten oben auf dem Bildschirm folgende Ausgabe:

```
5000 KOELN HAUPTSTR. 123
```

Wie Sie sehen, kann eine PRINT-Anweisung mehrere Variablen enthalten. Der Computer zeigt dann den Inhalt jeder dieser Variablen auf dem Bildschirm an. Beachten Sie, daß die Variablen durch ein Semikolon voneinander getrennt sind. Dies bewirkt, daß die Werte der Variablen in einer Zeile nebeneinanderstehend ausgegeben werden.

Geben Sie nun diese beiden PRINT-Anweisungen ein:

```
PRINT "ICH WOHNE IN"; O$; "."
PRINT "MEIN EINKOMMEN BETRAEGT DM"; EINKOMMEN; "."
```

In einer PRINT-Anweisung können also die Elemente beliebig miteinander kombiniert werden: Texte, numerische Variablen und String-Variablen. Die erste der beiden PRINT-Anweisungen sollte folgende Ausgabe im oberen Bildschirmteil ergeben:

```
ICH WOHNE IN KOELN.
```

Was bewirkt die zweite Anweisung? Als nächsten Versuch verändern Sie nun mit Hilfe der LET-Anweisung den Wert von O$ in den Namen Ih-

res Ortes und den Wert von EINKOMMEN in den Ihres eigenen Einkommens. Geben Sie anschließend die beiden PRINT-Anweisungen noch einmal ein, und sehen Sie sich an, was nun dabei herauskommt.

Zum Löschen von Variablen – und der darin gespeicherten Daten – gibt es verschiedene Möglichkeiten. Die einfachste Möglichkeit besteht natürlich darin, den Strom abzuschalten; allerdings verlieren Sie dann alles, was Sie gespeichert haben, einschließlich des Programmes. Benutzen Sie also lieber das Schlüsselwort

```
CLEAR
```

das über der X-Taste liegt. Dieses Kommando löscht alle Variablen und Daten im Speicher des Computers, jedoch nicht das Programm. Geben Sie dieses Kommando jetzt ein und anschließend die Anweisung

```
PRINT EINKOMMEN
```

Sie erhalten die Fehlermeldung 2/0, die besagt, daß die Variable EINKOMMEN nicht mehr definiert ist.

Außerdem werden bei jedem Programm-Start mit RUN sämtliche Variablen und Daten aus vorigen Programmabläufen gelöscht.

Neben der LET-Anweisung stellt die INPUT-Anweisung eine zweite Möglichkeit dar, Variablen zu definieren. Diese Anweisung wollen wir uns als nächstes ansehen.

Die Eingabe von Information über die Tastatur

Im ZX81-BASIC besitzt die INPUT-Anweisung nur eine einzige Form:

```
INPUT Name
```

Dabei kann der „Name" ein beliebiger Variablenname sein.

Wie Sie schon im Begrüßungsprogramm in Kapitel 2 gesehen haben, bewirkt INPUT, daß der Computer auf die Eingabe von Daten über die Tastatur wartet, die dann unter dem in der INPUT-Anweisung stehenden Variablennamen gespeichert werden. Dabei ist es gleichgültig, ob diese Variable schon vorher definiert war oder nicht. Im ersten Fall ginge dann der alte Wert verloren und würde durch den neuen Wert ersetzt, der über die Tastatur eingegeben wird.

Da INPUT nicht als direkte Anweisung verwendet werden kann, müssen wir ein kleines Programm schreiben, um damit herumprobieren zu können. Tippen Sie die folgenden sechs Zeilen ein:

```
10 PRINT AT 21,0; "NAME?"
20 INPUT N$
30 PRINT AT 21,0; "ALTER?"
40 INPUT A
50 CLS
60 PRINT N$;", ALTER ";A
```

Wenn Sie dieses Programm laufen lassen, sehen Sie, daß es zweimal anhält, damit Sie über die Tastatur etwas eingeben können. Geben Sie zuerst Ihren Namen und dann Ihr Alter ein. Daraufhin werden Sie eine Anzeige sehen, die ungefähr so aussehen könnte:

```
HANS DAMPF, ALTER 31
```

Aus diesem Programm und seinen Ergebnissen kann man viele wichtige Dinge lernen. Zunächst einmal: Was genau zeigt der Computer auf dem Bildschirm an, wenn er auf eine Eingabe von der Tastatur wartet? Wenn Sie während des Programmlaufs genau aufgepaßt haben, konnten Sie zwei verschiedene Arten der Bildschirmanzeige unterscheiden. Wenn der Computer auf eine String-Eingabe wartet, zeigt er den [L]-Cursor (natürlich invers dargestellt) zwischen Anführungszeichen an. Wartet er andererseits auf numerische Daten, so zeigt er nur den [L]-Cursor an. Auf diese Weise unterscheidet der Computer zwischen einer INPUT-Anweisung, die eine String-Variable enthält, und einer solchen, die eine numerische Variable enthält.

Die Zeilen 10 und 30 in diesem kurzen Programm veranschaulichen ein Hauptmerkmal für den Entwurf von übersichtlichen und wirkungsvollen Programmen. Wenn Sie in einem Programm INPUT-Anweisungen verwenden, denken Sie immer an die Person, die mit dem Programm umgehen soll. Dieser Person muß mitgeteilt werden, was für eine Information der Computer erwartet, wenn er eine Eingabe verlangt. Deshalb sollte der Computer immer eine entsprechende „Aufforderung" auf dem Bildschirm ausgeben, wenn er auf eine Eingabe wartet. Die Zeilen 10 und 30 enthalten Beispiele für solche Aufforderungen. Beachten Sie, daß die Ausgabe dieser Aufforderungen am unteren Bildschirmrand erfolgt. Dies wurde deshalb so programmiert, damit die Eingabe-Aufforderung möglichst nahe bei der Eingabe steht, die über die Tastatur erfolgt.

Zeile 50 in diesem Programm löscht den Bildschirm, so daß die Mitteilung aus Zeile 60 oben auf dem Bildschirm erscheint. Sehen Sie sich die Zeile 60 genau an; beachten Sie, daß der Text zwischen den Anführungszeichen („, ALTER ") durch ein Komma und zwei Leerzeichen von den beiden Variablen getrennt ist:

```
60 PRINT N$; ", ALTER ";A
```

Um noch einmal zusammenzufassen: Wir haben gesehen, daß Variablen auf zwei verschiedene Arten definiert werden können: mit der LET-Anweisung oder mit der INPUT-Anweisung. Im allgemeinen können Sie eine Variable in einem Programm nicht benutzen, bis sie auf die eine oder andere Art definiert worden ist. (Es gibt eine Ausnahme von dieser Regel, die wir im nächsten Kapitel kennenlernen werden.) Nachdem eine Variable einmal definiert worden ist und einen Wert erhalten hat, kann ihr Wert in einer PRINT-Anweisung auf dem Bildschirm ausgegeben werden. Sie können den Wert einer Variablen auch durch nachfolgende LET- oder INPUT-Anweisungen verändern. Damit werden wir uns im weiteren Verlauf dieses Kapitels noch beschäftigen.

Bevor wir fortfahren, wollen wir uns ein weiteres kurzes Programm ansehen, das den Gebrauch von Variablen illustriert. Geben Sie das Schlüsselwort NEW ein, um das im Speicher des Computers stehende Programm zu löschen. Tippen Sie nun die folgenden fünf Zeilen ein:

```
200 PRINT AT 21,0; "HORIZONTAL"
210 INPUT H
220 PRINT AT 21,0; "VERTIKAL"
230 INPUT V
240 PLOT H,V
```

Diese Zeilen sind Teil eines größeren Programmes, das wir im weiteren Verlauf erstellen werden, jedoch können Sie diese fünf Zeilen auch schon als selbständiges Programm betrachten und laufen lassen. Wenn Sie das Programm starten, sehen Sie zwei Eingabe-Aufforderungen auf dem Bildschirm. Zuerst:

```
HORIZONTAL
```

An dieser Stelle können Sie eine Zahl zwischen 0 und 63 eingeben; dann

```
VERTIKAL
```

worauf Sie eine Zahl zwischen 0 und 43 eingeben können. Wenn Sie beide Zahlen eingegeben haben, zeigt das Programm auf dem Bildschirm ein Pixel an, und zwar an der durch die beiden Zahlen angegebenen Adresse.

Machen Sie sich klar, wie das Programm funktioniert: die Zeilen 210 und 230 enthalten die INPUT-Anweisungen, durch die die numerischen Variablen H bzw. V definiert werden. In Zeile 240 werden dann die Werte von H und V als Adresse in einer PLOT-Anweisung benutzt:

```
240 PLOT H,V
```

In den PLOT- und UNPLOT-Anweisungen können nicht nur Zahlen für die Pixel-Adresse stehen, sondern auch Variablen, die definierte Werte in dem zulässigen Adressbereich für die Bildpunkte enthalten.

Als nächstes wollen wir die FOR/NEXT-Anweisungen untersuchen. Sie werden sich von Kapitel 2 her erinnern, daß diese Anweisungen dazu benutzt werden können, den Computer Anweisungen wiederholt ausführen zu lassen.

Anweisungen für Wiederholungen

Ein Beispiel für die einfachste Form einer FOR-Anweisung ist:

```
FOR I = 1 TO 20
```

Das I in dieser Anweisung kann als Lauf- oder Kontroll-Variable bezeichnet werden, da es kontrolliert, wie oft die FOR-Anweisung und die danach folgenden Anweisungen wiederholt werden. Der Name der Kontroll-Variablen darf nur aus einem Buchstaben bestehen. Die Kontroll-Variable muß vorher nicht durch eine LET- oder INPUT-Anweisung definiert werden. (Dies ist die Ausnahme von der Regel.)

Sehen wir uns doch einmal an, wie eine FOR-Anweisung ausgeführt wird. Da FOR und NEXT nicht als direkte Kommandos benutzt werden können, müssen wir sie in ein kleines Programm einbauen:

```
10 FOR I = 1 TO 20
20 PRINT "WIEDERHOLUNG"
30 NEXT I
```

Denken Sie daran, daß die Zeilen mit FOR und NEXT den Anfang und das Ende einer „Schleife" darstellen. Alle Zeilen dazwischen werden so lange wiederholt, bis die Kontroll-Variable den Endwert erreicht hat.

Im folgenden ist dargestellt, wie der Computer die Kontroll-Variable, in diesem Fall I, benutzt, um die Zahl der Wiederholungen zu bestimmen:

1. Zu Beginn erhält I den Wert, der direkt hinter dem Gleichheitszeichen steht, in diesem Fall 1.
2. Alle Anweisungen zwischen FOR und NEXT werden ausgeführt.
3. Wenn der Computer in der NEXT-Zeile ankommt, erhöht er den Wert von I um 1 und geht dann zurück zur FOR-Zeile.
4. Wenn I immer noch kleiner oder gleich der hinter dem Wort TO angegebenen Zahl ist (20 in diesem Fall), wiederholt der Computer wieder alle Zeilen bis NEXT. Wenn I größer ist als die Zahl hinter TO, wird die Schleife beendet.

Tippen Sie die drei Zeilen in den Computer und starten Sie das Programm. Abbildung 3.3 zeigt, was dabei herauskommt. Wie Sie sehen, ist die PRINT-Anweisung in Zeile 20 zwanzigmal ausgeführt worden.

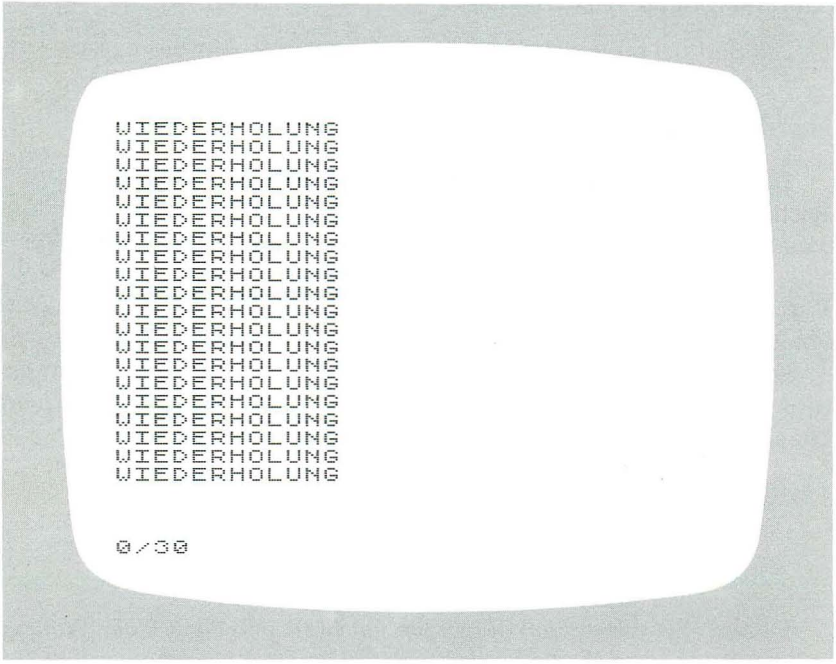


Abbildung 3.3: Wiederholung

Mit anderen Worten: die Kontroll-Variable I ist schrittweise von 1 bis 20 erhöht worden; nach jeder Erhöhung wurde die PRINT-Anweisung einmal ausgeführt.

Um den Wiederholungsbegriff noch weiter zu untersuchen, wollen wir zwei Variationen unseres kurzen Programmes betrachten. Dazu nehmen wir in der Zeile 20 jeweils kleine Änderungen vor. Benutzen Sie für die Änderungen die EDIT-Funktion Ihres Computers.

Ändern Sie zunächst Zeile 20 wie folgt:

```
20 PRINT "WIEDERHOLUNG ";
```

Zwei Veränderungen wurden vorgenommen: hinter dem Buchstaben G wurde ein Leerzeichen eingefügt, und an das Ende der Zeile wurde ein Semikolon angehängt. Nachdem Sie diese Änderungen gemacht haben, lassen Sie das Programm noch einmal laufen. Das Ergebnis sollte so aussehen, wie es in Abbildung 3.4 dargestellt ist. Diese kleine Übung

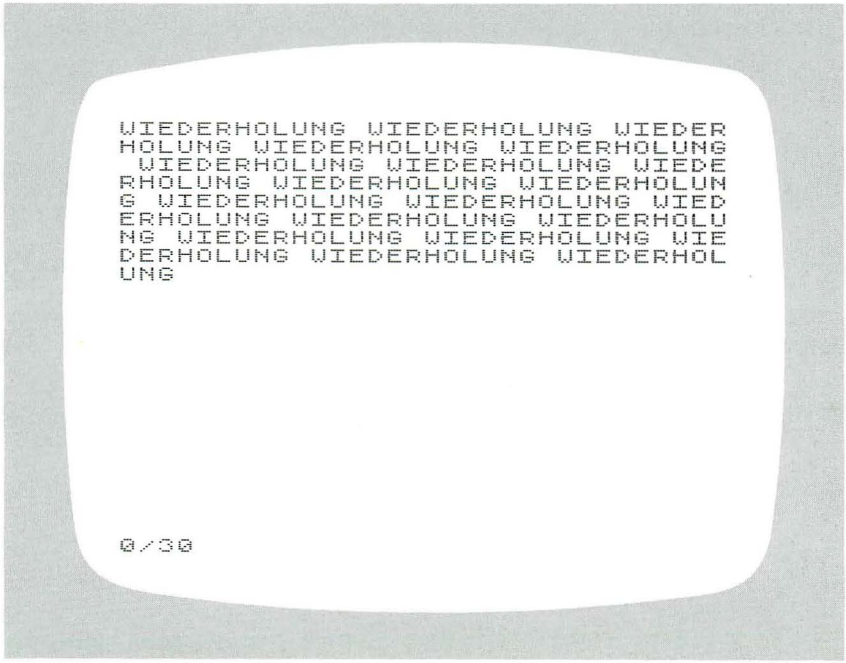


Abbildung 3.4: Weitere Wiederholungen

dient einfach dazu, Sie an die Funktion des Semikolons in einer PRINT-Anweisung zu erinnern. Das Semikolon weist den Computer an, nicht in eine neue Zeile zu gehen. (Man sagt: das Semikolon verhindert einen Zeilenvorschub.) Wenn also eine PRINT-Anweisung mit einem Semikolon endet, erfolgt die nächste Ausgabe mit PRINT an der Stelle, wo die vorige aufhörte.

Als abschließenden Versuch ändern Sie die Zeile 20 wie folgt:

```
20 PRINT AT I,I; "WIEDERHOLUNG"
```

und lassen Sie das Programm noch einmal laufen. Die Ergebnisse sind in Abbildung 3.5 dargestellt.

Die letzte Version des Programmes macht einen wesentlichen Punkt deutlich: Die Kontrollvariable aus der FOR-Anweisung kann auch in anderen Anweisungen innerhalb der Schleife benutzt werden. Die Tatsache, daß die Kontrollvariable bei jedem Schleifendurchgang um eins er-

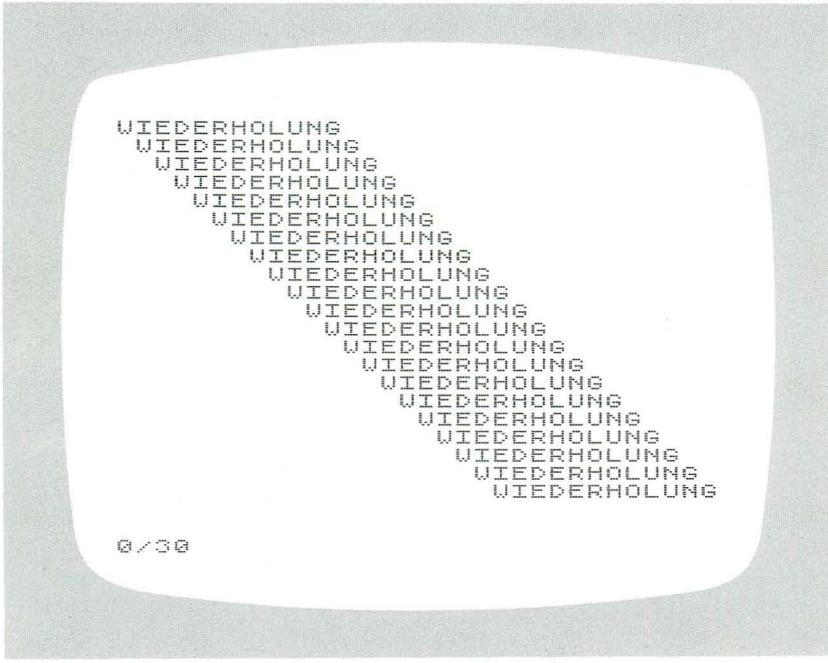


Abbildung 3.5: Noch mehr Wiederholungen

höht wird, kann für Aufgaben mit sich wiederholenden Anweisungen äußerst nützlich sein. In der Abbildung 3.5 sehen Sie, daß durch die Kontrollvariable I die Adresse für die einzelnen PRINT-Anweisungen gebildet wurde. Während sich der Wert von I in der Schleife von 1 auf 20 erhöhte, wurde die PRINT-Anweisung folgendermaßen ausgeführt:

```
PRINT AT 1,1; "WIEDERHOLUNG"
PRINT AT 2,2; "WIEDERHOLUNG"
PRINT AT 3,3; "WIEDERHOLUNG"
PRINT AT 4,4; "WIEDERHOLUNG"
```

. . . und so weiter.

Lassen Sie uns nun den letzten Punkt noch an einem weiteren Beispiel untersuchen. Ändern Sie dazu Ihr Drei-Zeilen-Programm, daß es so aussieht:

```
10 FOR I = 20 TO 40
20 PLOT I,I
30 NEXT I
```

In dieser Version benutzen wir also den Wert der Kontrollvariablen dazu, um die Adressen der Pixels zu bestimmen, die mit dem PLOT-Kommando erzeugt werden. Lassen Sie das Programm laufen, und Sie sehen eine Diagonale aus Pixeln, die sich von der Adresse 20,20 bis zur Adresse 40,40 erstreckt. Beachten Sie, daß die Kontrollvariable I in diesem Programm nicht bei 1 beginnt, sondern von 20 bis 40 läuft.

Wenn Sie Zeile 20 wie folgt ändern, können Sie eine horizontale Linie aus Pixeln erzeugen:

```
20 PLOT I,20
```

Entsprechend erhalten Sie eine vertikale Pixel-Linie, wenn Sie das Programm mit dieser Zeile laufen lassen:

```
20 PLOT 20,I
```

Probieren Sie am besten alle Programmversionen einmal aus.

In sämtlichen bisherigen Beispielen wurde die Kontrollvariable bei jedem Schleifendurchlauf um eins erhöht. In einigen Fällen kann es erwünscht sein, diese Schrittweise zu ändern. Dies können Sie mit dem Wort STEP erreichen, z. B.:

```
FOR I = 2 TO 40 STEP 2
```

(STEP ist eine Umschalt-Funktion, die sich auf der E-Taste befindet.)

Damit Sie sehen, wie STEP funktioniert, machen Sie sich die folgende Version unseres Programmes klar:

```
10 FOR I = 2 TO 40 STEP 2
20 PLOT I,20
25 PLOT 20,I
30 NEXT I
```

In diesem Programm hat die Kontrollvariable I nacheinander die Werte 2, 4, 6, 8, ..., 40. Das Ergebnis ist, daß die PLOT-Anweisungen eine durchbrochene Linie zeichnen, da jeweils ein Bildpunkt übersprungen wird. (Dieses Programm zeigt auch, wie einfach es ist, Zeilen in eine FOR-Schleife einzufügen.) Lassen Sie das Programm laufen. Abbildung 3.6 zeigt, was auf dem Bildschirm passiert. Sehen Sie sich das Programm und die Ergebnisse gut an, bevor Sie weitermachen. Machen Sie sich klar, warum das Programm das tut, was es tut.

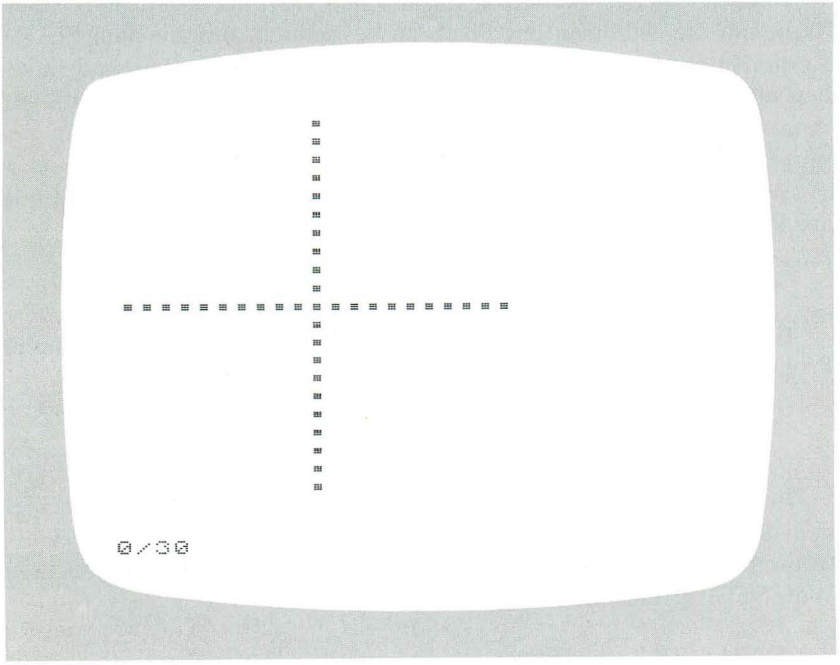


Abbildung 3.6: Versuche mit STEP

In allen Programmen, die wir uns bisher angesehen haben, wird die Reihenfolge, in der die einzelnen Anweisungen ausgeführt werden, ausschließlich durch die Zeilennummern bestimmt. Der Computer beginnt mit der kleinsten Zeilennummer und geht dann Zeile für Zeile weiter, bis die letzte Programmzeile ausgeführt ist. Manchmal ist ein solcher Programmaufbau angemessen, oft jedoch gibt es gewichtige Gründe, bestimmte Zeilen außerhalb dieser Reihenfolge auszuführen. Wir wollen einige dieser Gründe erörtern und dann im nächsten Abschnitt die Schlüsselwörter besprechen, die dies ermöglichen.

Wir steuern das Geschehen

Das Schlüsselwort GOTO (es liegt über der G-Taste) kann dazu benutzt werden, daß der Computer eine Zeile außerhalb der Reihenfolge ausführt. Hinter GOTO steht einfach die Nummer der Zeile, die der Computer als nächste ausführen soll. Mit der GOTO-Anweisung kann das Programm mit einer Zeile fortgesetzt werden, die eine höhere Zeilennummer als die direkt folgende Zeile hat:

```
90 GOTO 300
```

Es kann aber auch zu einer davor liegenden Zeile zurückgeschickt werden:

```
90 GOTO 10
```

Die GOTO-Anweisung ist zwar einfach, doch sind die Gründe, aus denen sie benutzt wird, oft kompliziert. Im allgemeinen ist es eine gute Idee, diese Anweisung nur sparsam zu verwenden, weil es durch die Verwendung von sehr vielen GOTOs fast unmöglich wird, die Ergebnisse eines Programmes sicher vorherzusagen. Jedoch gibt es einige Situationen, in denen es sich nicht vermeiden läßt, GOTO zu verwenden.

In den beiden größeren Programmen dieses Kapitels werden wir GOTO benutzen, um eine „Endlos-Schleife" zu erzeugen. Das Programm, oder ein bestimmter Teil des Programmes, wird fortlaufend wiederholt. Das Programm selbst besitzt keine Vorkehrung, um jemals aufzuhören. Ein solches Programm können Sie jedoch von außen – über die Tastatur – abbrechen. Dazu drücken Sie einfach die BREAK-Taste, die sich in der unteren rechten Ecke der Tastatur befindet.

Die folgenden Zeilen sind ein Beispiel für eine Endlos-Schleife:

```
10 PRINT AT 10,13; "BLITZ"
20 PRINT AT 10,13; "      "
30 GOTO 10
```

Sie können sich den Programmablauf klarmachen, bevor Sie das Programm laufen lassen. Zuerst werden die PRINT-Anweisungen in den Zeilen 10 und 20 ausgeführt, danach wird in Zeile 30 die Kontrolle über das Programm wieder an den Anfang zurückgeschickt. Dieser Vorgang läuft ununterbrochen weiter. Dies ist natürlich ein ziemlich unsinniges Beispiel; ein Durchlauf des Programmes lohnt sich nur, um das Ergebnis zu sehen. In anderen Programmen werden wir GOTO sinnvoller benutzen.

Eine andere BASIC-Anweisung, durch die der Computer die normale Reihenfolge verläßt, um an eine andere Stelle im Programm zu springen, heißt GOSUB. Diese Anweisung bedeutet: „Geh zu einem Unterprogramm" („go to a subroutine"). Ebenso wie bei der GOTO-Anweisung steht danach eine Zeilennummer, z. B.:

```
GOSUB 200
```

Damit Sie die GOSUB-Anweisung verstehen können, müssen Sie wissen, was Unterprogramme sind und wofür sie benötigt werden. Oft gibt es in einem umfangreicheren Programm bestimmte abgeschlossene Tätigkeiten, die an verschiedenen Stellen des Programmes immer wieder ausgeführt werden müssen. Wenn Sie ein solches Programm entwickeln, werden Sie sich bestimmt eine bequemere und weniger aufwendige Möglichkeit zur Bewältigung derartiger Aufgaben wünschen. In BASIC können Sie diese Aufgaben in einzelnen Programmteilen lösen, die als Unterprogramme bezeichnet werden. Nachdem Sie ein Unterprogramm einmal geschrieben haben, können Sie es von jeder Stelle des Programmes aus „aufrufen". Wenn das Unterprogramm seine Aufgabe erledigt hat, wird das Programm an der Stelle fortgesetzt, von der aus der Unterprogramm-„Aufruf" erfolgte.

Sehen wir uns ein Beispiel an! Erinnern Sie sich an das kurze Programm, das eine horizontale und eine vertikale Adresse von der Tastatur liest und dann an der entsprechenden Position ein Pixel auf dem Bildschirm ausgibt. Dieses Programm wird als Unterprogramm in einem größeren Programm verwendet, das wir gleich schreiben werden. Hier sehen Sie, wie es als Unterprogramm aussieht:

```
200 PRINT AT 21,0; "HORIZONTAL"
210 INPUT H
220 PRINT AT 21,0; "VERTIKAL"
230 INPUT V
240 PLOT H,V
250 RETURN
```

Beachten Sie, daß eine Zeile hinzugefügt worden ist: eine RETURN-Anweisung in Zeile 250. RETURN ist ein Schlüsselwort, das sich über der Y-Taste befindet. Jedes Unterprogramm wird mit RETURN abgeschlossen; die Anweisung besagt einfach, daß das Unterprogramm beendet ist. Wenn der Computer zu der RETURN-Anweisung gelangt, setzt er die Programmausführung automatisch an der Stelle fort, von der aus das Unterprogramm aufgerufen wurde.

Um dieses Unterprogramm aufzurufen, schreiben Sie einfach die folgende Anweisung:

```
GOSUB 200
```

Wir werden zwei kurze Programme betrachten, die dieses Unterprogramm benutzen. Sie sollten beide Programme – mit dem Unterprogramm – in den Computer eingeben und ausprobieren.

Hier ist das erste:

```
10 PRINT "SIE KOENNEN 5 BILDPUNKTE SETZEN"
20 FOR I = 1 TO 5
30 GOSUB 200
40 NEXT I
50 STOP
```

Dieses Programm erlaubt Ihnen, fünf Bildpunkte auf den Bildschirm zu setzen. Zunächst druckt es eine Mitteilung im oberen Bildschirmteil aus (Zeile 10). Dann beginnt eine Schleife. Die GOSUB-Anweisung in Zeile 30 ist ein Teil dieser Schleife. Während des Schleifendurchlaufs spielt sich folgendes ab:

1. In der FOR-Anweisung wird die Anzahl der Wiederholungen durch die Laufvariable I festgelegt.
2. Durch die GOSUB-Anweisung wird das Programm mit der Zeile 200 fortgesetzt, wo die Eingabe der Adressen und das Setzen eines Bildschirmpunktes erfolgt.

