MSX-

Wichert van Engelen

Computer-Anwenderhandbuch

McGraw-Hill

Wichert van Engelen · MSX-Computer-Anwenderhandbuch

Wichert van Engelen

MSX-Computer-Anwenderhandbuch

McGraw-Hill Book Company GmbH

Hamburg · New York · St. Louis · San Francisco · Auckland · Bogotá · Guatemala
Johannesburg · Lissabon · London · Madrid · Mexiko · Montreal · New Delhi
Panama · Paris · San Juan · São Paulo · Singapur · Sydney · Tokio · Toronto

Titel der Originalausgabe:
De MSX Gebruikersgids
Wichert van Engelen

Wichert van Engelen:
MSX-Computer-Anwenderhandbuch
Hamburg: McGraw-Hill Book Company GmbH, 1985
ISBN 3-89028-034-X

Umschlaggestaltung: Conny Zug
Übersetzung: Franz J. Lukassen, Erftstadt
Satz, Druck und Bindung: Druckerei Bitsch GmbH, Birkenau

INHALTSVERZEICHNIS

1. Einleitung 1
2. Der MSX-Computer 3
3. Die Bildschirmarten 17
4. Direkte Befehle 21
5. Ein Programm schreiben 27
6. Variablen 37
7. Eingabe und Bildwiedergabe 43
8. Schleifen 55
9. Logik und IF...THEN 65
10. Strings: Der Computer als Textverarbeitungsgerät 65
11. Listen mit Daten 91
12. Sortieren 103
13. Dateien bearbeiten 109
14. Abstecher 123
15. Einfache Grafik 135
16. Grafik und Farbe 149
17. Grafik: Hohe Auflösung 157
18. Grafik: Sprites 177
19. Geräusche und Musik 189
20. Die dritte Dimension 205

Anhang: ASCII-Code und Fehlermeldungen 255

Sachwortverzeichnis 233

INHALTSVERZEICHNIS

1. EINLEITUNG

Ein Programm schreiben

Bereits bei Ihrer ersten Begegnung mit der Welt der Mikrocomputer haben Sie viel über die Abfassung von Programmen gehört. Die MSX-Computer sollen dazu ja ganz besonders geeignet sein und über ein ausgefeiltes BASIC verfügen.

Aber was heißt das eigentlich: „Ein Programm schreiben" ? Weshalb wurden denn zahllose Bücher darüber geschrieben? Weshalb sollten wir überhaupt selber programmieren? Es gibt doch schließlich im Handel allerlei Kassetten mit den verschiedensten Programmen und Spielen. Sie ersparen uns eine Menge Tipparbeit, und nachdenken muß man auch so gut wie gar nicht mehr.

Hier haben Sie auch schon einen der besten Gründe, das Programmieren doch selber zu erlernen. Das Nachdenken. Wenn Sie mehr wissen wollen über das, was Computer überhaupt sind und was sie können, sollten Sie es nicht einfach dabei belassen, eine Kassette mit einem Spiel oder einem Buchführungsprogramm in den Rekorder zu schieben.

Aber selbst wenn Sie finden, daß es eigentlich Zeitverschwendung ist, programmieren zu lernen, werden Sie schon recht bald feststellen, daß Sie darüber sehr wohl ein bißchen Bescheid wissen müssen. In der Computerwissenschaft ist man noch nicht so weit, daß alle Programme auch wirklich „passen". Wenn Sie ein fix und fertiges Programm optimal anwenden wollen, müssen Sie entweder Ihre Anforderungen darauf abstimmen, oder aber Sie müssen das Programm Ihren Erfordernissen anpassen. Darüber hinaus ist es für die Anwendung vieler Programme unerläßlich, daß der Benutzer die Arbeitsweise seines Computers kennt. Selber Programme für Ihren MSX-Computer zu schreiben ist die beste Methode, mit dem Computer allmählich vertraut zu werden und zu verstehen, wie Ihr MSX funktioniert. Mit diesem Buch und anhand zahlreicher Übungen lernen Sie, wie die MSX-Computer arbeiten und was Sie damit alles machen können. Sie lernen, wie Sie eine Problemstellung in die Computersprache übersetzen müssen und wie Sie Probleme, die zu komplex oder zu vage sind, für einen Computer in verschiedene kleine und weniger vage Problemstellungen zergliedern können.

Für die Arbeit an diesem Buch wurde ein SONY HIT-BIT, Typ HB-75P benutzt. Aufgrund der weitgehenden Austauschbarkeit der MSX-Computer ist dieses Buch jedoch für jedes MSX-System geeignet.

Die Programme in diesem Buch sind meist recht einfach gehalten und Sie werden sie bald verstehen. Die Programme in den letzten Kapiteln setzen den vorher behandelten Stoff als bekannt voraus und sind infolgedessen komplizierter. Obgleich alle Programme ihre Aufgaben erfüllen, sind sie nicht wirklich „fertig". In jedem Programm gibt es immer noch Platz für Verbesserungen oder Angleichungen an den persönlichen Geschmack des Benutzers. Mittels des Textes zu jedem Programm soll verdeutlicht werden, wie das Programm aufgebaut ist und von welchen Tips und Tricks Gebrauch gemacht wurde. Auf diese Weise wird der Leser eventuelle Anpassungen selber rasch vornehmen können.

Kommunikation Mensch - Computer

Die Sprache, in der wir uns mit dem Computer verständigen, heißt BASIC. Dies ist eine Abkürzung für Beginners All-purpose Symbolic Instruction Code (Für alle Zwecke geeigneter symbolischer Befehlskode für Anfänger). Diese Sprache wurde in den 60er Jahren in den USA entwickelt. Seither sind verschiedene Dialekte dieser Sprache in Gebrauch, und einer der jüngsten ist MSX-BASIC. Es handelt sich hier um einen Dialekt des allgemeinen BASIC, das von der amerikanischen Firma Microsoft speziell für die MSX-Computer entwickelt wurde.

Neben BASIC gibt es in der Welt der Computer noch zahlreiche andere Computersprachen, die alle besonders für fortgeschrittene Computerbenutzer geeignet sind, welche das Gerät zu einem ganz bestimmten Zweck einsetzen: PASCAL (mathematische Aufgaben), FORTRAN (Verwaltung), LISP (künstliche Intelligenz), FORTH (Prozeßsteuerung). Daneben gibt es noch Computersprachen, die speziell für Kinder gedacht sind. Die bekannteste ist LOGO.

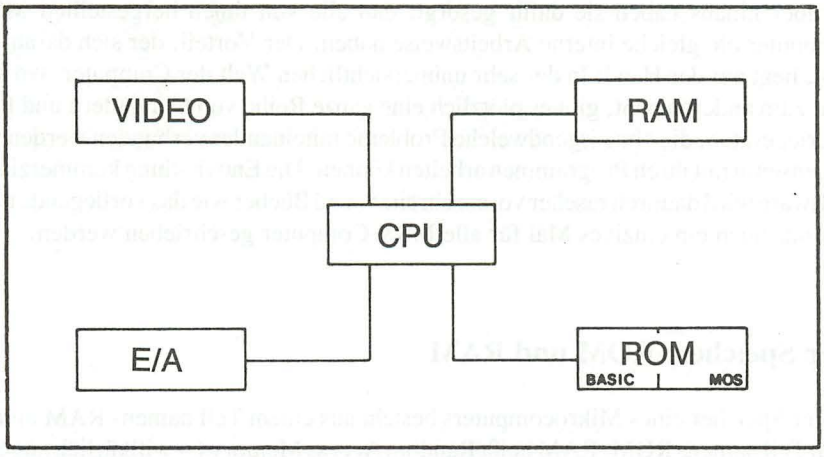
Mit BASIC können Sie alle Aufgaben erfüllen. Zuweilen ist das Programmieren in BASIC ein bißchen umständlicher oder weniger elegant als in einer der anderen Sprachen, aber der Vorteil, den die Einfachheit von BASIC bietet, ist so groß, daß so gut wie alle Mikrocomputer standardmäßig mit BASIC geliefert werden.

Auch die MSX-Computer werden mit einem BASIC-Interpreter (=Übersetzer) geliefert. Wie oben bereits erwähnt, weicht das BASIC des MSX-Computers in einigen Punkten vom STANDARDBASIC ab. Im Normalfalle ergeben sich daraus für den Benutzer keine Nachteile. MSX-BASIC kann alles, was auch STANDARDBASIC kann, und noch einiges mehr. Lediglich wenn Sie auf dem MSX Programme für andere (nicht MSX-) Computer schreiben wollen, müssen Sie darauf achten, daß Sie keine Befehle benutzen, die es in STANDARDBASIC nicht gibt.

2. DER MSX-COMPUTER

Das Innenleben

Der MSX-Computer setzt sich – wie jeder andere Mikrocomputer auch – aus folgenden Bestandteilen zusammen:



Stark vereinfachtes Schema eines MSX-Computers

Die CPU

Die CPU (Central Processing Unit = Zentraleinheit) ist der zentrale Teil des Computers. Er bestimmt, welche Informationen wohin gehen. Auf diese Weise verarbeitet er die Informationen. Jede Information, die die CPU erhält, wird zunächst untersucht, wonach die CPU festlegt, was weiter damit zu geschehen hat. Die CPU könnte vielleicht mit der Leitung einer großen Organisationsabteilung verglichen werden.

Die CPU eines jeden MSX-Computers ist eine Z-80A der amerikanischen Firma Zilog, und sie besteht aus einem einzigen (aber dennoch komplizierten) Chip.

Dies ist vielleicht genau die richtige Stelle, um mit einer häufig auftauchenden Sprachverwirrung aufzuräumen. Viele Leute machen keinen Unterschied zwischen Mikrocomputer und Mikroprozessor. Ein Mikroprozessor ist der wichtigste Baustein eines Mikrocomputers. Ein Mikrocomputer ist eine Maschine wie der MSX, den Sie möglicherweise in der Nähe stehen haben. Das heißt: eine Anordnung aus Tastatur, einem Kasten, einigen Chips, verschiedenen weiteren elektronischen Bauteilen, sowie Drähten und anderen Teilen. Dieser Mikrocomputer ist dazu in der

Lage, eine ganze Menge von Befehlen auszuführen, die ihm der Benutzer erteilt und die Resultate daraus mittels eines Kabels über Bildschirm, Printer, Kassettenrekorder oder andere Peripheriegeräte wiederzugeben.

Ein Mikroprozessor ist der zentrale und charakteristische Teil eines Mikrocomputers. Er bildet die CPU. Derzeit gibt es viele Fabrikate und Arten von Mikrocomputern auf dem Markt, aber nur einige wenige Mikroprozessoren. Deshalb haben die verschiedensten Computer die gleichen Mikroprozessoren.

Die japanischen Computerhersteller sind noch weiter gegangen und verwenden nicht nur alle die gleichen Prozessoren, sondern auch die gleichen Speicherchips; darüber hinaus haben sie dafür gesorgt, daß alle von ihnen hergestellten MSX-Computer die gleiche interne Arbeitsweise haben. Der Vorteil, der sich daraus ergibt, liegt auf der Hand: In der sehr unübersichtlichen Welt der Computer, wo kein Teil zum anderen paßt, gibt es plötzlich eine ganze Reihe von Computern und Peripheriegeräten, die ohne irgendwelche Probleme miteinander verbunden werden und gegenseitig mit ihren Programmen arbeiten können. Die Entwicklung kommerzieller Software wird dadurch rascher voranschreiten, und Bücher wie das vorliegende müssen nur noch ein einziges Mal für alle MSX-Computer geschrieben werden.

Der Speicher: ROM und RAM

Der Speicher eines Mikrocomputers besteht aus einem Teil namens RAM und einem Teil namens ROM. RAM heißt Random Access Memory (=willkürlich zugänglicher Speicher: der frei verfügbare Speicher), und ROM ist die Abkürzung für Read Only Memory (=nur Lesespeicher).

RAM ist derjenige Teil des Speichers, über den der Benutzer frei verfügen kann, um Informationen (dazu gehören auch Programme) abzulegen. Nicht alle MSX-Computer haben einen gleich großen RAM-Speicher. SONY hat Computer mit 16 K Bytes (erweiterbar auf 32 oder 64 K Bytes durch einen Speicherblock) oder 64 K Bytes freien Speicherplatz.

Ein K Byte ist die Einheit, in der Speicherplatz angegeben wird. Ein K Byte ist ein Kilo-Byte und das bedeutet 1024 Speicherplätze von der Größe eines Bytes. Ein Byte besteht aus 8 Bits (BInary digiTs). Jedes Bit ist eine Stelle im Computer, die an- oder abgeschaltet ist, ein Plus oder ein Minus, eine Eins oder eine Null angibt, kurz, eine Art ganz kleiner Schalter. Mit diesen riesigen Mengen von Schaltern kann der Computer allerlei Befehle für Sie ausführen.

ROM ist derjenige Teil (der Name sagt es bereits), der dem Benutzers dazu dienen kann, Informationen auszulesen. Neue Informationen können hier nicht untergebracht werden. ROM beinhaltet in jedem MSX-Computer den BASIC-Interpreter.

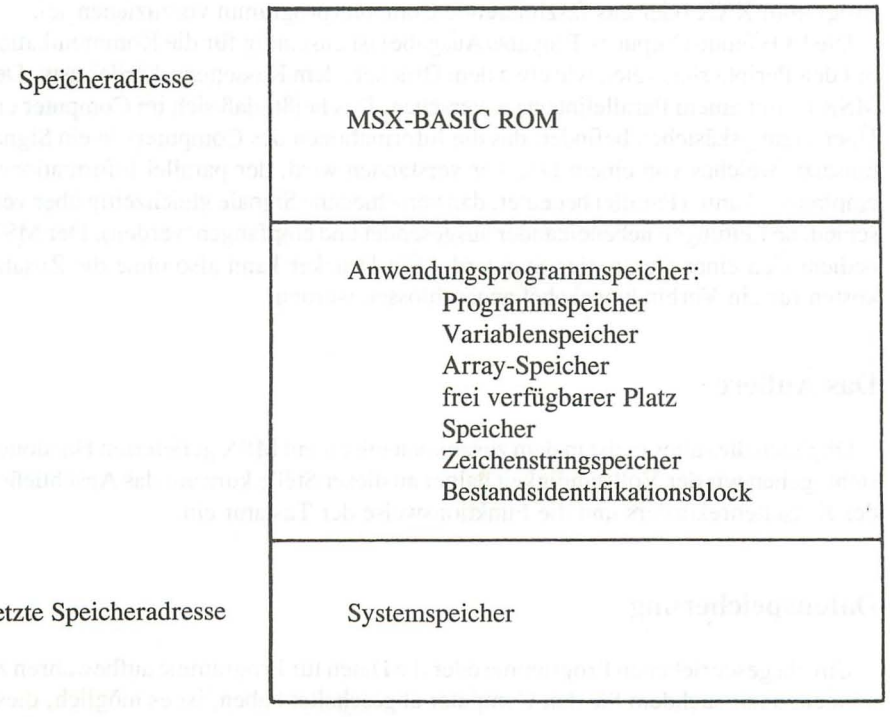
Darüber hinaus haben verschiedene MSX-Computer ein eingebautes Programm. So ein Programm (etwa die Datenbank des SONY) kann vom Benutzer nicht verändert werden (schließlich befindet es sich ja im ROM). Die neuen Informationen, die durch das Programm entstehen, werden vorübergehend im RAM abgelegt und später eventuell auf eine Kassette oder Diskette weggeschrieben.

Der BASIC-Interpreter ist der Dolmetscher des Computers. Der Interpreter ist die Grammatik und das Wörterbuch, mit denen der Computer die vom Benutzer in BASIC gegebenen Befehle verstehen kann. Ein Computer selber arbeitet binär. Erinnern Sie sich an die kleinen Schalter, die lediglich ein- oder ausgeschaltet sein können und nur 1 oder 0 kennen. Die Sprache BASIC wird nur deshalb verstanden, weil der Interpreter übersetzt. Es ist zwar möglich, den Computer direkt mit Einsen und Nullen anzusprechen, doch das ist für den Programmierer ziemlich lästig. Er beschäftigt sich ausschließlich mit dem Programmieren in BASIC.

Der BASIC-Interpreter hat einen Bedarf von 32 K Bytes Speicherplatz. Dies scheint sehr viel zu sein, aber dadurch, daß der Interpreter so breit angelegt ist, kann der Benutzer zahlreiche unterschiedliche Befehle eingeben, die der Computer alle versteht.

Der Systemplatz ist für den BASIC-Programmierer weniger von Bedeutung. Dieser Teil des Speichers versorgt die CPU mit den Informationen, die sie benötigt, um den internen Haushalt des Computers zu regeln.

Schematisch sieht der Speicheraufbau eines MSX-Computers folgendermaßen aus:



Der Umfang des RAM-Speichers ist abhängig vom Typ des MSX-Computers. Hinzu kommt, daß es durch die Möglichkeiten des Z-80A-Prozessors im MSX hinsichtlich des Speicherplatzes, der dem Benutzer den Herstellerangaben zufolge zur Verfügung steht, einige Haken und Ösen gibt. So kann es passieren, daß der Computer nach dem Einschalten über die Anzahl „freier" Bytes Angaben macht, die mit der Spezifikation des Computers nicht übereinstimmen. Es würde hier zu weit führen, darauf näher einzugehen. Es kommt hinzu, daß der MSX ziemlich sparsam mit seinem Speicherplatz umgeht, so daß Sie also recht lange Programme schreiben können, bevor der Speicher voll ist.

I/O und Video

Über die beiden anderen Teile des Schemas gibt es nicht so viel zu sagen.

Das VIDEO ist zuständig für die Wiedergabe auf dem Bildschirm. Die Wiedergabe beim MSX kann über einen Monitor (farbig oder schwarz/weiß) oder über ein einfaches Fernsehgerät (farbig oder schwarz/weiß) erfolgen. Die Töne, die der MSX erzeugen kann, werden über das Fernsehgerät hörbar gemacht. Wenn Sie einen Monitor benutzen, müssen Sie einen Typ nehmen, der gleichzeitig auch Töne wiedergeben kann. Ein Monitor liefert ein schärferes Bild, aber er ist auch kostspieliger. Ein Fernsehgerät gibt es meist schon im Haus, aber die umfassende Nutzung dieses Gerätes sorgt (zuweilen) für Turbulenzen bei der Entscheidung, ob nun das tolle TV-Programm XXX oder das faszinierende Computerprogramm vorzuziehen sei.