3. Die letzte Zeile des Unterprogrammes (250 RETURN) bewirkt die Fortsetzung des Programmes an der Stelle, von der aus das Unterprogramm aufgerufen wurde. In unserem Fall erfolgt der Rücksprung also zu der NEXT-Anweisung in Zeile 40.
4. NEXT erhöht den Wert der Laufvariablen, und die Schleife wird fortgesetzt.

Aus dieser Ablauffolge können Sie ersehen, daß das Unterprogramm in Zeile 200 fünfmal aufgerufen wird, so daß Sie fünf Bildpunkte auf den Bildschirm setzen können. Damit ist das Programm beendet. Die Zeile 50 ist sehr wichtig:

```
50 STOP
```

Das Schlüsselwort STOP (eine Umschalt-Funktion auf der A-Taste) teilt dem Computer mit, daß das Programm beendet ist. In unserem Fall stellen die Zeilen 10 bis 50 das Hauptprogramm dar. Wenn die FOR-Schleife mit ihren fünf Durchgängen fertig ist, soll der Computer ja nicht noch einmal nach Zeile 200 gelangen und dort das Unterprogramm nochmal ausführen. Deshalb ist es notwendig, eine ausdrückliche STOP-Anweisung vor der Zeile 200 anzugeben. Dies war das erste Mal, daß wir das Schlüsselwort STOP benutzt haben. Bisher hatten wir nur Programme gesehen, die Zeile für Zeile – von der ersten bis zur letzten Zeile – ausgeführt wurden und automatisch stoppten, wenn der Computer die letzte Zeile ausgeführt hatte. In diesem Programm aber bilden die letzten sechs Zeilen (200 bis 250) ein Unterprogramm, das nur über die GOSUB-Anweisung erreicht werden soll.

Hier ist das zweite Programm-Beispiel, das unser Unterprogramm benutzt:

```
10 PRINT "SETZEN VON BELIEBIG VIELEN BILDPUNKTEN"
20 GOSUB 200
30 GOTO 20
```

Mit diesem Programm können Sie so viele Punkte auf den Bildschirm setzen, wie Sie möchten. Die Zeilen 20 und 30 bilden eine Endlos-Schleife. Dies Programm funktioniert folgendermaßen:

1. Zeile 20 ruft das Unterprogramm auf.
2. Nachdem das Unterprogramm seine Aufgabe erledigt hat, wird das Programm in Zeile 30 fortgesetzt.

3. Durch die GOTO-Anweisung in Zeile 30 wird das Programm in Zeile 20 fortgesetzt.

Lassen Sie das Programm laufen, um genau zu sehen, was es macht. Sie können so lange Pixel-Adressen eingeben, wie Sie wollen. Nach der Eingabe einer Adresse setzt der Computer den entsprechenden Punkt auf den Bildschirm. Wenn Sie mit dem Programm aufhören wollen, müssen Sie das Schlüsselwort STOP anstatt einer Adresse eingeben. Die BREAK-Taste funktioniert nämlich nicht, wenn der Computer auf eine Eingabe von der Tastatur wartet.

Wir müssen noch ein weiteres Thema besprechen, bevor wir zu den beiden größeren Programmen in diesem Kapitel kommen. Im nächsten Abschnitt werden wir sehen, wie das Schlüsselwort IF benutzt wird, um den Computer während des Programmablaufes Entscheidungen treffen zu lassen.

Entscheidungen, Entscheidungen . . .

Sie können die Leistungsfähigkeit und Handhabung eines Programmes oft beträchtlich verbessern, wenn Sie Anweisungen benutzen, die es dem Computer erlauben, zwischen zwei verschiedenen Handlungs-Abläufen zu „entscheiden". Vielleicht haben Sie schon in dem Bildpunkte-Programm ein Problem gesehen, das sich durch im Programm eingebaute Entscheidungen lösen ließe. Sehen wir uns zunächst das Problem an, dann die Lösung.

Lassen Sie das Pixel-Programm noch einmal laufen. Sie wissen, wenn Sie die Eingabe-Aufforderung

```
HORIZONTAL
```

sehen, müssen Sie eine Zahl eingeben, die kleiner oder gleich 63 ist. Entsprechend dürfen Sie bei der Eingabe-Aufforderung

```
VERTIKAL
```

keine Zahl eingeben, die größer ist als 43, da die größte Adresse für die PLOT-Anweisung 63,43 lauten darf. Jede größere Adresse läge außerhalb des Bildschirmbereiches. Vielleicht haben Sie schon gesehen, was passiert, wenn Sie versehentlich eine unzulässige Adresse eingeben. Falls nicht, versuchen Sie's jetzt einmal. Geben Sie 64 für den horizontalen und 44 für den vertikalen Teil der Adresse ein. Beide Zahlen sind für die

PLOT-Anweisung zu groß. Demnach wird auch kein Punkt auf den Bildschirm gesetzt, sondern stattdessen erfolgt ein Programmabbruch mit der Fehlermeldung

```
B/240
```

in der unteren linken Bildschirmecke.

Wenn Sie im Anhang B nachsehen, finden Sie die Bedeutung dieser Fehlermeldung. Der Fehlercode „B" bedeutet, daß eine eingegebene Zahl außerhalb eines zulässigen Wertebereiches lag. In unserem Fall handelt es sich um den Bereich, der für das Schlüsselwort PLOT gilt. Beachten Sie, daß der zweite Teil der Fehlermeldung die Zeilennummer – 240 – angibt, in der die Programmausführung abgebrochen wurde. Wenn Sie sich das Unterprogramm ansehen, sehen Sie, daß es sich natürlich um die Zeile handelt, in der die PLOT-Anweisung steht.

Damit man dieses Problem vermeidet, wäre es sehr nützlich, wenn der Computer die eingegebenen Werte der Variablen H und V überprüfen könnte, bevor er sie in der PLOT-Anweisung verwendet. Wenn einer der für H und V eingegebenen Werte zu groß wäre, würde am besten in der INPUT-Anweisung ein neuer Wert angefordert. Genau dies werden wir dem Computer beibringen, indem wir zwei IF-Anweisungen benutzen.

Eine allgemeine Beschreibung der IF-Anweisung könnte etwa so aussehen:

```
IF (Bedingung) THEN (Schlüsselwort-Anweisung)
```

Wie Sie sehen, besteht die IF-Anweisung aus zwei Teilen. Der erste Teil beginnt mit dem Schlüsselwort IF (über der U-Taste) und der zweite Teil mit dem Wort THEN (eine Umschalt-Funktion auf der 3-Taste). Nach IF folgt ein Ausdruck, der entweder wahr oder falsch sein kann. Dieser Ausdruck wird gewöhnlich in Form einer Gleichung oder Ungleichung geschrieben. Wir werden gleich sehen, wie man diese Ausdrücke formuliert. Nach dem Wort THEN folgt in Form einer Schlüsselwort-Anweisung ein Befehl an den Computer.

Bei der IF-Anweisung sind zwei Ergebnisse möglich:

1. Wenn die Bedingung erfüllt ist, führt der Computer den hinter THEN angegebenen Befehl aus.
2. Wenn die Bedingung nicht erfüllt ist, übergeht der Computer den Teil der Zeile hinter THEN und macht einfach mit der nächsten Zeile weiter.

Sie sehen, wie nützlich die IF-Anweisung ist. Mit dieser Anweisung können Sie Programme schreiben, in denen Aufgaben gelöst werden müssen, die von bestimmten Entscheidungen abhängig sind.

Nun wollen wir uns genau ansehen, wie man eine IF-Anweisung schreibt. Dazu betrachten wir zunächst einige neue Symbole auf der Tastatur. Es handelt sich dabei um Gleichheits- und Ungleichheitszeichen:

```
=    (ist gleich)
<    (ist kleiner als)
>    (ist größer als)
<=   (ist kleiner oder gleich)
>=   (ist größer oder gleich)
<>   (ist ungleich)
```

Alle diese Zeichen erhält man durch Umschalten. Sie befinden sich auf den Tasten L, N, M, R, Y und T. Diese Symbole dienen dazu, zwei verschiedene Werte – numerische oder nicht-numerische – miteinander zu vergleichen. Hier sind einige Ausdrücke, in denen diese Symbole benutzt werden:

```
I > 5
K < J
M <= 16
S $ = "JA"
H <> 15
```

In all diesen Beispielen werden entweder die Werte zweier verschiedener Variablen miteinander oder der Wert einer Variablen mit einem konstanten Wert verglichen. Alle diese Ausdrücke sind entweder wahr oder falsch. Enthält z. B. im ersten Ausdruck die Variable I den Wert 4, so ist die Aussage falsch, weil I nicht größer ist als 5. Hat I jedoch den Wert 7, so ist die Aussage wahr, weil 7 größer ist als 5.

Hier sind Beispiele für einige IF-Anweisungen, in denen diese Ausdrücke benutzt werden:

```
IF I > 5 THEN PRINT "I IST ZU GROSS"
IF K < J THEN INPUT J
IF M < = 16 THEN GOTO 200
IF S$ = "JA" THEN STOP
IF H <> 15 THEN LET H = 100
```

Beachten Sie, daß in jedem Fall nach THEN ein Schlüsselwort steht. In einer IF-Anweisung können hinter THEN sämtliche Schlüsselwörter

stehen, die auf der Tastatur verfügbar sind, wenn auch einige häufiger als andere benutzt werden. Denken Sie daran, daß die Anweisung hinter THEN nur dann ausgeführt wird, wenn die davor stehende Bedingung erfüllt bzw. die Aussage wahr ist. Ansonsten geht der Computer einfach zur nächsten Programmzeile über.

Damit wir nun sinnvollere Beispiele für die Anwendung von IF erhalten, kehren wir jetzt zu dem oben erwähnten Problem zurück. Um das Problem zu lösen, müssen wir das Unterprogramm ändern, das bei Zeile 200 beginnt. Immer wenn der Computer einen Wert von der Tastatur einliest und ihn der Variablen H bzw. V zuweist, sollte ein Test durchgeführt werden, der überprüft, ob die Werte in dem zulässigen Bereich für die PLOT-Anweisung liegen. Diesen Test können wir leicht mit Hilfe einer IF-Anweisung formulieren, die direkt hinter die beiden INPUT-Anweisungen gesetzt wird.

Hier ist die erste Anweisung:

```
210 INPUT H
215 IF H > 63 THEN GOTO 210
```

In Zeile 210 wird ein Wert für H eingelesen. Zeile 215 überprüft die Aussage H > 63 und entscheidet dann, wie es weitergehen soll. Wenn der Wert der Variablen größer ist als 63, wird die GOTO-Anweisung ausgeführt, und der Computer geht zurück zur Zeile 210, um eine neue Eingabe zu erhalten. Ist der Wert von H jedoch kleiner oder gleich 63, so wird die GOTO-Anweisung übersprungen und das Programm normal mit der nächsten Anweisung fortgesetzt.

Eine ähnliche IF-Anweisung können wir hinter die INPUT-Anweisung für V schreiben. In dieser zweiten IF-Anweisung (in Zeile 235) wird die Aussage V > 43 überprüft: die Programmausführung wird mit Zeile 230 fortgesetzt, wenn V > 43 wahr ist.

Das vollständige Unterprogramm ist in Abbildung 3.7 wiedergegeben. Sie sollten es sorgfältig studieren und sich vergewissern, daß Sie verstehen, wie die beiden IF-Anweisungen funktionieren. Probieren Sie es dann mit einem der beiden oben beschriebenen Programme aus. Versuchen Sie insbesondere während des Programmlaufs, einige unzulässige Werte für den horizontalen bzw. vertikalen Teil der Adresse einzugeben. Statt zu einem Programmabbruch führen solche Eingabefehler nun dazu, daß der eingegebene Wert einfach nicht angenommen wird.

Wenn Sie die Eingabe-Aufforderung

```
HORIZONTAL
```

sehen, versuchen Sie z. B. die Zahl 66 einzugeben. In dem Fall bleibt die Aufforderung die gleiche, so daß Sie wissen, daß 66 als horizontale Adresse nicht möglich ist. Sie müssen einen anderen Wert eintippen.

Wir betrachten nun ein Grafik-Programm, das Rechtecke auf den Bildschirm zeichnet. In diesem Programm kommt all das vor, was Sie bisher in diesem Kapitel gelernt haben. Wir wollen es das „Rechteck-Programm" nennen.

Das Rechteck-Programm

Der Hauptteil dieses Programmes ist in Abbildung 3.8 wiedergegeben. Das Programm benutzt das Unterprogramm aus Abbildung 3.7, d. h. Sie müssen auch diese Zeilen eintippen. Geben Sie nun das ganze Programm in den Computer ein und lassen Sie es laufen.

Dieses Programm soll dazu dienen, verschiedene Programmier-Techniken zu illustrieren. Gleichzeitig können Sie damit aber auch einige interessante Bilder auf dem Bildschirm erzeugen. Es funktioniert folgen-

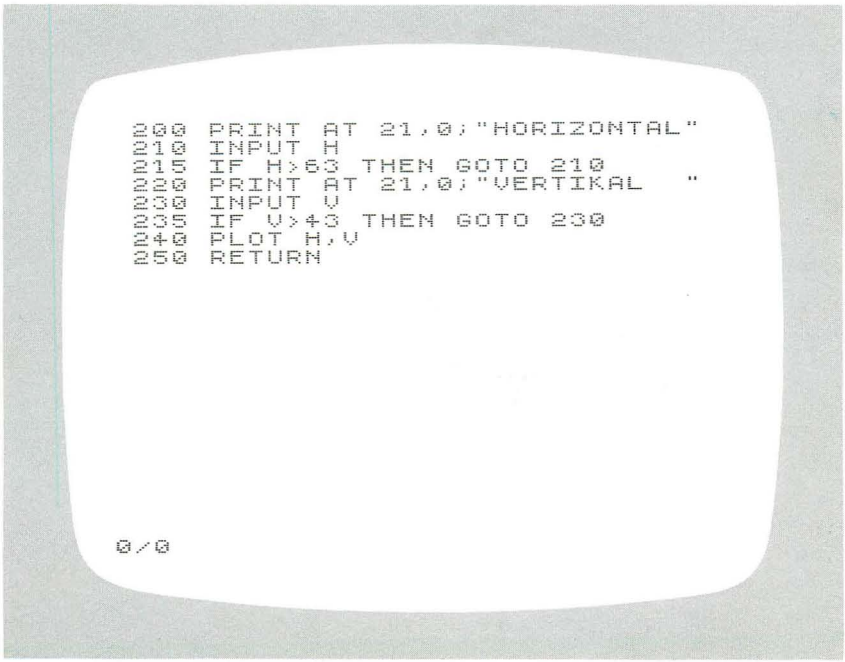


Abbildung 3.7: Das Pixel-Unterprogramm mit IF-Anweisungen

dermaßen: Zunächst werden Sie aufgefordert, die Adressen für zwei Bildschirmpunkte einzugeben, d. h. Sie sehen jeweils zweimal die Aufforderung, eine HORIZONTALe bzw. eine VERTIKALe Adresse einzugeben, und der Computer wartet jedesmal auf Ihre Eingabe. Sobald Sie eine vollständige und gültige Adresse eingegeben haben, erscheint auf dem Bildschirm der entsprechende Punkt. Wenn zwei Punkte festgelegt sind, zeichnet der Computer ein Rechteck auf den Bildschirm, bei dem die beiden Punkte zwei sich diagonal gegenüberliegende Ecken des Rechteckes bilden. Dies können Sie beliebig lange fortsetzen, so daß Sie viele unterschiedlich große und unterschiedlich geformte Rechtecke auf den Bildschirm zeichnen können. Abbildung 3.9 zeigt ein Beispiel dafür, wie der Bildschirm nach einiger Zeit aussehen kann. Wenn Sie das Programm beenden wollen, geben Sie das Schlüsselwort STOP ein.

Wenn Sie das Programm ausprobieren, beachen Sie, daß Sie damit Rechtecke in unterschiedlichen Richtungen zeichnen können. Um festzustellen was das heißt, geben Sie zunächst einmal die Adressen 5,5 und 25,25 ein. Da die erste Adresse kleiner ist als die zweite – und damit links

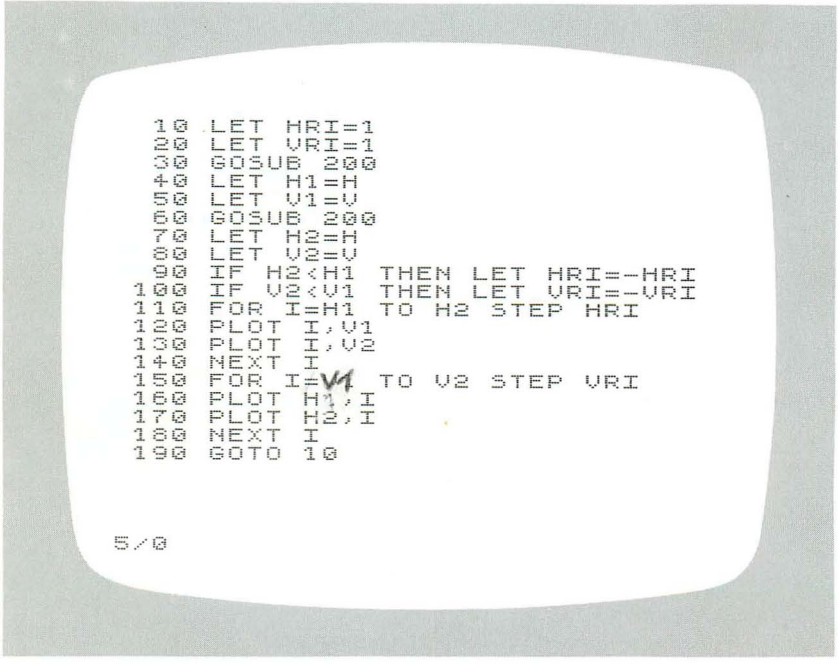


Abbildung 3.8: Das Rechteck-Programm

150 I=V1

von der zweiten liegt –, zeichnet der Computer das Ober- und Unterteil des Rechteckes von links nach rechts und die Seiten von unten nach oben. Geben Sie nun zwei andere Adressen ein: 45,43 und 25,23, in dieser Reihenfolge. In diesem Fall liegt die zuerst eingegebene Adresse über – und damit rechts von – der zweiten Adresse, so daß die Richtungen beim Zeichnen im Vergleich zum ersten Rechteck vertauscht sind. Ober- und Unterteil werden von rechts nach links und die Seiten von oben nach unten gezeichnet. Dies mag unbedeutend erscheinen, jedoch wurde das Programm absichtlich so entworfen. Sehen wir uns die Programmzeilen einmal an. Damit Sie sich das Programm auf Ihrem Bildschirm ansehen können, brechen Sie es mit STOP ab und drücken anschließend wieder die NEW LINE-Taste. Das Programm ist zu lang, um es gleichzeitig ganz auf dem Bildschirm anzuzeigen, doch wissen Sie ja, daß Sie das Schlüsselwort LIST benutzen können, um sich verschiedene Programmteile anzusehen. Wenn Sie z. B. das Unterprogramm bei Zeile 200 sehen wollen, geben Sie einfach ein:

```
LIST 200
```

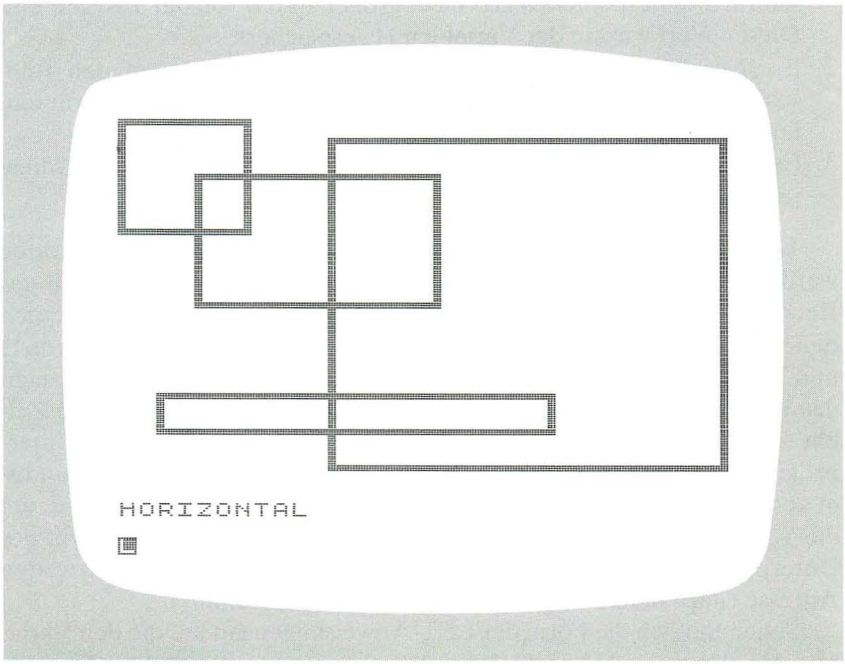


Abbildung 3.9: Ergebnisse des Rechteck-Programmes

In den beiden ersten Programmzeilen werden die Variablen HRI und VRI definiert:

```
10 LET HRI = 1
20 LET VRI = 1
```

Die Namen dieser Variablen stehen für „horizontale Richtung" bzw. „vertikale Richtung". Während des Programmlaufes enthalten diese Variablen entweder den Wert 1 oder den Wert −1, woraus dann die Richtung, in der die Rechtecke gezeichnet werden, abgeleitet wird. Wir werden gleich sehen, wie das geschieht.

Die Zeilen 30 und 60 enthalten jeweils einen Aufruf des Unterprogramms bei Zeile 200:

```
GOSUB 200
```

Erinnern wir uns, was dieses Unterprogramm tut:

1. Es schreibt die Eingabe-Aufforderung HORIZONTAL auf den Bildschirm und liest einen Wert (nicht größer als 63) von der Tastatur ein. Dieser Wert wird in der Variablen H gespeichert.
2. Es schreibt die Eingabe-Aufforderung VERTIKAL auf den Bildschirm und liest einen Wert (nicht größer als 43) von der Tastatur ein. Dieser Wert wird in der Variablen V gespeichert.
3. Es setzt mit Hilfe der PLOT-Anweisung einen Bildpunkt an die Adresse H,V.

Dieses Unterprogramm muß für jedes Rechteck zweimal aufgerufen werden – um die beiden diagonalen Ecken festzulegen, durch die das Rechteck definiert wird. Bei jedem Aufruf des Unterprogrammes werden neue Werte in den Variablen H und V gespeichert. Die alten Werte von H und V sind verloren. Um aber das Rechteck zeichnen zu können, braucht das Programm auch diese beiden Eckpunkte. Deshalb wurden die Zeilen 40, 50, 70 und 80 in das Programm aufgenommen. Nach jedem Unterprogramm-Aufruf speichert das Programm die neuen Werte von H und V in anderen Variablen, um sie zu „retten". So dienen H1 und V1 dazu, um die Werte der ersten Adresse zu speichern (nach dem ersten Aufruf des Unterprogramms), und H2 bzw. V2 speichern die Werte der zweiten Adresse (nach dem zweiten Aufruf des Unterprogramms).

Sehen Sie sich die Form der LET-Anweisungen gut an, die den Variablen H1, V1 sowie H2 und V2 ihre Werte zuweisen. Betrachten Sie z. B. die erste Anweisung:

```
40 LET H1 = H
```

Umschrieben lautet diese Anweisung: „Nimm den derzeitigen Wert der Variablen H und speichere ihn in der Variablen H1." Wir sehen hier zum ersten Mal einen Variablennamen in einer LET-Anweisung rechts vom Gleichheitszeichen. Es ist wichtig, daß Sie verstehen, was diese Anweisung bewirkt – und was sie nicht bewirkt. Sie definiert die Variable H1 und gibt ihr einen Wert. Aber sie verändert in keiner Weise die Variable H. Nachdem diese LET-Anweisung ausgeführt wurde, hat die Variable H weiterhin denselben Wert wie bisher.

Nachdem die beiden Adressen festgelegt sind, muß der Computer entscheiden, in welcher Richtung ein Rechteck gezeichnet werden soll. Diese Entscheidung wird in den Zeilen 90 und 100 getroffen. Denken Sie daran, daß HRI und VRI anfangs auf den Wert 1 gesetzt worden sind. Wie wir sehen werden, bedeutet dieser Wert, daß das Rechteck von links nach rechts und von unten nach oben gezeichnet wird. Diese Richtungen werden in den Zeilen 90 und 100 gegebenenfalls geändert. Wenn der zweite horizontale Wert – H2 – kleiner ist als der erste – H1 – (also links vom ersten liegt), soll das Rechteck von rechts nach links gezeichnet werden. Dies wird dadurch angezeigt, daß der Wert von HRI auf –1 gesetzt wird:

```
90 IF H2 < H1 THEN LET HRI = -HRI
```

Beachten Sie die LET-Anweisung hinter dem Wort THEN. Die Variable HRI erscheint auf beiden Seiten des Gleichheitszeichens. Diese LET-Anweisung bedeutet „Nimm den negativen Wert der Variablen HRI und speichere diesen neuen Wert wieder in HRI." Mit anderen Worten: da der ursprüngliche Wert von HRI 1 war, ist der neue Wert jetzt –1. Der ursprüngliche Wert geht natürlich verloren.

Als nächstes stehen im Programm zwei FOR-Schleifen. Die erste Schleife (Zeile 110 bis 140) zeichnet die obere und die untere Linie des Rechtecks, und die zweite Schleife (Zeile 150 bis 180) zeichnet die Seitenteile. Wenn Sie den Aufbau dieser Schleifen untersuchen, wird Ihnen die Bedeutung der Richtungsvariablen HRI und VRI völlig klar werden.

Jede Schleife enthält zwei PLOT-Anweisungen. Wir haben bereits gesehen, wie man mit zwei PLOT-Anweisungen in einer Schleife zwei getrennte Linien zeichnen kann, die in unserem Fall parallel verlaufen. In der ersten Schleife wird die Laufvariable vom Anfangswert H1 auf den Endwert H2 erhöht:

```
110 FOR I = H1 TO H2 STEP HRI
```

Die vertikalen Adressen der folgenden PLOT-Anweisungen sind mit V1 bzw. V2 festgelegt; als horizontale Adresse wird der Wert von I verwendet:

```
120 PLOT I,V1
130 PLOT I,V2
```

Deshalb werden die obere und untere Rechteckseite zuerst gezeichnet, wenn das Programm läuft.

In gleicher Weise zeichnet die zweite FOR-Schleife die Seitenteile des Rechtecks, wobei die horizontalen Adressen H1 und H2 fixiert sind und die vertikalen Adressen von V1 bis V2 variieren:

```
150 FOR I V1 TO V2 STEP VRI
160 PLOT H1,I
170 PLOT H2,I
180 NEXT I
```

Was bewirkt nun das Wort STEP in diesen beiden FOR-Anweisungen? Man kann sich das am einfachsten klarmachen, wenn man sich vorstellt, daß zwei Pixel-Adressen eingegeben werden, von denen die erste rechts oberhalb von der zweiten Adresse steht, z. B.:

```
Adresse Nr.1: 45,43
Adresse Nr.2: 25,23
```

Setzen Sie nun diese Adressen in die FOR-Anweisungen, und überlegen Sie, was passiert:

```
110 FOR I = 45 TO 25
150 FOR I = 43 TO 23
```

Es stellt sich heraus, daß diese FOR-Schleifen ihre Anweisungen nicht ein einziges Mal ausführen würden, da die erste Zahl in den FOR-Anweisungen größer als die zweite ist. Damit diese Schleifen laufen, müssen wir also die STEP-Anweisungen hinzufügen, durch die die Laufvariable I bei jedem Schleifendurchlauf vermindert (oder erhöht) wird. Aus diesem Grunde müssen die Richtungsvariablen HRI und VRI auf −1 gesetzt werden, wenn die erste Adresse größer als die zweite ist. Demnach würden die FOR-Schleifen für die obigen Adressen so aussehen:

```
110 FOR I = 45 TO 25 STEP −1
150 FOR I = 45 TO 23 STEP −1
```

Die STEP-Anweisung bewirkt also, daß die Laufvariable I bei jedem Schleifendurchlauf um 1 vermindert wird.

Falls diese Einzelheiten immer noch ein bißchen verwirrend erscheinen, lassen Sie das Programm wieder laufen, und denken Sie noch einmal über die beiden FOR-Schleifen nach, wenn Sie zusehen, wie die Rechtekke auf dem Bildschirm erscheinen. Zeichnen Sie mehrere Rechtecke und beobachten Sie die verschiedenen Richtungen, in die sie gezeichnet werden. Sie wissen jetzt, daß diese Richtungen durch die Reihenfolge der eingegebenen Pixel-Adressen bestimmt sind.

Die letzte Zeile dieses Hauptprogramms (als Gegensatz zum Unterprogramm) ist die Zeile 190:

```
190 GOTO 10
```

Durch diese Zeile wird eine Endlos-Schleife erzeugt. Sie können beliebig lange Rechtecke zeichnen. Das Programm selbst besitzt keine Möglichkeit um aufzuhören, so daß Sie also den STOP-Befehl von der Tastatur aus eingeben müssen, um den Programmlauf zu beenden.

Das letzte Programm in diesem Kapitel ist ebenfalls ein Grafik-Programm, das jedoch um einiges interessanter ist als das Rechteck-Programm. Wir werden es das „Zeichen-Programm" nennen, da Sie es benutzen können, um auf den Bildschirm zu zeichnen. Wir erklären zunächst, wie man das Programm benutzt, so daß Sie es nach dem Eintippen schon gleich ausprobieren können. (Das Programm ist lang, sehen Sie also zu, daß Sie es sofort nach dem Eingeben auf einer Kassette speichern.) Für dieses Programm wird die Speichererweiterung benötigt.

Sie finden aber auch eine ausführliche Erklärung, die Zeile für Zeile die einzelnen Programmschritte verdeutlicht. Das Programm soll dazu dienen, sämtliche BASIC-Anweisungen, die Sie bisher kennengelernt haben, zu vertiefen und zu erweitern. Die Methoden und Techniken, die in diesem Programm vorgestellt werden, sind schon etwas fortgeschrittener als das, was Sie bisher gesehen haben. Vielleicht werden Sie einige Zeit benötigen, um das ganze Programm vollständig zu verstehen. Aber lassen Sie sich ruhig Zeit: Sie können sich durchaus mit dem Programm vergnügen, auch wenn Sie die interne Logik noch nicht ganz verstanden haben.