Die I/O (Input/Output = Eingabe/Ausgabe) ist zuständig für die Kommunikation mit den Peripheriegeräten wie etwa dem Drucker, dem Kassettenrekorder usw. Der MSX ist mit einem Parallelinterface versehen. Das heißt, daß sich im Computer ein Übersetzungskästchen befindet, das die Informationen des Computers in ein Signal umsetzt, welches von einem Drucker verstanden wird, der parallel Informationen empfangen kann. (Parallel bedeutet, daß verschiedene Signale gleichzeitig über verschiedene Leitungen nebeneinander ausgesendet und empfangen werden). Der MSX bedient sich eines Centronics-Standards. Ein Drucker kann also ohne die Zusatzkosten für ein Verbindungskabel angeschlossen werden.

Das Äußere

Obgleich dies alles meist in dem zusammen mit einem MSX gelieferten Handbuch steht, gehen wir der Vollständigkeit halber an dieser Stelle kurz auf das Anschließen des Kassettenrekorders und die Funktionsweise der Tastatur ein.

Datenspeicherung

Um die geschriebenen Programme oder die Daten für Programme aufbewahren zu können, auch nachdem Sie den Computer abgeschaltet haben, ist es möglich, diese

auf einem Medium zu speichern, das nicht von einer kontinuierlichen elektrischen Spannung abhängig ist. Derzeit werden Daten fast immer auf magnetischen Medien gespeichert.

Magnetische Medien gibt es in vielerlei Form. Großcomputer (Main-frames) benutzen breite Magnetbänder. Billiger sind die sogenannten „Hard-Disks". Dies sind große Scheiben aus magnetischem Material, meist zusammengepackt zu jeweils fünf Stück in einer großen Wechselkassette, die von einer speziellen Apparatur gelesen und beschrieben werden kann. Noch ein bißchen preiswerter ist ein System, bei dem eine einzige Hard-Disk in einen Lese- und Schreibapparat fest eingebaut ist und infolgedessen nicht gewechselt werden kann. Diese letztere Form setzt sich immer mehr in Verbindung mit den Mikrocomputern durch. Am günstigsten sind Scheiben aus weichem Material (floppy disks = Disketten = weiche Scheiben), die auch kleiner sind. Für diese Scheiben ist ebenfalls ein besonderes Lese- und Schreibgerät notwendig. Diese Geräte kosten sie 900 DM. Es sieht nicht danach aus, als seien hier so bald Preissenkungen zu erwarten. Für professionellen Einsatz der Computer sind sie jedoch unerläßlich, da ihre Schreib-/Lesegeschwindigkeit um viele Male höher ist als die der preisgünstigeren Arten der magnetischen Speicherung: Bänder (Bandgeräte) und Kassetten (Kassettenrekorder). Die neueste Entwicklung auf diesem Sektor ist die speziell für die MSX-Computer auf den Markt gekommene neue Diskettenart. Diese Disketten sind wesentlich kleiner als eine normale Diskette, und sie stecken in einem Überzug aus Hartplastik. Man erwartet, daß diese Scheiben billiger werden als die herkömmlichen Disketten, wenn sie sich am Markt erst einmal durchgesetzt haben. Daneben versuchen verschiedene Hersteller, eine Zwischenform von Kassette und Diskette zu entwickeln und besteht aus einer Kassette mit sehr rasch laufendem Band. Diese Form ist sowohl hinsichtlich der Kosten als auch der Geschwindigkeit in der Mitte zwischen normalen Kassettensystemen und Diskettensystemen anzusiedeln.

Für den hobbymäßigen Computerbenutzer, dem es nicht in erster Linie auf die Schnelligkeit ankommt, sind Kassettenrekorder empfehlenswert. Die Diskettensysteme sind kostspielig, und bei den anderen Systemen ist es noch die Frage, ob sie sich allgemein durchsetzen: erst daraus ergibt sich ja eine gewisse Austauschbarkeit. Bei den Kassettenrekordern ist jede beliebige Marke geeignet. Im allgemeinen sind die billigeren Rekorder sogar besser als die teuren Modelle.

Das Anschließen des Kassettenrekorders, so daß dieser in der richtigen Weise Daten vom Computer übernimmt oder an ihn abgibt, ist beim MSX ganz einfach. Schließlich hat man ja für Standardisierung gesorgt! An der Rückseite des Computers finden Sie einen Anschluß mit der Aufschrift TAPE (Band). Hier hinein paßt der Stecker des mitgelieferten Kassettenkabels. Sie erkennen es an dem runden, achtpoligen DIN-Stecker an der einen Seite und den drei kleinen Steckern (einem weißen, einem roten und einem noch kleineren schwarzen) am anderen Ende. Den weißen Stecker schieben Sie in die Buchse für den Kopf- oder Ohrhörer (meist mit EAR bezeichnet) an der Seite des Kassettenrekorders. Der rote Stecker gehört in den Mikrofoneingang (MIC), und der schwarze kommt in die Buchse für die Fernbedienung (REMOTE) oder den zweiten (kleineren) Mikrofoneingang (MIC). Wenn Sie einen Kassettenrekorder mit nur einem runden DIN-Anschluß besitzen, müssen Sie sich bei

Ihrem Radio- und Fernsehfachhändler eine Verlängerungsschnur besorgen. Nehmen Sie das Kabel vom MSX zum Kassettenrekorder mit und erzählen Sie Ihrem Händler, wozu es benutzt werden kann. Nehmen Sie eventuell auch noch dieses Buch mit.

Hat Ihr Kassettenrekorder keine Fernbedienung, ist der schwarze Stecker sinnlos (Achtung: Bei einigen Kassettenrekordern ist die Fernbedienung in den kleinen Mikrofoneingang eingebaut!). Bei einem eventuellen Übergangsstück zu einem DIN-Stecker muß das berücksichtigt werden.

So, Ihr Computer ist nun komplett. Wir gehen davon aus, daß Sie einen Kassettenrekorder besitzen. Jedoch auch wenn Sie ein Diskettensystem haben, können Sie dieses Buch gebrauchen. Allerdings sind zum Lesen und Schreiben abweichende Befehle zu erteilen. Doch darauf wird gegebenenfalls noch gesondert eingegangen werden.

Wie Sie Ihre Kassetten einzusetzen haben, ist im Augenblick nicht weiter von Bedeutung. Sobald Sie Ihr erstes Programm geschrieben haben, kommen wir darauf noch einmal zurück.

Der Bildschirm

Im Normalfall wird ein Fernsehgerät als Bildschirm genommen. Hierbei ist der Anschluß des Bildschirmes äußerst einfach. Ein kleines Kabel wird zusammen mit dem Computer geliefert. Sie müssen nur noch den Stecker an der Rückseite des MSX einstecken und das andere Ende in die Antennenbuchse ihres Fernsehgerätes. Zusammen mit dem SONY HIT-BIT wird ein Antennenschalter geliefert. Der Stecker des Computerkabels gehört in diesen Schalter und daneben kommt der normale Antennenstecker hinein. Den Stecker des Schalters selbst schieben Sie in den Antenneneingang des Fernsehgerätes. Umschalten können Sie nun durch Betätigung des Schalters.

Mit den Knöpfen für die Senderwahl und die Feinabstimmung suchen Sie die bei Ihrem Fernsehgerät richtige Einstellung für den Computer. Das einfachste ist es, für diese Einstellung einen eigenen Sendervorwahlknopf zu nehmen. Den Ton des Fernsehgerätes sollten Sie nicht abschalten. Der MSX hat keinen eigenen Lautsprecher, sondern benutzt den des Fernsehgerätes. Soll das Bild schärfer sein, als es bei einem Fernsehgerät möglich ist (beispielsweise für Textverarbeitung), brauchen Sie einen Monitor. Achten Sie darauf, daß Sie einen Monitor mit Lautsprecher nehmen, sonst sind die Möglichkeiten des MSX zur Erzeugung von Geräuschen für Sie nutzlos. Das Verbindungskabel vom Monitor zum Computer erhalten Sie entweder zusammen mit dem Monitor oder aber Sie müssen es zusätzlich erwerben. Für einen Farbmonitor brauchen Sie ein 21-poliges PERI-Kabel. Bei einem Schwarzweißmonitor reicht ein 5-poliges DIN-Kabel (achten Sie auf die Anordnung der Pole!) oder aber einen Jack-Plug, wie Sie ihn für das Fernsehkabel brauchen. Weitere Details entnehmen Sie bitte der Bedienungsanleitung.

Wenn Sie den MSX einschalten, erscheint zunächst ein Hinweis auf das Copyright. Dieser Text teilt Ihnen mit, daß dieses BASIC im Jahre 1983 von der Firma Microsoft für diesen Computer entwickelt worden ist. Zugleich wird eine Versionsnummer genannt. Da jedes Computermodell fortlaufend verbessert wird, ist es möglich, daß es

irgendwann einmal verschiedene Versionen von MSX-BASIC gibt. Fast immer sorgen die Hersteller dafür, daß die jüngsten Versionen alles das können, was auch die älteren konnten und darüber hinaus noch einiges mehr. Die Nummer der Version kann also von Bedeutung sein, wenn Sie einmal wissen müssen, ob ein bestimmtes Programm auch auf Ihrem Computer ohne Probleme läuft.

Bei MSX-Computern, die ein fest installiertes Programm haben, erscheint nach kurzer Zeit die Ankündigung des Programmes. Beim SONY HIT-BIT etwa ist das die persönliche Datenbank. Da wir Ihnen in diesem Buch nicht zeigen wollen, wie Sie ein Programm benutzen, sondern wie Sie sich selber eines schreiben können, verlassen wir dieses Programm rasch wieder, indem wir das dunkle Pünktchen für die verschiedenen Möglichkeiten mit den Pfeiltasten ganz rechts auf der Tastatur viermal nach unten bewegen. Der Punkt befindet sich nun vor BASIC. Drücken Sie jetzt auf die RETURN-Taste.

Der MSX meldet sich mit:

```
MSX BASIC-Version 1.0
Copyright 1983 by Microsoft
28815 Bytes free
Ok
```

Die Zahlen dieser Meldung können sich voneinander unterscheiden: Vielleicht besitzen Sie eine andere Version, und der zur Verfügung stehende Speicherplatz ist nicht bei jedem Computer gleich groß.

Mit Ok zeigt der Computer, daß er bereit ist, Ihre Befehle entgegenzunehmen. Wir können also anfangen.

Die Tastatur

Die Kommunikation mit dem Computer erfolgt zumeist über die Tastatur. Obgleich sich die Tastaturen der MSX-Computer voneinander unterscheiden, gibt es eine Reihe von Übereinstimmungen.

Tastatur des SONY HIT BIT

Die Tasten sind nach dem QWERTY-Muster angeordnet. Das ist ein Muster, von dem man in der Anfangszeit der Schreibmaschine glaubte, daß es das am schnellsten zu bedienende sei. Auch heute noch sind die Tastaturen fast aller Schreibmaschinen nach diesem Muster angeordnet.

Obgleich die Tastaturen unterschiedlich aussehen können, finden Sie auf jeder Tastatur folgende Teile:

Buchstabentasten

Diese Tasten zeigen alle gebräuchlichen Buchstaben, Ziffern und Satzzeichen. Daneben können mit diesen Tasten verschiedene grafische Zeichen und Symbole erzeugt werden (Siehe unten).

Kontrolltasten

Auf der linken und der rechten Seite der Tastatur finden Sie eine Gruppe spezieller Tasten, die unter anderem festlegen, wie eingegebener Text auf dem Bildschirm aussieht und was die Benutzung der normalen Buchstabentasten ergibt.

Die ESC-Taste (ESCape = entkommen) stammt eigentlich von anderen Computern, und sie dient dort zum Anhalten des laufenden Programmes. In MSX-BASIC wird diese Taste nicht dazu benutzt. Sie können Ihr Programm jedoch so aufbauen, daß diese Taste funktioniert.

Die TAB-Taste (TABulator) sorgt dafür, daß die Wiedergabe der Zeichen vorrückt zu einem Punkt, den ein TAB-Stop markiert. Sie kennen das vielleicht von normalen Schreibmaschinen her. Die Tabulatorstops sind auf je acht Zeichen Zwischenraum voreingestellt. Die auf diese Weise „übersprungenen" Zeichen werden gelöscht.

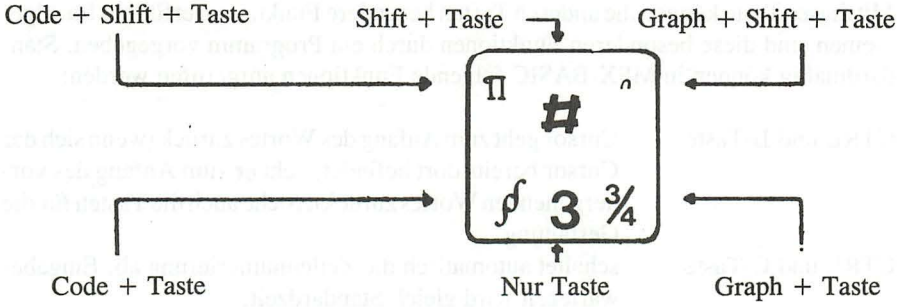
Die SHIFT-Taste (= verschiebe) dient zur Wiedergabe der jeweiligen Großbuchstaben oder des zuoberst dargestellten Zeichens. Hierzu müssen Sie SHIFT und die gewünschte Taste gleichzeitig drücken.

Mit CAP (CAPitals = Großbuchstaben) erreichen Sie das gleiche. Wenn Sie diese Taste drücken, leuchtet der CAP-Indikator (ein kleines Lämpchen) auf zum Zeichen dafür, daß von jetzt an alles, was Sie eingeben, in Großbuchstaben dargestellt wird. Bei den Tasten mit zwei Symbolen wird jedoch das zuunterst dargestellte Zeichen erzeugt! Um das obere Symbol zu erhalten, müssen Sie immer SHIFT drücken. Wollen Sie die Buchstaben wieder als Kleinbuchstaben darstellen, muß CAP abgeschaltet werden; dazu drücken Sie CAP ein weiteres Mal.

Die CODE-Taste dient zur Wiedergabe eines besonderen grafischen Symboles auf jeder Taste. Welche Taste welches Symbol erzeugt, kann man auf den Tasten selber (die linken Symbole) erkennen oder aber einer speziell dazu mitgelieferten Karte entnehmen. CODE zusammen mit der Taste gedrückt, ergibt das Symbol links unten; CODE mit SHIFT und der gewünschten Taste gedrückt, erzeugt das Symbol links oben.

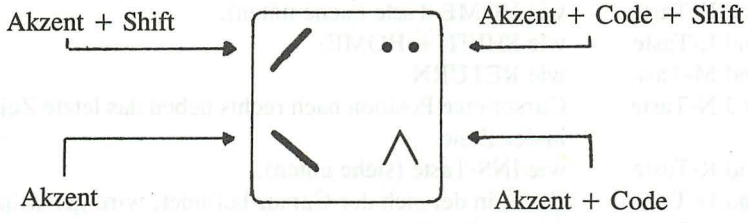
Die GRAPH-Taste (GRAPHics = Grafik) ermöglicht die Wiedergabe der grafischen Symbole auf der rechten Seite der Tasten, die auf der Karte abgebildet sind. Auch hierbei gilt, daß GRAPH zusammen mit der gewünschten Taste das grafische Symbol rechts unten ergibt und GRAPH plus SHIFT zusammen mit der gewünschten Taste das grafische Symbol rechts oben.

Eine einzige Taste vermag also die folgenden Zeichen zu erzeugen:



Alle in MSX-BASIC möglichen Zeichen sowie die Art und Weise, in der sie über die Tastatur eingegeben werden müssen, entnehmen Sie bitte Anlage 1.

Die AKZENT-Tasten ermöglichen es Ihnen, Buchstaben mit Akzenten zu versehen. Hierzu betätigen Sie zunächst die Akzenttaste auf folgende Weise:



Auf dem Bildschirm geschieht vorerst nichts. Wenn Sie anschließend die Taste mit dem Buchstaben drücken, über den der Akzent gehört, erscheint der Buchstabe zusammen mit dem Akzent auf dem Schirm.

Die SELECT-Taste (= aussuchen) ist allein zur Benutzung innerhalb von Programmen gedacht. Bei MSX-BASIC brauchen Sie diese Taste nicht.

Die RETURN-Taste (= zurück) ist die wichtigste Taste auf der gesamten Tastatur. Damit nicht gleich alles gemacht wird, was Sie in den Computer eingeben, wartet dieser mit der Übernahme des Befehles, bis Sie die RETURN-Taste betätigen (mit den wenigen Ausnahmen beschäftigen wir uns später). Die Betätigung der RETURN-Taste ist also für den Computer das Signal, daß Sie den Befehl abgeschlossen haben und er mit der Ausführung beginnen kann.

Die STOP-Taste spricht beinahe für sich. Indem Sie diese Taste drücken, stoppen Sie das Programm. Diese Taste kann auch dazu dienen, das „listen" (Auflisten aller Programmzeilen in der richtigen Reihenfolge) anzuhalten. Nach abermaligem Drücken wird das Listing des Programms fortgesetzt. Die STOP-Taste kann auch zugleich mit der CTRL-Taste (siehe unten) betätigt werden. In diesem Falle wird das Programm nur nach dem CONT-Befehl fortgesetzt. Das Listing ist endgültig unterbrochen und muß erneut mit LIST gestartet werden.

Die CTRL-Taste (ConTRoL = Steuerung) haben wir bis zum Schluß aufbewahrt. Mit dieser Taste können die anderen Tasten besondere Funktionen erfüllen. Im allgemeinen sind diese besonderen Funktionen durch ein Programm vorgegeben. Standardmäßig können in MSX-BASIC folgende Funktionen aufgerufen werden:

CTRL und B-Taste	Cursor geht zum Anfang des Wortes zurück (wenn sich der Cursor bereits dort befindet, geht er zum Anfang des vorhergehenden Wortes zurück). Siehe auch die Tasten für die Gestaltung.
CTRL und C-Taste	schaltet automatisch die Zeilennumerierung ab. Eingabewartezeit wird gleich Standardzeit.
CTRL und E-Taste	alle Buchstaben zwischen Cursor und Zeilenende werden gelöscht.
CTRL und F-Taste	Cursor an den Anfang des folgenden Wortes.
CTRL und G-Taste	Bliep ertönt.
CTRL und H-Taste	wie BS-Taste.
CTRL und I-Taste	wie TAB-Taste.
CTRL und J-Taste	Cursor eine Zeile nach unten.
CTRL und K-Taste	wie HOME-Taste (siehe unten).
CTRL und L-Taste	wie SHIFT + HOME.
CTRL und M-Taste	wie RETURN.
CTRL und N-Taste	Cursor eine Position nach rechts neben das letzte Zeichen in der Zeile.
CTRL und R-Taste	wie INS-Taste (siehe unten).
CTRL und U-Taste	Zeile, in der sich der Cursor befindet, wird gelöscht.
CTRL und X-Taste	wie SELECTL.
CTRL und /-Taste	Cursor eine Position nach rechts.
CTRL und]-Taste	Cursor eine Position nach links.
CTRL und SHIFT und 6	Cursor eine Position nach oben.
CTRL und SHIFT und -	Cursor eine Position nach unten.
CTRL und SHIFT und Pfund	Cursor eine Position nach oben.

Gestaltungstasten

Um dafür zu sorgen, daß der Text auch ansprechend aussieht, ist es zuweilen notwendig, ihn (im Nachhinein) zu gestalten. Hierzu dient eine Gruppe von Tasten auf der linken Seite der Tastatur. Durch ein Programm können diese Tasten eine bestimmte Bedeutung erhalten. Standardmäßig funktionieren sie in MSX-BASIC (also nach dem Einschalten des Computers) folgendermaßen:

Mit den vier PFEIL-Tasten ganz links außerhalb der Tastatur läßt sich der Cursor bewegen. Der Cursor ist ein kleiner weißer Block. Dieser kleine Block markiert die Stelle, an der das erste einzugebende Zeichen erscheinen wird. Auf diese Weise können Sie immer erkennen, wo Sie sich bei der Eingabe gerade befinden. Mittels der vier Pfeil-Tasten kann der Cursor an eine beliebige Stelle auf dem Bildschirm bewegt werden. Der Text auf dem Schirm wird dadurch nicht beeinflußt.

Die BS-Taste (Back-Space = Position zurück) bewegt den Cursor um eine Position nach links und löscht den Buchstaben, der dem Cursor im Weg steht. Befindet sich der Cursor ganz links auf dem Schirm, wird der Buchstabe an dieser Stelle gelöscht und die gesamte Zeile verschiebt sich um eine Position nach links.

Die DEL-Taste (DELete = löschen) hat eine vergleichbare Aufgabe. Der Cursor bleibt an der Stelle, an der er sich gerade befindet, aber das Zeichen an dieser Stelle wird gelöscht. Der gesamte Text rechts des Cursors wird um eine Position nach links verschoben.

Die INS-Taste (INSert = einfügen) verändert den Modus (Arbeitsniveau) des Computers. Normalerweise überschreibt der Computer beim Eintippen den alten Text. Durch Drücken der INS-Taste wird der Cursor kleiner, und der Computer schaltet auf Einfügemodus um. Das heißt, daß der Text, der jetzt eingegeben wird, den bereits vorhandenen Text vor sich herschiebt, ohne ihn zu löschen. Man kann den Einfügemodus wieder verlassen, indem man erneut INS drückt oder den Cursor mit einer Pfeiltaste bewegt. Der Cursor nimmt dann wieder seine normale Größe an.

Die HOME-Taste (= zu Hause) bringt den Cursor in seine Ausgangsposition zurück; das ist die linke obere Ecke. Aller Text auf dem Bildschirm bleibt unverändert. Wird die HOME-Taste zugleich mit SHIFT gedrückt, löscht man den gesamten Schirm.

Jetzt bleiben noch 6 Tasten übrig. RESET (= neu einstellen) ist eine Taste, die Sie eigentlich nie gebrauchen sollten. Mit dieser Taste wird der Computer veranlaßt, alles zu vergessen, was Sie ihm bislang eingegeben haben, und er versetzt sich in den Zustand zurück, in dem er sich direkt nach dem Einschalten befand. Sie sollten immer die STOP-Taste drücken, um ein Programm anzuhalten. Nur im äußersten Notfall kann es angebracht sein, sich einmal der RESET-Taste zu bedienen.

Die Funktionstasten

Die Tasten links oben sind Funktionstasten; sie haben alle eine eigene Aufgabe:

Funktionstaste		SHIFT + Funktionstasten	
F1	color	F6	color 15,4,4 RETURN
F2	auto	F7	cload"
F3	goto	F8	cont RETURN
F4	list	F9	list. RETURN
F5	run	F10	cls : run RETURN

Was Sie damit alles machen können, werden Sie im weiteren Verlauf der Lektüre von alleine herausfinden. Dadurch, daß ganze Funktionen durch Drücken einer einzigen Taste ausgelöst werden können, kann viel rascher programmiert werden.

Der Benutzer kann Befehle, die er häufig braucht, in einer Funktionstaste mittels des Befehls KEY ablegen.

Zu diesem Zwecke geben Sie ein:

KEY nr, ''xxxxxxxxxxxxxxx''

nr steht hier für die Nummer der zu definierenden Taste (von 1 bis 10 einschließlich) und ''xxxxxxxxxxxxxxx'' ist eine Reihe von maximal 15 beliebigen Zeichen. Wenn Sie also ganz schnell beim Programmieren den Befehl GOSUB erteilen wollen, können Sie ihn in Funktionstaste 3 unterbringen:

KEY 3, ''GOSUB'' (RETURN)

Nach Eingabe des Textes dürfen Sie nicht vergessen, auf RETURN zu drücken. Erst dann weiß der Computer, daß Sie mit der Eingabe des für ihn bestimmten Textes fertig sind.

Der Inhalt von Funktionstaste 3 steht nun zwischen all den anderen Umschreibungen unten am Bildrand. Um die Umschreibungen zu entfernen, um die unterste Zeile für andere Zwecke benutzen zu können, geben Sie ein:

KEY OFF (RETURN) (= Taste aus)

Wollen Sie die Definition der Tasten noch einmal sehen, reicht

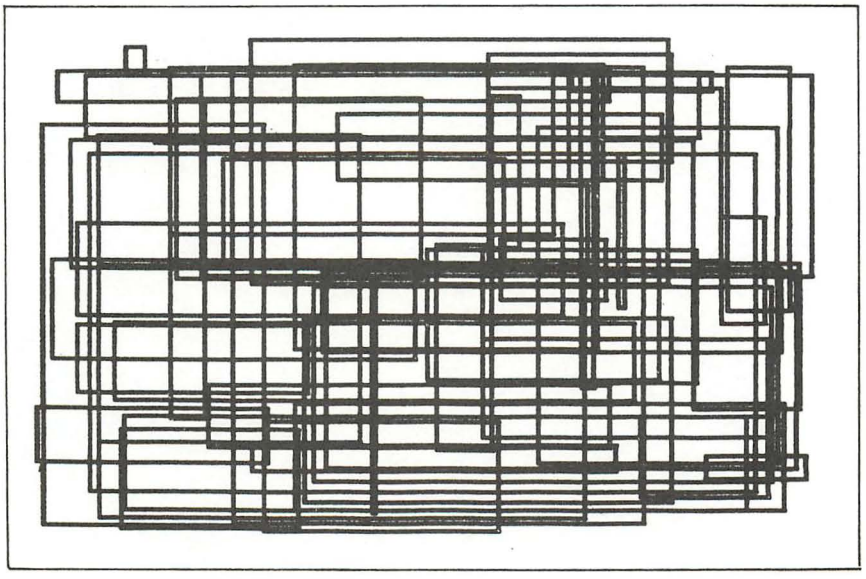
KEY ON (= Taste ein)

Um den Inhalt aller Tasten auf einen Blick erkennen zu können, geben Sie ein:

KEY LIST

Sie erhalten dann eine Auflistung aller Funktionstastendefinitionen. Die Nummern der Tasten müssen Sie sich selber dazudenken. Wenn Sie den obenstehenden Text eingegeben haben, sehen Sie in der dritten Zeile das getippte GOSUB.

Die Definitionen der Tasten gehen beim Abschalten des Computers verloren. Wenn Sie den Computer später wieder einschalten, erhalten Sie automatisch die standardmäßig programmierten Definitionen. Wenn Sie häufig eigene Definitionen gebrauchen, ist es ratsam, diese in einem eigenen Programm unterzubringen.



3. DIE BILDSCHIRMARTEN

Der Bildschirm, der an den MSX-Computer angeschlossen ist, kann auf vier verschiedene Arten aufgebaut sein; zwei Arten für Wiedergabe von Grafik und zwei Arten für Wiedergabe von Text.

Für eine bestimmte Bildschirmwiedergabe entscheiden Sie sich mit dem Befehl:

SCREEN nr

nr steht hier für eine Zahl von 0 bis 3 einschließlich.

Bildschirm 0

Das ist der Bildschirm, der standardmäßig erscheint, wenn Sie den MSX einschalten (und sich für BASIC entscheiden). Auf diesem Schirm können 40 Zeichen nebeneinandergesetzt werden. Für jeden Buchstaben oder jedes Symbol stehen 6 Punkte von links nach rechts und 8 Punkte von oben nach unten zur Verfügung. Alle normalen Buchstaben und Ziffern können hiermit mühelos wiedergegeben werden. Für einen Teil der grafischen Zeichen benötigt man jedoch ein Raster von 8 x 8 Punkten. Diese Zeichen werden auf diesem Schirm nicht vollständig wiedergegeben; an der rechten Seite fehlt ein Stückchen.

Es passen 24 Zeilen auf einen Schirm.

Schirm 1

Dieser Schirm dient zur besseren Wiedergabe von Buchstaben. Für jeden Buchstaben ist nun ein Raster von 8 x 8 Punkten reserviert. Da die Buchstaben nur 6 Punkte in der Breite benötigen, stehen sie etwas weiter auseinander und sind so besser voneinander zu unterscheiden.

Die verschiedenen Symbole und Zeichen können nun allesamt vollständig dargestellt werden.

Wegen des zusätzlichen Platzes, der für jede Zeichenposition jetzt zur Verfügung steht, kann eine Zeile nur noch 32 Zeichen lang sein.

Es passen 24 Zeilen auf einen Schirm.

Schirm 2

Dieser Schirm ist für die Wiedergabe von besonders fein gezeichneter Grafik gedacht. Detaillierte Erläuterungen dieser Art zu zeichnen entnehmen Sie bitte den Kapiteln über Grafik und Farbe.

Das Bild ist als ein Raster aus 256 (links-rechts) mal 192 (oben-unten) Punkten aufgebaut. Jeder Punkt kann einzeln eingefärbt werden. Die Färbung der Punkte erfolgt jedoch per Block. Bei Benutzung dieses Schirmes kann man auch nicht sicher sein, daß die Farben richtig wiedergegeben werden. Dies macht das folgende Programm besonders anschaulich, das Ihren MSX Purzelbäume schlagen läßt. Geben Sie das Programm genauso ein, wie es hier steht, und drücken Sie nach jeder Zeile (und keinesfalls vorher) auf RETURN. Wenn Sie alle Zeilen abgetippt haben, drücken Sie auf F5 (für RUN).

```
10 COLOR 15,1,1
20 SCREEN 2
30 X=INT(RND(1)*256)
40 Y=INT(RND(1)*192)
50 Q=INT(RND(1)*256)
60 R=INT(RND(1)*192)
70 C=INT(RND(1)*15)
80 LINE(X,Y)-(Q,R),C
90 IF INT(C/2)=C/2 THEN BEEP
100 GOTO 30
```

Dieses Programm zeichnet an willkürlichen Stellen eine Linie in einer willkürlichen Farbe. Zunächst geht noch alles gut, aber bald schon zeigt sich, daß der Computer zwar die Linien korrekt ausführt, die Farben aber durcheinanderzubringen beginnt. Wie das Programm arbeitet, werden wir später sehen. Sie können das Programm anhalten, indem Sie auf STOP drücken. Wollen Sie das Programm beenden, erreichen Sie das durch Drücken von CTRL und STOP.

Schirm 3

Auch dieser Schirm setzt sich aus 256 x 192 Punkten zusammen. Diesmal können jedoch nur Kästchen von 4 x 4 Punkten eingefärbt werden. Die graphische Wiedergabe ist also weniger fein. Demgegenüber kann aber jedes Kästchen seine eigene Farbe erhalten.

Sie können sich dieses Programm vorführen lassen, indem Sie im vorhergehenden Programm in Zeile 20 die 2 (von SCREEN) durch eine 3 ersetzen. Wenn das letzte Programm noch läuft, tippen Sie gleichzeitig auf CTRL und STOP. Der Computer antwortet mit:

```
Break in...

Ok
```

Auf diese Weise sagt Ihnen der Computer, daß er in der Zeile gestoppt hat, die durch die drei Punkte markiert wird. Mit Ok erklärt er, daß Sie an der Reihe sind, er erwartet Ihren Befehl. Drücken Sie auf F4 und RETURN oder auf LIST und RETURN. Der Computer zeigt Ihnen nun das komplette Programm. Bewegen Sie den

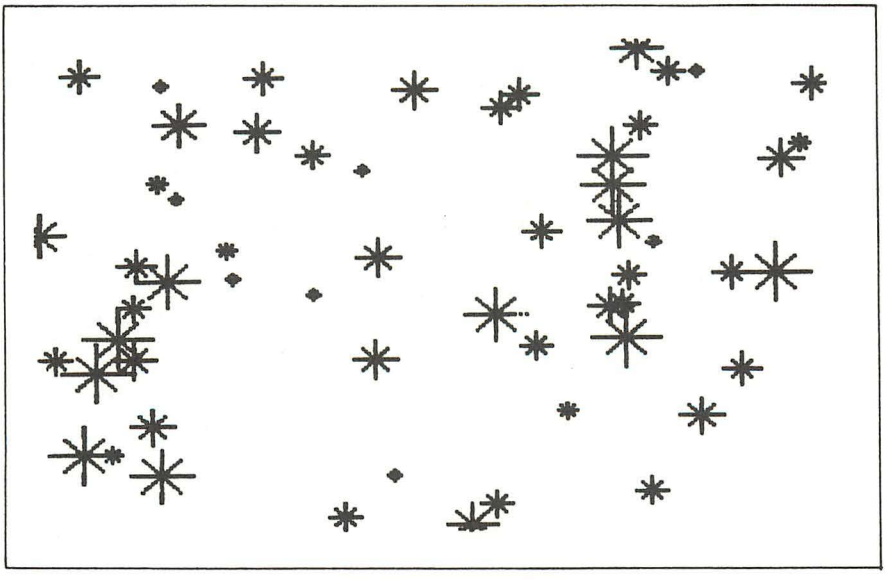
Cursor (den weißen Block, der andeutet, wo das nächste Zeichen erscheinen wird) mit den Pfeiltasten auf die 2 hinter SCREEN in Zeile 20 (bringen Sie den Cursor direkt über die 2). Geben Sie nun eine 3 ein und drücken RETURN, dann bewegen Sie den Cursor unter das Listing (das ist die Wiedergabe aller Programmzeilen). Die genaue Stelle ist völlig gleichgültig, wenn es sich nur um eine leere Zeile handelt. Sonst würde der Computer Sie nicht verstehen. Geben Sie jetzt wieder F5 ein oder tippen Sie RUN und drücken auf RETURN.

Die Striche, die nun gezogen werden, sind zwar viel kräftiger, doch die Farben sind nicht mehr vermischt.

Für alle Schirme gilt, daß oberhalb und unterhalb des Schirmes ein besonderer Rand erscheint. Dieser ist bei Schirm 1, 2 und 3 unabhängig von der jeweiligen Farbe des Schirmes.

Die grafischen Schirme (2 und 3) verschwinden nach Gebrauch von alleine wieder, und es erscheint automatisch Schirm 0. Um am Ende eines Programms nicht plötzlich den Bildschirm zu verlieren, muß der Programmierer dafür sorgen, daß der Schirm von Dauer wird.

Bei den Schirmen 1, 2 und 3 kann man mit SPRITES arbeiten. Dabei handelt es sich um farbige grafische Flächen, die „vor" dem Bild hin und her bewegt werden können. Mehr hierüber und über die Herstellung von Grafiken auf einem Textschirm oder von Text auf einem grafischen Schirm erfahren Sie in dem Kapitel über Grafik und Farbe.



4. DIREKTE BEFEHLE

Die einfachste Art, Ihren MSX zu benutzen ist, ihm Befehle zu geben (natürlich in BASIC). Einige direkte Befehle, die zu BASIC gehören, haben Sie bereits kennengelernt. So können Sie durch gleichzeitiges Drücken von CTRL und G-Taste dem Computer den Befehl geben, ein Bliep ertönen zu lassen. Dieser Befehl wird direkt ausgeführt. Die KEY-Befehle waren ebenfalls direkte Befehle. Sofort nach Betätigung von RETURN, wodurch Sie dem Computer mitteilen, daß Sie den Befehl erteilt haben, wird er ausgeführt, und die Funktionstaste erhält wieder ihre ursprüngliche Definition.

Um nun einmal einen BASIC-Befehl auszuprobieren, müssen Sie den Computer zunächst wieder in den Ausgangszustand versetzen. Drücken Sie auf RESET oder unterbrechen Sie die Stromzufuhr (Ein- /Aus-Schalter, Stecker) für 10 Sekunden. Gehen Sie anschließend nach MSX-BASIC (siehe Kapitel 2).

Tippen Sie jetzt:

PRINT „Ich gebe einen Satz wieder."

und drücken Sie auf RETURN.

Sie sehen, der Computer führt sofort aus, was Sie von ihm verlangen; er druckt (PRINT) einen kurzen Text.