Das Zeichen-Programm

Den Programm-Ausdruck finden Sie in den Abbildungen 3.10, 3.11 und 3.12. Wie Sie sehen, ist das Programm viel zu lang, um auf einmal auf dem Bildschirm angezeigt zu werden. Wenn Sie es in den Computer ein-

geben und der Bildschirm sich mit Anweisungen gefüllt hat, werden Sie feststellen, daß der Computer immer einen neuen Ausschnitt des Programms auf dem Bildschirm ausgibt, der bis zu der jeweils letzten Zeile reicht. Dadurch wird die Eingabe beträchtlich verlangsamt, vor allem, wenn Sie ein so langes Programm eingeben. Damit dieser Vorgang beschleunigt wird, können Sie den Computer in die schnelle Betriebsart umschalten. Geben Sie einfach das Schlüsselwort FAST als direkte Anweisung ein. Anschließend werden Sie feststellen, daß der Computer nun viel weniger Zeit braucht, um den entsprechenden Programmausschnitt auf den Bildschirm zu bringen. Wenn sie das Programm eingegeben haben, schalten Sie den Computer – bevor Sie das Programm laufen lassen – wieder in die langsame Betriebsart, indem Sie das Schlüsselwort SLOW direkt eingeben.

Überprüfen Sie alle Programmzeilen, um sicher zu gehen, daß Sie alles richtig eingetippt haben. Dann geben Sie das RUN-Kommando, um das Programm laufen zu lassen. Abbildung 3.13 zeigt, wie der Bildschirm aussieht, wenn Sie das Programm gestartet haben. In der Mitte des Bild-

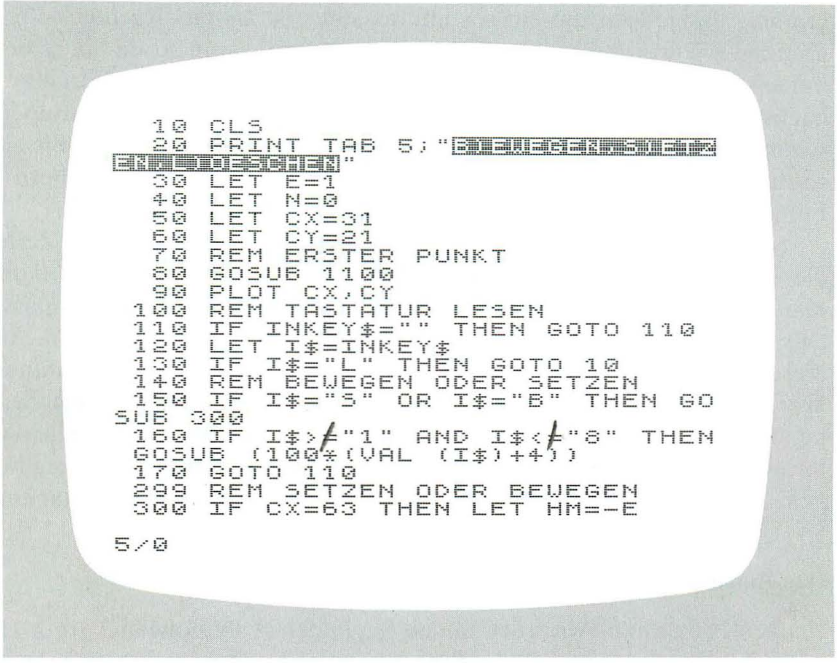
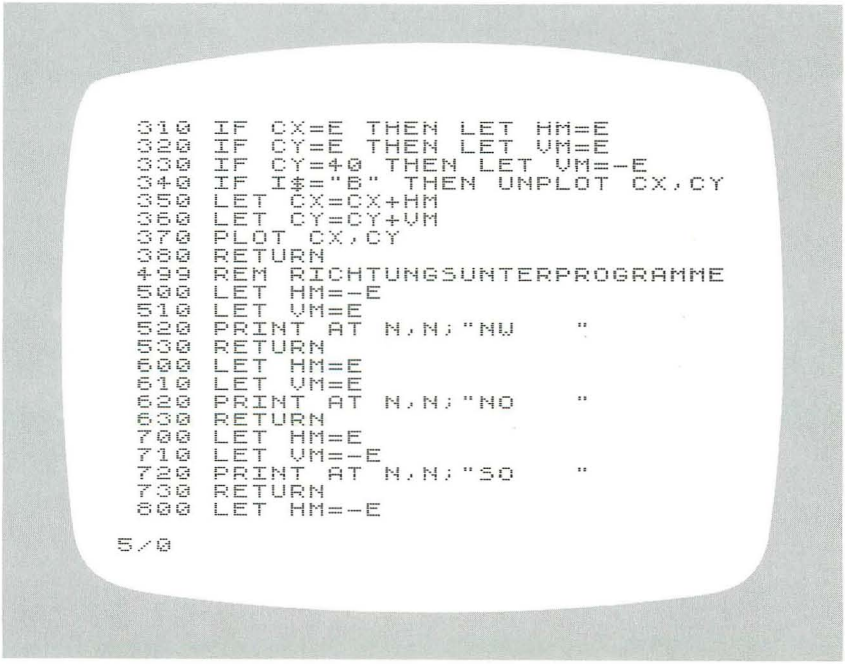


Abbildung 3.10: Das Zeichen-Programm

schirms befindet sich ein Bildpunkt, und im oberen Teil sehen Sie einige kurze Anweisungen. Sie können den Bildpunkt an eine beliebige Stelle auf dem Bildschirm bewegen, in insgesamt acht verschiedene Richtungen: nach oben („auf"), nach unten („ab"), nach links oder rechts, sowie in die diagonalen Richtungen, die (nach den Himmelsrichtungen) mit NW, NO, SO bzw. SW bezeichnet werden. Die augenblickliche Richtung wird immer in der oberen linken Bildschirmecke angegeben. (Zu Beginn des Programms ist die Richtung „AUF".) Sie können die Richtung ändern, indem Sie eine der Ziffern-Tasten von 1 bis 8 drücken. Den diagonalen Richtungen sind die Tasten 1 bis 4 wie folgt zugeordnet:

1: NW
2: NO
3: SO
4: SW

Die übrigen Richtungen entsprechen den Pfeilrichtungen auf den Tasten 5 bis 8:



```
310 IF CX=E THEN LET HM=E
320 IF CY=E THEN LET VM=E
330 IF CY=40 THEN LET VM=-E
340 IF I$="B" THEN UNPLOT CX,CY
350 LET CX=CX+HM
360 LET CY=CY+VM
370 PLOT CX,CY
380 RETURN
499 REM RICHTUNGSUNTERPROGRAMME
500 LET HM=-E
510 LET VM=E
520 PRINT AT N,N;"NW    "
530 RETURN
600 LET HM=E
610 LET VM=E
620 PRINT AT N,N;"NO    "
630 RETURN
700 LET HM=E
710 LET VM=-E
720 PRINT AT N,N;"SO    "
730 RETURN
800 LET HM=-E

5/0
```

Abbildung 3.11: Das Zeichen-Programm (Forts.)

5: links
6: ab
7: auf
8: rechts

Probieren Sie diese acht Tasten alle einmal aus. Bei jedem Tastendruck ändert sich das Wort in der oberen linken Bildschirmecke.

Nachdem Sie die Richtung, in der Sie sich bewegen wollen, gewählt haben, können Sie den Bildpunkt bewegen, indem Sie eine von zwei Tasten drücken. Die B-Taste bewegt den Punkt einfach. Drücken Sie diese Taste einmal, um den Punkt in die angegebene Richtung zu bewegen, oder halten Sie die Taste fest, um ihn weiterzubewegen.

Die S-Taste (für „setzen") hinterläßt eine Spur von Punkten hinter dem ursprünglichen Punkt. Auch diese Taste können Sie entweder nur einmal drücken oder aber auch festhalten, um eine längere Linie zu ziehen. Experimentieren Sie nun ein wenig mit diesen Tasten.

Wenn Sie einen Bildpunkt wieder löschen wollen, bewegen Sie sich einfach wieder über ihn zurück. Drücken Sie die entsprechende Richtungstaste und anschließend die B-Taste.

```
  810 LET VM=-E
  820 PRINT AT N,N;"SW    "
  830 RETURN
  900 LET HM=-E
  910 LET VM=N
  920 PRINT AT N,N;"LINKS"
  930 RETURN
 1000 LET HM=N
 1010 LET VM=-E
 1020 PRINT AT N,N;"AB    "
 1030 RETURN
 1100 LET HM=N
 1110 LET VM=E
 1120 PRINT AT N,N;"AUF   "
 1130 RETURN
 1200 LET HM=E
 1210 LET VM=N
 1220 PRINT AT N,N;"RECHTS"
 1230 RETURN


0/0
```

Abbildung 3.12: Das Zeichen-Programm (Forts.)

Wenn Sie mit einem neuen Bild anfangen wollen, drücken Sie die L-Taste, um den Bildschirm zu löschen.

Wenn der bewegliche Punkt den Bildschirmrand erreicht, wird er automatisch auf den Bildschirm zurückgeschickt. Wenn er sich gerade in einer diagonalen Richtung bewegt, wird er mit einem Winkel von 90 Grad abgelenkt. Bewegt er sich nach oben, unten, rechts oder links, wird er in die entgegengesetzte Richtung geschickt.

Sie werden feststellen, daß Ihre Bilder immer besser werden, je länger Sie mit dem Programm spielen. Sie können damit alles mögliche auf den Bildschirm zeichnen. Die größte Einschränkung liegt natürlich in der Auflösung, die von der Größe der Bildpunkte abhängig ist. Abbildung 3.14 zeigt ein Bild, das mit diesem Programm gezeichnet wurde.

Das Innenleben des Programms

Drücken Sie die BREAK-Taste, um das Programm abzubrechen, und anschließend NEW LINE, um das Programm auf dem Bildschirm aufzulisten. Das Programm besteht aus einem Hauptprogramm und verschie-

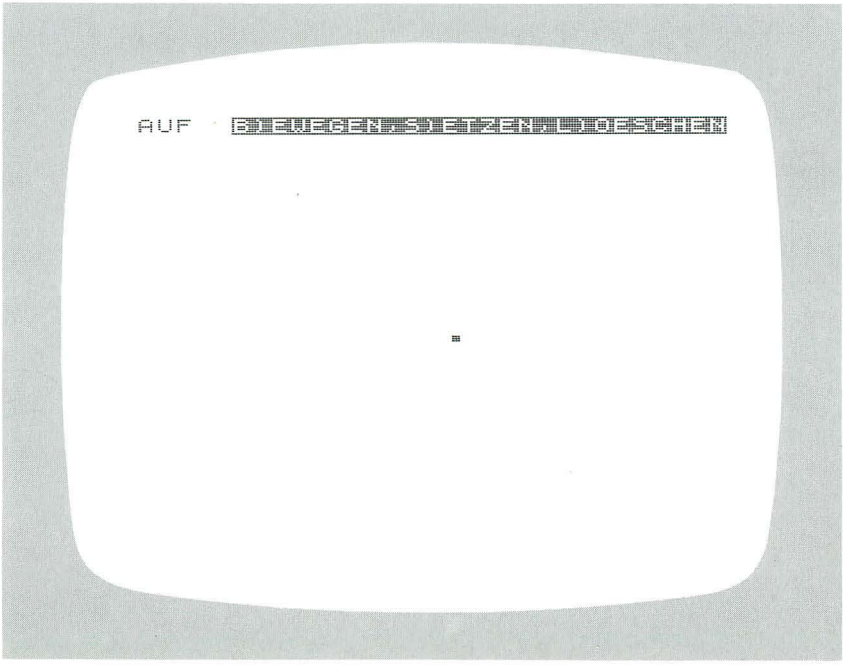


Abbildung 3.13: Start des Zeichen-Programms

denen Unterprogrammen. Wir sehen uns zunächst an, wie das Hauptprogramm mit den Unterprogrammen zusammenarbeitet. Danach betrachten wir einige Programmzeilen im einzelnen.

Dieses Programm demonstriert die enorme Leistungsfähigkeit von Unterprogrammen. Die Hauptaufgaben dieses Programms werden in eigenen Unterprogrammen abgewickelt. Dadurch ist das Programm leicht zu verstehen und einfach zu korrigieren oder zu ändern. Wenn Sie wissen, wozu die verschiedenen Unterprogramme dienen, wissen Sie genau, wo Sie nachschauen müssen, wenn Sie etwas an dem Programm ändern wollen.

Das Programm ist, kurz gesagt, folgendermaßen aufgebaut. Acht kurze Unterprogramme am Ende des Programms sind für das Ändern der Richtungen zuständig. Jedesmal wenn Sie eine Taste zur Richtungsänderung drücken, wird eines von diesen Unterprogrammen aufgerufen. Diese Unterprogramme kontrollieren die Werte zweier Variablen, die mit HM (für „horizontale Bewegung") und VM (für „vertikale Bewegung") bezeichnet sind. Diese beiden Variablen bestimmen dann die Richtung, in die sich der Bildpunkt bewegt, wenn Sie die P- oder M-Taste drücken.

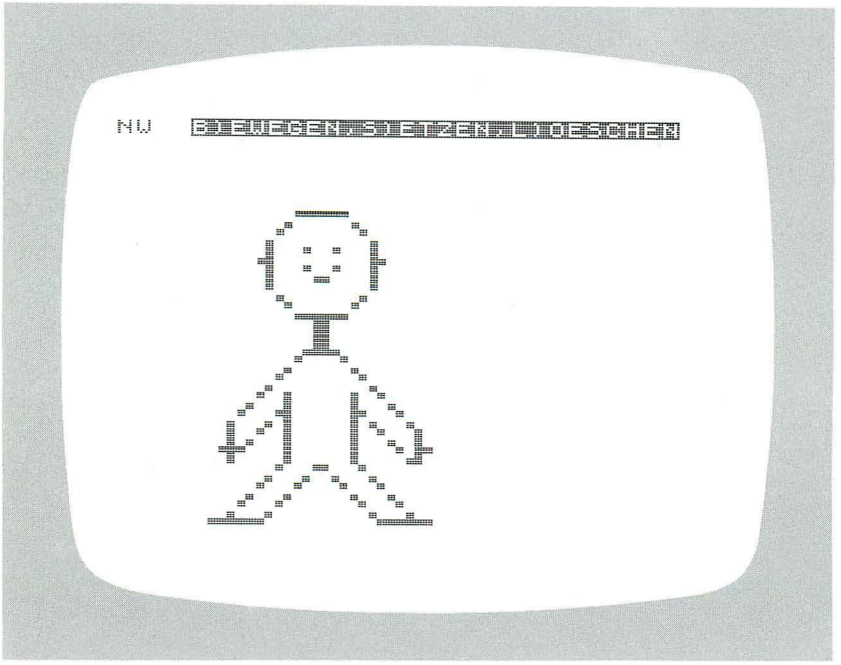


Abbildung 3.14: Ein Zeichen-Beispiel

Die Variablen HM und VM haben immer einen dieser drei Werte: 1, 0 oder −1. Außerdem wird in diesen Richtungs-Unterprogrammen auch noch die aktuelle Richtung in der oberen linken Bildschirmecke ausgegeben. Jedes dieser Unterprogramme ist nur vier Zeilen lang und beginnt mit einer Zeilennummer, die ein Vielfaches von 100 ist (500, 600, 700 ...). Dies ist für die aufrufende GOSUB-Anweisung von großer Bedeutung, wie wir gleich sehen werden.

Das Unterprogramm, das den Bildpunkt bewegt, befindet sich in Zeile 300. Es ist länger und komplizierter als die Unterprogramme für die Richtungen. Zuerst wird überprüft, ob die augenblickliche Punkt-Position am Bildrand liegt. Wenn ja, muß das Unterprogramm eine der Variablen HM oder VM verändern, um die Bewegungsrichtung umzudrehen. Dann muß das Unterprogramm feststellen, ob sich der Punkt lediglich bewegt, oder ob er eine Spur hinter sich herzieht (d. h. ob die B- oder die S-Taste gedrückt wird). Wenn der Punkt sich nur bewegt, muß die vorige Position (mit dem UNPLOT-Kommando) gelöscht werden. Zum Schluß bestimmt dieses Unterprogramm die neuen Punkt-Adressen, indem HM und VM zu der augenblicklichen Adresse hinzuaddiert werden, und setzt (mit PLOT) einen neuen Punkt auf den Bildschirm.

In Zeile 110 bis 170 erfolgt die Eingabe von der Tastatur und der Aufruf des entsprechenden Unterprogramms in Abhängigkeit von der gedrückten Taste. In diesem Programm wird nicht wie in bisherigen Programmen das Schlüsselwort INPUT verwendet, um eine Eingabe von der Tastatur zu machen. Stattdessen benutzen wir die mit INKEY$ bezeichnete Programmier-Funktion (unterhalb der B-Taste). Den Unterschied zwischen INKEY$ und INPUT besprechen wir bei der Erklärung der entsprechenden Zeilen.

Damit bleiben noch die Zeilen 10 bis 90 übrig, die eine Art „Einleitung" (oder „Initialisierung") darstellen. Zeile 10 löscht den Bildschirm (mit CLS). Zeile 20 gibt die invers dargestellte Anweisungszeile im oberen Bildschirmteil aus, wobei die TAB-Funktion benutzt wird, um den Text etwas einzurücken. In den Zeilen 30 bis 60 werden mehrere wichtige Variablen definiert (oder initialisiert, wie wir auch sagen könnten). Mit diesen Variablen soll jetzt die nähere Erläuterung des Programms beginnen.

In den Zeilen 30 und 40 werden die Variablen E (für „Eins") und N (für „Null") definiert:

```
30 LET E = 1
40 LET N = 0
```

Die Begründung, warum wir diese Variablen brauchen, ist eine ziemlich lange Geschichte. Sie hat eigentlich mehr mit den Eigenheiten Ihres Computers als mit diesem Programm zu tun. Wie Sie wissen, hat Ihr Computer eine begrenzte Speicherkapazität. Wenn Sie längere Programme schreiben (insbesondere solche mit vielen Ausgaben auf dem Bildschirm, wie z. B. das vorliegende), werden Sie mit dem Speicherplatz möglichst sparsam umgehen wollen. Vergessen Sie nicht, daß der Computer seinen Speicher für mehrere verschiedene Aufgaben braucht, wenn Sie ein Programm ablaufen lassen. Ein Teil des Speichers wird für die Programmanweisung benötigt; in einem anderen Bereich muß die Information verwaltet werden, die auf den Bildschirm geschickt wird; und noch mehr Speicher wird für andere, nicht sichtbare Aufgaben gebraucht. Offensichtlich besteht also ein Weg, um Speicher zu sparen, darin, daß für das Programm selbst möglichst wenig Platz benötigt wird. Dies ist der Grund für die Einführung der Variablen E und N in unserem Zeichen-Programm.

Der Computer braucht zur Speicherung eines Buchstabens viel weniger Platz als zur Speicherung einer Zahl. Der Computer verfügt über ein besonderes System, um Zahlen zu speichern; dieses System (es wird als Gleitkomma-Darstellung bezeichnet) ist so aufgebaut, daß es Zahlen möglichst genau bei möglichst wenig Speicherbedarf speichern kann. Die Genauigkeit spielt natürlich dann eine große Rolle, wenn Sie den Computer für Berechnungen verwenden wollen. Dieses System braucht für eine Zahl bis zu siebenmal mehr Speicher als für einen Buchstaben. Wenn also in Ihren Programmen eine Zahl auftaucht, sollten Sie sich überlegen, ob es möglich ist, diese Zahl durch einen Buchstaben zu ersetzen.

In unserem Zeichen-Programm werden die Zahlen 0 und 1 mehrfach benutzt. Wenn wir das wissen, können wir einen hübschen kleinen Trick anwenden, durch den wir eine Menge Speicherplatz sparen. Wir können die Werte 0 und 1 zwei Variablen zuweisen und dann im weiteren Programm diese Variablen anstatt der Zahlen verwenden. Genau dies geschieht in den Zeilen 30 und 40. Wenn Sie das Programm lesen, sollten Sie deshalb immer, wenn Sie die Variablen N und E sehen, daran denken, daß es sich um die „konstanten" Werte 0 und 1 handelt. Dadurch ist es für Sie vielleicht etwas schwieriger, das Programm zu entziffern, doch macht es sich am Ende bezahlt. Immerhin ist Ihr Gedächtnis erheblich größer als das des Computers.

Zur Erklärung der beiden anderen Variablen, die am Anfang des Programms definiert werden, brauchen wir nicht annähernd so viel Zeit. Die Variablen CX und CY stellen die augenblickliche horizontale bzw. vertikale Position dar. (Wenn Sie mathematisch orientiert sind, wissen Sie, daß die Buchstaben X und Y üblicherweise als Variablennamen für die

horizontale und vertikale Richtung in einem Koordinatensystem benutzt werden.) In den Zeilen 50 und 60 werden diese Variablen initialisiert:

```
50 LET CX = 31
60 LET CY = 21
```

Dies ist die Adresse des Bildpunktes, den Sie zu Beginn des Programms etwa in der Mitte des Bildschirmes sehen.

Zeile 70 ist die erste von mehreren REM-Zeilen, die in diesem Programm stehen. Das Schlüsselwort REM steht für „remark", d. h. Bemerkung, Kommentar. (Es liegt über der E-Taste.) REM-Zeilen werden vom Computer nicht ausgewertet. Während des Programmlaufes werden sie vom Computer ignoriert. Die Nützlichkeit dieser Anweisung besteht darin, daß Sie auf diese Weise Ihr Programm innerhalb der Programmzeilen „dokumentieren" können. Nach dem Schlüsselwort REM können Sie einen beliebigen Text schreiben. Im allgemeinen werden Sie in kurzen Worten beschreiben, was das Programm an bestimmten Stellen tut. Wenn Sie sich dann später wieder einen Ausdruck Ihres Programms ansehen, werden Ihnen die REM-Zeilen dabei helfen, Ihr Programm wieder zu verstehen.

Leider benötigen REM-Zeilen auch Speicherplatz. Deshalb sollten Sie sparsam damit umgehen. Im übrigen können Sie durch das Kürzen oder Löschen von Kommentar-Zeilen natürlich auch wieder etwas Platz für andere Programmzeilen schaffen, wenn der Speicherplatz knapp wird. Im allgemeinen aber werden Sie feststellen, daß ein guter Kommentar (in Form einer REM-Zeile) an der richtigen Stelle viel Zeit sparen hilft, wenn Sie ein Programm verändern, korrigieren oder einfach verstehen wollen.

Der Initialisierungsteil des Programms besteht aus noch zwei weiteren Zeilen. Zeile 80 initialisiert die Bewegungs-Richtung, indem sie das Richtungs-Unterprogramm aufruft:

```
80 GOSUB 1100
```

Das Unterprogramm bei Zeile 1100 ist dasjenige, das die Richtung nach oben („AUF") festsetzt.

In Zeile 90 wird dann der erste Bildpunkt ausgegeben:

```
90 PLOT CX,CY
```

Dies ist der Punkt in der Mitte des Bildschirms, wenn das Programm gestartet wird.

Der Initialisierungsteil des Programms wird einmal zu Beginn des Programms ausgeführt, sowie jedesmal, wenn Sie die L-Taste drücken, um den Bildschirm zu löschen. In Zeile 130 wird die GOTO-Anweisung benutzt, um das Programm neu zu initialisieren, wenn das eingegebene Zeichen der Buchstabe „L" war:

```
130 IF I$ = "L" THEN GOTO 10
```

In Zeile 110 erfolgt die Tastatur-Eingabe über die INKEY$-Funktion. Bei jeder Ausführung von INKEY$ „durchkämmt" der Computer die Tastatur, um zu sehen, ob eine Taste gedrückt wurde. Wenn sie genau eine Taste drücken, liefert INKEY$ das Zeichen, das durch diese Taste dargestellt wird – einen Buchstaben von „A" bis „Z" oder eine Ziffer von „0" bis „9". Beachten Sie, daß INKEY$ die Ziffern als Zeichen und nicht als Zahlen behandelt. Dies ist ein bedeutender Unterschied. Es besagt, daß der Computer diese Ziffern nicht als Zahlen speichert, mit denen gerechnet werden kann (d.h. nicht in Gleitkomma-Darstellung), sondern einfach als Zeichen, wie beliebige andere Zeichen auch.

Da wir wollen, daß der Computer fortwährend darauf achtet, ob eine Taste gedrückt wurde, müssen wir die Anweisung INKEY$ in eine Schleife einbauen:

```
110 IF INKEY$ = "" THEN GOTO 110
```

Die GOTO-Anweisung nach dem THEN schickt die Programmkontrolle direkt wieder zum Beginn derselben Zeile! Diese Zeile bedeutet – umschrieben – folgendes: „Wenn INKEY$ kein Zeichen liefert (das bedeutet, daß keine Taste gedrückt wurde), geh zum Zeilenanfang zurück und führe INKEY$ noch einmal aus." Solange Sie keine Taste drücken, führt der Computer immer wieder die Zeile 110 aus. Sobald Sie jedoch eine Taste drücken, trifft die Bedingung

```
INKEY$ = ""
```

nicht mehr zu, und das Programm wird mit der folgenden Zeile fortgesetzt. Da Sie das Programm schon laufen gelassen haben, wissen Sie, daß sich INKEY$ völlig anders als INPUT verhält. Bei INPUT erscheint jedes Zeichen, das Sie eingeben, in der unteren linken Bildschirmecke. INPUT wartet auch darauf, daß Sie die NEW LINE-Taste drücken, und speichert die eingegebenen Daten erst dann in einer Variablen. INKEY$ hingegen tut nichts dergleichen: es liest einfach ein einziges Zeichen von der Tastatur und übergibt dies Zeichen dem Programm.

Wenn INKEY$ ein Zeichen gefunden hat, wird es in Zeile 120 in der String-Variablen I$ gespeichert:

```
120 LET I$ = INKEY$
```

Der Wert von I$ wird dann in den Zeilen 130, 150 und 160 mit drei verschiedenen IF-Anweisungen überprüft, um zu entscheiden, was als nächstes passieren soll. Wir haben schon gesehen, daß in Zeile 130 der Computer wieder zurück zur Zeile 10 geschickt wird, wo der Bildschirm für ein neues Bild gelöscht wird.

In den Zeilen 150 und 160 steht eine Form der Bedingung nach IF, die wir bisher noch nicht besprochen haben – darin werden die Wörter AND (und) und OR (oder) benutzt. Mit diesen beiden Wörtern können Sie mehrere Gleichungen oder Ungleichungen in einer einzigen IF-Anweisung überprüfen.

In Zeile 150, in der gefragt wird, ob eine der Tasten S oder B gedrückt wurde, steht das Wort OR:

```
150 IF I$ = "S" OR I$ = "B" THEN GOSUB 300
```

Mit anderen Worten: Wenn die Variable I$ das Zeichen S oder das Zeichen B enthält, dann wird in Zeile 150 das Unterprogramm bei Zeile 300 aufgerufen. Eine Aussagenverknüpfung, die das Wort OR enthält, ist dann wahr, wenn eine oder auch beide Teilaussagen wahr sind.

Zeile 160 ist die schwierigste Zeile im ganzen Programm. Darin wird überprüft, ob eine der Richtungstasten – von „1" bis „8" – gedrückt wurde:

```
160 IF I$ >= "1" AND I$ < "8" THEN
                    GOSUB (100 * (VAL (I$) + 4))
```

Eine zusammengesetzte Aussage, die das Wort AND enthält, ist nur dann wahr, wenn beide Teilaussagen wahr sind. Deshalb wird in dieser Zeile nur dann ein Unterprogramm aufgerufen, wenn der Wert von I$ eins der Zeichen von „1" bis „8" ist. Die GOSUB-Anweisung in Zeile 160 unterscheidet sich von anderen GOSUBs darin, daß die Zeilennummer des aufzurufenden Unterprogramms berechnet wird. Wir werden sehen, daß mit dieser einen Zeile jedes der acht Unterprogramme aufgerufen werden kann, eine sehr leistungsfähige Eigenschaft der GOSUB-Anweisung also. Der Schlüssel zu dieser Fähigkeit liegt in dem Ausdruck, der nach dem Wort GOSUB steht:

```
(100 * (VAL (I$) + 4))
```

Sie werden diesen Ausdruck im einzelnen besser verstehen, nachdem Sie das Kapitel 4 gelesen haben. (Tatsächlich könnten Sie an Zeile 160 für einige Zeit zu beißen haben. Machen Sie sich nichts daraus, wenn Sie nicht gleich alles auf einmal verstehen.) Kurz gesagt erzeugt dieser Ausdruck ein Vielfaches von 100, das zwischen 500 und 1200 liegt. Wie Sie sich erinnern, sind dies die Zeilennummern der Unterprogramme für die Richtungen. Die Berechnung der Zeilennummern erfolgt in drei Schritten:

1. Die Funktion VAL macht aus dem Wert von I$ (d.i. ein Zeichen zwischen „1" und „8") eine Zahl (zwischen 1 und 8).
2. Zu dieser Zahl wird 4 hinzugezählt, so daß eine Zahl zwischen 5 und 12 herauskommt.
3. Das Ergebnis von Schritt 2 wird mit 100 multipliziert, so daß sich eine Zahl zwischen 500 und 1200 ergibt.

Das wichtigste an der ganzen Sache ist, sich klarzumachen, daß Sie eine einzige GOSUB-Anweisung schreiben können, die dann eins von verschiedenen Unterprogrammen aufrufen wird. Diese Anweisung kann auf Grund der Berechnungen mit einer Variablen gewissermaßen „entscheiden", welches Unterprogramm aufgerufen werden soll.

Zusammenfassend gesagt: von den Zeilen 130, 150 und 160 aus kann ein Unterprogramm aufgerufen werden, dies jedoch nur dann, wenn bestimmte Tasten gedrückt worden sind, die für dieses Programm eine Bedeutung haben, also: L, B, S oder 1 bis 8. Sollten Sie irgendeine andere Taste drücken, wird keins dieser Unterprogramme aufgerufen, und das Programm wird einfach mit Zeile 170 fortgesetzt:

```
170 GOTO 110
```

Das heißt, irgendwie kommt das Programm also immer wieder zu der Eingabe mit INKEY$ in Zeile 110 zurück.