Haben Sie den PRINT-Befehl mit Hilfe der SHIFT- und der CAPS-Tasten in Großbuchstaben geschrieben? Das ist gut für die Fingerfertigkeit! Doch der MSX stellt in Rechnung, daß es mit Ihrer Tippkunst noch nicht besonders weit her ist. BASIC-Befehle können sowohl in Groß- als auch in Kleinbuchstaben eingegeben werden.

Versuchen Sie es ruhig mal:

print „Ich gebe einen Satz wieder."

Mittels der RETURN-Taste zeigen Sie der Maschine, daß Sie mit der Eingabe einer Befehlszeile fertig sind und daß der Computer sie ausführen (bei direkten Befehlen) oder in seinen Speicher übernehmen soll (bei indirekten Befehlen).

Die Anführungszeichen zu beiden Seiten des Textes sagen der Maschine, daß die Angaben dazwischen nicht als BASIC-Befehl gemeint sind, sondern als Text, der ganz ausgegeben werden soll. Geben Sie jetzt einmal zur Kontrolle ein:

PRINT Ich gebe einen Satz wieder.

und drücken Sie auf RETURN.

Jetzt versteht der Computer nicht, was Sie eigentlich wollten. Er führt zwar den Befehl aus, aber das Resultat ist ganz anders, als Sie es sich vorgestellt hatten. Der MSX druckt nämlich eine 0 anstelle des Satzes. Dies kommt dadurch, daß der Computer wegen der fehlenden Anführungszeichen den Satz nicht als eine Reihe von Zeichen (einen String) ansieht, sondern als eine Variable (siehe Kapitel 6). Der Wert der Variablen ist 0, und das wird ausgedruckt.

Der MSX führt nur korrekt erteilte Befehle aus. Vergleichen Sie einmal die beiden folgenden Befehle miteinander:

```
BEEP
BEEEP
```

oder

```
COLOR 15, 6, 6
CLOR 15, 6, 6
```

Bei dem unteren dieser beiden Befehle ertönt ein Bliep, und es erscheint der Text:

```
Syntax error                    (Grammatikfehler)
```

Das bedeutet, daß der MSX den Befehl nicht versteht und deshalb auch nicht ausführt. Das Bliepzeichen dient dazu, Sie auf diese Tatsache besonders hinzuweisen. Der BEEP-Befehl bewirkt ein Bliep und der COLOR-Befehl legt die Farbe von Buchstaben, Schirm und Umrandung fest. Im Beispiel oben wurde die Farbe des Schirmes und des Randes von Dunkelblau in Dunkelrot geändert.

Fehler korrigieren

Bei der Korrektur von Fehlern sind die Pfeiltasten rechts auf der Tastatur sowie die anderen Gestaltungstasten besonders hilfreich. Der Cursor (der weiße Block, der angibt, wo das nächstfolgende Zeichen erscheinen wird) befindet sich nach Eingabe der oben genannten Beispiele unter dem Wörtchen Ok, das die Bereitschaft des Computers zur Entgegennahme weiterer Eingaben anzeigt. Wenn Sie den Cursor mit der großen Pfeiltaste nach oben verrücken, kommt er auf das O von Ok zu stehen. Dieser Buchstabe ist nun invers (in umgekehrten Farben) wiedergegeben. Drücken Sie noch 7 mal auf diese Taste, und der Cursor befindet sich über dem B von BEEEP. Bewegen Sie den Cursor mit der Pfeil-rechts-Taste auf eines der Es. Löschen Sie es mit der DEL-Taste (DELETE = Löschen). Drücken Sie nun auf RETURN. Der Befehl wird ausgeführt, ein Bliep ertönt, und unter dem Befehl erscheint ein auf den ersten Blick unverständlicher Text:

```
Okntax error
```

Sie werden bereits bemerkt haben, daß der alte Text „Syntax error" stehen bleibt und der Computer den neuen Text „Ok" darüberschreibt. Ein Teil des alten Textes bleibt sichtbar. Versuchen Sie auf die gleiche Weise, den mißlungenen COLOR-Befehl zu verbessern. Benutzen Sie anstelle von DEL, um einen Buchstaben zu entfernen, jetzt die INS-Taste (INSERT = einfügen), um ein Zeichen einzufügen (nach Benutzung wird INS wieder abgeschaltet, indem Sie diese Taste noch einmal

drücken). Versuchen Sie danach mit Hilfe der Tasten eine andere Farbe auf dem Schirm zu erzeugen (ersetzen Sie jede 6 durch eine 4 für dunkelblau, fügen Sie eine 8 für mittelrot und eine 10 für dunkelgelb ein). Beim Überschreiben von Zeichen dürfen Sie weder die DEL- noch die INS-Taste benutzen.

Versuchen Sie es einmal mit den folgenden drei Beispielen:

```
PRINT ''Ich gebe einen Satz wieder.
PRINT Ich gebe einen Satz wieder.''
? Ich gebe einen Satz wieder.''
```

Bedienen Sie sich hierbei auch der Gestaltungstasten, um das Eintippen zu verkürzen.

Beim ersten Fehlerbeispiel reagiert der Computer überhaupt nicht. Der Satz wird einfach abgedruckt. Das ist ausgezeichnet, wenn Sie mal etwas vergessen, aber in einer Programmzeile mit mehr als einem Befehl kann das verheerende Folgen haben. Im zweiten Beispiel erkennt der Computer wieder eine Variable. Denken Sie daran, einen Zeichenstring immer in Anführungszeichen zu schreiben. Das ist die einzige Art und Weise, die es dem Gerät ermöglicht, fehlerfrei zu arbeiten.

Das dritte Beispiel zeigt, daß anstelle des Befehles PRINT auch das kürzere ? verwandt werden kann. Auch nach dieser Form des PRINT-Befehles müssen die Anführungszeichen gesetzt werden!

Lassen Sie bei der Eingabe von Befehlen und Programmzeilen mit Befehlen folgende Punkte nicht außer acht:

Alle Tasten sind auto-repeat (= selbstwiederholend). Das heißt, daß jede Taste das zu ihr gehörige Zeichen so lange wiederholt, wie Sie diese Taste niedergedrückt halten. Das gilt auch für die Pfeiltasten.

Wenn Sie den Ton Ihres Fernsehgerätes oder Monitors eingeschaltet haben, hören Sie, daß jeder Anschlag ein Klinken erzeugt. Auf diese Weise können Sie mit Ihrem Gehör kontrollieren, ob Sie die Taste auch korrekt gedrückt haben. Jede Fehlermeldung wird vom Computer mit einem Bliep angezeigt.

Jeder Befehl oder jede Befehlszeile muß mit einem RETURN abgeschlossen werden. Dadurch wird der Befehl dem Computer eingegeben und er kann mit der Ausführung beginnen bzw. den Befehl oder die Befehlszeile speichern.

Um es kürzer zu machen, wird von jetzt an das Drücken der RETURN-Taste nicht mehr extra erwähnt. Nach jedem Befehl oder nach jeder Befehlszeile müssen Sie diese Taste drücken!

Befehle

Wir versuchen es mit einem weiteren Befehl:

```
PRINT 13+6
```

Ohne daß Sie es Ihrem MSX irgendwie mitgeteilt haben, kann er bereits addieren. Und nicht nur das: er kann alles, was auch Ihr Taschenrechner kann. (Allerdings soll-

ten Sie ihn deshalb nicht gleich ausrangieren, schließlich ist er um einiges kompakter und handlicher als ein MSX mit Bildschirm.)

Versuchen Sie es mal mit:

```
PRINT 12-6
PRINT 12/6
PRINT 12*6
PRINT 12^6
```

Sie sehen: perfekt! Allerdings müssen Sie die ein wenig abweichende Notation in BASIC beachten. Das Multiplikationszeichen ist ein Sternchen (*). Als Divisionszeichen benutzt man einen Schrägstrich (/). Potenzieren kann man nicht, indem man den Exponenten schräg über die zu potenzierende Zahl setzt, sondern man muß einen kleinen Pfeil oder ein kleines Dach (^) vor den Exponenten stellen.

Die Zahlen werden alle um eine Leerstelle nach rechts versetzt. Dadurch wird es möglich, daß dort das Zeichen der Zahl gesetzt werden kann. Bei positiven Zahlen (größer als Null) wird das Pluszeichen nicht dargestellt. Zahlen kleiner als Null wird ein Minuszeichen (-) vorangestellt, um auf diese Weise anzudeuten, daß es sich hier um negative Zahlen handelt. Versuchen Sie doch selber einmal 12 von 6 zu subtrahieren:

Natürlich kann der MSX auch schwierigere Berechnungen anstellen:

```
PRINT 12*6+2*5^1/4+13^2*13
```

Hat Ihr MSX als Ergebnis auch zweitausendzweihunderteinundsiebzigeinhalb errechnet?

Der MSX hält sich bei Rechenoperationen genau an die Reihenfolge, die auch sonst in der Mathematik gilt: zunächst Potenzieren (^), dann Multiplizieren und Dividieren (* und /), schließlich Wurzelziehen (SQR) und zum Schluß Addieren und Subtrahieren (+ und -). Bei gleichwertigen Operationen wie etwa Addieren und Subtrahieren geht der MSX von links nach rechts vor.

Wenn Sie eine bestimmte Rechenoperation vor einer anderen ausführen lassen wollen, können Sie das durch Klammern erreichen:

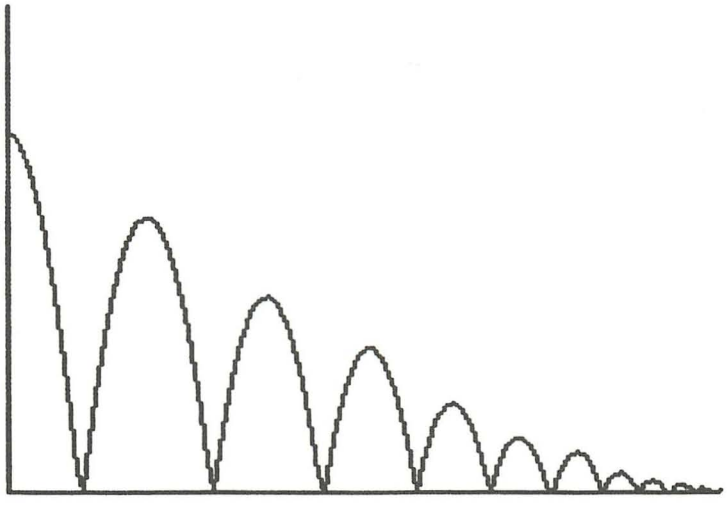
```
PRINT 3*5^2
PRINT (3*5) ^2
```

Im ersten Falle wird die Zahl 5 quadriert (25) und anschließend mit 3 multipliziert (75). Im letzteren Falle wird 5 mit 3 multipliziert (15) und das Resultat dann quadriert (225).

Es gibt neben dem PRINT-Befehl noch weit mehr direkte Befehle. Wenn Sie den Schirm nach all den Eingaben ein wenig chaotisch finden, tippen Sie einfach:

```
CLS
```

Sofort wird der ganze Schirminhalt gelöscht und links oben erscheinen der Hinweis Ok sowie der Cursor. Den gleichen Effekt ohne Ok-Meldung erzielen Sie durch gleichzeitiges Drücken von SHIFT und HOME. HOME alleine gedrückt bewirkt, daß der Cursor in die Stellung links oben springt, ohne den Bildschirm zu löschen.



5. EIN PROGRAMM SCHREIBEN

Ein Programm

Geben Sie das folgende Programm ein. Nach jeder Zeile drücken Sie natürlich auf RETURN. Einige Mikrocomputer akzeptieren keine Zeilen mit offensichtlichen Fehlern. Der MSX macht das sehr wohl. Programmzeilen (also keine direkten Befehle) werden immer akzeptiert. Erst wenn Sie den RUN-Befehl eingeben (der Computer also alle Programmzeilen der Reihe nach ausführen soll), meldet der MSX die eventuell gemachten Fehler.

Nach Eingabe einer jeden Zeile erscheint links im Bild der Cursor, und Sie können die nächste Zeile tippen.

```
10 REM EINE MULTIPLIKATION
20 LET A=99
30 LET B=12
40 LET P=A*B
50 PRINT A;"*";B;"=";P
```

Der Cursor steht vor der nächsten Zeile bereit. Sie tippen:

RUN (+ RETURN)

(Anstatt RUN in einzelnen Buchstaben einzutippen, können Sie auch F5 (RUN) drücken.)

Es zeigt sich, daß der MSX auch verschiedene Befehle nacheinander ausführen kann. Das hätten Sie natürlich auch nicht anders erwartet, wozu hat man denn schließlich einen Computer.

Achten Sie bei der Erteilung von Befehlen auf die folgenden Punkte:

In einem BASIC-Programm fängt jede Zeile mit einer positiven ganzen Zahl (der Zeilennummer) an.

Der Computer führt die verschiedenen Zeilen in der Reihenfolge der Zeilennummer aus.

In einer Zeile können verschiedene Befehle stehen (siehe etwa Zeile 50: der MSX muß dieser Zeile zufolge 5 verschiedene Dinge auf dem Schirm ausgeben. Dies ist eigentlich ein Befehl mit verschiedenen Untergliederungen. Später werden Sie richtigen sogenannten Multistatement-Zeilen (Zeilen mit mehr als einem Statement = Befehl, Funktion oder Feststellung) begegnen.

Es ist üblich, beim Schreiben von Programmen die Zeilennummern um jeweils 10 zu erhöhen. Auf diese Weise können Sie später immer noch einmal eine Zeile einfügen.

Sie müssen die Zeilen nicht in der richtigen Reihenfolge eingeben. Der MSX bringt sie jedoch in der korrekten Reihenfolge ins Programm und führt sie in dieser Reihenfolge auch aus.

(Der Befehl LET weist einem allgemeinen Wert – einer sogenannten Variablen – einen bestimmten Wert zu. LET A=99 heißt: von jetzt ab ist A gleichbedeutend mit 99. LET P=A*B heißt: von jetzt ab ist P gleichbedeutend mit A multipliziert mit B. Im Kapitel über die Variablen kommen wir darauf noch einmal zurück.)

Hilfen beim Herstellen von Programmen

Es sind verschiedene kleine Hilfen fest in den MSX installiert, die es ermöglichen, ein Programm rascher und einfacher zu schreiben.

Um das gesamte eingegebene Programm auf einen Blick übersehen zu können, tippen Sie:

LIST

(oder F4) und anschließend ein RETURN. Ist das Programm so lang, daß es nicht vollständig auf den Bildschirm paßt, verschwinden die oberen Zeilen. Versuchen Sie es einmal, indem Sie ein Programm mit 28 Zeilen schreiben. Tippen Sie in jede Zeile einfach ein ?. Wie Sie sehen, verwandelt der Computer beim AufLISTen des Programmes die ? in den Befehl PRINT. Dadurch, daß die oberen Zeilen wieder so schnell verschwinden, ist es sehr schwer, ein langes Programm zu lesen. Um einen Teil in aller Ruhe überschauen zu können, nehmen Sie die STOP-Taste zu Hilfe. Wenn Sie sie drücken, hört das Weiterrollen (scrollen) des Listings auf. Drücken Sie noch einmal auf STOP, wird es fortgesetzt. So können Sie jeden Teil des Listings genau lesen und gegebenenfalls auch verbessern.

Wollen Sie lediglich eine einzelne Zeile aus dem Programm noch einmal sehen, geben Sie ein:

LIST Zeilennummer

Für Zeilennummer tippen Sie die Nummer der Zeile, die Sie sehen wollen. Um einige Zeilen zusammen zu sehen, können Sie die folgenden Befehle tippen:

LIST Anfangszeile - Endzeile
LIST - Endzeile
LIST Anfangszeile -

Diese verschiedenen Befehle bewirken, daß das Programm von der ersten bis zur letzten Zeile aufgelistet wird, daß es vom Anfang des Programms bis zur letzten Zeile aufgelistet wird, daß es von der ersten Zeile bis zum Ende des Programms aufgelistet wird.

Wenn Sie bei der Eingabe eines Programms die Zeilennummern zwar jeweils um 10 erhöht haben, Ihnen die Numerierung durch nachträgliche Ergänzungen und Verbesserungen zu unübersichtlich erscheint, können Sie das Programm neu numerieren. Sie bedienen sich hierfür eines der folgenden Befehle (von RENUMBER = neu numerieren):

RENUM
RENUM erste neue Zeilennummer
RENUM erste neue Zeilennummer, erste alte Zeilennummer
RENUM , , Schrittlänge
RENUM 1-e neue Zlnr, 1-e alte Zlnr, Schrittlänge

Hinter dem Befehl RENUM können drei Werte eingegeben werden (das muß aber nicht gemacht werden). Der erste Wert gibt die Höhe der ersten neuen Zeilennummer an. Der zweite Wert sagt etwas über die neu zu numerierende alte Zeile, und der dritte Wert gibt die Schrittlänge zwischen den neuen Zeilennummern an.

RENUM alleine numeriert alle Zeilen eines Programmes neu, beginnend bei 10 und jeweils in Zehnerschritten. Das heißt, daß die neuen Zeilennummern 10, 20, 30, 40 usw. sein werden.

Nachdem Sie den zweiten Wert eingegeben haben, beginnen die neuen Zeilennummern nicht mehr bei 10, sondern bei einer genannten Zahl. So ergibt RENUM 200 die Zeilennummern 200, 210, 220, 230, usw.

Durch Einsetzen des zweiten Wertes wird die erste neu zu numerierende Zeile angegeben. Die Zeilen davor werden nicht neu numeriert, aber alle folgenden. Als erste neue Zeilennummer wird der erste angegebene Wert genommen.

Durch Angabe des dritten Wertes wird die Schrittlänge bestimmt. So ergibt RENUM , , 100 die Zeilennummern 100, 200, 300, 400, usw. (Die beiden Kommata hinter RENUM zeigen, daß die ersten beiden Werte nicht eingegeben worden sind.)

Die beiden eingegebenen Werte dürfen nicht höher sein als der erste eingegebene Wert. In diesem Falle würde der Computer über die alten Zeilennummern hinwegnumerieren.

Um wiederholtes Eingeben einer Zeilennummer zu sparen, kann der Computer veranlaßt werden, selber jedesmal eine Zeilennummer zu generieren. Dazu dient folgender Befehl:

AUTO erste Zeilennummer , Schrittlänge

Mit der ersten Zeilennummer geben Sie an, welche Zeilennummer der Computer zuerst generieren soll. Geben Sie nichts ein, wählt der Computer 10. Mit der Schrittlänge geben Sie an, wie groß die Schritte zwischen den einzelnen Nummern sein sollen. Geben Sie nichts ein, entscheidet sich der Computer für 10.

Sie stoppen das automatische Numerieren durch Drücken CTRL und STOP oder CTRL und C.

Wenn der Computer eine Zeilennummer für eine Zeile generiert, in der bereits etwas geschrieben steht, dann wird dies durch ein Sternchen (*) rechts der Zeilennummer angedeutet. Wollen Sie die Zeile nicht verändern, drücken Sie einfach auf RETURN, worauf die nächste Zeile erscheint. Wollen Sie die Zeile jedoch verändern, entfernen Sie das Sternchen mit der Pfeiltaste und geben die neue Zeile ein. Um die Zeile zu sehen, befehlen Sie einfach LIST.

Ein Programm erstellen

Nun, da wir die Möglichkeit haben, verschiedene Befehle nacheinander ausführen zu lassen, wobei wir wissen, daß sich der Computer an die Zeilennummern hält, steht uns ein System mit mehreren Vorteilen zur Verfügung. Tippen Sie:

```
20 LET A=67891.3
```

Nach Drücken von RETURN ist die alte Zeile 20 durch die neue ersetzt. Nichts weist jedoch darauf hin. Um hier eine Kontrolle zu haben, müssen Sie LIST eingeben.

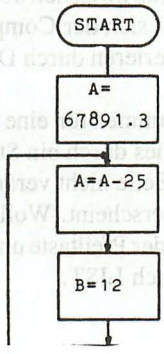
Durch das Numerieren der Zeilen werden diese aufrufbar und erkennbar. Wir können nun immer mal wieder eine Zeile verändern. Wir können auch auf eine bestimmte Zeile verweisen. Beispiele hierzu sind die bereits erwähnten Hilfen LIST, RENUM und AUTO. Aber auch innerhalb eines Programmes können Verweise angebracht werden. Geben Sie ein:

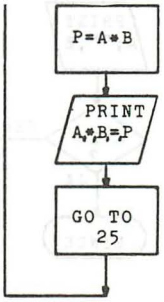
```
25 LET A=A-25
60 GOTO 25
```

Lassen Sie das Programm ablaufen (RUN, RETURN). Der MSX liefert eine unendliche Zahlenreihe. Sie geben ja schließlich in Zeile 60 den Befehl, wieder nach Zeile 25 zurückzugehen (GOTO = gehe nach). Dort angelangt, beginnt der MSX seinen Weg wieder durch das Programm. Dabei kommt er auch wieder zur Zeile 60, die ihn zurück nach Zeile 25 schickt. Um das Programm zu stoppen, drücken Sie CTRL und STOP. Wollen Sie es nur kurz anhalten, drücken Sie lediglich auf STOP.

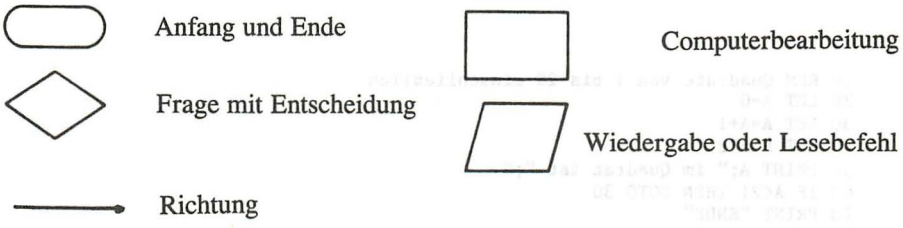
In der normalen englischen Sprache sind „go" und „to" zwei eigene Worte. MSX-BASIC macht jedoch keinen Unterschied zwischen „GOTO" und „GO TO".

Durch die Numerierung der Zeilen sind sogenannte Schleifen möglich geworden. Wenn wir das Programm oben in ein Flußdiagramm umschreiben (ein Schema, das den Verlauf des Programmes wiedergibt), erhalten wir:

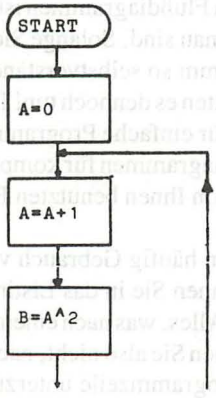


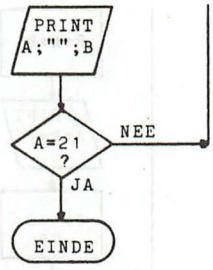


Es handelt sich hierbei um ein einfaches Flußdiagramm, aber sobald Sie ein etwas komplizierteres Problem lösen müssen, ist es sinnvoll, so ein Diagramm zu zeichnen, bevor Sie das Programm schreiben. Die wichtigsten Zeichen, die in einem Flußdiagramm verwandt werden, sind:



Als Beispiel nehmen wir einmal das folgende Problem: Erstelle eine Liste der Quadrate aller Zahlen von 1 bis 21. Das Flußdiagramm kann dann so aussehen:





Wenn daraus ein Programm gemacht wird, erhalten Sie das, was hierunter steht.

Das alte Programm entfernen Sie mit dem Befehl NEW (= neu) aus dem Speicher des MSX. Durch diesen Befehl werden alle Programme und alle Informationen aus dem Speicher des MSX entfernt; das gleiche erreichen Sie auch mit Hilfe der RESET-Taste. Doch dauert es dann länger, und es hat den Nachteil, daß bei dieser Methode auch eventuell im Speicher vorhandene Maschinensprachprogramme gelöscht werden. Mit NEW bleiben diese erhalten.

```
10 REM Quadrate von 1 bis 20 einschlieBlich
20 LET A=0
30 LET A=A+1
40 LET B=A^2
50 PRINT A;" im Quadrat ist ";B
60 IF A<21 THEN GOTO 30
70 PRINT "ENDE"
```

In Zeile 60 testet der MSX, ob A noch immer kleiner ist als 20 (A<20). Wenn nicht, dann ist A gleich 20, und das Quadrat daraus wurde gerade errechnet; wenn ja, dann muß das Programm noch einen Durchlauf machen, und die Maschine wird zurückgeschickt.

Die Verwendung von Flußdiagrammen ist kein überflüssiger Luxus für Menschen, die noch genauer als genau sind. Solange Sie kurze, einfache Programme schreiben, scheint das Flußdiagramm so selbstverständlich zu sein, daß ein Aufschreiben einfach unnötig ist. Sie sollten es dennoch tun! Die Übung, die Sie durch die Anfertigung von Flußdiagrammen für einfache Programme bekommen, wird Ihnen später bei der Anfertigung von Flußdiagrammen für kompliziertere Programme nützlich sein. Versuchen Sie, zu jedem von Ihnen benutzten Programm aus diesem Buch ein Flußdiagramm zu erstellen.

Machen Sie vor allem häufig Gebrauch von dem Befehl REM (REMARK = Bemerkung). Hiermit können Sie in das Listing Ihres Programmes Anmerkungen zur Erläuterung einfügen. Alles, was nach einem REM-Befehl steht, wird vom Computer nicht beachtet. Versuchen Sie also nicht, nach einem REM-Befehl einen weiteren Befehl in der gleichen Programmzeile unterzubringen.

10 REM lösche den Schirm: CLS	Das geht nicht. Der Computer negiert den CLS-Befehl.
10 CLS : REM lösche den Schirm	Das geht. Der Computer löscht den Schirm.

Achten Sie darauf, wie man mehr als nur einen Befehl in eine Programmzeile bekommt: Die Befehle werden voneinander durch einen Doppelpunkt (:) getrennt.

Sie können das Wort REM auch durch einen Apostroph (') ersetzen. Wenn Sie das Programm nun später aufLISTen, wird der Apostroph nicht in das Wort REM umgewandelt (vergleiche das ? für PRINT). Wenn Sie den REM-Befehl als letzten Befehl innerhalb einer Programmzeile benutzen, können Sie den Doppelpunkt (:) zur Trennung der einzelnen Befehle auch fortlassen, falls Sie sich ersatzweise für den Apostroph entscheiden:

10 CLS REM lösche den Schirm	Nicht möglich, (Syntax error).
10 CLS ' lösche den Schirm	Möglich.

Oben sahen Sie ein einziges Programm mit einem einzigen Verweis. Unten finden Sie ein Programm mit mehreren Verweisen. Dieses Programm wandelt Grad Fahrenheit in Grad Celsius um und umgekehrt. Die benutzten Formeln lauten:

Das Programm sieht dann so aus:

```
10 REM Fahrenheitgrade in Celsiusgrade umrechnen
20 CLS
30 PRINT : PRINT "Dieses Programm rechnet Grade F in
   Grade C um und umgekehrt."
40 PRINT : PRINT
50 PRINT "Moechten Sie F in C umrechnen (0)"
60 PRINT "oder C in F (1)?"
70 PRINT "Bitte 0 oder 1 RETURN eingeben."
80 REM Wahl
90 INPUT X
100 IF X=0 THEN GOTO 180
110 REM wenn X=1 faehrt das Programm hier fort
120 INPUT "Wieviel Grad? ";C
130 LET F=(9/5)*C+32
140 PRINT : PRINT C;"Grad Cels.= ";F;"Grad Fahr."
150 PRINT
160 REM beide Teile werden zusammen gefuehrt
170 GOTO 230
180 REM jetzt von F nach C
190 INPUT "Wieviel Grad? ";F
200 LET C= (5/9)*(F-32)
210 PRINT : PRINT F;"Grad Fahr.=";C;"Grad Cels."
220 REM jetzt folgt automatisch 230
```

```
230 PRINT : PRINT "Wuenschen Sie einen weiteren Durchgang?"
240 INPUT "Bitte 1 (ja) oder 0 (nein) RETURN eingeben";Y
250 REM bei Wahl 1 (ja) zurueck zum Anfang
260 IF Y=1 THEN CLS : GOTO 40
270 REM bei Wahl 0 (nein) Ende
280 PRINT : PRINT
290 PRINT "Bis zum naechsten Mal!"
```

Tippen Sie dieses Programm ein und speichern es auf einer Kassette. Bitte befolgen Sie dabei die Anweisungen Ihres Handbuches. Geben Sie folgendes ein:

CSAVE ''xxxxxx'' (sichern)

Setzen Sie zwischen die Anführungszeichen den Namen, den Sie diesem Programm geben wollen. Dieser Name darf aus maximal 6 Zeichen bestehen, und er muß mit einem Buchstaben anfangen.

Drücken Sie auf dem Kassettenrekorder die REC-Taste (Aufnahme) oder die SAVE-Taste (sichern) und danach den PLAY-Knopf. Das Kassettenband beginnt zu laufen.

Jetzt drücken Sie auf RETURN! Sobald diese Taste gedrückt wird, schickt der Computer Signale zum Bandgerät hinüber.

Sobald der Computer das gesamte Programm hinübergeschickt hat, erscheint die vertraute Nachricht Ok.

Nun kann die STOP-Taste des Kassettenrekorders gedrückt werden.

Wenn Sie einen Kassettenrekorder mit Fernbedienung haben und diese mittels eines kleinen schwarzen Steckers angeschlossen ist, wird der Computer das Abstellen des Bandes regeln. Sie müssen lediglich darauf achten, ob die REC- und PLAY-Knöpfe gedrückt sind. Das einzige, was der Computer steuern kann, ist der Lauf des Motors. Die Stellung der einzelnen Knöpfe (Abspielen oder Aufnehmen) kann vom Computer nicht geregelt werden.

Um ein Programm später zur Benutzung wieder in den MSX zu laden, spielen Sie die Kassette zurück zu der Stelle, an der sich das Programm befindet. Die beiden folgenden Methoden sind möglich:

1. Sie nehmen für jedes Programm ein neues Band. Dadurch haben Sie nur wenig Arbeit mit dem Umspulen, aber dafür einige Kosten für Bänder.
2. Sie notieren sich auf einem Zettel, bei welcher Nummer des Bandzählers welches Programm anfängt.

Um das Programm zu laden (um es also vom Band in den Speicher des Computers zu bringen), tippen Sie:

CLOAD''xxxxxx'' (laden)

Nun stellen Sie den Kassettenrekorder auf Wiedergabe (passen Sie auf, daß Sie nicht versehentlich den Aufnahmeknopf drücken!). Der Computer empfängt nun das vom Kassettenrekorder kommende Signal und registriert den Programmnamen. Wenn es das richtige Programm ist (Sie haben ihm ja schließlich den Namen des Pro-

grammes mitgeteilt anstelle von xxxxxx), wird er es in seinen Arbeitsspeicher laden.

Sind Sie sich nicht sicher, welches Programm auf der bereitliegenden Kassette ist, oder wissen Sie den genauen Programmnamen nicht mehr, geben Sie einfach ein:

```
CLOAD
```

Der Computer wird nun das erste Programm laden, das er auf der Kassette findet.

Sie können CLOAD benutzen, um sofort nach dem Sichern (SAVE) eines Programmes noch einmal zu kontrollieren, ob das Programm auch korrekt aufgenommen worden ist. Sie spulen die Kassette zurück zum Anfang der Aufnahme und geben ein:

```
CLOAD? "xxxxxx"                    (überprüfen)
```

Der Computer vergleicht nun das Programm xxxxxx auf dem Band mit dem Programm xxxxxx in seinem Speicher. Wenn die Programme identisch sind, ist das Programm gut auf der Kassette angekommen, und der Computer meldet Ok. Wenn die Programme jedoch nicht übereinstimmen, meldet der Computer:

```
Device I/O error                   (Fehler bei Ein- oder
                                    Ausgabe)
```

Sie müssen jetzt das Band wieder zum Anfang zurückspulen und das Programm noch einmal aufnehmen. Siehe oben.

Wenn der Name des zu kontrollierenden Programmes fortgelassen wird, vergleicht der Computer das erste Programm, das er auf dem Band findet, mit dem Programm in seinem Speicher.

Das Laden und Lesen von Programmen auf eine und von einer Diskette geht ähnlich vor sich. Sorgen Sie zunächst für eine formatierte Diskette (siehe Anleitung bei der Diskette). Der Befehl SAVE "xxxxxx" speichert das Programm auf der Diskette. Der Programmname darf nicht fortgelassen werden! Der Befehl LOAD "xxxxxx" liest das Programm von der Diskette in den Speicher. Auch hierbei muß der Programmname angegeben werden

Bei einem Kassettensystem ist es nicht schlimm, wenn für verschiedene Programme die gleichen Namen verwandt werden. Das ist zwar nicht gerade empfehlenswert, denn so kann es leicht zu Verwirrung kommen, aber der Computer wird die Programme ganz normal lesen und abspeichern. Bei einem Diskettensystem müssen alle Programme auf einer Diskette verschiedene Namen haben. Wenn Sie ein Programm unter einem bereits vorhandenen Namen speichern, wird das alte Programm überschrieben. Damit man sehen kann, welche Programme sich auf einer Diskette befinden, gibt es den Befehl FILES (Bestände). Alle Aufzeichnungen und Programme werden jetzt wiedergegeben.

In unserem Gradprogramm kommen einige neue Instruktionen vor. Unbekannt sind Ihnen bisher INPUT und IF...TEN...

Die Funktion von INPUT ist es, den Anwender selber eine Zahl, Zahlen, einen Buchstaben oder mehrere Buchstaben eingeben zu lassen.

Bei IF...THEN... wechselt der Computer, abhängig von einer bestimmten Bedingung, in einen anderen Teil des Programms. Auf diese Weise können wir die beiden Wahlmöglichkeiten des Programms (F oder C sowie „noch einmal") ausnutzen. Da es hier jedoch nur um Verweise ging (vorgeführt mit einem hübschen Programm), soll zur Erläuterung dieser Instruktionen auf die folgenden Kapitel hingewiesen werden.

Noch ein Tip. Durch einen einzelnen PRINT-Befehl erhalten Sie eine besser lesbare Ausgabe. Dieser Befehl bewirkt zusätzlichen Raum im Programmlisting, wodurch es besser zu überschauen ist. Wollen Sie jedoch lediglich etwas mehr Platz im Programmlisting haben, können Sie sich dazu des Apostrophs (=REM) bedienen, ohne jeden Kommentar. Sie können auf diese Weise die verschiedenen Teile eines Programmes voneinander trennen.

6. VARIABLEN

Dieses Kapitel ist vor allem für diejenigen gedacht, die nur wenig oder überhaupt keine mathematischen Kenntnisse haben. Ihnen bietet dieses Kapitel einen Überblick über den Gebrauch von Variablen, die beim Programmieren unerläßlich sind. Wenn Sie bereits genügend darüber wissen, lesen Sie das Kapitel flüchtig durch, es enthält einige wissenswerte Dinge über den Unterschied im Gebrauch der Variablen in der Mathematik und beim Programmieren.

Über das Rechnen haben Sie bereits einiges erfahren. Hier ein paar Beispiele:

```
2 + 2 = 4
6 - 3 = 3
2 x 2 = 4
6 : 2 = 3
```

Es läßt sich auch mit dem Computer eine Menge ausrechnen. Aber es wird doch lästig, wenn sich bestimmte Zahlen innerhalb eines Programmes immer wieder ändern. Jedes Mal müßte dazu das Programm aufgeLISTet werden, damit Sie die Zahlen austauschen können.

Die Summe 5 x 4 = 20 ist richtig errechnet, aber es ist nicht sinnvoll, das durch den Computer ausrechnen zu lassen. Interessanter wird die Summe für Sie, wenn Sie wissen, daß die Zahlen eigentlich stehen für:

Länge x Breite = Oberfläche

Mit dieser Formel ist es möglich, wesentlich mehr Zahlenbeispiele zu benutzen. Die erste Summe sagt lediglich, daß 5 mal 4 gleich 20 ist. Die zweite Summe sagt wesentlich mehr. Für jedes Rechteck gilt, daß Länge mal Breite gleich Oberfläche ist. Ein Rechteck mit einer Länge von 5 und einer Breite von 4 hat eine Oberfläche von 20. Was haben wir hier eigentlich gemacht? Die Begriffe Länge, Breite und Oberfläche sind vorübergehend durch 4, 5 und 20 ersetzt worden.