Jetzt wollen wir uns die Unterprogramme kurz ansehen, und zwar zuerst die, die für die Richtungen zuständig sind. Diese kurzen Unterprogramme haben alle den gleichen Aufbau. In jedem stehen zwei LET-Anweisungen, die den Variablen HM und VM Werte zuweisen. In dem Unterprogramm, das den Bildpunkt bewegt, werden die Werte dieser beiden Variablen dazu benutzt, um eine neue Adresse aus der bisherigen zu berechnen, wobei die gegenwärtige Richtung berücksichtigt wird. Damit Sie sehen, wie das funktioniert, sehen Sie sich das Unterprogramm für die „NW"-Richtung bei Zeile 500 an:

```
500 LET HM = -E
510 LET VM = E
```

Sie sehen: zieht man vom horizontalen Teil der Punkt-Adresse 1 ab, bewegt sich der Punkt nach links (d. h. „West"). Entsprechend bewegt er sich nach oben („Nord"), wenn man zu dem vertikalen Teil der Adresse 1 hinzufügt.

Zuletzt gibt jedes dieser Richtungs-Unterprogramme die neue Richtung in der oberen linken Bildschirmecke (an der Adresse 0,0) aus:

```
520 PRINT AT N,N; "NW"
```

In den Zeilen 350 und 360 des Unterprogramms, das den Bildpunkt bewegt, wird die augenblickliche Adresse (CX,CY) durch die Werte HM und VM geändert:

```
350 LET CX = CX + HM
360 LET CY = CY + VM
```

Zeile 350 kann so umschrieben werden: „Addiere den Wert von HM zu dem Wert von CX und speichere das Ergebnis in der Variablen CX." Der alte Wert von CX ist verloren, ebenso wie der alte Wert von CY in Zeile 360.

Nachdem die neue Adresse berechnet worden ist, kann der Punkt auf den Bildschirm gesetzt werden:

```
370 PLOT CX,CY
```

Das Unterprogramm bei Zeile 300 erledigt noch zwei weitere Dinge, die Sie sich näher ansehen sollten: In Zeile 300 bis 330 wird überprüft, ob die derzeitige Adresse sich an einem der vier Bildschirmränder befindet. Wenn dies der Fall ist, muß der Wert von HM oder VM so geändert werden, daß sich der Punkt wieder in den Bildschirmbereich bewegt. Wenn z B. die horizontale Adresse den Wert 63 erreicht (die rechte Bildschirmseite), so muß HM, die Variable für die horizontale Bewegung, auf −1 gesetzt werden, um den Punkt nach links zu bewegen:

```
300 IF CX = 63 THEN LET HM = -E
```

In Zeile 340 wird dann geprüft, ob der Punkt an der gegenwärtigen Position gelöscht werden soll, bevor ein neuer Punkt gesetzt wird. Wenn Sie

die Taste „B" gedrückt haben, löscht UNPLOT den Punkt an der Stelle CX,CY aus:

```
340 IF I$ = "B" THEN UNPLOT CX,CY
```

Haben Sie dagegen die Taste „S" gedrückt, wird das UNPLOT-Kommando nicht ausgeführt, und der momentane Punkt bleibt zusammen mit dem neuen auf dem Bildschirm.

Wenn Sie dies Programm sorgfältig studieren, werden Sie einige BASIC-Anweisungen sicherlich besser verstehen lernen, wie z. B. LET, IF und GOSUB. Dazu werden in diesem Programm mehrere neue Wörter eingeführt, wie REM, INKEY$, TAB, AND, OR und VAL. Falls Sie soviel nach einmaligem Lesen noch nicht behalten können, sollten Sie später noch einmal auf dieses Programm zurückkommen, vielleicht dann, wenn Sie die übrigen Kapitel gelesen haben.

Zusammenfassung

Sie haben in diesem Kapitel viele Dinge kennengelernt. Die folgende Übersicht soll Ihnen helfen, sich an die bisher vorgekommenen BASIC-Anweisungen zu erinnern:

- Variablen definieren und ihnen Werte zuweisen – LET, INPUT
- Informationen über die Tastatur eingeben – INPUT, INKEY$
- Informationen auf dem Bildschirm anzeigen – PRINT, AT, TAB, CLS
- Grafische Zeichen auf dem Bildschirm darstellen – PLOT, UNPLOT
- Anweisungen wiederholen, Schleifen bilden – FOR, TO, STEP, NEXT, GOTO
- Die Reihenfolge der Programmausführung steuern – GOTO, GOSUB, RETURN
- Entscheidungen treffen – IF, AND, OR, THEN
- Programmdokumentation – REM

Kapitel 4

Wir können nicht nur bis drei zählen . . .

Einführung

In diesem Kapitel lernen Sie, wie Sie mit Ihrem Computer Berechnungen mit Zahlen durchführen können. Das Kapitel beginnt mit der Beschreibung der arithmetischen Operationen. Es folgt ein kurzer Überblick über einige der gebräuchlicheren arithmetischen Funktionen, die in verschiedenen Beispielen vorgeführt werden.

Außerdem finden Sie zwei wichtige und nützliche Programme in diesem Kapitel. Das erste Programm verwandelt Ihren Computer in einen „Super-Rechner" mit vielen Operationen, Funktionen und Datenspeichern. Mit dem zweiten Programm, dem längsten in diesem Buch, können Sie Balken-Diagramme für beliebige numerische Daten erstellen. Dieses Programm läuft wiederum nur, wenn Ihr ZX81 mit einem RAM-PACK ausgerüstet ist. Sie werden lernen, wie man diese beiden Programme benutzt, und Sie werden Gelegenheit dazu bekommen, Ihre BASIC-Kenntnisse zu erweitern, indem Sie den Aufbau und die logische Struktur der Programme studieren. Im zweiten der beiden Programme werden Sie vor allem auch *Felder* kennenlernen. Ein Feld ist eine wesentliche Strukturierungsmöglichkeit in BASIC-Programmen, die Ihnen erlaubt, viele Daten unter einem einzigen Variablen-Namen zu speichern.

Berechnungen mit Ihrem Computer

Ihr Computer kann folgende arithmetische Operationen ausführen: Addition, Subtraktion, Multiplikation, Division und Exponentiation (oder Potenzierung). Die Symbole für diese Operationen sind:

+ Addition
− Subtraktion
* Multiplikation
/ Division
** Exponentiation

Eine Möglichkeit, den Computer rechnen zu lassen, ist durch die PRINT-Anweisung gegeben. Der Computer berechnet den Ausdruck in der PRINT-Anweisung und zeigt dann die Ergebnisse auf dem Bildschirm an. Damit Sie sehen, wie das geht, geben Sie die folgenden Anweisungen direkt ein:

```
PRINT 15.193 + 6.5
PRINT 1235 - 872
PRINT 18 * 97
PRINT 183 / 5
PRINT 9 ** 3
```

Jedesmal wenn Sie eine dieser Anweisungen eingeben, sehen Sie das Ergebnis der Berechnung fast sofort oben auf dem Bildschirm angezeigt. Die letzte dieser fünf PRINT-Anweisungen ist ein Beispiel für eine Exponentiation oder Potenzierung. Der Ausdruck „9 ** 3" bedeutet 9 hoch 3 oder 9 Kubik. Das Ergebnis ist, wie Sie sehen, gleich 9 x 9 x 9 oder 729. (Das „**"-Symbol wird durch Umschalten der H-Taste erreicht.)

Sie können auch mehrere arithmetische Operationen in einer einzigen Anweisung schreiben. Z. B. ergibt die folgende Anweisung die Summe zweier Produkte:

```
PRINT 9 * 8 + 5 * 3
```

Die auf dem Bildschirm angezeigte Zahl ist 87, die Summe von 72 und 15. Beachten Sie, daß der Computer zuerst die beiden Multiplikationen durchgeführt und dann erst die Summe aus den beiden Produkten gebildet hat. Dies ist ein wesentlicher Punkt. Wenn der Computer mehrere Operationen ausführt, geschieht dies in einer bestimmten Reihenfolge. Sie müssen sich über diese Reihenfolge immer im klaren sein, da Sie sonst riskieren, fehlerhafte Ergebnisse zu erhalten.

Die Reihenfolge der Operationen ist die folgende:

1. Potenzierung (Exponentiation), von links nach rechts;
2. Multiplikation und Division, von links nach rechts;
3. Addition und Subtraktion, von links nach rechts.

Geben Sie zum Ausprobieren die folgende Zeile ein:

```
PRINT 1 + 2 ** 3 * 4 - 5
```

Hätten Sie das richtige Ergebnis – 28 – vorhergesagt? Dieser arithmetische Ausdruck wird folgendermaßen berechnet: Zuerst die Potenzie-

rung, 2 ** 3, die 8 ergibt; dann das Produkt 8 * 4, woraus sich 32 ergibt; schließlich die Addition von 1 und die Subtraktion von 5, wonach man das Ergebnis 28 erhält.

Vermutlich wären Sie sehr erstaunt gewesen, wenn Sie die Zeile eingegeben und erwartet hätten, daß der Computer die Berechnungen von links nach rechts durchführen würde. In dem Fall wäre nämlich das Ergebnis 103. (Sie können nachrechnen: 1 plus 2 ist 3; 3 hoch 3 ist 27; 27 mal 4 ist 108; 108 minus 5 ist 103.) Es versteht sich, daß Sie bei der Formulierung von arithmetischen Berechnungen immer sorgfältig überlegen müssen, in welcher Reihenfolge der Computer die Berechnungen ausführt.

Häufig jedoch möchten Sie Berechnungen durchführen, die nicht so gut in diese schematische Reihenfolge passen. Glücklicherweise gibt es in BASIC die Möglichkeit, die normale Reihenfolge zu umgehen und eine eigene Reihenfolge zu wählen. Dazu müssen Sie die arithmetischen Ausdrücke in Klammern setzen.

Ist ein Teil eines arithmetischen Ausdrucks in Klammern eingeschlossen, so wird dieser Teil zuerst berechnet. Geben Sie z. B. nacheinander diese beiden Anweisungen ein:

```
PRINT 1 + 2 * 3
PRINT (1 + 2) * 3
```

In der ersten Anweisung wird die Multiplikation zuerst ausgeführt und dann die Addition, so daß sich 7 ergibt. In der zweiten Anweisung berechnet der Computer zuerst die in Klammern eingeschlossene Summe und multipliziert diese Summe dann mit 3; das Ergebnis ist 9.

Sie können auch Anweisungen schreiben, die verschiedene Klammerebenen haben. (Klammern innerhalb von Klammern bezeichnet man auch als *geschachtelte* Klammern.) Der Computer beginnt mit der *innersten* Klammer und arbeitet sich dann bis zur äußersten Klammerebene vor. Die folgende Anweisung würde also wegen der Klammern von links nach rechts abgearbeitet:

```
PRINT (((1 + 2) ** 3) * 4) - 5
```

Das Ergebnis ist, wie schon erwähnt, 103. Achten Sie insbesondere bei verschachtelten Klammern darauf, daß zu jeder öffnenden Klammer „(eine schließende Klammer)" gehört. Wenn Sie aus Versehen einen Ausdruck schreiben, der nicht genau so viele öffnende wie schließende Klammern enthält, wird der Computer die Berechnung nicht ausführen.

Wir haben schon gesehen, daß arithmetische Ausdrücke Variablen

enthalten können. Ein solcher Ausdruck ist gültig, sofern die darin enthaltenen Variablen bereits definiert sind. Dies gilt für die LET-Anweisungen, die PRINT-Anweisungen und die meisten anderen BASIC-Anweisungen, in denen numerische Werte vorkommen. Damit Sie das selbst überprüfen können, geben Sie die folgende Anweisungsfolge direkt ein:

```
LET A = 1
LET B = A + 2
LET C = B ** 3
LET D = C * 4
LET E = D - 5
PRINT A,B,C,D,E
```

Jede LET-Anweisung in dieser Folge verwendet den Wert der Variablen aus der vorigen Anweisung. Das PRINT-Kommando zeigt dann die Werte sämtlicher Variablen an. Beachten Sie, daß die letzte Variable – E – den Wert 103 hat, das Ergebnis aus dem Ausdruck $(((1 + 2) ** 3) * 4) - 5$. Die anderen Variablen enthalten die Zwischenergebnisse.

Die obige PRINT-Anweisung verdeutlicht ein weiteres Merkmal von BASIC, das wir noch nicht besprochen haben: den Gebrauch von Kommas zur Trennung von Elementen bei der Ausgabe. Wenn ein *Komma* in einer PRINT-Anweisung steht, erfolgt die nächste Ausgabe im Abstand einer halben Bildschirmbreite. Deshalb erzeugt die angegebene PRINT-Anweisung folgende Ausgabe:

```
1                    3
27                   108
103
```

Der Unterschied zwischen einem Komma und einem Semikolon in einer PRINT-Anweisung ist sehr bedeutend. Wie Sie sich erinnern, bewirkt das Semikolon, daß der Computer die Elemente alle nebeneinander ausgibt.

Sie sollten noch einen weiteren Punkt im Zusammenhang mit Berechnungen berücksichtigen: Die Division durch Null ist für den Computer nicht definiert. Eine Anweisung, die zu einem solchen Ergebnis führt, wird mit einer Fehlermeldung abgebrochen. Wenn Sie das wissen, werden Sie sicherlich nicht absichtlich Anweisungen wie diese schreiben:

```
LET E = 1/0
```

Doch kann es oft problematisch sein, wenn Sie Anweisungen mit Divisionen schreiben, die im Nenner Variablen enthalten, wie z. B.:

```
LET E = 1/D
```

Falls die Variable D jemals den Wert Null hat, wenn diese Anweisung ausgeführt wird, bringt der Computer eine Fehlermeldung. Dies kann besonders problematisch sein, wenn Sie vorher nicht wissen, welche Werte D annehmen wird. Eine Möglichkeit, diesem Problem aus dem Weg zu gehen, besteht darin, der LET-Anweisung eine IF-Anweisung vorausgehen zu lassen:

```
IF D < > 0 THEN LET E = 1/D
```

Dadurch wird sichergestellt, daß die LET-Anweisung nur dann ausgeführt wird, wenn der Wert der Variablen D ungleich Null ist.

Sie haben gesehen, daß die Tastatur Ihres Computers mehrere nützliche mathematische Funktionen bietet, die Sie nach dem Umschalten in die Funktions-Betriebsart eingeben können. Jetzt wollen wir diese Funktionen in Berechnungen ausprobieren.

Arithmetische Funktionen

Wir werden die folgenden Funktionen betrachten:

- ABS (Absolutwert; liegt unterhalb der G-Taste);
- INT (Integralwert; liegt unterhalb der R-Taste);
- SGN (Signumfunktion; liegt unterhalb der F-Taste);
- SQR (Quadratwurzel; liegt unterhalb der H-Taste).

Die Wirkung dieser Funktionen könnten Sie durchaus auch selbständig durch Probieren herausfinden, selbst wenn Sie nicht mathematisch veranlagt sind. Der Computer bietet auch noch andere, speziellere Funktionen. Dazu gehören die trigonometrischen Funktionen (SIN, COS, TAN, ARCSIN, ARCCOS, ARCTAN) und die Exponential-Funktionen (EXP, LN). Wenn Sie sich für diese Funktionen interessieren, können Sie Näheres darüber im Anhang A lesen.

Die Absolutwert-Funktion, ABS, liefert den positiven Wert einer beliebigen Zahl. Mit anderen Worten, wenn eine Zahl negativ ist, macht ABS sie positiv. Wenn eine Zahl *positiv* ist, wird sie durch ABS nicht verändert. Geben Sie die beiden folgenden Anweisungen ein, damit Sie sehen, wie ABS funktioniert:

```
PRINT ABS −45
PRINT ABS 45
```

Beide Anweisungen liefern das Ergebnis 45. Beachten Sie, daß das Minus-Zeichen in der ersten Anweisung einfach das Vorzeichen von 45 ist und *keine* Operation zwischen zwei verschiedenen Zahlen.

Die INT-Funktion schneidet den Bruchteil einer Dezimalzahl ab. Probieren Sie z. B. diese Anweisungen:

```
PRINT INT 19.34
PRINT INT 19.81
```

In beiden Fällen lautet das Ergebnis 19. Beachten Sie, daß INT nicht rundet. Es *schneidet* einfach den Teil *ab*, der rechts vom Dezimalpunkt (der Computer kennt kein Dezimalkomma!) steht, so daß sich die nächst niedrigere ganze Zahl ergibt. Im weiteren Verlauf dieses Kapitels werden wir sehen, wie man mit Hilfe der INT-Funktion eine Rundungs-Funktion erstellen kann.

Die Funktion SGN liefert als Resultat immer einen von drei möglichen Werten, der vom Vorzeichen der angegebenen Zahl abhängt:

```
−1 wenn die Zahl negativ ist
 0 wenn die Zahl gleich Null ist
 1 wenn die Zahl positiv ist.
```

Die Anweisung

```
PRINT SGN −45, SGN 0, SGN 45
```

ergibt somit −1, 0 und 1.

Die SQR-Funktion berechnet die Quadratwurzel aus einer positiven Zahl. Z. B. erhält man aus

```
PRINT SQR 81
```

als Ergebnis 9. Wenn Sie eine negative Zahl nach SQR eingeben, meldet der Computer einen Fehler, da die Quadratwurzel aus einer negativen Zahl nicht definiert ist.

Funktionen können in Anweisungen vorkommen, die mehrere arithmetische Operationen enthalten, so daß Sie auch in dem Fall wieder auf die Reihenfolge der Operationen achten müssen. Funktionen werden vor

allen anderen fünf arithmetischen Operationen berechnet (sofern Sie keine Klammern setzen). Geben Sie z. B. diese Anweisung ein:

```
PRINT 10 * SQR 81 + 19
```

Das Ergebnis ist 109. Der Computer berechnet zuerst die Quadratwurzel aus 81, also 9. Dann multipliziert er 9 mit 10, erhält also 90. Dazu addiert er schließlich noch 19, woraus sich dann 109 ergibt. Vergleichen Sie dies mit der folgenden Anweisung:

```
PRINT 10 * SQR (81 + 19)
```

Das Ergebnis dieser Anweisung ist 100. Der Computer führt zuerst die Addition innerhalb der Klammern aus, berechnet daraus die Quadratwurzel und führt dann die Multiplikation aus.

Funktionen können außerdem auch auf die Ergebnisse anderer Funktionen angewendet werden. Mit anderen Worten, Sie können Funktionen auch in komplexen Ausdrücken benutzen, wie z. B.:

```
SQR INT 9.73
```

Dieser Ausdruck besagt: „Berechne die Quadratwurzel des ganzzahligen Teils von 9.73." Wenn Sie diesen Ausdruck in einer PRINT-Anweisung angeben, sehen Sie, daß das Ergebnis 3 ist.

Im nächsten Abschnitt werden wir mehr über den Gebrauch der INPUT-Anweisung in Berechnungen lernen und ein kurzes Programm laufen lassen, um damit zu experimentieren.

Berechnungen mit der Input-Anweisung

Geben Sie zunächst NEW ein, um den Programmspeicher zu löschen; dann die drei folgenden Programm-Zeilen:

```
10 INPUT V
20 PRINT V
30 GOTO 10
```

Auf den ersten Blick sieht dieses Programm nicht besonders nützlich aus. Zeile 10 liest einen numerischen Wert von der Tastatur und speichert ihn in der Variablen V. Zeile 20 zeigt den Wert von V auf dem Bildschirm an, und in Zeile 30 geht das Ganze wieder von vorne los. Wenn Sie das

Programm laufen lassen, wartet der Computer auf die Eingabe einer Zahl, die er dann auf dem Bildschirm anzeigt.

Wie wir gleich sehen werden, kann der Computer jedoch tatsächlich auch arithmetische Ausdrücke während des Programmlaufs über die Tastatur einlesen. Dies ist eine einzigartige und sehr nützliche Eigenschaft des ZX81-BASIC. Anstatt nur Zahlen einzutippen, können Sie beliebige zulässige arithmetische Ausdrücke über die Tastatur eingeben, mit allen fünf arithmetischen Operationen sowie sämtlichen numerischen Funktionen. Der Computer führt die angegebenen Berechnungen sofort aus und speichert dann das *Ergebnis* der Berechnungen in der INPUT-Variablen, in diesem Fall also in V.

Lassen Sie das Programm jetzt laufen und probieren Sie es aus. Geben Sie zunächst einfach eine Zahl ein, sagen wir 10. Die Zahl erscheint oben auf dem Bildschirm. Sie sehen, daß das Programm läuft.

Als nächstes geben Sie einen Ausdruck ein, z. B. diesen:

```
10 * SQR 49 + 16/8
```

Oben auf dem Bildschirm erscheint die Zahl 72. Der Computer hat den Ausdruck berechnet und das Ergebnis – 72 – in der Variablen V gespeichert.

Geben Sie nun diesen Ausdruck ein:

```
V/9
```

Der Computer antwortet, indem er die Zahl 8 anzeigt. Der Ausdruck V/9 bedeutet: „Dividiere den augenblicklichen Wert der Variablen V durch 9." Da V aus der vorigen Berechnung den Wert 72 hatte, ist das Ergebnis der Division 8.

Fassen wir zusammen: Wir haben gesehen, daß der Computer in einer INPUT-Anweisung einen zulässigen arithmetischen Ausdruck annimmt. Dieser Ausdruck kann sogar Variablen enthalten, sofern diese vorher definiert worden sind.

Jetzt sind wir soweit, daß wir uns das Programm ansehen können, das aus unserem Computer einen Super-Rechner macht.

Wir bauen einen Super-Rechner

Das Super-Rechner-Programm ist in Abbildung 4.1 aufgelistet. Geben Sie es sorgfältig in den Computer ein und lassen Sie es laufen. Sie sehen dann die Meldung:

Super-Rechner

in inverser Darstellung sowie den [L]-Cursor in Anführungszeichen; d. h. der Computer wartet auf eine String-Eingabe.

Sie können mit diesem Programm beliebige Berechnungen anstellen. Der eingegebene Ausdruck kann enthalten:

- die fünf arithmetischen Operationen;
- beliebige mathematische Funktionen von der Tastatur;
- komplexe Ausdrücke mit geschachtelten Klammern und Funktionen;
- zwei Variablen: M1 und M2, die für Speicher-Funktionen zur Verfügung stehen. (Wir kommen darauf gleich zurück.)

Mit anderen Worten: Sie können jeden beliebigen Ausdruck eingeben, den der Computer berechnen kann. Das Programm läuft in einer Endlos-Schleife, so daß Sie so viel rechnen können, wie Sie wollen.

Probieren wir's einmal aus. Geben Sie den folgenden Ausdruck über die Tastatur ein:

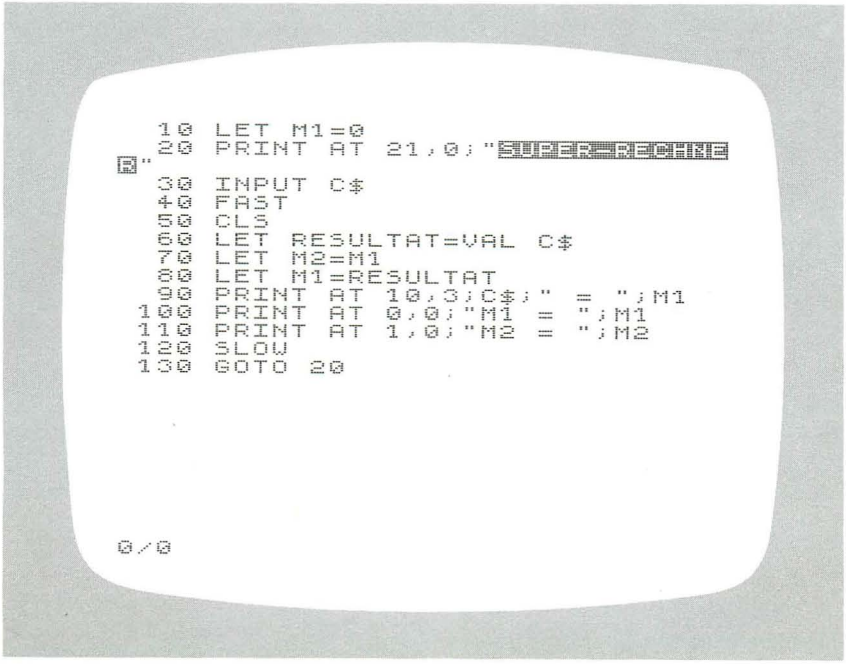


Abbildung 4.1: Das Super-Rechner-Programm

```
1 + INT (SQR (2 * 3) *4)
```

Drücken Sie die NEW LINE-Taste und sehen Sie zu, was passiert. Der Bildschirm wird kurz gelöscht und sollte anschließend so aussehen, wie in Abbildung 4.2 angegeben. Schauen Sie sich den Bildschirm gut an, damit Sie nichts übersehen.

Zunächst einmal steht der von Ihnen eingegebene Ausdruck in der Mitte des Bildschirms. Das ist durchaus wichtig: Wie oft haben Sie eine Reihe von Berechnungen auf einem Taschenrechner durchgeführt und sich anschließend verunsichert gefragt, ob Sie überhaupt alle Tasten richtig gedrückt hatten? Eingabefehler sind wahrscheinlich die häufigste Ursache für fehlerhafte Ergebnisse, wenn wir mit Maschinen rechnen. Deshalb teilt Ihnen das Programm erst einmal mit – sogar noch vor dem Ergebnis –, welchen Ausdruck Sie dem Computer eingegeben haben. Sie können sich dieses „Echo" ansehen, wenn es Ihnen paßt, um sich zu vergewissern, daß tatsächlich die von Ihnen gewünschten Berechnungen durchgeführt worden sind. Nach diesem Echo gibt Ihnen der Computer die Antwort auf das Problem. In diesem Fall lautet die Antwort 10.

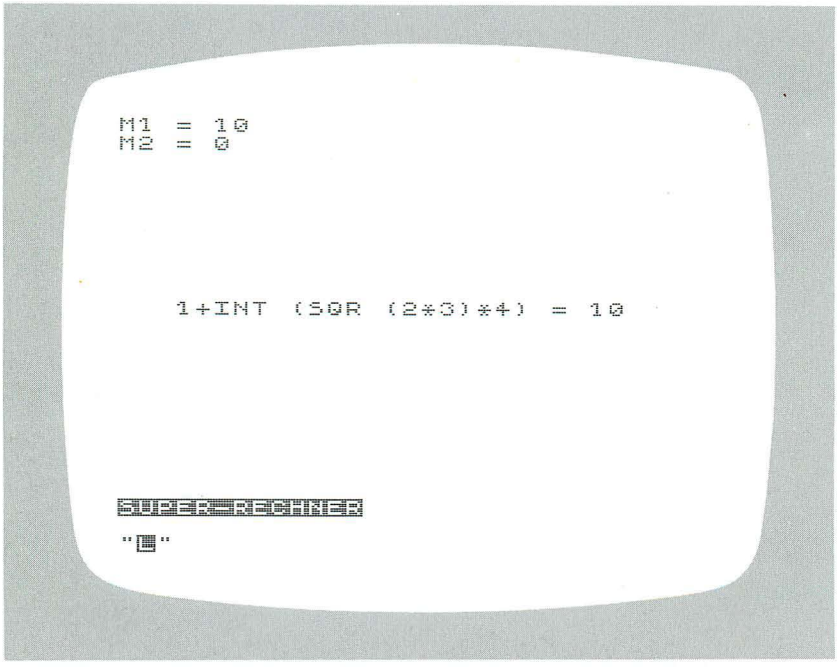


Abbildung 4.2: Eine Antwort des Super-Rechner-Programms

Zusätzlich hält das Programm zwei Speicher-Variablen bereit, die mit M1 und M2 bezeichnet sind. Nach der ersten Berechnung werden ihre augenblicklichen Werte immer in der oberen linken Bildschirmecke angezeigt. M1 ist das Ergebnis der zuletzt durchgeführten Berechnung, M2 das der vorigen. Wie Sie auf dem Bildschirm sehen können, sind die derzeitigen Werte von M1 und M2:

```
M1 = 10
M2 = 0
```

Der Wert von M1 ist das Ergebnis der Berechnung, die im Augenblick in der Bildschirmmitte angezeigt wird. Da dies die erste Berechnung nach dem Programmstart war, ist M2 noch gleich Null. (Zu Beginn des Programms haben sowohl M1 als auch M2 den Wert Null.)

Geben Sie nun einen zweiten Ausdruck zur Berechnung ein:

```
(18 ** 2)/5
```

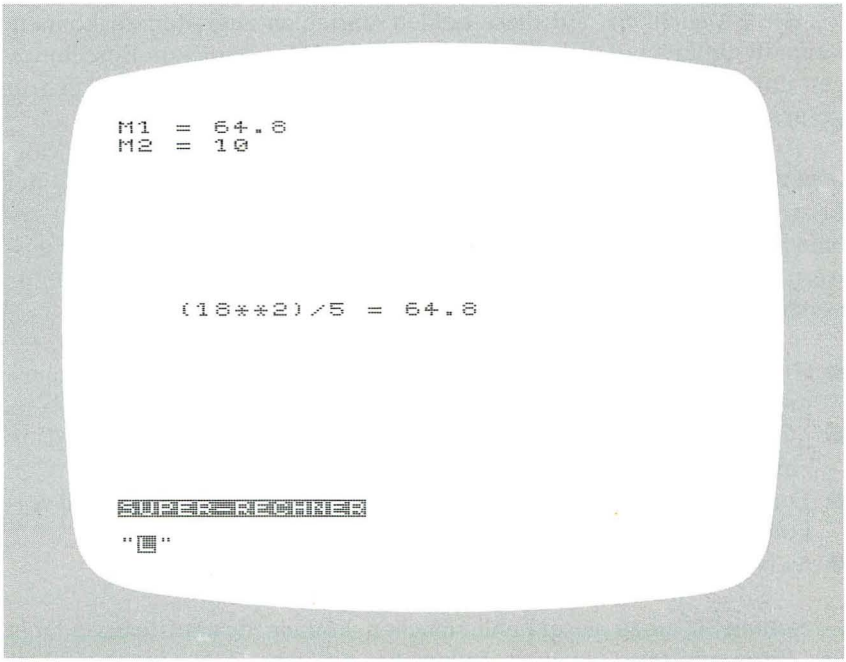


Abbildung 4.3: Eine zweite Berechnung

Die Antwort lautet, wie Abbildung 4.3 zeigt, 64.8. Sehen Sie sich an, was mit den beiden Speicher-Variablen passiert ist:

```
M1 = 64.8
M2 = 10
```

Das letzte Ergebnis steht nun in M1, das Resultat der vorigen Berechnungen in M2. Diese beiden Speicher-Variablen können nach Belieben in den Ausdrücken verwendet werden, die Sie eingeben. Geben Sie z. B. als nächstes den folgenden Ausdruck ein:

```
M1/M2
```

Das bedeutet: „Dividiere den augenblicklichen Wert von M1 durch den augenblicklichen Wert von M2." Der Computer gibt darauf die folgende Antwort:

```
M1/M2 = 6.48
```

Die Möglichkeit, auf diese beiden Variablen zugreifen zu können, kann oft äußerst nützlich sein, wenn Sie eine lange Serie von Berechnungen durchführen müssen. Der momentane Wert wird bei jedem Schritt gespeichert und angezeigt.