Es können für die gleichen Begriffe auch andere Zahlen eingesetzt werden:

6 (Länge) x 2 (Breite) = 12 (Oberfläche)

Wenn wir nun die Summe in der Form Länge x Breite = Oberfläche dem Computer eingeben und anschließend jedes Mal dem Computer mitteilen, wie groß Länge und Breite sind, kann der Computer immer wieder aufs Neue die Oberfläche eines Rechteckes berechnen.

Auf diese Weise können die festen Zahlen durch „Zahlen" ersetzt werden, die variieren, die sich verändern (sogenannte Variablen). Diese Formel kann nun allgemein angewandt werden. Es muß dem Computer nur ein einziges Mal mitgeteilt werden, was mit den Variablen zu geschehen hat; fortan braucht man der Maschine nur noch die jeweils neuen Werte für die Variablen anzugeben.

Länge x Breite = Oberfläche

Der Computer erhält nun die Angabe, daß Länge=5 und Breite=4.
Er setzt die Zahlen ein. In seinem Speicher steht jetzt:

5 x 4 = Oberfläche

So wie Sie mit Zahlen rechnen können, können Sie auch mit Variablen rechnen. Man nennt das dann Algebra.

Noch ein Beispiel für die Anwendung von Variablen.
Nehmen wir einmal an, daß Sie jede Woche an einem Kiosk eine Reihe von Zeitungen und Zeitschriften kaufen. Jede Zeitung kostet 2 DM und jede Zeitschrift 5 DM. Der Preis, den Sie zu bezahlen haben, hängt natürlich von der Anzahl der Zeitungen und Zeitschriften ab, die Sie kaufen. Für 3 Zeitungen und 4 Zeitschriften bezahlen Sie:

3 x 2 DM + 4 x 5 DM = 26 DM

Für 4 Zeitungen und 7 Zeitschriften bezahlen Sie:

4 x 2 DM + 7 x 5 DM = 43 DM

Wenn Sie nun mit den Computer errechnen wollen, wieviel Sie zu zahlen haben, müssen Sie nicht jedes Mal wieder die gesamte Aufgabe eintippen. Sie teilen dem Computer einfach mit, mit welcher Formel er den Preis errechnen muß, und darüber hinaus tippen Sie nun noch jede Woche die Anzahl der Zeitungen und Zeitschriften ein, die Sie kaufen wollen. Der Computer errechnet dann von alleine den Betrag. Sie geben ihm folgende Formel an:

Preis = Anzahl Zeitungen x 2 + Anzahl Zeitschriften x 5

oder ein wenig kürzer:

PR = ZU x 2 + ZS x 5

Wenn Sie anschließend den Preis für 3 Zeitungen und 4 Zeitschriften wissen wollen, geben Sie ein: ZU=3 und ZS=4. Der Computer setzt nun von alleine die Werte ein: PR = 3 x 3 + 4 x 5, und er nennt die Antwort: PR = 26. ZU und ZS sind Variablen: Wir können dafür immer wieder andere Zahlen einsetzen.

Achten Sie bei diesen Beispielen auf das Multiplikationszeichen. Dem Computer muß dies als ein Sternchen (*) angegeben werden.

Einige Punkte erfordern bei der Verwendung von Variablen besondere Aufmerksamkeit:

Beim Programmieren darf eine Variable nicht irgendeinen Namen erhalten. Es dürfen im Namen keine Leerstellen vorkommen. Der Name kann maximal 255 Zei-

chen lang sein, aber nur die ersten beiden Zeichen werden vom Computer auch gebraucht. Das zweite Zeichen des Variablennamens darf irgendein Zeichen, das erste Zeichen jedoch muß ein Buchstabe des Alphabets sein. Der Variablenname darf nicht mit einem für BASIC identisch sein; für BASIC reservierte Worte sind Begriffe, die in BASIC eine Funktion erfüllen. So ist beispielsweise der Variablenname TONNE nicht zulässig, weil er mit dem BASIC-Wort TO beginnt.

In der Algebra können Sie das Multiplikationszeichen beim Gebrauch von Variablen fortlassen. Beim Programmieren ist das nicht möglich. Der Computer betrachtet „ab" als die Variable mit den Namen „ab"; „a*b" wird von ihm als das Produkt aus den Variablen „a" und „b" angesehen.

Sie weisen einer Variablen einen Wert zu mittels des BASIC-Befehls

LET

Etwa LET b = 5. So wird der Variablen „b" der Wert „5" zugewiesen.

Daneben kann einer Variablen auch während des Programmlaufes (RUN) ein Wert mittels verschiedener, später erklärter Befehle zugewiesen werden; ein solcher Befehl etwa wäre:

INPUT

Beispielsweise INPUT b. So wird der Variablen „b" der Werte zugewiesen, den der Benutzer nach Erscheinen eines Fragezeichens auf dem Bildschirm eintippt.

Eine Variable kann auch einen variablen Wert erhalten. Dazu verwendet man den Befehl LET. Etwa LET a = b + 3. Dadurch erhält die Variable „a" den Wert „b + 3". Achtung! Das ist nur möglich, wenn der Computer bereits weiß, welchen Wert „b" hat. Sie müssen dies dem Computer also vorher mitteilen, und zwar durch einen direkten Befehl oder dadurch, daß die Wertzuweisung an „b" innerhalb des Computerprogramms für die Wertzuweisung an „a" erfolgt.

Im allgemeinen gilt dies für alle Mikrocomputer. Fast jeder Mikrocomputer reagiert mit einer Fehlermeldung (meist: no such variable = eine solche Variable gibt es nicht oder: variable unknown = Variable unbekannt) auf die Benutzung einer Variablen, bevor ihr ein Wert zugewiesen worden ist. Wenn Sie also Computerprogramme schreiben, die auch noch von anderen benutzt werden, sollten Sie das beachten.

Der MSX belastet sich mit dieser Regel allerdings nicht. Wenn Sie eine Variable gebrauchen, der noch kein Wert zugewiesen worden ist, geht der MSX davon aus, daß diese Variable den Wert 0 hat. Löschen Sie alle Eingaben durch RESET und tippen Sie:

```
LET A = B + 3
PRINT A
```

Der MSX gibt Ihnen die Antwort 3. Einerseits ist das sehr bequem, denn Sie haben nie Probleme mit der Wertzuweisung an Variable, aber andererseits kann es auch sehr

gefährlich sein. Machen Sie es sich zur Gewohnheit, zuerst allen Variablen einen Wert zuzuweisen, bevor Sie sie weiter gebrauchen.

Bei der Zuweisung eines Wertes darf das Gleichheitszeichen (=) nicht als Teil einer Behauptung angesehen werden:

```
LET A = A + 1
```

Dies muß gelesen werden als: Der Wert der Variablen „a" ist von jetzt an gleich dem alten Wert von „a" plus 1. Oder anders ausgedrückt: Zähle zum Wert der Variablen „a" 1 hinzu und betrachte das Resultat als den neuen Wert von „a". Es darf nicht gelesen werden als: „a" ist gleich „a+1".

Das Wörtchen LET ist in MSX-BASIC nicht absolut notwendig, um den Wert einer Variablen anzugeben. Sie können also auch tippen:

```
A = 10
```

Die Werte von Variablen können sowohl Zahlen (numerische Variablen) als auch Buchstabenreihen (Stringvariablen) sein. Über Stringvariablen erfahren Sie mehr im Kapitel über Strings.

MSX kennt drei Arten numerischer Variablen: Variablen, für die nur ganze Zahlen eingesetzt werden können (Integer), und Variablen, für die reale Zahlen eingesetzt werden können, wobei noch einmal zwischen Speicherung mit einfacher Genauigkeit und Speicherung mit doppelter Genauigkeit unterschieden wird.

Für eine integere Variable kann nur eine ganze Zahl eingesetzt werden. Der kleinste einzusetzende Wert ist -32768. Der höchste einzusetzende Wert ist 32767. Eine Integer-Variable unterscheidet sich von anderen numerischen Variablen durch das Prozentzeichen (%) hinter dem Namen der Variablen.

Beispielsweise:

```
A% = 8
```

Einer numerischen Variablen mit einfacher Genauigkeit kann eine Zahl mit maximal 6 Ziffern (vor oder hinter dem Komma) zugewiesen werden. Sind die Zahlen zu groß, werden die überzähligen Ziffern abgerundet und der MSX notiert die Zahl in Form eines Exponenten mit der Grundzahl 10. Etwa:

```
A! = 9.69974E+17
```

Der Exponentialteil kann sich zwischen -64 und +64 bewegen. Variablen einfacher Genauigkeit werden mit einem Ausrufungszeichen (!) hinter dem Namen der Variablen angedeutet.

Numerische Variablen mit doppelter Genauigkeit werden mit dem Nummernzeichen (#) hinter dem Variablennamen gekennzeichnet. Diese Variablen können Zahlen einfacher Genauigkeit mit maximal 14 Ziffern beiinhalten. Genau wie bei den Zahlen mit einfacher Genauigkeit werden die überschüssigen Ziffern abgerundet, und die Zahl mit einem E notiert:

```
A# = 9.6997445234213E+17
```

Wenn Sie dem MSX nichts anderes mitteilen, wird von jeder numerischen Variablen angenommen, daß sie eine Variable mit doppelter Genauigkeit sei. Versuchen Sie es einmal selbst:

```
RESET                          (Nach MSX-BASIC)
LET A=1
LET B=A/3
? B
```

Weiter oben haben Sie gelesen, daß bestimmte Variablen mit bestimmten Zeichen angegeben werden: integer (%), einfache Genauigkeit (!), doppelte Genauigkeit (#). Die Stringvariable wird mit einem Dollarzeichen angeben ($), wie Sie noch sehen werden. Wenn Sie hinter dem Namen einer Variablen nichts angeben, nimmt der Computer an, daß es sich um eine numerische Variable mit doppelter Genauigkeit handelt.

Es ist jedoch auch möglich, Variablen zu benennen. Danach brauchen Sie das Zeichen hinter dem Namen der Variablen nicht mehr anzugeben, weil der Computer sich merkt, zu welcher Gruppe von Variablen die betreffende Variable gehört. Die Instruktionen hierfür sehen folgendermaßen aus:

```
DEFINT Buchstabe (-Buchstabe)   Integer = ganze Zahl
DEFSNG Buchstabe (-Buchstabe)   Single = einfache Genauigkeit
DEFDBL Buchstabe (-Buchstabe)   Double = doppelte Genauigkeit
DEFSTR Buchstabe (-Buchstabe)   String = Zeichenkette
```

Bei Buchstabe setzen Sie den ersten Buchstaben des Namens der Variablen ein, die Sie benennen wollen. Eventuell können Sie mit Hilfe eines Striches und eines zweiten Buchstabens eine ganze Gruppe von Variablen beginnend mit einem der angegebenen Buchstaben benennen. Etwa:

```
DEFINT A
```

Dieser Befehl sorgt dafür, daß jede Variable, die mit einem A anfängt, eine Integer-Variable ist.

```
DEFINT A - L
```

Dieser Befehl bewirkt, daß jede Variable, die mit einem der Buchstaben zwischen A und L beginnt, eine Integer-Variable ist.

```
DEFSNG R - W
```

Mit diesem Befehl können Sie festlegen, daß jede Variable, die mit einem Buchstaben zwischen R und W beginnt, eine Variable einfacher Genauigkeit ist.

Wenn auch eine Variable mittels eines DEF-Befehls wie oben benannt ist, kann sie dennoch eine andere Funktion erhalten, und zwar durch Verwendung eines der Zeichen hinter dem Variablennamen. Beispielsweise:

```
DEFINT A - L
LET E = 1.456
? E                          Ergebnis: 1
LET E! = 1.456
? E!                         Ergebnis: 1.3
```

Die Variablenzeichen gehen der Benennung durch DEF vor, und die Benennung von DEF geht der allgemeinen Annahme vor, daß alle Variablen doppelt genaue Variablen sind.

Ein Überschuß an Ziffern wird bei der Zuweisung eines Wertes an eine Variable der falschen Art entfernt. Zu wenig Ziffern werden nicht ergänzt.

Vergleichen Sie:

```
LET G% = 1.9555
? G%                         Ergebnis: 1
LET Q# = 1.3
? Q#                         Ergebnis: 1.3
```

Die sicherste und eleganteste Art mit Variablen zu arbeiten ist, zu Anfang eines Programmes alle Variablen zu benennen. Bei jeder zu benutzenden Variablen liegt somit fest, von welcher Art sie ist. Auf diese Weise können Irrtümer vermieden werden.

7. EINGABE UND BILDWIEDERGABE

Input

Um mit dem Computer arbeiten zu können, müssen wir etwas eingeben (INPUT). Damit wir sehen können, was der Computer gerade macht oder was er gemacht hat, ist es notwendig, daß die Maschine uns etwas mitteilen kann (OUTPUT). Der MSX-Computer gibt seine Mitteilungen meist über ein Fernsehgerät oder über einen Monitor aus.

Einige der Möglichkeiten, Daten einzugeben, haben Sie bereits in den Kapiteln „Einleitung" und „Ein Programm schreiben" kennengelernt, über die Eingabemöglichkeiten während eines Programmlaufes haben Sie allerdings noch nichts gehört. Diese Möglichkeiten bringen verschiedene Vorteile mit sich. Es ist nicht mehr nötig, bei jeder Informationsänderung das Programm anzuhalten, damit die Änderungen eingegeben werden können, und anschließend dann das Programm wieder zu starten. Bei Spielprogrammen ist das sogar unentbehrlich, denn es muß eine Wechselwirkung zwischen Ihnen und dem Computer geben. Während des Programmlaufes müssen Sie auf Befehle oder Signale des Computers reagieren können, indem Sie bestimmte Eingaben machen. In MSX-BASIC gibt es hierzu den Befehl INPUT (=Eingabe).

Tippen Sie:

```
NEW
10 INPUT A
20 PRINT A;^ 2
```

Lassen Sie das Programm laufen (tippen Sie RUN oder drücken Sie auf F5). Links auf dem Schirm erscheint ein Fragezeichen, hinter dem sich der Cursor befindet. Der MSX erfragt von Ihnen einen Wert. Tippen Sie 12. (Selbstverständlich drücken Sie danach auf RETURN).

Auf dem Bildschirm erscheinen prompt die Zahlen 12 und 144, durch zwei Leerstellen voneinander getrennt. Der MSX trennt zwei Zahlen immer durch eine Leerstelle, die zweite Leerstelle ist für ein mögliches negatives Vorzeichen reserviert.

Der INPUT-Befehl funktioniert ähnlich wie der LET-Befehl. Aufgrund des INPUT-Befehles wartet der Computer, bis Sie einen Wert eingeben. Dieser Wert wird einer Variablen zugewiesen, die Sie hinter dem INPUT-Befehl nennen. Es ist auch möglich, verschiedenen Variablen mittels INPUT Werte zuzuweisen.

Tippen Sie:

```
10 INPUT A, B, C
20 PRINT A;''*'';B;''*''C;''=;A*B*C
```

Die Sternchen (=Multiplikationszeichen) zwischen den Anführungszeichen werden nicht durch den Computer ausgerechnet, sondern sie müssen ordentlich zwischen

die Zahlen auf den Schirm gesetzt werden. Sie stehen in Anführungszeichen, damit der Computer nicht mit ihnen rechnet.

Wenn Sie dieses Programm laufen lassen, erscheint wieder ein Fragezeichen mit dem Cursor dahinter. Sie können die drei Zahlen jetzt einsetzen. Die Zahlen, die Sie eingeben, können Sie auf zwei Arten voneinander trennen: durch Kommata oder durch RETURNs. In beiden Fällen muß der letzte Befehl ein RETURN sein. Sie können also eingeben:

```
1, 2, 3                    (RETURN)
```

oder

```
1                          (RETURN)
2                          (RETURN)
3                          (RETURN)
```

oder

```
1, 2                       (RETURN)
3                          (RETURN)
```

oder

```
1                          (RETURN)
2, 3                       (RETURN)
```

Sie sehen, es ist völlig gleichgültig, wie Sie die verschiedenen Zahlen eingeben. Die einzige Bedingung ist, daß die Zahlen voneinander getrennt sind, entweder durch ein Komma oder durch ein RETURN.

Sie müssen immer die richtige Menge an Information eingeben. Das heißt, es müssen genau soviele Angaben sein, wie der Computer für das Programm benötigt. Wenn Sie zu wenig Angaben machen, wird der Computer nur mit einem Fragezeichen antworten und auf weitere Informationen warten. Machen Sie zuviele Angaben, bleibt der Überschuß unberücksichtigt. Geben Sie in das oben stehende Programm einmal 1, 2, 3, 4 ein. Dies ist das Ergebnis:

```
?Extra ignored             (Überschuß ignoriert)
```

Leerstellen, die Sie (aus Versehen) vor, hinter oder innerhalb einer Zahl mit eingeben, werden vom Computer gleichfalls ignoriert. Geben Sie zum Beispiel einmal diese Zahlen ein: 2, 20, 3 10.

Erläuterungen

Bei dieser Art des Umgangs mit dem INPUT-Befehl gibt es einen Nachteil. Bei kleinen Programmen wie dem vorigen ist noch klar, was vom Benutzer eingegeben werden muß. Aber bei einem längeren Programm ist es nicht ausgeschlossen, daß der Benutzer einmal vergißt, welche Art von Eingaben der Computer genau erwartet. Im Kapitel über die Variablen haben Sie gesehen, daß es verschiedene Arten von Variablen gibt, die alle verschiedene Arten von Informationen beinhalten können. Die Ein-

gabe eines Wertes, der normalerweise nicht durch eine Variable ersetzt werden kann, führt deshalb zu ganz unerwarteten Ergebnissen.

Aus diesem Grunde gibt es die Möglichkeit, dem INPUT-Befehl Erläuterungen beizufügen. Geben Sie ein:

```
10 REM Wieviele cm faehrt ein Fahrrad
   bei einer Radumdrehung?
20 CLS
30 DEFINT A
40 INPUT "Raddurchmesser (ganze Zahl)?";A
50 PRINT
60 PRINT "Bei einem Raddurchmesser von ";A;" cm"
70 PRINT "kommt Ihr Fahrrad bei einer Umdrehung "
80 PRINT "des Rades ";A*3.14;" cm vorwaerts."
90 PRINT : GOTO 40
```

Jetzt fragt der Computer nicht mehr einfach nur nach einem Wert, sondern auf dem Bildschirm steht, nach welchem Wert gefragt wird, nämlich nach dem Reifendurchmesser, einzugeben als ganze Zahl.

Eine Erläuterung hinter dem INPUT-Befehl macht es erforderlich, zwischen die Erläuterung und den Variablennamen immer ein Semikolon (;) zu setzen. Hinter der Erläuterung kann mehr als nur eine Variable genannt werden. Beispiel:

```
INPUT ''Nenne vier Zahlen'' ; A, B, C, D
```

Die Variablen, die ihren Wert durch einen INPUT-Befehl erhalten, müssen nicht von der gleichen Art sein. Etwa:

```
INPUT ''Nenne drei Zahlen (Integer, real, integer)'' ; A%, B,C%
```

Neben der Benennung von numerischen Variablen mit numerischen Angaben können auch Stringvariablen mit Stringangaben belegt werden. Auch hierbei muß auf den Unterschied zwischen den verschiedenen Variablenarten geachtet werden:

```
INPUT ''Name, Alter'' ; A$;B%
```

Für die Eingabe einer ganzen Kette von Schriftzeichen gibt es bei MSX den INPUT-Befehl (sprich: Inputstring) sowie den LINE INPUT-Befehl. Diese Befehle werden im Kapitel über Strings eingehender behandelt werden.

Darstellung auf dem Bildschirm

Das Gegenstück der Dateneingabe ist die Darstellung der Daten durch den Computer auf dem Bildschirm. Einigen solcher Zeilen sind Sie ja schon begegnet. So wird vor jeder Zahl grundsätzlich eine Leerstelle für ein mögliches Minuszeichen freige-

halten. Daneben sorgt der MSX dafür, daß zwei Zahlen nie direkt aneinanderstehend auf dem Bildschirm dargestellt werden. Diese Zahlen würden sonst nicht mehr voneinander zu unterscheiden sein.

Die Darstellung von Daten und Mitteilung durch den Computer auf dem Bildschirm erfolgt meist mit dem PRINT-(=darstellen) Befehl. Es ist Ihnen natürlich bereits aufgefallen, daß der darzustellende Text bei diesem Befehl nicht immer im gleichen Abstand vom linken Rand beginnt. Denken Sie nur an die Zahlen mit der freigehaltenen Stelle davor für das Minuszeichen. Die Wiedergabe des Textes kann auf verschiedene Arten festgelegt werden. Zunächst indem verschiedene Zeichen zwischen die darzustellenden Texte gesetzt werden. Versuchen Sie es einmal mit dem folgenden Programm (tippen Sie zuerst NEW):

```
10 A=4
20 B=A^2
30 PRINT A;B;B^2
40 PRINT "MSX";"MICRO";"COMPUTER"
```

Wenn Sie dieses Programm laufen lassen, erkennen Sie, daß die Buchstaben direkt nebeneinander stehen und die einzelnen Zahlen durch einen Zwischenraum getrennt sind.

```
4     16     256
MSXMICROCOMPUTER
```

Wenn der MSX nicht beachten würde, daß Zahlen nie direkt nebeneinander stehen dürfen, hätten auch die Zahlen eine Reihe gebildet. Es zeigt sich, daß ein Semikolon (;) zwischen den PRINT-Daten bewirkt, daß die Daten ohne jeglichen Zwischenraum auf dem Schirm erscheinen. Das gleiche läßt sich auch mit einer Leerstelle ( ) erreichen, die das Semikolon ersetzt. Das allerdings ist nur bei darzustellenden Strings möglich! Bei Variablen muß grundsätzlich ein Semikolon eingesetzt werden. Ersetzen Sie im obenstehenden Programm zunächst die Semikola in Zeile 40 durch Leerstellen und lassen Sie das Programm dann laufen. Anschließend ersetzen Sie auch die Semikola in Zeile 30 durch Leerstellen und lassen das Programm wieder laufen. Achten Sie einmal auf den Unterschied.

Ein anderes Zeichen, das zwischen die PRINT-Daten gesetzt werden kann, ist das Komma (,). Setzen Sie im letzten Programm Kommata zwischen die PRINT-Daten und beobachten Sie, was dann passiert. Das Ergebnis sieht so aus:

```
 4                    16
 256
MSX                  MICRO
COMPUTER
```

Da scheint wohl etwas durcheinandergeraten zu sein. Der MSX setzt Daten, die durch ein Komma voneinander getrennt sind, immer in die nächste Spalte. Wenn es

sich also um mehr als zwei Daten handelt, wird die dritte Angabe wieder in die erste Spalte gesetzt, dann allerdings in der nächstfolgenden Zeile. Aus diesem Grund stehen 256 und COMPUTER in der ersten Spalte. Die Versetzung der Zahlen läßt sich wiederum mit dem für die Minuszeichen freigehaltenen Platz erklären.

Tabulation

Es gibt unterschiedliche Arten, die Darstellung auf dem Bildschirm zu beeinflussen. Wichtig dabei ist immer der Unterschied zwischen den verschiedenen Schirmen (screens). Der MSX verfügt über zwei Textschirme: SCREEN 0 und SCREEN 1.

SCREEN 0 hat eine Breite von 40 Zeichen. Die einzelnen Positionen sind von 0 bis 39 numeriert. Wenn Sie Ihren Computer allerdings einschalten und 40 Zeichen auf den Schirm bringen wollen, werden Sie schnell feststellen, daß tatsächlich nur 37 Zeichen in eine Zeile passen. Die ersten beiden Zeichenpositionen und die letzte Zeichenposition bleiben ungenutzt. Der MSX läßt standardmäßig an beiden Seiten des Schirmes einen breiten Rand. Wollen Sie alle 40 Zeichenpositionen ausnutzen (auch mit LOCATE, siehe weiter unten), müssen Sie zunächst ausdrücklich festhalten, daß der Schirm auch wirklich 40 Zeichen breit ist. Das machen Sie mit folgendem Befehl:

```
WIDTH 40
```

SCREEN 1 hat eine Breite von 32 Zeichen. Die Positionen sind von 0 bis 31 numeriert. Auch für diesen Schirm gilt, daß die Standardbreite nicht der tatsächlich möglichen Breite entspricht. Befehlen Sie

```
WIDTH 32
```

um den gesamten Schirm ausnutzen zu können.

Die Höhe eines jeden Textschirmes beträgt 24 Zeilen. Die Zeile ganz unten wird allerdings für die Wiedergabe der Funktionstasten freigehalten. Sie können das mit

```
KEY OFF
```

abschalten.

Und nun läßt sich auch die letzte Zeile noch beschriften. Wollen Sie die Wiedergabe der Funktionen wieder vor Augen haben, befehlen Sie einfach:

```
KEY ON
```

Mit dem WIDTH-Befehl läßt sich die Breite des Bildschirmes auch ganz neu festlegen. Der allgemeine Befehl dazu lautet:

```
WIDTH xxx                    (Breite)
```

An die Stelle der x setzen Sie die von Ihnen gewünschte Schirmbreite in Zeichenpositionen. So bewirkt etwa WIDTH 20 eine Schirmbreite von 20 Zeichen. Wenn Sie das gemacht haben, verändert sich allerdings auch die Numerierung der Zeichenpositionen! Der WIDTH-Befehl verkürzt den Schirm nämlich nicht nur an einer Seite, sondern entfernt zu beiden Seiten des Schirmes die gleiche Anzahl von Positionen. Auf diese Weise erscheint der neue Schirm genau in der Mitte. Die erste Position des neuen, nicht mehr so breiten Schirmes trägt die Nummer 0. Die letzte Position ist gleich der von Ihnen angegebenen Breite minus 1 (aufgrund der Tatsache, daß die Numerierung mit 0 beginnt). Sie können jede gewünschte Breite eingeben; Minimum ist eine Breite von 1, und Maximum ist die Breite des aufgerufenen SCREENs. Beide der unten behandelten Befehle berücksichtigen die Breite, die Sie mit WIDTH eingegeben haben.

Um den Computer zu veranlassen, einen Text oder eine Zahl nicht direkt am linken Rand des Schirmes darzustellen, sondern ein wenig weiter rechts, befehlen Sie:

TAB( ) (TABulator)

Dieser Befehl wird immer zusammen mit dem PRINT-Befehl verwendet. In der Klammer geben Sie die Zeichenposition an, an der der Textanfang stehen soll. Beispielsweise:

```
PRINT TAB(5) ''Position fünf''
```

Durch diesen Befehl wird der Text ''Position fünf'' an der sechsten Position vom linken Rand an beginnen (denn die Numerierung fängt ja mit 0 an). Das gleiche können Sie auch mit Zahlen machen:

```
PRINT TAB(9) 5*3
```

Nun wird die Zahl an Position 10 erscheinen; vor der Zahl wird ja eine Leerstelle für ein mögliches Minuszeichen freigehalten.

Die Gestaltung des PRINT TAB-Befehles ist nicht sonderlich kompliziert. Zwischen die Klammer und den zu druckenden Text kann eine Leerstelle eingeschoben werden, aber das ist keine Voraussetzung, ebenso zwischen PRINT und TAB. Zwischen die Klammer und das Anführungszeichen des zu druckenden Strings darf eine Leerstelle oder ein Semikolon eingeschoben werden, man kann es aber auch bleiben lassen. Wenn Sie dorthin ein Komma setzen, hat der TAB-Befehl nicht die gewünschte Wirkung, denn der Text wird in der nächsten Schirmspalte erscheinen.

Als einzige Voraussetzung muß darauf geachtet werden, daß zwischen TAB und der ersten Klammer keine Leerstelle liegt. Die Klammern mit der Zahl würden dann als Variablenname verstanden werden.

Sie können verschiedene TAB-Befehle hintereinander verwenden:

```
PRINT TAB(5) ''fünf'' TAB(15) ''fünfzehn''
```

Der TAB-Befehl springt nicht zurück. Wenn Sie in dem Beispiel oben aus der Fünfzehn eine Zwei machen, springt der Cursor nicht zurück, sondern der zweite Text wird hinter dem ersten erscheinen.

Geben Sie eine Zahl ein, die die Breite des Schirmes übersteigt, zählt der Computer einfach in der nächsten Zeile weiter.

LOCATE

Der Befehl LOCATE (plaziere) bringt den Cursor in eine bestimmte Position. Von dieser Position an kann anschließend gedruckt werden. Dieser Befehl ähnelt dem TAB-Befehl, aber es gibt doch einige Unterschiede:

Der LOCATE-Befehl darf nicht vor dem PRINT-Befehl erfolgen.

Der LOCATE-Befehl kann eine Position nicht nur horizontal (links-rechts), sondern auch vertikal (oben-unten) festlegen.

Wie bereits erwähnt, hat der Schirm nach einem WIDTH 40-Befehl eine Breite von 40 Positionen und eine Höhe von 24 Positionen. Mit dem LOCATE-Befehl kann der Cursor an einen beliebigen Punkt in einer beliebigen Zeile auf dem Schirm versetzt werden. Dies funktioniert folgendermaßen:

```
10 CLS
20 LOCATE 5, 4
30 PRINT ''Position fünf, Zeile 4''
```

Wie Sie sehen, setzen Sie beim LOCATE-Befehl keine Klammern, sondern trennen LOCATE und die Ziffern mit einem Leerzeichen (dieses Leerzeichen ist keine Bedingung). Die erste Ziffer bezeichnet die Buchstabenposition von der linken Seite aus, die zweite Ziffer definiert sie von oben. Jede dieser Ziffern kann fortgelassen werden, worauf der Computer annimmt, daß die augenblickliche Cursorposition auch weiterhin gilt:

```
10 CLS
20 LOCATE 5, 4
30 PRINT ''Position fünf, Zeile 4''
40 LOCATE , 20
50 PRINT ''Zeile 20'';
60 LOCATE 30
70 PRINT ''Position 30''
```

Wenn es darum geht, daß eine Zeilennummer auch weiterhin erhalten bleibt, ist es notwendig, den Computer nach Darstellung des Textes nicht automatisch in die nächste Zeile springen zu lassen. Aus diesem Grunde findet sich in Zeile 50 das Semikolon.

Wenn Sie die Zeichenposition fortlassen, müssen Sie dem Computer mitteilen, daß Sie mit der verbliebenen Zahl die Zeilennummer meinen. Deshalb bleibt das Komma in Zeile 40 vor der Zeilennummer stehen.

LOCATE kann nicht als direkter Befehl verwendet werden, da der Computer nach einem direkten Befehl immer auf die linke Seite zurückkehrt, um einen neuen Befehl entgegenzunehmen.

LOCATE kann auch dazu dienen, den Cursor ab- und wieder anzuschalten:

```
LOCATE , , 0          (Cursor aus)
LOCATE , , 1          (Cursor an)
```

Anstelle der Kommata können in dem gleichen Befehl auch Zeichenposition oder Zeilen angegeben werden.

Im folgenden Programm soll Ihnen der LOCATE-Befehl ermöglichen, sich an die Gestaltung eines Bildschirmes zu gewöhnen. Tippen Sie das Programm ein und üben Sie.

```
10 SCREEN 0 : WIDTH 40
20 PRINT "01234567890123456789012345678901234567890";
30 KEY OFF
40 FOR A=1 TO 23
50 LOCATE 0,A
60 PRINT A;
70 NEXT A
80 LOCATE 20,21
90 INPUT "SPALTE ";X
100 LOCATE 20,22
110 INPUT "ZEILE ";Y
120 LOCATE X,Y,0
130 PRINT "*(";X;",";Y;")"
140 LOCATE 27,21
150 PRINT "   "
160 LOCATE 20,22
170 PRINT "                "
180 GOTO 80
```

Versuchen Sie einmal herauszufinden, wozu das Semikolon in Zeile 60 da ist und was mit den Zeilen 140 bis 170 einschließlich beabsichtigt wird. Lassen Sie versuchsweise mal irgendetwas fort und achten Sie auf den Unterschied.

Gestaltung von Eingaben mit PRINT USING

Neben den oben genannten Möglichkeiten kann die Gestaltung eines jeden zu druckenden Textes auch individuell beeinflußt werden. Dazu verwenden Sie den folgenden Befehl:

```
PRINT USING ''Symbol'';
```

Anstelle von „Symbol" können verschiedene Symbole eingesetzt werden, die jeweils eine andere Wiedergabe des zu druckenden Textes oder der zu druckenden Zahlen bewirken.

```
PRINT USING ''!''
```

Durch dieses Symbol wird nur der erste Buchstabe des abzudruckenden Textes wiedergegeben.

```
PRINT USING ''!'' ; ''MSX''                    Ergebnis: M
```

Dieses Symbol ist nur für Strings zu gebrauchen (nicht für numerische Variablen).

```
PRINT USING ''\Leerstellen\''
```

Hiermit wird die Anzahl der Zeichen durch die Anzahl der Leerstellen plus 2 wiedergegeben. Wenn die abzudruckenden Angaben kürzer sind als die Anzahl der Leerstellen plus 2, wird der verbleibende Rest mit Leerstellen aufgefüllt.

```
PRINT USING ''\      \'';''MSX-COMPUTER''    Ergebnis: MSX-
```

Es ist möglich, hinter PRINT USING verschiedene zu druckende Angaben zu machen:

```
PRINT USING ''\      \'';''MSX'', ''COMPUTER'' Ergebnis: MSX COMP
```

Jede Angabe wird nun entsprechend dem jeweiligen Symbol dargestellt. Die Leerstellen zwischen MSX und COMP entstehen dadurch, daß die Leerstellen zwischen den Schrägstrichen zuzüglich zwei zusammen vier sind und MSX aus lediglich drei Zeichen besteht. Der Rest wird mit Leerstellen ausgefüllt.

```
PRINT USING ''..&..''
```

Der angegebene Test wird an der Stelle eingefügt, die mit & bezeichnet ist.

```
PRINT USING ''..&'';''MSX''                    Ergebnis: ..MSX..
PRINT USING ''& MSX''; ''HIER'', ''DA''        Ergebnis: Hier MSX Da
                                               MSX
```

```
PRINT USING ''#.## #''
```

Jedes Nummernzeichen (#) steht für eine Ziffer einer zu druckenden Zahl. Der Punkt (.) bezeichnet die Stelle des Dezimalkommas. Dies kann vor allem bei der Wiedergabe von Kolonnen mit Geldbeträgen nützlich sein, bei denen die DM-Beträge or-

dentlich untereinanderstehen müssen. Wenn die darzustellende Zahl mehr Ziffern hat, als mit den Nummernzeichen angegeben sind, wird ein % anstelle der Zahl erscheinen; gibt es weniger Ziffern, wird die Zahl zentriert. Das heißt, daß zu beiden Seiten der Zahl gleich viel Leerstellen erscheinen. Dies gilt jedoch nur für Zahlen vor einem Komma oder Zahlen ohne Komma. Hat die Zahl hinter dem Komma weniger Ziffern als durch Nummernzeichen angegeben, wird der Rest mit Nullen aufgefüllt; sind es zuviele Ziffern, wird der Rest abgerundet:

```
PRINT USING "# # #.# #";123.45          Ergebnis: 123.45
PRINT USING "# # #.# #";123.457         Ergebnis: 123.46
PRINT USING "# # #.# #";1234.57         Ergebnis: %1234.57
PRINT USING "# # #.# #";123.4           Ergebnis: 123.40
PRINT USING "# # #.# #";1.57            Ergebnis: 1.57
PRINT USING "# # #.# #";12              Ergebnis: 12
```

Bei der Darstellung der Zahlen wird das Pluszeichen (+) nicht mitgezählt. Das Minuszeichen (−) dagegen zählt als Ziffer mit.

```
PRINT USING "+# # # #"
```

Die Zahlen werden mit dem Minuszeichen (negativ, kleiner als Null) oder dem Pluszeichen (positiv, größer als Null) vor der Zahl dargestellt. Die Anzahl der Nummernzeichen ist bei Verwendung dieses Symbols nicht festgelegt. Zugleich darf der Dezimalpunkt oder eines der anderen Symbole verwandt werden.

```
PRINT USING "# # # #+"
```

Die Zahlen werden mit dem Zeichen hinter der Zahl dargestellt.

```
PRINT USING "+# # # #";123,-123         Ergebnis: +123-123
PRINT USING "# # # #+";123,-123         Ergebnis: 123+123-
```

```
PRINT USING "# # #-"
```

Negative Zahlen werden mit dem Minuszeichen hinter der Zahl dargestellt.

```
PRINT USING "**# # #"
```

Der Platz vor einer numerischen Eingabe wird mit Sternchen (*) aufgefüllt. Die Anzahl der zu verwendenden Nummernzeichen steht frei.

```
PRINT USING "**# # #";123, 12, 1        Ergebnis: **123***12
                                                  ****1
PRINT USING "£# #"
```

Die Zahl wird mit dem Pfundzeichen (£) davor dargestellt. Dieses Symbol wird als Ziffer bei der Gestaltung entsprechend einem der anderen Symbole mitgezählt.

```
PRINT USING ''£# #.# #";12.34          Ergebnis: £12.34
```

Durch Einsetzen eines Sternchens vor dem Pfundzeichen wird das Pfundzeichen direkt vor die darzustellende Zahl plaziert, und die Stellen davor werden mit Sternchen aufgefüllt.

```
PRINT USING ''*£#,#''
```

Wenn das Komma irgendwo vor dem Dezimalpunkt erscheint, wird zwischen jede der drei Ziffern ein Komma gesetzt. Dies ist die amerikanische Methode, Tausender darzustellen.

```
PRINT USING ''#,###.##";1234.567       Ergebnis: %1, 234.57
```

Verwenden Sie das Komma nicht als erstes Symbol.

```
PRINT USING ''# #^^^^''
```

Diese Symbole bestimmen die Gestaltung des Exponentialteiles bei wissenschaftlicher Schreibweise. Es wird Platz geschaffen für E+..

```
PRINT USING ''# #.# #^^^^";123456.78    Ergebnis: 1.23E+05
```

Die meisten der oben genannten Symbole können kombiniert werden. Sie haben das ja bereits bei dem Symbol # gesehen, das fast überall dazu dient, die Anzahl der Zifferstellen anzugeben.