Arbeiten Sie mit diesem Programm so lange, bis Sie alle seine Möglichkeiten verstehen. Vielleicht finden Sie sehr schnell heraus, daß der Programmlauf unterbrochen und eine Fehlermeldung angezeigt wird, sobald Sie einen Ausdruck eingegeben haben, den der Computer nicht auswerten kann. Dies kann verschiedene Gründe haben, wie die folgende Aufzählung zeigt:

- Eingabe eines Ausdrucks, der eine ungerade Anzahl von Klammern enthält.
- Verwendung einer undefinierten Variablen in einem Ausdruck (d. h. eine andere Variable als M1 und M2).
- Durchführung einer Berechnung, in der eine Division durch Null vorkommt.
- Verwendung einer Funktion, die keine Zahl als Ergebnis hat.

Sollten Sie einen dieser Fehler machen, können Sie natürlich wieder in das Programm zurückgehen, indem Sie einfach RUN eingeben. Der Nachteil dabei liegt jedoch darin, daß Sie die derzeitigen Werte der bei-

den Variablen M1 und M2 verlieren. (Wie Sie wissen, löscht das Kommando RUN erst alle Variablen, bevor es mit der Programmausführung beginnt.) Eine bessere Möglichkeit bietet in diesem Fall die direkte Anweisung:

```
GOTO 100
```

Wenn Sie dies eingeben, werden die Werte von M1 und M2 wieder unverändert auf dem Bildschirm angezeigt, und das Programm ist bereit für die nächste Berechnung. Wenn wir uns den Programmaufbau näher betrachten, sehen wir, wie diese GOTO-Anweisung funktioniert. Grundsätzlich gilt: wir können jedes Programm wieder neu starten, indem wir eine GOTO-Anweisung anstatt RUN verwenden. Dabei bleiben die Werte aller Variablen im Speicher des Computers erhalten.

Lassen Sie uns nun einen Blick auf das Programm selbst werfen. Sie können das Programm auf dem Bildschirm auflisten, indem Sie das STOP-Kommando eingeben, um das Programm abzubrechen, und anschließend die Taste NEW LINE drücken. Oder Sie sehen sich wieder die Abbildung 4.1 an, in der das Programm ausgedruckt ist.

Das Super-Rechner-Programm aus der Nähe betrachtet

Die ersten drei Zeilen des Programms sind völlig unkompliziert. Zeile 10 setzt die Speicher-Variable M1 auf Null, Zeile 20 zeigt den Programmtitel auf dem Bildschirm an, und Zeile 30 liest die Eingabe von der Tastatur. Beachten Sie, daß die Eingabe in der String-Variablen C$ gespeichert wird:

```
30 INPUT C$
```

Der Ausdruck ist bisher noch *nicht* ausgewertet, wie das bei einer numerischen Variablen der Fall wäre. Im Moment stellt C$ für den Computer eine ganz gewöhnliche String-Variable dar.

Zeile 40 schaltet den Computer in die schnelle Betriebsart:

```
40 FAST
```

Wir haben das FAST-Kommando zwar schon in einigen anderen Programmen in diesem Buch benutzt, jedoch nie genau geklärt, was es eigentlich tut. In der schnellen Betriebsart schickt der Computer keine Informationen auf den Bildschirm, während er Berechnungen durchführt.

Das Ergebnis ist, daß die Berechnungen sehr viel schneller beendet werden. Wenn Sie ein Programm laufen lassen, bei dem der Bildschirm zeitweilig gelöscht wird, können Sie davon ausgehen, daß der Computer sich in der schnellen Betriebsart befindet. Sie erinnern sich, daß in dem Super-Rechner-Programm der Bildschirm direkt nach der Eingabe eines Ausdrucks gelöscht wird. Der Computer gebraucht seine Fähigkeiten nur für das Rechnen, so daß die Angelegenheit schnell erledigt ist.

Löschen Sie einmal probeweise die Zeile 40, und lassen Sie das Programm dann laufen, um den Unterschied festzustellen. Sie werden sehen, daß sich die Ausführungszeit nun beträchtlich verlängert hat. (Fügen Sie die Zeile 40 wieder ein, bevor Sie weitermachen.)

Zeile 50 löscht den Bildschirm mit dem CLS-Kommando. In Zeile 60 wird die eigentliche Arbeit dieses Programms geleistet:

```
60 LET RESULT = VAL C$
```

Wie wir in Kapitel 3 gesehen haben, ergibt die VAL-Funktion das numerische Gegenstück eines Strings. Damit VAL überhaupt funktionieren kann, muß der String natürlich aus Zeichen bestehen, die in Zahlen umgewandelt werden können. Aus diesem Grunde muß der Ausdruck, den Sie ins Programm eingeben, ein zulässiger numerischer Ausdruck sein. Wenn Sie einen Tippfehler machen und einen unzulässigen Ausdruck eingeben, erhalten Sie die Fehlermeldung:

```
C/60
```

Die Zahl 60 bedeutet, daß das Programm in Zeile 60 beendet wurde. Wie Sie in Anhang B nachlesen können, bedeutet der Fehlercode C, daß Sie der VAL-Funktion einen unzulässigen String-Ausdruck übergeben haben.

Wenn alles gut geht, speichert die LET-Anweisung in Zeile 60 das Ergebnis des numerischen Ausdrucks in der numerischen Variablen RESULT AT. In den Zeilen 70 und 80 wird den Speicher-Variablen M1 und M2 ihr neuer Wert zugewiesen. Zuerst wird der Wert aus der vorigen Berechnung, der zur Zeit in M1 steht, in M2 gespeichert:

```
70 LET M2 = M1
```

Dann wird das Ergebnis der letzten Berechnung in M1 gespeichert:

```
80 LET M1 = RESULT AT
```

Die drei nächsten Zeilen erledigen die Bildschirmausgabe. Zeile 90 zeigt den Originalausdruck (der noch immer in der String-Variablen C$ steht) und die (in M1 gespeicherte) Antwort:

```
90 PRINT AT 10,3; C$;" = "; M1
```

In den Zeilen 100 und 110 erfolgt die Anzeige der Werte der beiden Speicher-Variablen in der oberen linken Bildschirmecke:

```
100 PRINT AT 0,0; "M1 = "; M1
110 PRINT AT 1,0; "M2 = "; M2
```

Hier setzt das Programm also fort, wenn Sie mit der Anweisung

```
GOTO 100
```

das Programm nach einem Fehlerabbruch fortsetzen möchten.

In Zeile 120 schließlich schaltet der Computer wieder in die langsame Betriebsart, und in Zeile 130 wird mit dem Sprung zur Zeile 20 eine endlose Schleife gebildet.

Bevor wir zum letzten Thema dieses Kapitels kommen, lassen Sie das Super-Rechner-Programm noch einmal laufen, um ein letztes Experiment durchzuführen. Tippen Sie den Ausdruck

```
5 ** 20
```

ein, und drücken Sie die Taste NEW LINE. Sie haben sich wahrscheinlich schon gedacht, daß der Ausdruck „5 hoch 20" eine sehr große Zahl darstellt, aber vielleicht sind Sie von der Anzeige auf Ihrem Bildschirm doch überrascht. Die sieht nämlich so aus:

```
5 ** 20 = 9.5367432E + 13
```

Das Ergebnis ist in der sogenannten *wissenschaftlichen* oder Exponential-*Darstellung* formuliert. Für sehr große Zahlen, so wie diese, schaltet der Computer automatisch in diese Schreibweise um. Diese Schreibweise ist nicht so schwierig zu lesen, wie es Ihnen auf den ersten Blick vielleicht vorkommen mag, sofern Sie nicht ohnehin damit vertraut sind. Der Buchstabe E (an der rechten Seite des Ausdrucks) steht für Exponent, und der Ausdruck E + 13 bedeutet deshalb „10 hoch 13" oder

```
10,000,000,000,000
```

Deshalb lautet die ausgeschriebene Version des Exponential-Ausdrucks:

9.5367432 x 10,000,000,000,000

oder

95,367,432,000,000

Wenn Sie bedenken, daß der Computer mit Zahlen umgehen kann, die noch viel größer sind als diese, werden Sie vermutlich den Nutzen der wissenschaftlichen Notation allmählich verstehen. Es ist einfach eine abkürzende Schreibweise zur Darstellung großer Zahlen.

Falls Sie dieses Thema weiter verfolgen möchten, können Sie das Super-Rechner-Programm dazu benutzen, weitere Fragen, die Ihnen in den Sinn kommen, zu beantworten. Mit etwas Erfahrung können Sie viel über die Art und Weise lernen, wie der Computer mit Zahlen umgeht:

- Wie groß muß eine Zahl sein, damit der Computer in die Exponential-Darstellung umschaltet?
- Welches ist die größte Zahl, die der Computer darstellen kann?
- Wie genau ist die Darstellung sehr großer Zahlen? (Geben Sie z. B. 5**15 ein. Sie wissen, daß jede Potenz von 5 als letzte Ziffer eine 5 hat. Trifft dies für die vom Computer dargestellte Zahl zu?)

Wir machen Balken-Diagramme

Das nächste Programm, das wir uns ansehen, dient dazu, Balken-Diagramme auf dem Bildschirm darzustellen. Durch solche grafischen Darstellungen kann man eine Reihe von Zahlen bequem und sehr anschaulich miteinander vergleichen. Jede Zahl wird dabei durch einen Balken dargestellt. Je größer die Zahl, desto länger ist der Balken. Wenn Sie sich so eine Grafik anschauen, sehen Sie auf einen Blick die relativen Anteile der einzelnen dargestellten Kategorien. Deshalb bietet sich das Balken-Diagramm-Programm immer dann an, wenn Sie beliebige wichtige Daten darstellen wollen und es Ihnen mehr um die *relativen* Werte, als um die einzelnen Beträge geht.

Da in der Balken-Grafik Daten-Listen vorkommen, sollen Sie jetzt die in BASIC mögliche Datenstruktur kennenlernen, mit der man solche Listen handhaben kann. Diese Struktur wird als *Feld* bezeichnet. Bevor

wir zu dem Programm kommen, sehen wir uns diese Felder an und lernen, wie man sie definiert.

Felder

Sie wissen jetzt, daß eine einfache Variable jeweils genau einen Wert speichern kann. Wenn Sie der Variablen einen neuen Wert zuweisen – mit LET oder INPUT –, ist der alte Wert für immer verloren. Einfache Variablen sind dann angebracht, wenn Daten gespeichert werden müssen, die nichts Besonderes miteinander zu tun haben, abgesehen davon, daß sie alle in demselben Programm stehen.

Wenn Sie mit Ihrem Computer Daten verarbeiten, haben Sie es jedoch häufig mit Listen oder Tabellen von zusammengehörigen Daten zu tun, die Sie im Computer speichern wollen. Das können einige Dutzend oder auch Hunderte von Daten sein, so daß es äußerst lästig wäre, für jedes Datenelement eine eigene Variable zu definieren.

BASIC bietet besondere Datenstrukturen, sogenannte *Felder*, die hervorragend dazu geeignet sind, große Datenmengen zu speichern. Felder besitzen sogenannte Dimensionen. In einem eindimensionalen Feld kann man eine Datenliste speichern. Ein zweidimensionales Feld kann eine *Tabelle* speichern, in der die Daten in Zeilen und Spalten angeordnet sind. Auf dem ZX81 werden Sie wahrscheinlich selten Felder mit mehr als zwei Dimensionen gebrauchen, obwohl BASIC auch mehrdimensionale Felder zuläßt. Das Balken-Grafik-Programm benutzt eindimensionale Felder, deshalb beschränken wir unsere Diskussion auf das Speichern von *Datenlisten*.

Angenommen, Sie haben eine Liste mit 15 Zahlen, die Sie im Computer speichern wollen. Sie möchten ein Programm entwickeln, das drei Aufgaben bewältigt:

1. während des Programmlaufs Zahlen von der Tastatur liest;
2. mit den Zahlen Berechnungen durchführt;
3. die Zahlen mit den Ergebnissen der Berechnungen auf dem Bildschirm anzeigt.

Sie können nun ein Feld definieren, das alle 15 *Zahlen* gleichzeitig speichert. Nennen wir das Feld z. B. N, so sähe die Definition dieses Feldes so aus:

```
10 DIM N(15)
```

Das Schlüsselwort DIM (über der D-Taste) steht für Dimension. Eine

DIM-Anweisung enthält vier wesentliche Angaben über die Merkmale eines Feldes:

1. den Namen des Feldes,
2. den darin zu speichernden Datentyp (numerisch oder String),
3. die Zahl der Dimensionen,
4. die Zahl der Datenelemente in jeder Dimension. (Diese letzte Eigenschaft wird als Länge bezeichnet.)

Hier sind die entsprechenden Merkmale für das oben in Zeile 10 definierte Feld N:

1. der Name ist N,
2. es ist ein numerisches Feld (da der Name nicht mit einem $-Zeichen endet),
3. es hat eine Dimension (da in den Klammern nur eine Zahl angegeben ist),
4. die *Länge* des Feldes beträgt 15.

Wenn der Computer zu dieser Programmzeile kommt, reserviert er im Speicher automatisch Platz für 15 Zeilen unter dem Feldnamen N.

Jetzt brauchen Sie noch eine Möglichkeit, um die einzelnen Feldelemente ansprechen zu können. Nachdem Sie dem Feld Zahlen zugewiesen haben, wollen Sie z. B. auf das 7. Element der Liste zugreifen. Dazu müssen Sie jetzt dem Computer auf möglichst einfache Weise mitteilen, welches Element Sie interessiert.

Das Ansprechen von Feldelementen ist sehr einfach. Dazu geben Sie den Feldnamen sowie – in Klammern – die Nummer des betreffenden Elementes an. So heißt z. B. das 7. Feldelement:

```
N(7)
```

Entsprechend sind die ersten fünf Elemente:

```
N(1)
N(2)
N(3)
N(4)
N(5)
```

Die Zahl in den Klammern wird als *Index* des Feldes bezeichnet. Sie können z. B. folgende Anweisungen schreiben:

```
LET N(7) = 162
INPUT N(5)
PRINT N(2)
```

In der ersten Anweisung wird die Zahl 162 als 7. Element des Feldes N gespeichert. Die zweite Anweisung liest eine Zahl von der Tastatur, die als 5. Element von N gespeichert wird. Die dritte Anweisung zeigt den augenblicklichen Wert des zweiten Elementes von N auf dem Bildschirm an.

Zwischen den Klammern hinter dem Feldnamen darf auch ein Variablenname stehen, z. B.:

```
N(I)
```

In dem Fall ist die Wahl der Feldelemente vom Wert des Index I abhängig. Wenn I gleich 5 ist, bezieht sich N(I) auf das 5. Element von N. Sie können also FOR-Schleifen benutzen, in denen die Laufvariable der Schleife gleichzeitig als Index eines Feldes verwendet wird, z. B.:

```
10 DIM N(15)
20 FOR I = 1 TO 15
30 INPUT N(I)
40 NEXT I
```

In diesem Programm ist die Laufvariable I gleichzeitig der Index des Feldes N. Mit diesen einfachen Zeilen können Sie 15 Zahlen in den Computer eingeben, die alle unter dem Feldnamen N gespeichert werden. Sie sehen allmählich, was für wertvolle Instrumente wir in den Feldern und FOR-Schleifen besitzen.

Damit Sie diese Leistungsfähigkeit selbst beurteilen können, tippen Sie nun die oben stehenden Zeilen 10 bis 40 ein. Fügen Sie die folgenden drei Zeilen hinzu, die alle 15 Werte von N auf dem Bildschirm anzeigen:

```
50 FOR I = 1 TO 15
60 PRINT N(I)
70 NEXT I
```

Lassen Sie das Programm laufen. Geben Sie 15 beliebige Zahlen in den Computer ein. Anschließend werden diese sofort wieder auf dem Bildschirm ausgegeben. Denken Sie daran, daß alle 15 Zahlen im Computer gespeichert sind und für beliebige Anwendungen benutzt werden

können. Sämtliche Berechnungen, die für eine ganze Reihe von Elementen durchgeführt werden sollen, können in eine FOR/NEXT-Schleife eingebaut werden.

Sie können mit Ihrem Computer auch String-Felder definieren; allerdings ist dies etwas komplizierter als bei Zahlen. Wir betrachten wieder ein eindimensionales Feld. Bei der Dimensionierung müssen Sie nach der Angabe der Feldlänge eine weitere Zahl in der Klammer hinzufügen. Diese Zahl gibt die Länge der einzelnen Elemente des Stringfeldes an, also die Zahl der Zeichen eines Strings. Für alle Elemente eines Stringfeldes wird die gleiche Länge vorgegeben. Sehen wir uns ein Beispiel dazu an.

Angenommen, Sie haben eine Liste von 15 Wörtern, die Sie für eine weitere Verwendung im Computer speichern wollen. Das längste Wort ist acht Zeichen lang. Dann können Sie ein String-Feld für diese Liste wie folgt definieren:

```
10 DIM W$(15,8)
```

Der Name des String ist W$; er kann 15 Strings enthalten, von denen jeder maximal 8 Zeichen lang sein darf.

Sie können mit diesem Feld experimentieren, indem Sie das folgende kurze Programm eingeben:

```
10 DIM W$(15,8)
20 FOR I = 1 TO 15
30 INPUT W$(I)
40 PRINT W$(I);
50 NEXT I
```

Halten Sie zunächst einmal fest, daß sie die Elemente dieses String-Feldes genauso ansprechen können wie die eines numerischen Feldes. Das 7. Wort des Feldes W$ wird z. B. so angesprochen:

```
W$(7)
```

und das I. Wort ist:

```
W$(I)
```

Lassen Sie das Programm nun laufen, und achten Sie darauf, was es tut. Geben Sie zuerst ein Wort ein, das genau 8 Zeichen lang ist:

```
COMPUTER
```

Das Wort wird in W$(1) gespeichert und – wegen Zeile 40 – auch oben auf dem Bildschirm angezeigt. Geben Sie nun zwei Wörter ein, die in W$(2) bzw. W$(3) gespeichert werden sollen; wählen Sie diesmal Wörter, die weniger als 8 Zeichen haben:

```
BUCH
LADEN
```

Das erste Wort erscheint auf dem Bildschirm direkt hinter dem Wort COMPUTER. (Beachten Sie, daß Zeile 40 mit einem Semikolon aufhört, das den Zeilenvorschub verhindert.) Das Wort LADEN jedoch steht im Abstand von vier Leerzeichen hinter dem Wort BUCH. Sie sehen die folgende Zeile auf dem Bildschirm:

```
COMPUTERBUCH    LADEN
```

Woher kommen die vier Leerzeichen zwischen BUCH und LADEN? Das liegt daran, daß Sie in der Felddimensionierung eine Stringlänge von acht Zeichen definiert haben. Da BUCH aus nur vier Zeichen besteht, füllt der Computer W$(2) mit Leerzeichen auf, um auf acht Zeichen zu kommen.

Versuchen Sie jetzt einmal, ein Wort mit *mehr* als acht Zeichen einzugeben:

```
INTERESSANT
```

Es werden nur die ersten acht Zeichen – INTERESS – auf dem Bildschirm angezeigt. Da das ganze Wort nicht in ein Element von W$ hineinpaßt, werden die drei letzten Zeichen einfach abgeschnitten.

Nach dieser kurzen Einführung in numerische und String-Felder sind Sie so weit,daß Sie sich das Balken-Grafik-Programm ansehen können. Dieses Programm verwendet zwei verschiedene Felder, die in zwei DIM-Anweisungen definiert werden, um alle für die Balken-Grafik benötigten Informationen zu speichern.

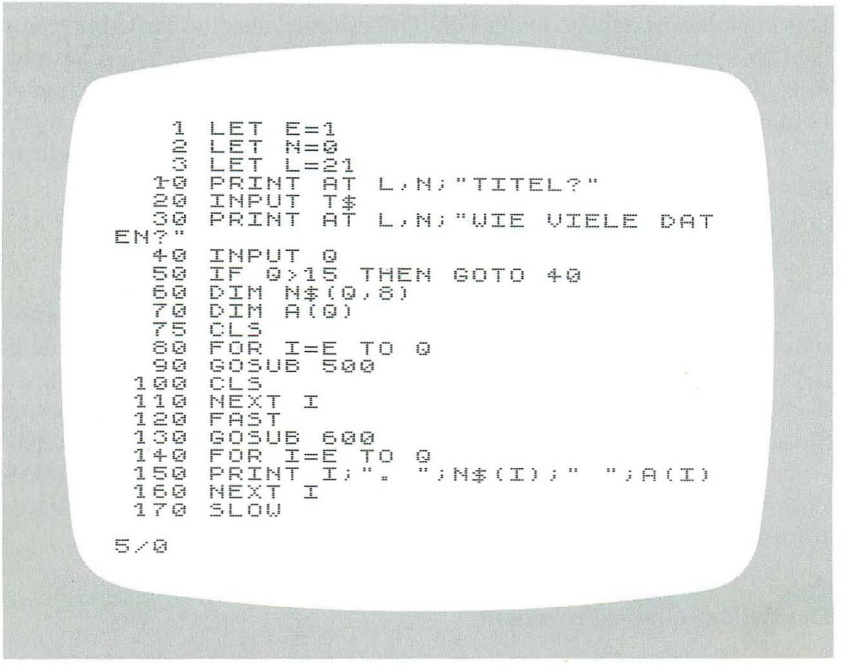
Das Balken-Grafik-Programm

Den Ausdruck des Balken-Grafik-Programms finden Sie in den Abbildungen 4.4 bis 4.7. Wenn Sie ein so langes Programm in den Computer

eingeben, ist es eine gute Idee, es abschnittweise auf einer Kassette zu speichern. Sobald Sie etwa eine Bildschirmseite beschrieben haben, sollten Sie diesen Programmteil sichern. Geben Sie dann den nächsten Programmteil ein. Wenn Sie jetzt das bisherige Programm speichern, können Sie die alten Versionen auf der Kassette überspielen, da diese nicht mehr benötigt werden. Fahren Sie so schrittweise fort, bis Sie das ganze Programm eingegeben und gespeichert haben. Dadurch können Sie vermeiden, daß Ihre ganze Arbeit umsonst war. Das schlimmste, was Ihnen passieren kann, ist, daß Sie die nach dem letzten Speichern eingegebenen Teile verlieren.

Wenn Sie das Programm vollständig eingegeben haben, sehen Sie es noch einmal durch, um sicherzugehen, daß Sie keine Fehler gemacht haben. Dann lassen Sie das Programm laufen.

Dieses Programm erzeugt eine Balken-Grafik für eine beliebige Liste von bis zu 15 Zahlen. In Abbildung 4.8 ist ein Beispiel für eine Programmausgabe wiedergegeben. In dieser Grafik werden die Lebensmittel-Ausgaben für den Monat Dezember miteinander verglichen. Sofern es



```
  1 LET E=1
  2 LET N=0
  3 LET L=21
 10 PRINT AT L,N;"TITEL?"
 20 INPUT T$
 30 PRINT AT L,N;"WIE VIELE DAT
EN?"
 40 INPUT Q
 50 IF Q>15 THEN GOTO 40
 60 DIM N$(Q,8)
 70 DIM A(Q)
 75 CLS
 80 FOR I=E TO Q
 90 GOSUB 500
100 CLS
110 NEXT I
120 FAST
130 GOSUB 600
140 FOR I=E TO Q
150 PRINT I;". ";N$(I);" ";A(I)
160 NEXT I
170 SLOW

5/0
```

Abbildung 4.4: Das Balken-Grafik-Programm

sich dabei um Ihre tatsächlichen Ausgaben handeln würde, hätten Sie damit eine übersichtliche Darstellung Ihrer Ausgaben für die einzelnen Monate. (Falls Sie sich jetzt an den Kopf fassen und sich fragen, was Sie denn mit einer Balken-Grafik über Lebensmittel-Ausgaben anfangen sollen, sorgen Sie sich nicht länger. Das Programm kann für *beliebige* Datenmengen verwendet werden, die Gegenstände spielen keine Rolle. Zu Beginn des Programms geben Sie einen Titel für die Balken-Grafik ein. Außerdem legen Sie auch die Bezeichnungen für die einzelnen Gruppen fest und geben die zugehörigen Zahlenwerte ein.)

Das Programm läuft in drei verschiedenen Phasen ab: Zuerst müssen Sie sämtliche Daten eingeben – Titel, Datenbezeichnungen und Datenwerte. Das Programm zeigt jeweils in Klarschrift an, welche Eingabe es erwartet. Die zweite Phase dient zur Korrektur eventueller Eingabefehler. Dazu werden zunächst alle Daten, die Sie eingegeben haben, in Form einer Tabelle dargestellt. Sie können sich diese Informationen in Ruhe ansehen und haben anschließend Gelegenheit, Fehler in den Eingabedaten zu korrigieren. In der dritten Phase schließlich sehen Sie das aus Ihren

```
 180 PRINT AT L-E,N;"KORRIGIEREN
? (J ODER N) "
 190 INPUT A$
 200 IF A$<>"J" THEN GOTO 280
 210 PRINT AT L,N;"DATEN-NUMMER?
     "
 220 INPUT I
 230 IF I>0 THEN GOTO 220
 240 PRINT AT L-E,N;
 245 GOSUB 700
 250 GOSUB 700
 260 GOSUB 500
 261 PRINT AT I+E,N;
 263 GOSUB 700
 265 PRINT AT I+E,N;I;". ";N$(I)
;" ";A(I)
 270 GOTO 180
 280 FAST
 290 LET MAX=N
 300 LET SUM=N
 310 FOR I=E TO 0

5/0
```

Abbildung 4.5: Das Balken-Grafik-Programm (Forts.)

Daten erstellte Balken-Diagramm. Wir wollen diese Phasen jetzt im einzelnen durchgehen und dazu die Lebensmittel-Ausgaben für einen Probelauf verwenden.

Zu Beginn erwartet das Programm, daß Sie einen Titel für das Balken-Diagramm eingeben:

```
TITEL?
```

Sie können einen beliebigen Titel von maximal 32 Zeichen Länge eingeben. Geben Sie den Titel ein, der oben in der Abbildung 4.8 steht. Anschließend wird Ihnen eine weitere Frage gestellt:

```
WIEVIELE DATEN?
```

Sie müssen jetzt die Anzahl der Daten für Ihre Grafik angeben (maximal 15). Bei den Lebensmittel-Ausgaben sind 15 Posten angegeben; geben Sie deshalb 15 ein und drücken Sie NEW LINE.

```
  320 IF A(I)>MAX THEN LET MAX=A(
I)
  330 LET SUM=SUM+A(I)
  340 NEXT I
  350 LET MIT=SUM/0
  360 LET FAK=L/MAX
  370 CLS
  380 GOSUB 600
  390 FOR I=E TO 0
  400 PRINT N$(I);">";
  410 FOR J=E TO INT (A(I)*FAK+.5
)
  420 PRINT "::::";
  430 NEXT J
  440 PRINT
  450 NEXT I
  455 PRINT
  460 PRINT "[GRAPHICS]";SUM;" [GRAPHICS]
[GRAPHICS]";INT (MIT*100+.5)/100
  470 SLOW
  480 STOP
  500 PRINT AT L-E,N;"DATEN ";I;

  0/0
```

Abbildung 4.6: Das Balken-Grafik-Programm (Forts.)

Als nächstes fragt das Programm nach den einzelnen (hier: 15) Daten-Bezeichnungen und -Werten. Die Eingabe-Aufforderung für den Namen des ersten Elementes sieht so aus:

```
ELEMENT I => NAME:
```

Geben Sie zuerst das Wort FLEISCH ein. Gleich darauf sehen Sie:

```
ELEMENT 1 => NAME: FLEISCH
BETRAG:
```

Jetzt müssen Sie den in diesem Monat für Fleisch ausgegebenen Betrag eintippen. Geben Sie 27.53 ein.

Das Programm fragt nun weiter nach den Namen und Beträgen für die übrigen Daten, bis Sie alle 15 Datenpaare eingegeben haben. In Abbildung 4.9 sind die Daten aufgelistet, die Sie nach FLEISCH eingeben sollten. (Sofern ein Wort länger ist als 8 Zeichen, wird es für dieses Programm abgekürzt.)

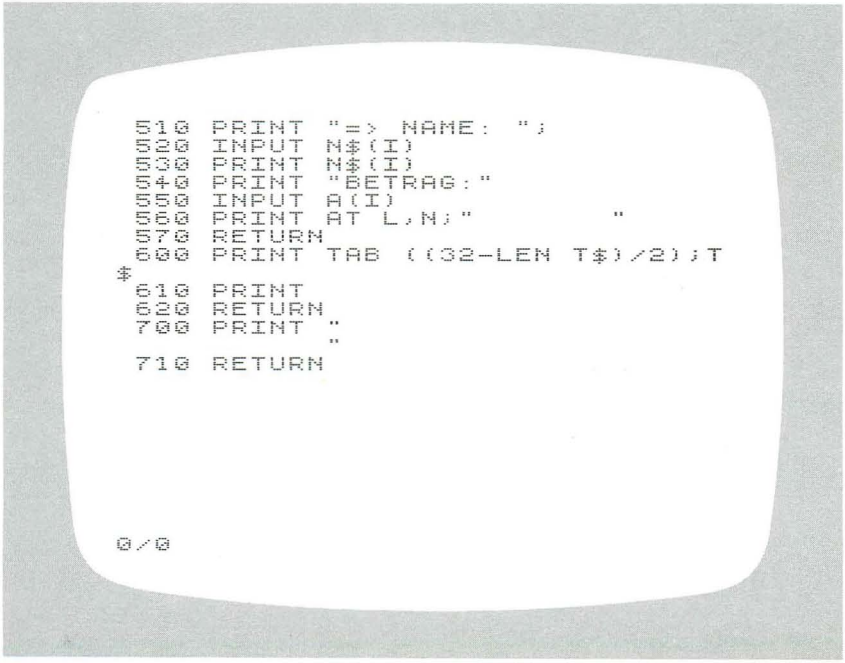


Abbildung 4.7: Das Balken-Grafik-Programm (Forts.)

Wenn Sie die letzten Daten eingegeben haben, wird der Bildschirm für einen Augenblick gelöscht. Anschließend werden die Daten in Form einer Tabelle (vgl. Abbildung 4.9) noch einmal auf dem Bildschirm ausgegeben. Dies ist das „Echo" auf die von Ihnen eingegebenen Daten. Am unteren Bildschirmrand erscheint dazu die Frage:

KORRIGIEREN? <J> ODER <N>

Jetzt haben Sie Gelegenheit, die Daten auf Tippfehler hin zu überprüfen. Wenn Sie einen Fehler finden, brauchen Sie nur J (für „ja") zu tippen, worauf das Programm zurückfragt:

WELCHE ELEMENT-NR.?

An dieser Stelle müssen Sie die Nummer des Elementes eingeben, das Sie ändern wollen. Damit wir sehen, wie das Korrigieren vor sich geht, wollen wir den Betrag für Nr. 9 (OBST) ändern. Geben Sie die Zahl 9 ein. Das Programm fragt dann wieder nach dem Namen und dem Betrag.

Abbildung 4.8: Probelauf des Balken-Grafik-Programms: Lebensmittel-Ausgaben

(Sie müssen wieder beide Daten eingeben, auch wenn Sie nur eine Angabe ändern wollen.) Geben Sie für den Namen wieder OBST und für den Betrag den neuen Wert 7.15 ein. Sie sehen den korrigierten Wert in der angezeigten Tabelle, so daß Sie ihn noch einmal überprüfen können. Sie können beliebig viele Korrekturen machen, so lange, bis Sie den Buchstaben N (für „nein") als Antwort auf die Eingabe-Aufforderung eingeben.

Sobald Sie mit dem Korrigieren fertig sind, wird der Bildschirm für einige Augenblicke gelöscht. Danach erscheint die Balken-Grafik auf dem Bildschirm. Sehen Sie sich noch einmal Abbildung 4.8 an, die Balken-Grafik für die Lebensmittel-Ausgaben. Beachten Sie, daß außer der Grafik auch noch die Summe der monatlichen Ausgaben sowie der Mittelwert angezeigt werden.

Abbildung 4.10 zeigt ein weiteres Beispiel für die Balken-Grafik. Diesmal heißt die Überschrift „Büroausgaben – 1982", und die Balken stellen die monatlichen Büroausgaben von Januar bis Dezember dar. Die Gesamtausgaben betragen DM 7439.86, der Durchschnittswert pro Monat ist DM 619.99.

Abbildung 4.9: Die Lebensmittel-Daten

Im nächsten Abschnitt dieses Kapitels betrachten wir den Aufbau dieses Programms. Bevor Sie weiterlesen, lassen Sie vielleicht das Programm noch einmal mit Daten laufen, die Sie persönlich interessieren – Daten aus Ihrem Haushalt, Geschäft oder aus dem Schulbereich.

Der Aufbau des Balken-Grafik-Programms

Beim Lesen dieser Beschreibung können Sie den Ausdruck des Programms in den Abbildungen 4.4 bis 4.7 verfolgen oder aber das Programm auf Ihrem eigenen Bildschirm auflisten.

Dies Programm ist umfangreicher als die bisherigen. Um Platz zu sparen, wurden deshalb drei Variablen am Programmanfang definiert, die während des ganzen Programmlaufs konstant bleiben. Wie wir in Kapitel drei gesehen haben, spart man Speicherplatz, wenn man Zahlen durch eine Variable ersetzt, die aus nur einem Buchstaben besteht:

```
1 LET E = 1
2 LET N = 0
3 LET L = 21
```

Die Variable L dient als Bildschirm-Adresse für die Eingabe-Aufforderungen.

Um Speicherplatz zu sparen, wurde auch auf Kommentarzeilen (REM) im Programm verzichtet. Dies ist gerade für ein so langes Programm natürlich bedauerlich, jedoch werden Sie sich manchmal wohl oder übel dazu entscheiden müssen. Zum Ausgleich werden hier die einzelnen Programmabschnitte besprochen:

- Initialisierungs-Teil (Zeile 1 bis 70).

In diesem Abschnitt werden die drei „konstanten" Variablen definiert, die Überschrift und die Zahl der Daten eingelesen, und die beiden Fehler N$ und A definiert, in denen die Daten für das Balken-Diagramm gespeichert werden sollen.

- Eingabe-Teil (Zeile 75 bis 110).

In diesem Teil erfolgt die Dateneingabe. Für jedes Datenelement wird das Unterprogramm bei Zeile 500 einmal aufgerufen. Dieses Unterprogramm zeigt die Eingabe-Aufforderungen an und liest die Daten ein.

- Echo-Teil (Zeile 120 bis 170).

In diesem Teil werden die eingegebenen Daten noch einmal in Form einer Tabelle auf dem Bildschirm angezeigt. Durch das Unterprogramm bei Zeile 600 wird die Überschrift oben auf dem Bildschirm ausgegeben.

- Korrektur-Teil (Zeile 180 bis 270).

In diesem Abschnitt erfolgt der Korrektur-Dialog. Jedesmal wenn Sie Daten ändern, werden sie hier den entsprechenden Elementen von N$ und A zugewiesen. Die neuen Daten werden dann auf dem Bildschirm angezeigt. Vom Korrektur-Teil aus werden zwei Unterprogramme aufgerufen: das Unterprogramm bei 500, das die Dateneingabe besorgt, und das Unterprogramm bei Zeile 700, das eine Bildschirmzeile löscht, damit die neue Information angezeigt werden kann.

- Zwischenberechnungs-Teil (Zeile 280 bis 360).

Hier werden vier Werte aus den in Feld A gespeicherten Daten berechnet. MAX ist die größte Zahl in diesem Feld, SUM ist die Summe al-

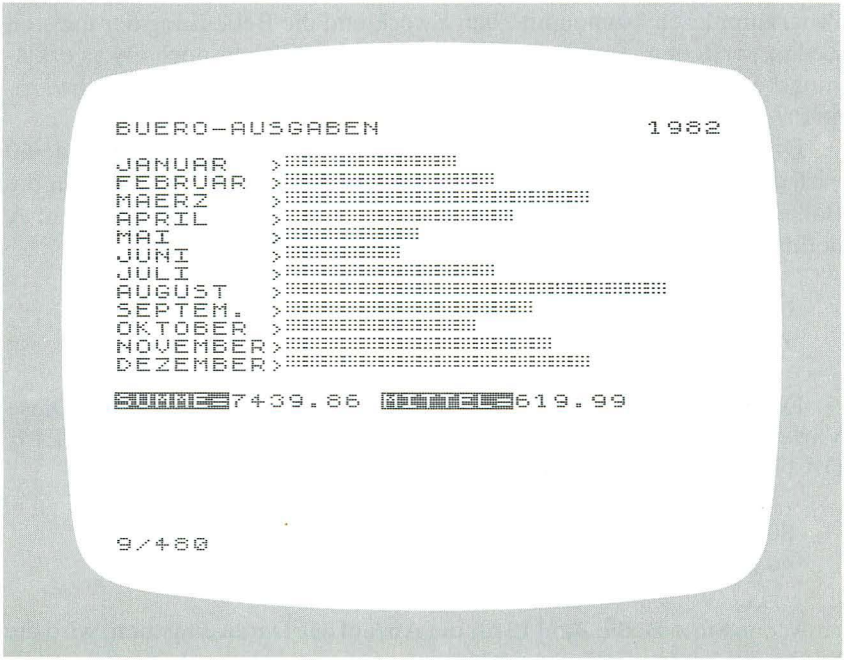


Abbildung 4.10: Zweites Beispiel zur Balken-Grafik: Büroausgaben

ler Werte und MIT ist der Mittelwert. FAK ist ein Umrechnungsfaktor, durch den die Balkenlänge so festgelegt wird, daß sie der zugehörigen Zahl entspricht.

- Der Balken-Grafik-Teil (Zeile 370 bis 480).

Dieser Teil beginnt mit einem Aufruf des Unterprogramms bei Zeile 600, in dem die Überschrift für das Balkendiagramm angezeigt wird. Dann werden die einzelnen Balken gezeichnet, wobei die Länge durch den Umrechnungsfaktor FAK bestimmt wird. Anschließend werden die Summe und der Mittelwert der Daten ausgegeben.

- Die Unterprogramme

- bei Zeile 500: das Eingabe-Unterprogramm;
- bei Zeile 600: das Überschrift-Unterprogramm;
- bei Zeile 700: das Unterprogramm zum Zeilenlöschen.

Mit Hilfe der vorstehenden Übersicht müßten Sie eigentlich das ganze Programm lesen können und den Zweck und die Bedeutung der meisten Zeilen verstehen. Einige Zeilen sind jedoch vielleicht noch etwas erklärungsbedürftig. In den folgenden Anmerkungen werden einige schwierige Programmzeilen erklärt:

Die Zeilen 60 und 70 enthalten eine DIM-Anweisung. In Zeile 60 wird das String-Feld N$ definiert, in dem die Namen der Elemente für die Balken-Grafik gespeichert werden. In Zeile 70 wird das Zahlen-Feld A definiert, das die dazugehörigen Zahlen speichern soll:

```
60 DIM N$(Q,8)
70 DIM A(Q)
```

Die Länge dieser Felder wird durch die Variable Q bestimmt. Diese Variable wird während des Programmlaufs auf die Frage „WIEVIELE DATEN?“ von Ihnen eingegeben:

```
30 PRINT AT L,Z; "WIEVIELE DATEN?"
40 INPUT Q
```

Wenn Sie z. B. die Zahl 12 für die Anzahl der Daten eingeben, wird die Länge der beiden Felder auf 12 festgesetzt, so als ob die Zeilen 60 und 70 wie folgt geschrieben wären:

```
60 DIM N$(12,8)
70 DIM A(12)
```

Durch die Verwendung der Variablen Q wird sichergestellt, daß der für die beiden Felder reservierte Speicherbereich nicht größer als notwendig ist.

In den Zeilen 80 bis 110 werden den beiden Feldern in einer FOR-Schleife Werte zugewiesen:

```
 80 FOR I = E TO Q
 90 GOSUB 500
100 CLS
110 NEXT I
```

Durch die FOR-Anweisung erhält die Variable I nacheinander die Werte von 1 bis Q. Das Unterprogramm in Zeile 500 verwendet diese Laufvariable, um die jeweilige Element-Nummer auf dem Bildschirm anzuzeigen:

```
500 PRINT AT L-E,Z; "ELEMENT ";I;
```

Die Variable I wird außerdem dazu benutzt, die einzelnen Daten-Elemente in den entsprechenden Feld-Elementen zu speichern:

```
520 INPUT N$(I)
.
.
.
550 INPUT A(I)
```

Diese Eingabe-Zeilen und die davor stehenden Eingabe-Aufforderungen wurden deshalb als Unterprogramm geschrieben, weil sie im Programm an zwei verschiedenen Stellen gebraucht werden: zunächst bei der ersten Eingabe und später im Korrektur-Teil.

Eine andere FOR-Schleife – in den Zeilen 140 bis 160 – bringt den Inhalt der beiden Felder auf den Bildschirm:

```
140 FOR I = E TO Q
150 PRINT I; "."; N$(I); " "; A(I)
160 NEXT I
```

In Zeile 150 wird die Laufvariable I dreimal benutzt: zur Anzeige der laufenden Nummer, zur Ausgabe der entsprechenden Datenbezeichnung aus dem Feld N$ und zur Ausgabe des Zahlenwertes aus dem Feld A.

In den Zeilen 310 bis 340 werden der größte Wert des Feldes A (MAX) und die Summe aller Werte (SUM) bestimmt. Die beiden dazu verwendeten Variablen werden zuerst auf Null gesetzt:

```
290 LET MAX = N
300 LET SUM = N
```

Dann läuft eine FOR-Schleife durch das ganze Feld von 1 bis Q:

```
310 FOR I = E TO Q
320 LF A(I) > MAX THEN LET MAX = A(I)
330 LET SUM = SUM + A(I)
340 NEXT I
```

Zeile 320 vergleicht jeden Wert des Feldes mit dem augenblicklichen Wert von MAX. Jedesmal wenn ein größerer Wert als MAX gefunden wird, wird dies der neue Wert von MAX. In Zeile 330 werden die einzelnen Werte aufaddiert.

Der Mittelwert ergibt sich aus der Division der Summe durch die Anzahl der Werte, Q:

```
350 LET MIT = SUM/Q
```

Der proportionale Umrechnungsfaktor wird aus dem größten Wert des Feldes, aus MAX, abgeleitet:

```
360 LET FAK = L/MAX
```

Die Variable L ist, wie Sie sich erinnern, ein „konstanter" Wert, der gleich 21 ist, d. h. der längste Balken in der Grafik wird 21 Zeichen lang. Wie wir gleich sehen werden, ergibt sich durch Multiplikation der einzelnen Feldelemente von A mit dem Umrechnungsfaktor FAK eine Zahl zwischen Null und 21, durch die dann die Länge der einzelnen Balken bestimmt wird.

Die Balken-Grafik selbst wird in den Zeilen 390 bis 450 dargestellt. Diese Zeilen sollten Sie sich sorgfältig ansehen. Sie finden darin ein Beispiel für eine FOR-Schleife *innerhalb* einer anderen FOR-Schleife. Dies wird auch als *geschachtelte* Schleife bezeichnet. Es gibt zwei grundlegende

Regeln, die Sie bei der Bildung von geschachtelten Schleifen beachten müssen:

1. Die Schleifen müssen verschiedene Laufvariablen verwenden.
2. Sowohl die FOR- als auch die NEXT-Anweisung der inneren Schleife müssen innerhalb der äußeren Schleife liegen.

Denken Sie an diese Regeln, wenn Sie die Zeilen betrachten, die die Balken-Grafik erzeugen.

Die äußere Schleife, die bei Zeile 390 beginnt, kontrolliert die Daten-Nummer, indem die Laufvariable von 1 bis Q erhöht wird:

```
390 FOR I = E TO Q
```

Zuerst wird der Name des Daten-Elements angezeigt:

```
400 PRINT N$(I); "> ";
```

Dann erzeugt die innere Schleife einen Balken, der dem Wert der Zahl in A(I) entspricht:

```
410 FOR J = E TO INT(A(I) * FAK + .5)
```

Aus dem Ausdruck A(I) * FAK ergibt sich, wie schon erwähnt, eine Zahl zwischen 0 und 21. Durch Addition von .5 und Anwenden der INT-Funktion wird diese Zahl auf den nächsten ganzzahligen Wert gerundet:

```
INT(A(I) * FAK + .5)
```

Auf diese Weise wird in der inneren Schleife die Laufvariable J von 1 bis zu der entsprechenden Balkenlänge erhöht. Jedesmal wenn J erhöht wird, wird die Balkenlänge um ein Zeichen verlängert. (Das als „Baustein" verwendete Grafik-Zeichen befindet sich auf der S-Taste.)

Fassen wir zusammen: Sie sehen, daß bei jedem Durchlauf der äußeren Schleife die innere Schleife einen Balken erzeugt. Dies sind wahrscheinlich die wichtigsten Zeilen, die Sie in diesem Programm verstehen müssen. Wenn Ihnen nicht ganz klar ist, was in diesen Zeilen passiert, hilft Ihnen vielleicht der folgende Vergleich, den Begriff der geschachtelten Schleife zu verstehen.

Stellen Sie sich die drei Zeiger einer Uhr vor. Wenn Sie einen Umlauf um das Ziffernblatt als „Schleife" bezeichnen, haben Sie insgesamt drei

Schleifen. Die innerste Schleife wird durch den Sekundenzeiger dargestellt, der sich um 60 Stellen weiterbewegt – d. h. eine Schleife durchläuft –, bevor der Minutenzeiger um eine Stelle weiterrückt. Der Minutenzeiger wiederum bewegt sich um 60 Stellen weiter, ehe der Stundenzeiger um eine Einheit weiterrückt. Es handelt sich also um drei Wiederholungsebenen. Die innerste Schleife läuft am schnellsten und die äußerste Schleife am langsamsten. Ensprechend verhält es sich mit den Schleifen in unserem Programm: die äußere Schleife bei Zeile 390 läuft langsamer als die innere Schleife bei Zeile 410.

Zusammenfassung

Ihr Computer verfügt über fünf arithmetische Operationen (+, −, *, /, **) und eine Anzahl von gebräuchlichen mathematischen Funktionen. Diese Operationen und Funktionen können zu komplexen Ausdrücken zusammengesetzt werden. Wenn Sie solche Ausdrücke schreiben, müssen Sie die Reihenfolge berücksichtigen, in der der Computer die Operationen durchführt. Wenn Sie diese Reihenfolge ändern wollen – oder einen Ausdruck einfach übersichtlicher gestalten wollen –, können Sie Klammern in den arithmetischen Ausdrücken benutzen.

Die schnelle Betriebsart des Computers kann dann von Nutzen sein, wenn Berechnungen möglichst schnell durchgeführt werden sollen, wie wir in dem Super-Rechner-Programm gesehen haben.

Ein *Feld* ist eine wichtige Datenstruktur in BASIC, die es ermöglicht, viele Werte unter einem Variablennamen zu speichern. Felder können am besten innerhalb von FOR-Schleifen genutzt werden; die Laufvariable einer FOR-Schleife kann als *Feldindex* verwendet werden.

Das in diesem Kapitel vorgeführte Balken-Grafik-Programm enthält ein gutes Beispiel für die Leistungsfähigkeit von geschachtelten FOR-Schleifen sowie mehrere Beispiele für den effektiven Gebrauch von Feldern in FOR-Schleifen.

Kapitel 5

Wörter, nichts als Wörter
Strings und Stringfunktionen

Einführung

In diesem letzten Kapitel wollen wir uns Strings und das, was man mit ihnen machen kann, näher ansehen. Wie Sie wissen, ist ein String ein Datenelement, das aus einem oder mehreren Zeichen besteht. Sie haben zwar schon einige Erfahrungen mit Strings und Stringvariablen aus den vorhergehenden Kapiteln, aber es gibt noch verschiedene wichtige Funktionen, die Sie kennenlernen sollten.

Wir sehen uns zunächst den vom ZX81 verwendeten Zeichencode an. Dabei lernen Sie, wie man die Funktionen CODE und CHR$ benutzt, um die einem Zeichen entsprechende Zahlencode-Darstellung zu erhalten, bzw. umgekehrt das einer Zahl entsprechende Zeichen zu ermitteln. Sie können diese Funktionen an etlichen kurzen Programmen ausprobieren und erfahren auch, was eine Verkettung ist und wie man mit der LEN-Funktion umgeht.

Zur Vorbereitung auf das letzte Programm in diesem Buch werden Sie dann feststellen, was passiert, wenn man einen String „auseinandernimmt" – und wozu das gut sein kann. Das letzte Programm – das „Entscheidungs-Programm" – macht aus Ihrem Computer einen persönlichen Berater.

Der ZX81 Zeichen-Code

Der Computer verwendet eine spezielle Form der Verschlüsselung, um die Tastaturzeichen zu speichern und zu unterscheiden. In manchen Fällen kann es bei der Programmierung nützlich sein, wenn Sie über diesen Code Bescheid wissen.

Die Verschlüsselung erfolgt dadurch, daß jedem Zeichen eine Zahl zwischen 0 und 255 zugeordnet wird. Der Computer betrachtet übrigens alle Angaben auf der Tastatur als ein *Zeichen* – auch die Schlüsselwörter,

Funktionen und sonstigen Wörter auf der Tastatur. In den Abbildungen 5.1 bis 5.3 ist dieser Code aufgelistet. Es ist keineswegs notwendig, daß Sie diesen Code auswendig lernen, jedoch sollten Sie sich schon etwas Zeit nehmen, um einen Überblick über die Lage der verschiedenen Zeichengruppen zu gewinnen. Die Grafik-Zeichen z. B. liegen in zwei verschiedenen Teilen des Codes (von 1 bis 10 und 128 bis 138); die Ziffern und Buchstaben befinden sich am Anfang des Codes (28 bis 63), und die invers dargestellten Zeichen liegen ungefähr in der Mitte des Codes (139 bis 191). Ebenso enthält der Code auch alle Schlüsselwörter, Funktionen, Operationszeichen und andere Symbole.

Sie werden feststellen, daß ein großer Teil des Codes (von 67 bis 127) fehlt. Die meisten dieser fehlenden Code-Zahlen sind einfach unbenutzt. Einige von ihnen werden für „Kontroll-Zeichen" benutzt, wie die Richtungs-Tasten und die RUBOUT-Taste, die keine Zeichen auf dem Bildschirm darstellen.

Die CHR$-Funktion liefert das einer Codezahl zugeordnete Zeichen. Diese Zahl muß natürlich zwischen 0 und 255 liegen. CHR$ ergibt den

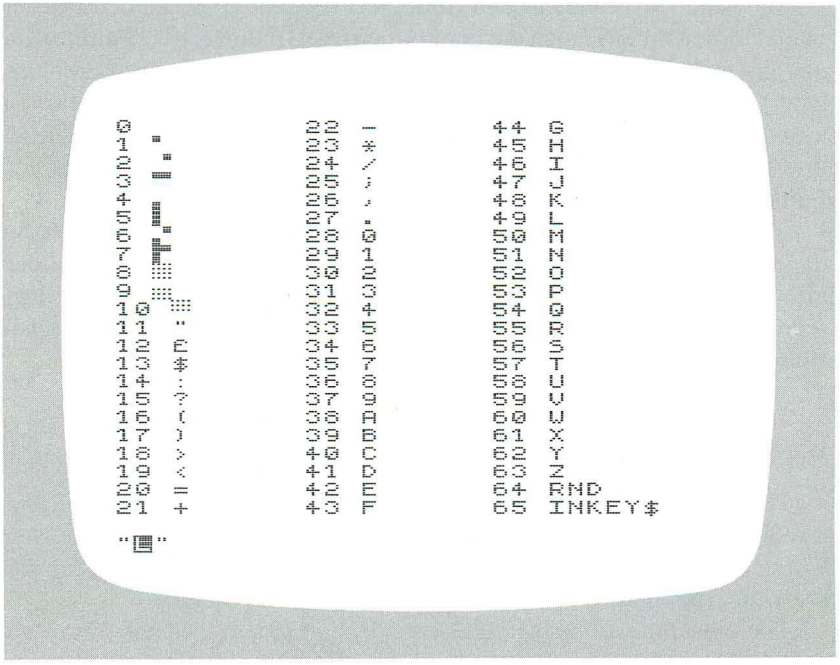


Abbildung 5.1: Der Zeichen-Code

entsprechenden Buchstaben, das Schlüsselwort, die Funktion oder das Symbol.

Das folgende sechs Zeilen lange Programm liest eine Code-Zahl von der Tastatur ein und bringt das entsprechende Zeichen auf den Bildschirm:

```
10 PRINT AT 21,0; "WELCHE CODE-ZAHL?"
20 INPUT C
30 IF C > 255 THEN GOTO 20
40 CLS
50 PRINT AT 8,8; "CODE ";C; " = ";CHR$ C
60 GOTO 10
```

Zeile 30 überprüft, ob die eingegebene Zahl eine zulässige Code-Zahl ist. Zeile 50 verwendet die CHR$-Funktion, um das Zeichen anzuzeigen.

Umgekehrt liefert die CODE-Funktion die Code-Zahl eines Zeichens. Sie erinnern sich vielleicht aus Kapitel 2 an CODE und CHR$,

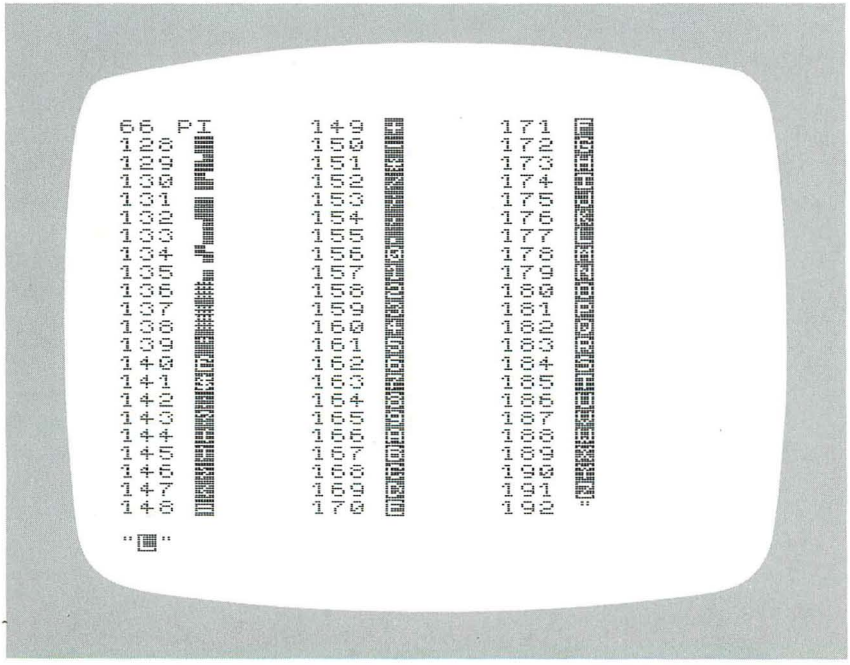


Abbildung 5.2: Der Zeichen-Code (Forts.)

denn das erste Programm, das Sie in den Computer eingegeben haben, benutzte diese beiden Funktionen, um eine Zeichenkette invers darzustellen.

Das in Abbildung 5.4 wiedergegebene Programm gibt einen eingegebenen String in inverser Darstellung wieder aus. Das Programm dient hauptsächlich zur Übung. Sie können daran einige interessante Techniken zur String-Verarbeitung kennenlernen, wenn Sie sich die Zeit nehmen, das Programm sorgfältig zu studieren.

Einige Beispiele für die Programmausgabe sind in Abbildung 5.5 angegeben. Wie Sie sehen, verlangt das Programm fortwährend nach einer neuen Eingabe. Nach jeder String-Eingabe gibt das Programm diesen String invers wieder aus. Der interessante Teil dieses Programms spielt sich in den Zeilen 50 bis 80 ab, in denen die Umformung in die inverse Darstellung erfolgt.

Die String-Variable W$ enthält das eingegebene Wort, die Variable R$ die inverse Darstellung des Wortes. In Zeile 50 wird R$ ein *leerer String* zugewiesen, d. h. ein String, der keine Zeichen enthält.

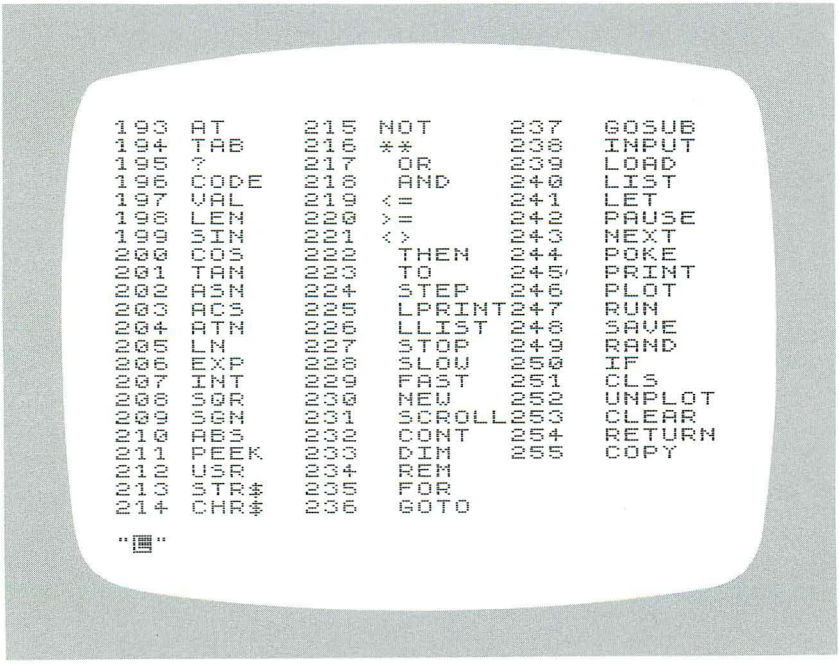


Abbildung 5.3: Der Zeichen-Code (Forts.)

```
50 LET R$ = ""
```

Dann wird in der FOR-Schleife in den Zeilen 60 bis 80 die inverse Darstellung der *einzelnen Zeichen* von W$ ermittelt, die an das Ende von R$ angehängt werden. In der FOR-Anweisung (Zeile 60) wird die LEN-Funktion benutzt, um die Länge des Eingabe-Strings W$ in Zeichen zu ermitteln. Wie Sie sehen, läuft die Kontroll-Variable I von 1 bis LEN W$:

```
60 FOR I = 1 TO LEN W$
```

So wird der Reihe nach jedes Zeichen von W$ umgeformt. BASIC ermöglicht Ihnen, ein Zeichen eines Strings auszuwählen, indem Sie eine Zahl eingeben, die die Position dieses Zeichens angibt. Diese Zahl steht in Klammern hinter der String-Variablen. Somit stellt W$(I) das Zeichen von W$ an der Stelle I dar.

Wenn Sie z. B. einen drei Zeichen langen String eingeben, würde I in der FOR-Schleife von 1 bis 3 laufen. Die drei Zeichen dieses Strings sind W$(1), W$(2) und W$(3).

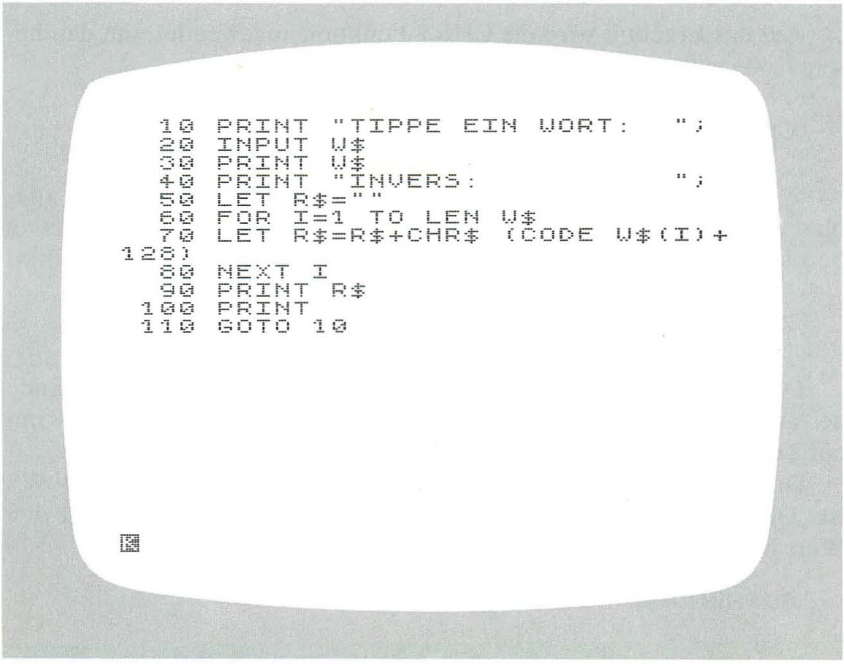


Abbildung 5.4: Das Programm zur inversen Zeichendarstellung

Sehen wir uns nun den Ausdruck in Zeile 70 an, der ein Zeichen in das entsprechende inverse Zeichen umformt:

```
CHR$(CODE W$(I)+128)
```

Die CODE-Funktion ergibt die Code-Zahl des Zeichens W$(I). Sehen Sie sich noch einmal den Zeichen-Code in den Abbildungen 5.1 bis 5.3 an. Wenn der eingegebene String aus Interpunktionszeichen, Ziffern oder Buchstaben besteht, liegt die Code-Zahl für ein solches Zeichen zwischen 11 und 63. Wenn Sie zu einer dieser Code-Zahlen 128 addieren, ergibt sich die Code-Zahl für die inverse Darstellung des entsprechenden Zeichens. Ein Beispiel: Der Zeichen-Code für den Buchstaben D ist 41. Durch Addieren von 128 erhält man 169. Die inverse Darstellung des Buchstaben D hat also den Code 169.

Sie können jetzt also genau verfolgen, wie die Umrechnung in Zeile 70 vor sich geht. Zunächst wird 128 zu der Code-Zahl des betreffenden Zeichens addiert:

```
(CODE W$(I)+128)
```

Auf das Ergebnis wird die CHR$-Funktion angewendet, um das diesem Code entsprechende Zeichen zu ermitteln:

```
CHR$(CODE W$(I)+128)
```

Wenn das neue Zeichen bestimmt ist, wird es an den String der Variablen R$ angehängt. Diese Verbindung zweier Strings wird als *Verkettung* bezeichnet. Der Vorgang wird durch das Plus-Zeichen (+) dargestellt:

```
70 LET R$ = R$ + CHR$(CODE W$(I) + 128)
```

Diese LET-Anweisung besagt: „Füge das neue Zeichen an das Ende des augenblicklichen Strings in R$ an; dann speichere den neuen – um ein Zeichen erweiterten – String wieder in R$."

Die Umwandlung des ganzen Strings ist beendet, wenn I = LEN W$ ist. Damit ist die FOR-Schleife abgeschlossen, und Zeile 90 zeigt den Wert von R$ auf dem Bildschirm an:

```
90 PRINT R$
```

Somit können Sie aus den Zeilen 50 bis 80 dieses Programms einige wichtige Dinge über Strings und String-Verarbeitung lernen:

- Sie können einer String-Variablen einen *leeren String* zuweisen, d.h. einen String, der keine Zeichen enthält (Zeile 50).
- Die Funktion LEN liefert die Länge eines Strings in Zeichen (Zeile 60).
- Sie können ein einzelnes Zeichen aus einem String auswählen, indem Sie die Position eines Zeichens in Klammern angeben (Zeile 70).
- Die Code-Nummern der inversen Zeichen sind alle um 128 größer als die Code-Nummern der normalen Zeichen (Zeile 70).
- Sie können zwei Strings verketten, indem Sie das „+"-Symbol verwenden (Zeile 70).

Im nächsten Abschnitt werden Sie sehen, daß Ihr Computer auch eine Möglichkeit bietet, auf mehrere aufeinander folgende Zeichen eines Strings, sogenannte *Teilstrings*, zuzugreifen.

Teilen eines Strings

Aus verschiedenen Gründen ist es oft wünschenswert, einen *Teilstring*

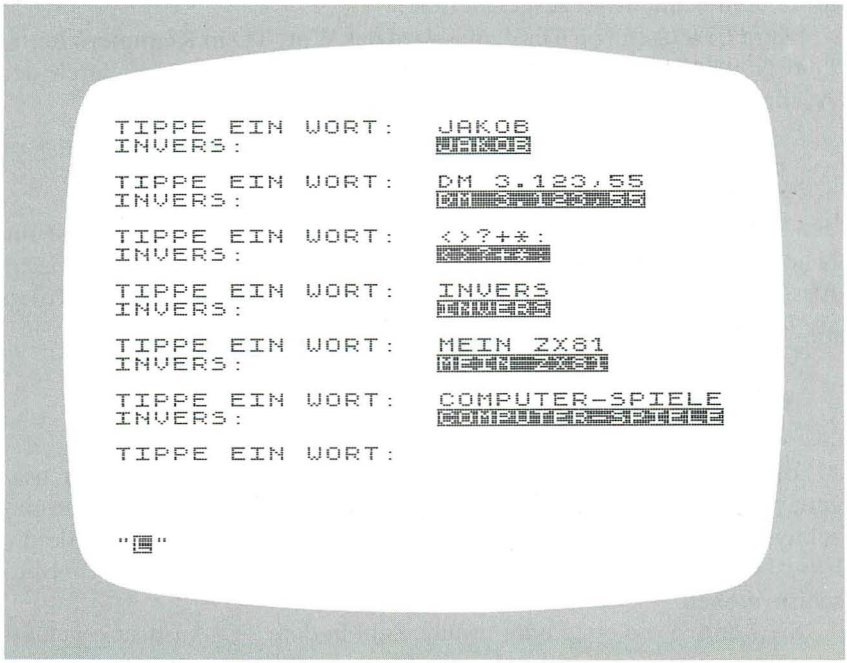


Abbildung 5.5: Ausgabebeispiel für das Programm zur inversen Zeichendarstellung

aus einem größeren String herausnehmen zu können. Vielleicht möchten Sie feststellen, ob ein String eine bestimmte Zeichenfolge enthält, oder Sie möchten die Werte bestimmter Zeichen innerhalb eines Strings verändern. In dem Entscheidungsprogramm am Ende dieses Kapitels wollen wir auf ziemlich kleine Teile eines langen Strings *zugreifen*, um sie anzuzeigen. Mit Hilfe von Teilstrings können Sie solche Dinge einfach und bequem erledigen.

Wir probieren zunächst eine Reihe von direkten Anweisungen aus. Damit Sie sehen, was passiert, sollten Sie alle Anweisungen in Ihren Computer eingeben.

Definieren Sie zunächst einen String W$ wie folgt:

```
LET W$ = "FUNDAMENTAL"
```

Geben Sie nun das Kommando

```
PRINT W$
```

ein, um sich zu überzeugen, daß das Wort FUNDAMENTAL tatsächlich unter dem Namen W$ gespeichert ist.

Zum Erzeugen von Teilstrings wird das Wort TO in Klammern hinter dem Namen der String-Variablen benutzt. Allgemein wird durch den Ausdruck

```
S$(M TO N)
```

der Teil des Strings S$ angesprochen, der zwischen den Positionen M und N liegt. M und N können Zahlen, numerische Variablen oder numerische Ausdrücke sein. Wir bezeichnen einen solchen Ausschnitt eines Strings als Teilstring.

Geben Sie zunächst folgende Anweisung:

```
PRINT W$(5 TO 8)
```

Bevor Sie die NEW LINE-Taste drücken, versuchen Sie vorherzusagen, wie das Ergebnis dieser Anweisung aussehen wird, indem Sie die einzelnen Zeichen des Wortes FUNDAMENTAL abzählen. Wenn Sie die Eingabe abgeschlossen haben, sollte das Wort AMEN oben auf dem Bildschirm stehen.

Sie können die erste oder zweite Zahl in dem TO-Ausdruck weglassen. Wenn Sie die erste weglassen, fängt der Teilstring mit dem ersten Zeichen des Strings an:

```
PRINT W$(TO 3)
```

Diese Anweisung ergibt das Wort FUN. Wenn Sie die letzte Zahl weglassen, reicht der Teilstring bis zum Ende des Strings, z. B.:

```
PRINT W$(6 TO)
```

Wenn Sie diese Anweisung eingeben, sollten Sie das Wort MENTAL auf dem Bildschirm sehen.

Sie können einen Teilstring auch *austauschen*. Dazu müssen Sie diesen Teilstring in einer LET-Anweisung wie folgt angeben:

```
LET W$(TO 8) = "BERG U. "
```

Geben Sie dieses Kommando und anschließend

```
PRINT W$
```

ein, um zu sehen, was aus dem String geworden ist. Sie sollten die Wörter BERG U. TAL sehen. Die ersten 8 Zeichen des Strings sind durch die neuen Zeichen ersetzt worden.

Das Entscheidungsprogramm wählt mit Hilfe von Teilstrings eine Antwort aus einem langen String aus. Das desillusionierende Geheimnis dieses Programms besteht darin, daß die Antworten auf Ihre Fragen völlig zufällig ausgewählt werden. Bevor wir das Programm untersuchen, müssen wir also noch kurz besprechen, wie der Computer Zufallszahlen erzeugen kann.

Zufallszahlen

Die Funktion RND (unterhalb der T-Taste) erzeugt eine *Zufallszahl* zwischen 0 und 1. Bei jedem Aufruf liefert sie eine andere Zahl. Sie können die Arbeitsweise von RND mit dem folgenden Programm ausprobieren:

```
10 PRINT RND
20 GOTO 10
```

Eine Reihe von Zufallszahlen erscheint auf dem Bildschirm, bis die Ausgabe am unteren Bildschirmrand angelangt ist. Zufallszahlen sind grundsätzlich Zahlen, die in einer völlig unvorhersehbaren Weise ausge-

wählt werden. Tatsächlich benutzt der Computer aber mathematische Formeln, um die Zahlen für die RND-Funktion zu liefern. D.h. daß die Zahlen nicht wirklich zufällig sind. Die Formel ist jedoch so aufgebaut, daß die Zahlen immer wie Zufallszahlen *aussehen*. Für die Programme, die Sie auf Ihrem Computer schreiben werden, wird diese Pseudo-Zufälligkeit wahrscheinlich immer ausreichen.

Auf so kleinen Computern wie dem ZX81 wird die Zufallszahlen-Funktion vermutlich hauptsächlich dazu benutzt, um zufällige Ereignisse in Spielprogrammen zu simulieren. Beispielsweise können Sie mit RND einen rollenden Würfel oder eine Handvoll Spielkarten simulieren.

Damit wir diese Simulationen erzeugen können, müssen wir den Bereich der von RND gelieferten Zufallszahlen umformen können. Dazu sind die Zufallszahlen zwischen 0 und 1 vielleicht nicht besonders brauchbar; wenn wir sie jedoch z.B. in zufällige ganze Zahlen zwischen 1 und 6 umformen, können wir sie etwa in Würfelspielen verwenden.

Die Formel für solche Umformungen ist verhältnismäßig einfach. Wenn Sie eine Zufallszahl haben, die größer ist als 0 und kleiner als 1, und

```
   5 LET R=INT (RND*6)+1
  10 PRINT AT 20,0;"ICH DENKE MI
R EINE ZAHL"
  20 PRINT "VON 1 BIS 6. KANNST
DU SIE RATEN?"
  30 INPUT G
  40 CLS
  50 IF R=G THEN PRINT AT 8,3;"R
ICHTIG";
  60 IF R<>G THEN PRINT AT 8,3;"
BEDAURE";
  70 PRINT ".DIE ZAHL WAR ";R;".
"
  80 PRINT AT 21,0;"TIPPE [NEW LINE]
 FUER EINE WEITERE RUNDE."
  90 INPUT A$
 100 CLS
 110 GOTO 5
```

Abbildung 5.6: Das Ratespiel-Programm

wenn Sie diese Zahl mit 6 multiplizieren, erhalten Sie eine Zufallszahl zwischen 0 und 6:

```
RND * 6
```

Wenn Sie aus dieser Zahl (mit Hilfe der INT-Funktion) eine ganze Zahl machen, ergibt sich eine ganzzahlige Zufallszahl zwischen 0 und 5:

```
INT(RND * 6)
```

Addieren Sie zu diesem Ergebnis noch 1 hinzu, so erhalten Sie eine ganzzahlige Zufallszahl zwischen 1 und 6:

```
INT(RND * 6) + 1
```

Damit Sie diese Formel ausprobieren können, geben Sie das kurze Programm aus Abbildung 5.6 ein und lassen es laufen. Dieses Programm stellt ein sehr elementares Ratespiel dar. Der Computer zeigt zunächst den folgenden Text auf dem Bildschirm an:

```
ICH DENKE AN EINE ZAHL
ZWISCHEN 1 UND 6. RATEN SIE!
```

Sie geben nun eine ganze Zahl zwischen 1 und 6 ein. Der Computer sagt Ihnen dann, ob Sie richtig oder falsch geraten haben. Wenn Sie z. B. die 4 eingegeben haben, sehen Sie entweder eine Meldung wie diese

```
RICHTIG! DIE ZAHL WAR 4.
```

oder

```
LEIDER FALSCH! DIE ZAHL WAR 3.
```

Dieses Spiel ist zwar sehr einfach, zeigt aber, wie man mit Hilfe von RND unvorhersagbare Ergebnisse mit dem Computer erzeugen kann. Beachten Sie, daß die Zeile 5 des Programms die Zufallszahl mit Hilfe der oben entwickelten Formel erzeugt:

```
5 LET R = INT(RND * 6) + 1
```

Eine ähnliche Formel werden Sie in dem Entscheidungsprogramm finden.

Das Entscheidungs-Programm

Sie können das Entscheidungs-Programm als Gesellschaftsspiel benutzen, um Ihre Freunde mit den wunderbaren Fähigkeiten des Computers zu unterhalten, die Zukunft vorherzusagen, persönliche Schwierigkeiten zu lösen oder schnelle geschäftliche Entscheidungen zu treffen. Dabei müssen Sie nur eine beliebige mit JA oder NEIN beantwortbare Frage eingeben. Der Computer macht eine kurze Denkpause und zeigt dann seine „wohlüberlegte" Antwort auf dem Bildschirm an.

Sie und Ihr Computer kennen natürlich das Geheimnis und wissen, daß die Antworten einfach zufällig ausgewählt werden und keinerlei Beziehungen zu den Fragen haben. Behalten Sie aber dieses Geheimnis für sich.

Abbildung 5.7 zeigt, wie der Bildschirm bei diesem Programm aussehen kann. Nachdem Sie eine Frage eingegeben haben, zeigt der Computer diese Frage oben auf dem Bildschirm an und gibt die „passende" Antwort dazu. Dann wartet der Computer auf die nächste Frage, wenn Sie wollen, den ganzen Tag lang.

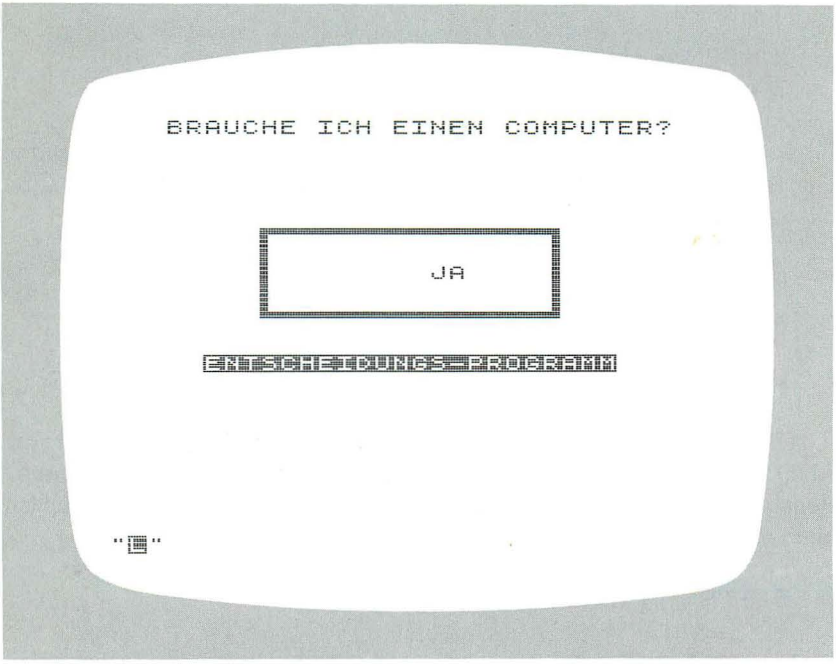


Abbildung 5.7: Beispiel zum Entscheidungs-Programm

Das Programm ist in Abbildung 5.8 aufgelistet. Wir werden die drei wichtigen Techniken, die in diesem Programm angewandt wurden, besprechen: den Gebrauch von RND, um eine zufällige Auswahl aus verschiedenen Antworten zu treffen; die Verwendung von Teilstrings, um auf die richtige Antwort zuzugreifen und schließlich den Gebrauch der LEN-Funktion, um einen String horizontal auf dem Bildschirm zu zentrieren.

Sehen Sie sich zunächst die Zeile 10 an. Darin wird der „Antwort-String" genau definiert. Dieser String ist 60 Zeichen lang und ist zusammengesetzt aus 6 Abschnitten zu jeweils 10 Zeichen, die die 6 verschiedenen Antworten des Computers enthalten:

JA
NEIN
VIELLEICHT
BESTIMMT
NA KLAR!
TJA ...

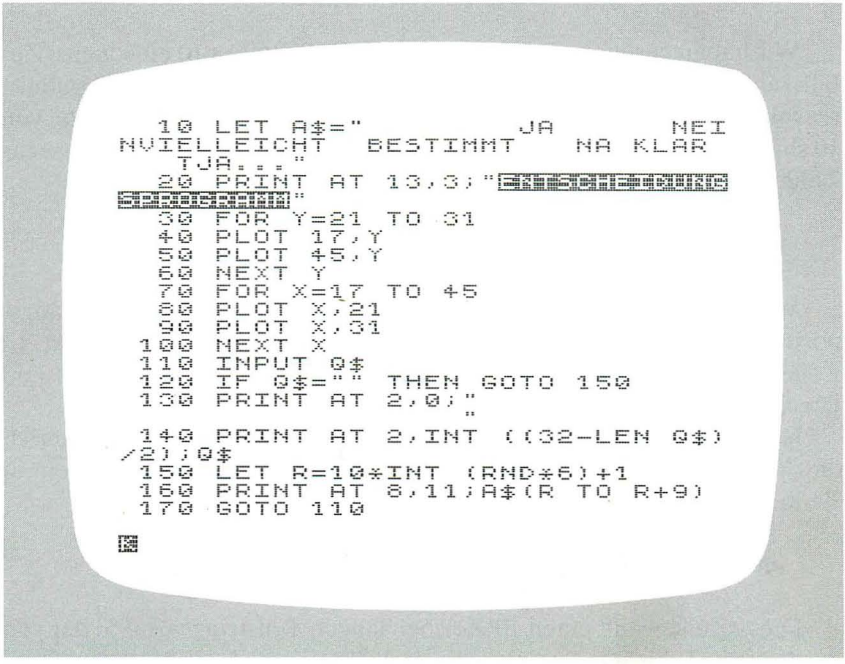


Abbildung 5.8: Das Entscheidungs-Programm

Da diese Antworten unterschiedlich lang sind, enthält der String Leerzeichen, um alle Teilstrings auf 10 Zeichen Länge aufzufüllen. Somit beginnt die erste Antwort bei A$(1) und reicht bis A$(10); die zweite Antwort reicht von A$(11) bis A$(20); die dritte von A$(21) bis A$(30) usw. Wir werden sehen, daß es durch diesen gleichmäßigen Aufbau möglich ist, Antworten aus dem String auszuwählen.

Zeile 20 zeigt die Programmüberschrift an. In den Zeilen 30 bis 100 wird mit Hilfe von zwei FOR-Schleifen und vier PLOT-Anweisungen der Rahmen für die Antwort auf den Bildschirm gezeichnet.

Zeile 110 liest die Frage von der Tastatur ein und weist sie der String-Variablen Q$ zu:

```
110 INPUT Q$
```

In den Zeilen 150 und 160 wird die Antwort ausgewählt. Zeile 150 benutzt die RND-Funktion in einer für unsere Bedürfnisse „maßgeschneiderten" Formel:

```
150 LET R = 10 * INT(RND * 6) + 1
```

Wir haben gesehen, daß der Ausdruck INT(RND * 6) eine ganze Zufallszahl zwischen 0 und 5 erzeugt. Durch Multiplikation dieses Ergebnisses mit 10 erhalten wir Zufallszahlen zwischen 0 und 50, die Vielfache von 10 sind. Wenn wir 1 dazu addieren, erhalten wir eine der folgenden sechs Zahlen:

```
1   11   21   31   41   51
```

Wie wir gesehen haben, stellen diese Zahlen die jeweiligen Anfangspositionen der sechs verschiedenen Antworten in dem String A$ dar. In Zeile 150 wird also einer dieser Anfangspunkte zufällig ausgewählt und der Variablen R zugewiesen.

Nachdem dieser Anfangspunkt festgelegt ist, wird die Antwort durch die Auswahl des entsprechenden Teilstrings bestimmt. Dies geschieht in Zeile 160:

```
160 PRINT AT 8,11; A$(R TO R + 9)
```

Diese Zeile wählt einen 10 Zeichen langen Teilstring aus A$, der von der Position R bis R + 9 reicht. Dieser Teilstring wird dann auf dem Bildschirm innerhalb des zuvor gezeichneten Rahmens angezeigt. So wird mit

Hilfe der Variablen R in Zeile 160 eine zufällig ausgewählte Antwort ausgegeben. In Zeile 170 fängt dann eine neue Spielrunde an, indem das Programm wieder mit der INPUT-Anweisung in Zeile 110 fortgesetzt wird:

```
170 GOTO 110
```

Eine andere Programmzeile verdient ebenfalls Aufmerksamkeit. Zeile 140, in der die Frage oben auf dem Bildschirm angezeigt wird, enthält eine interessante Formel, um den String Q$ horizontal zu *zentrieren*:

```
140 PRINT AT 2, INT ((32 – LEN Q$)/2); Q$
```

(Wenn Sie sehr aufmerksam waren, haben Sie vielleicht eine ähnliche Formel in dem Balken-Grafik-Programm von Kapitel 4 gesehen.) Durch den AT-Teil dieser PRINT-Anweisung wird ein String in Zeile 2 ausgegeben; die Spalten-Position wird in einer Formel berechnet, die auf der String-Länge beruht:

```
INT(32 – LEN Q$)/2)
```

Diese Formel setzt voraus, daß der betreffende String – Q$ – nicht länger als 32 Zeichen lang ist. Die LEN-Funktion ermittelt die Zahl der Zeichen in Q$. Da der Bildschirm 32 Spalten breit ist, liefert der Ausdruck (32 – LEN Q$)/2 die Adresse der Spalte, in der die Stringausgabe beginnen muß, um horizontal zentriert werden zu können. Nehmen wir z. B. an, daß Sie eine Frage eingeben, die 20 Zeichen lang ist. Um diesen String zu zentrieren, brauchen Sie 6 Leerzeichen auf beiden Seiten. Wie Sie sehen, liefert die Formel (32 – 20)/2 den richtigen Anfangswert.

Zusammenfassung

Ihr Computer verschlüsselt (codiert) jedes Tastatur-Zeichen durch eine Zahl zwischen 0 und 255. In den meisten Fällen haben Sie beim Programmieren jedoch mit diesem Code nichts zu tun. Der Computer verwendet diesen Code nur ganz privat, und Sie brauchen nicht einmal zu wissen, daß es ihn gibt. Für solche Gelegenheiten aber, in denen Sie den Code direkt benutzen wollen, gibt es die Funktionen CODE und CHR$, mit denen Zahlen und Symbole umgewandelt werden können.

Mit Verketten wird das Verknüpfen von Strings bezeichnet, um einen neuen String zu bilden. Außerdem gibt es die Möglichkeit, aus einem

String Teilstrings auszuwählen. In den Programmen dieses Kapitels haben wir Beispiele für diese beiden String-Operationen kennengelernt.

Inzwischen sollten Sie sich mit Ihrem Computer gut verstehen und mit seinen Operationen und Rechenfähigkeiten vertraut sein. Sie können nun alle möglichen im Handel erhältlichen Programme auf Ihrem Computer laufen lassen oder sich Ihre eigenen schreiben. Sie beherrschen nun die Grundlagen von BASIC, einer leistungsfähigen und vielseitigen Programmiersprache. Kurz gesagt, Sie sind nun nicht mehr nur ein Computer-*Besitzer*, sondern ein Computer-*Benutzer*.

Anhang A

Der BASIC-Wortschatz

In diesem Anhang werden die Wörter definiert, die auf der Tastatur des ZX81 aufgedruckt sind. In vielen Fällen werden dazu Beispiele von BASIC-Anweisungen angegeben, in denen diese Wörter vorkommen. Außerdem ist die Lage auf der Tastatur angegeben sowie die Gruppe, zu der das Wort gehört (Schlüsselwort, Funktion oder Umschalt- bzw. SHIFT-Funktion).

ABS (Funktion; G-Taste). Ergibt den *Absolutwert* einer Zahl.

AND (SHIFT; 2-Taste). Erzeugt zusammengesetzte Wahrheitswert-Aussagen für IF-Anweisungen, z. B.:

```
IF I > 0 AND I < 100 THEN GOSUB 300
```

In diesem Beispiel wird die GOSUB-Anweisung nach THEN nur dann ausgeführt, wenn die beiden Anweisungen I > 0 *und* I < 100 wahr sind.

ARCCOS (Funktion; S-Taste). Ergibt den Arcuscosinus einer Zahl zwischen −1 und 1. Das Ergebnis wird im Bogenmaß dargestellt. (1 im Bogenmaß = 57.3 Grad.) ARCCOS wird auf dem Bildschirm als ACS angezeigt.

ARCSIN (Funktion; A-Taste). Ergibt den Arcussinus einer Zahl zwischen −1 und 1. Das Ergebnis wird im Bogenmaß dargestellt. ARCSIN wird auf dem Bildschirm als ASN angezeigt.

ARCTAN (Funktion; D-Taste). Ergibt den Arcustangens im Bogenmaß. ARCTAN wird als ATN angezeigt.

AT (Funktion; C-Taste). Wird in Verbindung mit der PRINT-Anweisung benutzt, um die Bildschirmposition für ein Zeichen, einen String oder eine Ziffer anzugeben. Hinter AT muß die Adresse in der Form: *Zeile, Spalte* stehen, wobei *Zeile* eine Zahl zwischen 0 und 21

und *Spalte* eine Zahl zwischen 0 und 31 ist. So wird z. B. durch die Anweisung

```
PRINT AT 8,10; „HALLO"
```

das Wort HALLO in Zeile 8, beginnend in Spalte 10, angezeigt. (Die Adresse 0,0 stellt die obere linke Bildschirmecke dar.)

BREAK (Leer-Taste). Unterbricht ein laufendes Programm. BREAK kann nicht benutzt werden, wenn der Computer auf eine Eingabe von der Tastatur wartet.

CHR$ (Funktion; U-Taste). Ergibt das einer Codezahl zwischen 0 und 255 entsprechende Zeichen. Der Zeichen-Code des Computers enthält sowohl Schlüsselwörter als auch Funktionen, Buchstaben, Ziffern und andere Symbole.

CLEAR (Schlüsselwort; X-Taste). Löscht alle Variablen und ihre Werte im Speicher. Die Programmzeilen werden *nicht* gelöscht.

CLS (Schlüsselwort; V-Taste). Löscht den Bildschirm; die Ausgabe einer darauffolgenden PRINT-Anweisung beginnt in der oberen linken Bildschirmecke (Position 0,0).

CODE (Funktion; I-Taste). Ergibt die Code-Nummer des Computers für das gegebene Zeichen. Die Code-Nummern reichen von 0 bis 255.

CONT (Schlüsselwort; C-Taste). Setzt ein unterbrochenes Programm fort.

COPY (Schlüsselwort; Z-Taste). Schickt den gesamten Bildschirminhalt auf den Drucker.

COS (Funktion; W-Taste). Ergibt den Cosinus einer Zahl, die einen Winkel darstellt. Die COS-Funktion setzt voraus, daß der Winkel im Bogenmaß angegeben ist. (1 Grad = .01745 im Bogenmaß.)

DIM (Schlüsselwort; D-Taste). Definiert den Namen, Typ, die Dimensionen und die Länge eines Feldes. Z. B. wird durch

```
DIM N(10)
```

ein eindimensionales Feld N mit der Länge 10 definiert. In String-Feldern müssen alle String-Elemente gleich lang sein. Die Länge muß in der DIM-Anweisung angegeben werden. Z. B. wird durch DIM A$(10,5) ein eindimensionales String-Feld A$ definiert. A$ kann bis zu 10 String enthalten, von denen jeder bis zu 5 Zeichen lang sein darf.

EDIT (SHIFT; 1-Taste). Zeigt eine Kopie der *aktuellen Zeile* am unteren Bildschirmrand an, damit sie korrigiert werden kann. Die aktuelle Zeile ist mit einem invers dargestellten „>"-Zeichen in der Liste der Programmzeilen markiert.

EXP (Funktion; X-Taste). Ergibt den „natürlichen Exponenten" einer Zahl. (Berechnet einen Exponenten von e, wobei e = 2.7182818 ist.)

FAST (SHIFT; F-Taste). Schaltet den Computer in die schnelle Betriebsart.

FOR (Schlüsselwort; F-Taste). Bildet eine Wiederholungs-Schleife; die Zeilen innerhalb der Schleife werden entsprechend oft wiederholt. (Das Ende der Schleife wird durch eine NEXT-Anweisung gekennzeichnet.) Zum Beispiel:

```
FOR I = 2 TO 20 STEP 2
```

I ist die *Laufvariable* in dieser FOR-Anweisung. Bei jedem Schleifen-Durchgang wird I in Zweier-Schritten von 2 bis 20 erhöht.

FUNCTION (SHIFT; NEW LINE-Taste). Schaltet die Tastatur in die Funktions-Betriebsart um. Der nächste Tastendruck wird als Funktionstaste interpretiert.

GOSUB (Schlüsselwort; H-Taste). Springt zu dem Unterprogramm bei der angegebenen Zeilennummer. Die Angabe der Zeilennummer kann durch eine Zahl, eine numerische Variable oder einen numerischen Ausdruck (aus dem die Zeilennummer erreicht wird) erfolgen:

```
GOSUB 300
GOSUB T
GOSUB 100 * (VAL I$ + 4)
```

GOTO (Schlüsselwort; G-Taste). Springt zu der angegebenen Zeile. Die Angabe der Zeilennummer kann durch eine Zahl, eine numerische Variable oder einen numerischen Ausdruck erfolgen (siehe GOSUB).

GRAPHICS (SHIFT; 9-Taste). Schaltet die Tastatur in die Grafik-Betriebsart. Die nachfolgenden Tastendrücke erzeugen Grafik- oder inverse Zeichen. Das Ausschalten der Grafik-Betriebsart erfolgt ebenfalls mit der GRAPHICS-Taste.

IF (Schlüsselwort; U-Taste). Leitet eine Entscheidungs-Anweisung ein. Eine IF-Anweisung enthält eine Bedingung, die wahr oder falsch sein kann. Ist die Bedingung erfüllt, wird die hinter THEN stehende Anweisung ausgeführt; ansonsten wird das Programm mit der darauffolgenden Zeile fortgesetzt. Beispiel:

```
IF D < > 0 THEN LET Q = I/D
```

**INKEY$** (Funktion; I-Taste). Fragt die ganze Tastatur einmal ab und enthält, falls eine Taste gedrückt wird, den Wert dieser Taste. Wenn keine Taste gedrückt wurde, enthält INKEY$ einen leeren String („").

INPUT (Schlüsselwort; I-Taste). Liest Daten von der Tastatur ein und speichert sie in der angegebenen Variablen; z. B. liest

```
INPUT S$
```

einen String von der Tastatur ein und speichert diesen Wert in der Variablen S$. Die eingegebenen Daten werden gleichzeitig auf dem Bildschirm angezeigt.

INT (Funktion; R-Taste). Ergibt den ganzzahligen Wert einer Zahl.

LEN (Funktion; K-Taste). Ergibt die Länge eines Strings in Zeichen.

LET (Schlüsselwort; L-Taste). Weist einer Variablen einen Wert zu. Die Variable kann dadurch neu eingeführt werden, oder einer schon benutzten Variablen kann ein neuer Wert zugewiesen werden. Auf der linken Seite des Gleichheitszeichens muß ein Variablenname stehen. Die rechte Seite kann konstante Werte, Variablen oder Ausdrücke enthalten. Die auf der rechten Seite stehenden Variablen müssen vorher definiert sein; ihre Werte werden durch die LET-Anweisung nicht verändert. Beispiele:

```
LET V = 5
LET A = B
LET X = (5 * Y)/2
```

LIST (Schlüsselwort; K-Taste). Listet das im Computer gespeicherte Programm auf dem Bildschirm. Steht hinter LIST eine Zeilennummer, so wird das Programm von dieser Zeile ab gelistet. Beispiel:

```
LIST 100
```

LLIST (SHIFT; G-Taste). Listet das Programm auf dem Drucker.

LN (Funktion; Z-Taste). Ergibt den natürlichen Logarithmus (zur Basis e) einer positiven Zahl.

LOAD (Schlüsselwort; J-Taste). Lädt ein auf der Kassette gespeichertes Programm in den Computer. Es sind zwei Formen möglich, mit Angabe des Programmnamens:

```
LOAD "NAME"
```

oder ohne Angabe eines Programmnamens:

```
LOAD ""
```

Die Anführungszeichen sind in beiden Fällen erforderlich. Im zweiten Fall lädt der Computer das erste Programm, das er auf der Kassette findet.

LPRINT (SHIFT; S-Taste). Druckt eine Zeile auf dem Drucker.

NEW (Schlüsselwort; A-Taste). Löscht das im Speicher des Computers stehende Programm.

NEW LINE schließt eine Eingabe in den Computer ab.

NEXT (Schlüsselwort; N-Taste). Kennzeichnet das Ende einer FOR-Schleife. Die in der FOR-Anweisung definierte Laufvariable muß hinter dem Wort NEXT stehen, z. B.:

```
NEXT I
```

NOT (Funktion; N-Taste). Verändert den Wahrheitswert einer Anweisung. Beispiel:

```
IF NOT D > 0 THEN GOTO 10
```

Diese Anweisung ist der folgenden gleichwertig:

```
IF D <= 0 THEN GOTO 10
```

OR (SHIFT; W-Taste). Kann Bedingungen in IF-Anweisungen miteinander verknüpfen. Im folgenden Beispiel werden zwei Vergleiche miteinander verknüpft:

```
IF X = 5 OR Y = 3 THEN GOSUB 300
```

Die hinter THEN stehende GOSUB-Anweisung wird dann ausgeführt, wenn eine oder beide Gleichungen erfüllt sind.

PAUSE (Schlüsselwort; M-Taste). Verzögert die Programmausführung in Abhängigkeit von der angegebenen ganzen Zahl; z. B. bewirkt

```
PAUSE 50
```

eine Pause von etwa einer Sekunde. Wenn die hinter PAUSE stehende Zahl größer ist als 32766, bedeutet die Anweisung „unbegrenzt lange Pause". Eine Pause kann durch Drücken einer beliebigen Taste abgebrochen werden, so daß das Programm fortgesetzt wird.

PEEK (Funktion; 0-Taste). Ergibt den Inhalt einer Speicherzelle von 0 bis 65535.

PLOT (Schlüsselwort; Q-Taste). Bringt ein „Pixel" (Bildpunkt) an einer bestimmten Stelle auf den Bildschirm. Die Elemente einer PLOT-Adresse sind andersherum angeordnet als die Elemente einer PRINT AT-Adresse. Die allgemeine Form der PLOT-Anweisung ist:

```
PLOT h,v
```

wobei h die horizontale Adresse (von 0 bis 63) und v die vertikale Adresse (von 0 bis 43) ist. Die PLOT-Adresse 0,0 befindet sich in der unteren linken Bildschirmecke.

POKE (Schlüsselwort; O-Taste). Schreibt einen Wert in die angegebene Speicheradresse zwischen 0 und 65535. Beispiel:

```
POKE a,w
```

wobei a die Speicheradresse und w der Wert ist. Der absolute Wert von w kann nicht größer sein als 255.

PRINT (Schlüsselwort; P-Taste). Gibt eine Zeile auf dem Bildschirm aus. Werden in einer PRINT-Anweisung mehrere Elemente durch Semikolon getrennt angegeben, so erscheinen sie ohne Zwischenräume auf dem Bildschirm. Ein Komma zwischen den Elementen bewirkt eine Ausgabe am linken Bildschirmrand oder in der Bildschirmmitte, je nachdem, welche Position näher liegt.

RAND (Schlüsselwort; T-Taste). Setzt den Beginn einer Folge von durch RND erzeugten Zufallszahlen auf einen neuen Anfangswert. Bei den folgenden Anweisungen sind die Ergebnisse der Folgen nicht vorhersagbar:

```
RAND
RAND 0
```

Durch die folgende Anweisung, in der n größer ist als 0 und kleiner als 65536, wird ein bestimmter Anfangswert festgesetzt, so daß jedesmal die gleiche Folge von Zufallszahlen entsteht:

```
RAND n
```

REM (Schlüsselwort; E-Taste). Kennzeichnet eine Kommentar-Zeile, die vom Computer während des Programmlaufs nicht beachtet wird. Hinter dem Schlüsselwort REM kann ein beliebiger Text stehen.

RETURN (Schlüsselwort; Y-Taste). Kennzeichnet das Ende eines Unterprogramms. Das Programm wird mit der Zeile fortgesetzt, die *hinter* der aufrufenden GOSUB-Anweisung steht.

RND (Funktion; T-Taste). Erzeugt eine „Zufalls"-Zahl, die größer oder gleich Null und kleiner als eins ist.

RUBOUT (SHIFT; 0-Taste). Löscht das unmittelbar links neben dem Cursor stehende Zeichen, Schlüsselwort oder die Funktion; kann sowohl bei der Eingabe einer neuen Zeile als auch beim Korrigieren einer Zeile benutzt werden.

RUN (Schlüsselwort; R-Taste). Beginnt mit der Ausführung der im Speicher des Computers stehenden Programmzeilen. (Zuvor werden alle Variablen gelöscht.) Ist hinter RUN eine Programmzeile angegeben, so beginnt die Programmausführung mit dieser Zeile.

SAVE (Schlüsselwort; S-Taste). Speichert ein Programm auf einer Tonband-Kassette. Die SAVE-Anweisung erfordert einen Programmnamen, der in Anführungszeichen stehen muß:

```
SAVE "NAME"
```

SCROLL (Schlüsselwort; B-Taste). Schiebt den gesamten Bildschirminhalt um eine Zeile nach oben. Die oberste Zeile geht verloren, dafür ist unten eine Zeile freigeworden.

SGN (Funktion; F-Taste). *Vorzeichen*-Funktion, ergibt die Werte −1, 0 oder 1, je nachdem ob das Argument negativ, gleich Null oder positiv ist.

SLOW (SHIFT; D-Taste). Schaltet den Computer in die langsame Betriebsart.

SPACE erzeugt ein Leerzeichen.

SQR (Funktion; H-Taste). Berechnet die Quadratwurzel aus einer nichtnegativen Zahl.

STEP (SHIFT; E-Taste). Gibt die (postive oder negative) Schrittweite der Laufvariablen in einer FOR-Anweisung an, z. B.:

```
FOR I = 5 TO 25 STEP 5
```

In dieser FOR-Schleife wird die Laufvariable I bei jedem Schleifendurchgang um 5 erhöht. Wenn die STEP-Anweisung fehlt, wird eine Schrittweite von 1 angenommen.

STOP (SHIFT; A-Taste). Beendet die Ausführung eines Programms. Das Wort STOP kann in einer Programmzeile stehen oder auch von der Tastatur eingegeben werden, um ein Programm zu beenden.

**STR$** (Funktion; Y-Taste). Macht aus einer Zahl einen String. Zum Beispiel bewirkt die folgende Anweisung, daß ein String „1234" erzeugt und in der Variablen S$ gespeichert wird:

```
LET S$ = STR$ 1234
```

TAB (Funktion; P-Taste). Wird zusammen mit der PRINT-Anweisung benutzt, um eine Ausgabe in einer bestimmten Spalte einer Zeile beginnen zu lassen. Das folgende Beispiel beginnt mit der Ausgabe des Wortes HALLO in der Spalte 15:

```
PRINT TAB 15; "HALLO"
```

TAN (Funktion; E-Taste). Ergibt den Tangens einer Zahl, die einen Winkel darstellt. TAN setzt voraus, daß der Winkel im Bogenmaß angegeben wird (1 Grad = .01745 Bogengrad).

THEN (SHIFT; 3-Taste). Bestandteil einer IF-Anweisung. Nach THEN steht immer ein Schlüsselwort. Die nach THEN folgende Anweisung wird nur dann ausgeführt, wenn die hinter IF stehende Bedingung erfüllt ist.

TO (SHIFT; 4-Taste). Bestandteil einer FOR-Schleife, z. B.:

```
FOR I = 1 TO 10
```

TO wird auch bei der Verarbeitung von Teilstrings benutzt. Die

folgende LET-Anweisung ersetzt die 2. bis 4. Position des Strings S$ durch den String „ABC“:

```
LET S$(2 TO 4) = "ABC"
```

UNPLOT (Schlüsselwort; W-Taste). Löscht an der angegebenen Stelle ein Pixel (Bildpunkt) vom Bildschirm. Die allgemeine Form der UNPLOT-Anweisung lautet:

```
UNPLOT h,v
```

wobei h die horizontale Adresse (von 0 bis 63) und v die vertikale Adresse (von 0 bis 43) ist. Die UNPLOT-Adresse 0,0 befindet sich in der unteren linken Bildschirmecke.

USR (Funktion; L-Taste). Aufruf eines Maschinensprache-Unterprogramms an der angegebenen Speicheradresse. Die allgemeine Form lautet:

```
USR a
```

wobei a die Speicheradresse ist, an der das Programm beginnt.

VAL (Funktion; J-Taste). Ergibt den numerischen Wert eines Strings. Die Zeichen des Strings müssen aus Ziffern, Funktionen, Rechenoperationen oder Variablen bestehen, die einen zulässigen numerischen Ausdruck darstellen.

Anhang B

Die Fehlermeldungen des ZX81

Nach jedem Programmlauf erscheint in der unteren linken Bildschirmecke eine Meldung in der Form:

m/n

wobei n die Nummer der Zeile angibt, mit der das Programm beendet wurde, und m einen der folgenden Fehler-Codes darstellt:

0 Kein Fehler; Programm beendet.

1 Unterschiedliche Variablennamen in FOR- und NEXT-Anweisung.

2 Undefinierter Variablenname verwendet.

3 Feld-Index liegt außerhalb des in der DIM-Anweisung angegebenen Bereichs.

4 Speicherplatz vollständig belegt.

5 Bildschirm voll.

6 Das Ergebnis einer Berechnung ist eine für den Computer zu große Zahl.

7 RETURN-Anweisung ohne GOSUB-Anweisung.

8 INPUT wurde als direktes Kommando benutzt.

9 Programm wurde mit einer STOP-Anweisung beendet.

A Unzulässiges Argument in einer Funktion (z. B. SQR −1).

B Unzulässige Adresse, Code-Nummer oder andere Zahlenangabe in Anweisungen wie PRINT AT, PLOT, CHR$ usw.

C Der String-Ausdruck hinter VAL ergibt keinen gültigen numerischen Ausdruck.

D Programm wurde durch BREAK oder STOP von der Tastatur aus abgebrochen.

E Wird nicht verwendet.

F SAVE-Anweisung ohne Programmname.

Literatur zum ZX81

R. Zaks: Programmierung des Z80, Sybex-Verlag

E. Flögel: Programmierung des ZX81 in BASIC und Maschinencode, Hofacker-Verlag

R. Hurley: The Sinclair ZX81, Programming for Real Applications, Macmillan Press Ltd.

R. Hurley: More Real Applications for the ZX81 and ZX Spectrum, Macmillan Press Ltd.

Sinclair: ZX81 BASIC Programmier-Handbuch

T. Hartnell: 49 explosive Spiele für den Sinclair ZX81, Verlag COOPERATION

A. Gourlay: 34 1K-Superspiele für den Sinclair ZX81, Verlag COOPERATION

T. Toms: Das ZX81 Buch, Verlag COOPERATION

T. Hartnell: Entdecken Sie die unendlichen Dimensionen Ihres ZX81, Verlag COOPERATION

Bei einigen Geräten findet sich lediglich eine kurzgefaßte, etwas dürftige Bedienungsanleitung. Das Original-Handbuch in Deutsch erhält man bei:

Jürgen Schumpich
Internationale Industrievertretungen GmbH
Ottostraße 28
8012 Ottobrunn

Stichwortverzeichnis

ABS 113, 159
Addition 109
Adresse 66
aktuelle Zeile 54
AND 105, 108, 159
Anschlußmöglichkeiten 15
Anweisung 31
ARCCOS 113, 159
ARCSIN 113, 159
ARCTAN 113, 159
arithmetische Funktionen 109, 113
arithmetische Operationen 109
AT 34, 65, 78, 108, 159
Ausgabe 13

Balken-Diagramme 109, 130
BASIC 10, 11, 24
BASIC-Programmierer 25
Begrüßungsprogramm 30
Benutzung der Tastatur 30
Berechnung beim INPUT 115
Betriebsart 33, 34, 44
Bildschirm 19
BREAK 58, 81, 85, 99, 160
Buchstaben-Betriebsart 46
Buchstaben-Taste 33
Byte 22

Chip 17
CHR$ 30, 49, 143, 144, 145, 160
CLEAR 72, 160
CLS 30, 45, 122, 160
CODE 49, 143, 145, 160
Computer 18
Computersprache 24
CONT 160

COPY 24, 160
COS 113, 160
Cursor 33
Cursor-F 36
Cursor-G 46
Cursor-K 43
Cursor-L 36, 46, 51
Cursor-S 39, 44

Dezimalpunkt 114
DIM 125, 138, 160
direkte Kommandos 64
Division 109
Division durch Null 112
Dokumentation 61, 103
Dollar-Zeichen 41
Drucker 17, 23

Edit 44, 161
Edit-Taste 54
Editieren 19, 42
Eingabe 13
Entscheidungen 85
Ergebnisse 51
EXP 113, 161
Exponential-Darstellung 123
Exponentiation 109

FAST 45, 96, 121, 161
Fehler 19, 25, 42
Fehlermeldungen 12, 70
Felder 109, 125, 126
Feld-Elemente 126
Feld-Index 126
Feld-Laenge 126
Feld-Name 126

Fernsehgerät 15, 18
FOR 45, 75, 94, 108, 139, 161
Funktion 34, 161
Funktions-Betriebsart 44
Funktions-Eingabe-Taste 36

Gleichheitszeichen 87
Gleichung 86, 105
Gleitkomma-Darstellung 102
GOSUB 82, 101, 105, 108, 161
GOTO 81, 104, 108, 121, 161
Grafik 46, 161
Grafik-Betriebsart 45
grafische Zeichen 45

Hardware 10

IF 86, 105, 108, 162
INKEY$ 101, 104, 108, 162
INPUT 41, 72, 73, 101, 104, 108, 162
INT 113, 153, 162
Inverse Darstellung 20, 33, 49

Kalkulationsprogramme 59
Kassetten-Recorder 15, 55
Kauf von Software 61
Klammerebene 111
Komma 66, 112
Kommando 15, 43
Kontroll-Variable 75

Länge 147
langsame Betriebsart 46
Laufvariable 93, 161
leerer String 146
LEN 49, 143, 162
LET 69, 92, 108, 162
LIST 19, 54, 62, 91, 162
LLIST 24, 163
LN 113, 163
LOAD 56, 64, 163
LPRINT 24, 163

Mikroprozessor 18
Mittelwert 135
Modulator 20
Multiplikation 109

Netzstecker 16
NEW 64, 163
NEW LINE 39, 43, 44, 99, 163
NEXT 45, 46, 108, 163
nichtnumerisch 11
NOT 163
Null 36
numerisch 41

OR 105, 108, 163

PAUSE 164
PEEK 164
Personal-Computer 9
Pfeile 42
Pixel 67
Platine 17
PLOT 67, 75, 108, 164
POKE 164
PRINT 33, 64, 65, 70, 108, 164
Programm 19, 23
Programmeingabe 11
Programmstart 19, 23

RAM 21, 22, 23
RAND 164
rechnen 110
Reihenfolge der Operationen 110
REM 103, 108, 165
RETURN 83, 108, 165
RND 151, 165
ROM 21
RUBOUT 37, 43, 44, 51, 165
RUN 50, 64, 165
Rundungs-Funktion 114

SAVE 56, 64, 165
SCROLL 165
Schleife 45, 93
Schleifenschachtelung 141
schnelle Betriebsart 45
Semikolon 66, 71, 77
SGN 113, 165
Shift 36, 43
Simulation 152
SIN 113, 165
SLOW 46, 96, 165
Software 11, 59
SPACE 166
Spalte 65
Speicher 11, 22

Speichererweiterung 13
Speicherung 102
SQR 113, 166
STEP 79, 94, 108, 166
STOP 84, 166
STR$ 166
String 41
Stringfelder 128
Stringvariable 71
Stringverarbeitung 146
Stromversorgung 15
Subtraktion 109
Syntax-Fehler 39, 44

TAB 101, 108, 166
Tabelle 125
TAN 113, 166
Tastatur 9, 13
Teilstrings 149
Texte 41
THEN 86, 87, 104, 108, 166
TO 76, 108, 166

Umschaltmöglichkeiten 15
Ungleichung 86, 105
Ungleichheitszeichen 87
UNPLOT 67, 108, 167
Unterprogramme 82
USR 167

VAL 106, 122, 167
Variablen-Name 41
Variablentypen 69
Verkettung 148
Verschlüsselung 143
VU-CALC 59

wahr und falsch 86
Wert 69

Z80-Mikroprozessor 17
Zeichen 143
Zeichenketten 11
Zeile hinzufügen 53
Zeile löschen 53
Zeilen 65
Zeilennummer 31, 34
zentrieren 155
Zufallszahl 151
Zusatz-Speicher 17

Die SYBEX Bibliothek

MEIN ERSTER COMPUTER (2. überarbeitete Ausgabe)
von Rodnay Zaks – eine Einführung für alle, die den Kauf oder die Nutzung eines Mikrocomputers erwägen. 305 Seiten, 150 Abbildungen, Format DIN A5, Ref.Nr.: **3020** (1982)

CP/M HANDBUCH MIT MP/M
von Rodnay Zaks – ein umfassendes Lehr- und Nachschlagewerk für CP/M, das Standard Betriebssystem für Mikrocomputer. 310 Seiten, 100 Abbildungen, Format DIN A5, Ref.Nr. **3002** (1981)

MIKROPROZESSOR INTERFACE TECHNIKEN (3. überarbeitete Ausgabe)
von Rodnay Zaks/Austin Lesea – Hardware und Software Verbindungstechniken samt Digital/Analog-Wandler, Peripheriegeräte, Standard-Busse und Fehlersuchtechniken. 435 Seiten, 400 Abbildungen, Format DIN A5, Ref.Nr.: **3012** (1982)

PROGRAMMIERUNG DES 6502 (2. überarbeitete Ausgabe)
von Rodnay Zaks – Programmierung in Maschinensprache mit dem Mikroprozessor 6502, von den Grundkonzepten bis hin zu fortgeschrittenen Informationsstrukturen. 350 Seiten, 160 Abbildungen, Format DIN A5, Ref.Nr.: **3011** (1982)

PROGRAMMIERUNG DES Z80
von Rodnay Zaks – ein kompletter Lehrgang in der Programmierung des Z80 Mikroprozessors und eine gründliche Einführung in die Maschinensprache. 630 Seiten, 200 Abbildungen, Format DIN A5, Ref.Nr.: **3006** (1982)

EINFÜHRUNG IN PASCAL UND UCSD/PASCAL
von Rodnay Zaks – das Buch für jeden, der die Programmiersprache PASCAL lernen möchte. Vorkenntnisse in Computerprogrammierung werden nicht vorausgesetzt. Eine schrittweise Einführung mit vielen Übungen und Beispielen. 540 Seiten, 130 Abbildungen, Ref.Nr.: **3004** (1981)

DAS PASCAL HANDBUCH
von Jacques Tiberghien – ein Wörterbuch mit jeder Pascal Anweisung und jedem Symbol, reservierten Wort, Bezeichner und Operator, für beinahe alle bekannten Pascal-Versionen. 510 Seiten, 270 Abbildungen, Format 23 x 18 cm, Ref.Nr.: **3005** (1982)

PASCAL PROGRAMME FÜR WISSENSCHAFTLER UND INGENIEURE
von Alan Miller – eine Sammlung von 60 der wichtigsten wissenschaftlichen Algorithmen samt Programmauflistung und Musterdurchlauf. Ein wichtiges Hilfsmittel für Pascal Benutzer mit technischen Anwendungen. 320 Seiten, 120 Abbildungen, Format 23 x 18 cm, Ref.Nr.: **3007** (1982)

POCKET MIKROCOMPUTER LEXIKON
– die schnelle Informations-Börse! 1300 Definitionen, Kurzformeln, technische Daten, Lieferanten-Adressen, ein englisch-deutsches und französisch-deutsches Wörterbuch. 176 Seiten, Format DIN A6, Ref.Nr. **3008** (1982)

BASIC COMPUTER SPIELE/Band 1
herausgegeben von David H. Ahl – die besten Mikrocomputerspiele aus der Zeitschrift „Creative Computing" in deutscher Fassung mit Probelauf und Programmlisting. 208 Seiten, 59 Abbildungen, Ref.Nr. **3009**

BASIC COMPUTER SPIELE/Band 2
herausgegeben von David H. Ahl – 84 weitere Mikrocomputerspiele aus „Creative Computing". Alle in Microsoft-BASIC geschrieben mit Listing und Probelauf. 224 Seiten, 61 Abbildungen, Ref.Nr.: **3010**

BASIC PROGRAMME FÜR WISSENSCHAFTLER UND INGENIEURE
von Alan Miller – eine Bibliothek von Programmen zu den wichtigsten Problemlösungen mit numerischen Verfahren, alle in BASIC geschrieben, mit Musterlauf und Programmlisting. 320 Seiten, 120 Abbildungen, Ref.Nr.: **3015** (Erscheint Herbst 1983)

EINFÜHRUNG IN DIE TEXTVERARBEITUNG
von H. Glatzer – aus was eine Textverarbeitungsanlage besteht, wie man sie nutzen kann und zu was sie fähig ist. Beispiele verschiedener Anwendungen und Kriterien für den Kauf eines Systems. 208 Seiten, 67 Abbildungen, Ref.Nr. **3018** (1983)

EINFÜHRUNG IN WORDSTAR
von A. Naiman – eine klar gegliederte Einführung, die aufzeigt, wie WORDSTAR funktioniert, was man damit tun kann und wie es eingesetzt wird. 208 Seiten, 30 Abbildungen, Ref.Nr.: **3019** (1983)

6502 ANWENDUNGEN
von Rodnay Zaks – das Eingabe-/Ausgabe-Buch für Ihren 6502-Mikroprozessor. Stellt die meistgenutzten Programme und die dafür notwendigen Hardware-Komponenten vor. 288 Seiten, 205 Abbildungen, Ref.Nr.: **3014** (1983)

BASIC ÜBUNGEN FÜR DEN APPLE
von J.-P. Lamoitier – das Buch für APPLE-Nutzer, die einen schnellen Zugang zur Programmierung in BASIC suchen. Abgestufte Übungen mit zunehmendem Schwierigkeitsgrad. 240 Seiten, 185 Abbildungen, Ref.Nr.: **3016** (1983)

CHIP UND SYSTEM: Einführung in die Mikroprozessoren-Technik
von Rodnay Zaks – eine sehr gut lesbare Einführung in die faszinierende Welt der Computer, vom Microprozessor bis hin zum vollständigen System. 560 Seiten, 325 Abbildungen, Ref.Nr.: **3017** (Erscheint Herbst 1983)

VORSICHT! Computer brauchen Pflege
von Rodnay Zaks – das Buch, das Ihnen die Handhabung eines Computersystems erklärt – vor allem, was Sie damit nicht machen sollten. Allgemein gültige Regeln für die pflegliche Behandlung Ihres Systems. 224 Seiten, 96 Abbildungen, Ref.Nr.: **3013** (Erscheint April 1983)

MEIN SINCLAIR ZX81
von D. Hergert – eine gut lesbare Einführung in diesen Einplatincomputer und dessen Programmierung in BASIC. 180 Seiten, 20 Abbildungen, Ref: **3021** (Erscheint Februar 1983)

FORDERN SIE EIN GESAMTVERZEICHNIS UNSERER VERLAGSPRODUKTION AN:

SYBEX-VERLAG GmbH
Heyestraße 22
4000 Düsseldorf 12
Tel.: (02 11) 28 70 66
Telex: 8 588 163

SYBEX-EUROPE
4 Place Felix Eboué
75583 Paris Cedex 12
Tel.: 1/347-30-20
Telex: 211801

SYBEX INC.
2344 Sixth Street
Berkeley, CA 94710, USA
Tel. (415) 848-8233
Telex: 336311

Mein Sinclair ZX81

Dieses Buch erklärt Ihnen von Anfang an mit einfachen Beispielen wie Sie Ihren ZX81 voll nutzen können. Schritt für Schritt lernen Sie, wie ...

- Ihr ZX81 an den Fernseher und den Cassettenrecorder angeschlossen wird, so daß sie als System zusammenarbeiten
- Sie die Tastatur nutzen, um Ihrem Computer Befehle zu geben
- Sie für Graphik-, Rechnungs-, Spiel- und andere Anwendungen Ihre eigenen Programme schreiben können
- Sie mit Ihrem ZX81 die Programmiersprache BASIC am besten einsetzen können
- Sie die bereits lauffähigen Programme in diesem Buch sofort zur Anwendung bringen.

Verwandeln Sie Ihren Sinclair in eine Super-Rechenmaschine, entwickeln Sie Balkendiagramme, um Ihre Zahlen bildhaft zu machen, oder zeichnen Sie Bilder auf Ihren Fernsehschirm. **Mein Sinclair ZX81** hilft Ihnen, aus Ihrem Computer das Beste herauszuholen.

Über den Autor

D. Hergert ist Mitglied des SYBEX-Lektorats und Autor der Bücher „Basic for Business" und „Mastering Visicalc". Aufbauend auf seiner breiten Computererfahrung hat er an mehreren SYBEX-Titeln als Lektor und Übersetzer mitgewirkt.

3-88745-021-3

SYBEX