Angeln

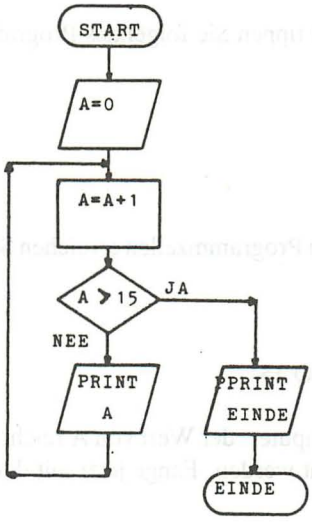
Zum Schluß sehen Sie hier noch ein nettes Programm, in dem zahlreiche LOCATE-Befehle vorkommen. Versuchen Sie, das Programm zu verstehen, wenn Sie lediglich das Listing lesen. Fertigen Sie sich ein Flußdiagramm an. Es finden sich in diesem Programm eine Reihe von Befehlen, die Sie bisher noch nicht gesehen haben. Sollten Sie mit dem Programm zunächst nichts anfangen können, setzen Sie einfach die Lektüre des Buches fort. Nach und nach werden Ihnen alle hier vorkommenden Befehle verständlich werden.

```
5 REM angeln
10 SCREEN 0 : WIDTH 40 : KEY OFF
15 DEFINT A-Z
20 LET S=0
30 LET PU=0
```

```
40 PRINT "Schwierigkeitsgrad (30-5)"
50 PRINT "30 = leicht"
60 PRINT "5  = schwer";
70 INPUT M
80 INPUT "GESCHWINDIGKEIT (1-10) ";V
90 CLS
100 FOR ST=1 TO 20
110 LOCATE 25,23 : PRINT "score:";PU;
120 LOCATE M+2,0 : PRINT"<<^>>";
130 LET RA=INT(RND(1)*20)+1
140 FOR X=0 TO 37
150 LOCATE X,RA : PRINT" *";
160 LOCATE M+4,S : PRINT":";
170 LOCATE M+4,S+1 : PRINT" ";
180 C=STICK(0)
190 IF S=0 THEN GOTO 220
200 IF C=1 THEN S=S-1
210 IF S=23 THEN GOTO 230
220 IF C=5 THEN S=S+1
230 LOCATE M+4,S : PRINT"#";
240 IF M+4=X AND S=RA THEN :BEEP : BEEP: BEEP : PU=PU+1 :GOTO 270
250 FOR Z=1 TO 50-V*5 : NEXT Z
260 NEXT X
270 NEXT ST
280 PRINT "Ende"
```

8. SCHLEIFEN

In fast jedem Programm ist es sinnvoll, den Computer eine bestimmte Aufgabe mehrere Male verrichten zu lassen, jedesmal entweder mit den gleichen oder mit anderen Daten. Nehmen wir einmal an, daß folgendes Problem vom Computer gelöst werden soll: die Zahlen von 1 bis 15 einschließlich sollen auf dem Bildschirm wiedergegeben werden, ohne daß dazu alle fünfzehn Ziffern eingegeben werden müssen. Wir können dazu zunächst einmal ein Flußdiagramm anfertigen:



Auf diese Weise wurde ein Loop (=Schleife) in das Programm eingebaut. Der Computer dreht, macht innerhalb des Programmes seine Runden und verläßt es erst, wenn A größer als 15 ist. Wenn der Computer die Schleife zum ersten Male durchläuft, ist der Wert von A gleich mit 0+1=1. Der Computer zeigt das an und geht über zum nächsten Befehl, demzufolge kontrolliert werden muß, ob A größer als 15 ist. 1 ist kleiner als 15, also lautet die Antwort: nein. Der Computer wird zurückgeschickt, zählt wieder 1 zu A hinzu. A wird nun 2. Dies wird wieder angezeigt, mit 15 verglichen usw. Diese Runde, diese Schleife also, macht der Computer solange, bis die Antwort auf die Frage „ist A größer als 15?" positiv ist. Wenn A=16, ist A größer als 15, und der Computer kehrt nicht mehr in das Programm zurück, sondern geht zur nächsten Zeile über, die den Abschluß des Programmes anzeigt. Hier also haben wir ein Beispiel für ein Programm, bei dem der Computer zwar auf die Reihenfolge der Zeilennummern achtet, aber in dem der Programmierer dafür gesorgt hat, daß der Computer bestimmte Programmteile verschiedene Male durchläuft.

Das Programm zu dem obenstehenden Flußdiagramm sieht so aus:

```
10 LET A=0
20 LET A=A+1
30 IF A>15 THEN GOTO 60
40 PRINT A
50 GOTO 20
60 PRINT "ENDE"
```

Aber bei einem Computer wie dem MSX kann man hierfür natürlich einen Trick erwarten. Und den gibt es dann auch. Um den Ablauf oben zu beschleunigen, verwenden Sie

FOR...NEXT... (von...als Nächstes...)

Geben Sie NEW ein und tippen Sie folgendes Programm:

```
10 FOR A = 1 TO 15
20 PRINT A
30 NEXT A
```

Mit weniger und kürzeren Programmzeilen erreichen Sie das gleiche Ziel. Der Befehl funktioniert so:
In der ersten Zeile steht

FOR A = 1 TO 15

Damit sagt man dem Computer: der Wert von A reicht von 1 bis 15 einschließlich und soll immer um 1 erhöht werden. Fange jetzt mit dem ersten Wert an.
Zeile 20:

PRINT A

Diese Zeile sagt dem Computer, was er mit A machen muß. In diesem Falle nur PRINT, aber alle anderen BASIC-Anweisungen sind möglich, auch BASIC-Befehle, bei denen A überhaupt keine Rolle spielt. Es gibt keine Begrenzung der Befehlsanzahl, aber sie müssen alle zwischen der Zeile mit FOR und der Zeile mit NEXT stehen.
Zeile 30:

NEXT A

Diese Zeile verlangt vom Computer, den nächsten Wert für A zu nehmen, zu der Zeile zurückzukehren, in der FOR steht, und dort nachzusehen, ob der Wert nicht größer geworden ist als der angegebene Bereich (TO 15). Ist das nicht der Fall, geht der Computer zur nächsten Zeile über und führt dort die Bearbeitung mit dem neuen Wert für A aus. Anschließend kehrt er zu Zeile 30 zurück, wo A wieder um 1 erhöht wird usw. usw.

Sobald A den Wert 16 erreicht, führt der Computer den Auftrag aus Zeile 20 nicht mehr aus und springt in die Zeile, die dem NEXT-Statement (=Formulierung eines Befehles) folgt. In unserem Programm gibt es keine nächste Zeile, und so ist das Programm zu Ende. Mit dem FOR...NEXT-Befehl können Sie den Computer einen bestimmten Befehl mehrere Male ausführen lassen, wobei Sie die Anzahl der Wiederholungen festlegen, und der Computer den Wert einer genannten Variablen verändert.

Aber mit dem FOR...NEXT-Befehl kann man noch mehr anfangen. Die einzelnen Schritte, die der Computer bei der Veränderung des Wertes macht, können Sie beeinflussen. Geben Sie einmal das folgende Demonstrationsprogramm ein:

```
10 REM Demonstration FOR...NEXT
20 CLS
30 PRINT "FOR X=16 TO 25 ergibt: "
40 PRINT
50 FOR X=16 TO 25
60 PRINT X
70 NEXT X
80 PRINT
90 PRINT
100 REM Pause
110 FOR X=1 TO 5000 : NEXT X
120 REM auch groessere Schritte moeglich
130 PRINT "FOR X=1 TO 21 STEP 2 ergibt: "
140 PRINT
150 FOR X=1 TO 21 STEP 2
160 PRINT X
170 NEXT X
180 PRINT : PRINT
190 REM Pause
200 FOR X=1 TO 5000 : NEXT X
210 REM abwaerts
220 PRINT "FOR X=634 TO 517 STEP -9 ergibt: "
230 PRINT
240 FOR X=634 TO 517 STEP -9
250 PRINT X
260 NEXT X
270 PRINT : PRINT
280 REM Pause
290 FOR X=1 TO 5000 : NEXT X
300 REM auch nicht-ganze Zahlen
310 PRINT "FOR X=117.2 TO 3 STEP -15.7 ergibt: "
320 PRINT
330 FOR X=117.2 TO 3 STEP -15.7
340 PRINT X
350 NEXT X
360 PRINT : PRINT
370 REM Pause
380 FOR X=1 TO 5000 : NEXT X
390 REM Variablen benutzen
400 PRINT "A=7, B=87, C=8"
410 PRINT "und FOR X=A TO B STEP C ergibt: "
420 PRINT
430 LET A=7 : LET B=87 : LET C=8
440 FOR X=A TO B STEP C
450 PRINT X
460 NEXT X
```

Wenn Sie dieses Programm laufen lassen, erhalten Sie auf dem Bildschirm eine Auswahl von Beispielen möglicher Variationen mit der FOR...NEXT-Schleife.

Darüber hinaus erkennen Sie im Listing noch einige andere wichtige Dinge. Es wurde schon früher darauf hingewiesen, aber achten Sie auf die Verwendung des REM-Statements. Alles, was dahinter steht, wird vom Computer überschlagen. Sie können damit Ihr Listing klarer machen.

Zeile 100 nennt als Kommentar nur „Pause". In Zeile 110 finden Sie eine Möglichkeit, die FOR...NEXT-Schleife als Bremse fungieren zu lassen. Der einzige Befehl, den Sie eingeben, dient dazu, den Wert der Variablen X viele Male zu erhöhen. Damit ist der Computer einige Zeit beschäftigt, und so verlangsamt sich das Programm soweit, daß Sie die Ergebnisse in Ruhe lesen können. Diese Art der Verzögerung ist jedoch nicht für genaue Zeitbestimmungen geeignet. Um eine präzisere Zeitmessung zu erläutern, verlassen wir kurz unsere FOR...NEXT-Schleifen.

Zeit messen

Der MSX bedient sich sehr oft sogenannter Interupts oder Unterbrechungen. Das heißt, daß dasjenige, womit sich die Z-80A-Zentraleinheit gerade beschäftigt, kurz für eine andere Aufgabe unterbrochen wird. Etwa dann, wenn die Z-80A Ihr BASIC-Programm ausführt, während Sie eine Taste der Tastatur drücken. Der Prozessor stoppt dann für einen Augenblick das BASIC-Programm, um nachzuschauen, welche Taste Sie gedrückt haben. Die Eingabe legt er in einem besonderen Puffer (Speicherraum) zur späteren Verwendung ab.

Neben diesen gelegentlichen Unterbrechungen stoppt die Z-80A jede fünfzigstel Sekunde, um ein Signal von einer internen „Uhr" zu empfangen und den Wert einer besonderen Vaiablen TIME (Zeit) um eins zu erhöhen. So gibt es verschiedene Dinge, die die Z-80A zwischendurch noch erledigen muß. Das bedeutet, daß die Geschwindigkeit, mit der Ihr BASIC-Programm arbeitet, nicht immer genau die gleiche ist. Die FOR...NEXT-Schleife erlaubt daher also keine präzise Zeitmessung.

Die genaue Zeitbestimmung kann allerdings mittels der Variablen TIME erfolgen. Jede fünfzigstel Sekunde (das entspricht der Frequenz des Stromnetzes) wird der Wert der Variablen TIME – wie bereits erwähnt – um eins erhöht. Diese Anhebung des Wertes beginnt in dem Augenblick, in dem Sie Ihren MSX einschalten und wird kontinuierlich fortgesetzt. Nur in den Fällen, in denen Unterbrechungen nicht möglich sind (unter anderem beim Lesen oder Beschreiben) einer Kassette, wird die Uhr zeitweise abgeschaltet. Tippen Sie:

```
PRINT TIME
```

Sie erhalten jetzt die Zeit, in fünfzigstel Sekunden. Das Abschalten des MSX bewirkt, daß TIME wieder auf Null zurückspringt. TIME kann einen Maximalwert von 65535 annehmen; danach beginnt die Variable wieder bei 0.

Um mit einem Programm eine bestimmte und präzise Zeitdauer bestimmen zu können, verwenden Sie folgendes Programm:

```
10 TIME = 0
20 ENDE = TIME + 500
30 REM wir warten
40 PRINT ''Wir warten.....'';
50 PRINT ''.'';
60 IF TIME < ENDE THEN GOTO 50
70 PRINT : PRINT ''10 Sekunden.''
```

In Zeile 20 geben wir ein, daß ENDE gleich mit TIME plus 500 (fünfzigstel einer Sekunde) ist. Solange ENDE größer ist als TIME, macht der MSX Punkte auf den Bildschirm. Da das schiefgehen kann, wenn TIME zu Beginn dieses Programmteiles fast 65535 beträgt, stellen wir in Zeile 10 den Wert von TIME auf 0. Nach 10 Sekunden (genau) ist TIME 500 mal um eins erhöht worden und ist nun gleich groß mit ENDE.

Mit folgender Rechnung können Sie die fünfzigstel Sekunden in Sekunden umsetzen:

```
M =  FIX(TIME/3000)
S =  FIX(TIME/50 - M*60)
```

Es passen nämlich 50 Fünfzigstel einer Sekunde in eine Sekunde, aber nach 60 Sekunden muß die Uhr wieder bei 1 anfangen. Deshalb wird jedes sechzigste Mal die Anzahl der Minuten von der Gesamtzahl der Sekunden abgezogen.

FIX (fest)

gibt ganze Zahlen an. Ziffern hinter dem Komma bleiben unberücksichtigt. Das Statement sieht aus wie

INT (integer)

Aber dieses Statement gibt die größte Zahl an, die noch kleiner ist als die eingegebene Zahl. Vergleichen Sie einmal genau folgende Beispiele:

? FIX(5)	ergibt	5	? INT(5)	ergibt	5
? FIX(4.9)	ergibt	4	? INT(4.9)	ergibt	4
? FIX(−5)	ergibt	−5	? INT(−5)	ergibt	−5
? FIX(−4.9)	ergibt	−4	? INT(−4.9)	ergibt	−4

Wollen Sie auch Stunden erhalten, wird die Sache schon schwieriger. Es ergeben schließlich 50*60*60 Fünfzigstel einer Sekunde eine Stunde. Das macht 180 000. Die Variable TIME reicht jedoch nur bis 65 535 und fängt dann wieder von vorne an zu zählen. Deshalb müssen die Stunden in Minuten ausgedrückt werden. Eine Digitaluhr können Sie wie folgt darstellen:

```
10 KEY OFF
20 SCREEN 1
30 TIME =0
40 U=0
50 M=FIX(TIME/3000)
60 S=FIX(TIME/50-M*60)
70 IF M>0 AND INT(M/60)=M/60 THEN U=U+1
80 LOCATE 6,8 : PRINT "***************"
90 LOCATE 8,10 : PRINT "Uhr ";U
100 LOCATE 8,11 : PRINT "Min ";M
110 LOCATE 8,12 : PRINT "Sek ";S
120 LOCATE 6,14 : PRINT "***************"
130 GOTO 50
```

Zurück zu den Schleifen

Die FOR...NEXT-Schleife kann auch für andere Zwecke als für die Veränderung von Variablen eingesetzt werden. Sie können den Computer mit diesem Befehl bestimmte unveränderliche Aufgaben mehrfach wiederholen lassen. In diesem Falle dient die Schleife lediglich als Zähler. Ein Beispiel hierzu haben Sie als Pausenschleife bereits kennengelernt:

```
10 FOR A=1 TO 22
20 PRINT ''.....(Ihr Name).....''
30 NEXT A
```

Auf diese Weise zählt der Computer zwar jedesmal zu A 1 hinzu, aber er benutzt A zu nichts anderem als zum Beibehalten der Zählung.

Die Anzahl der Befehle, die wiederholt werden müssen, bestimmen Sie selber. Jeder Befehl, der zwischen der FOR-Zeile und der NEXT-Zeile steht, wird genau so oft wiederholt, wie Sie den Computer diese Schleife durchlaufen lassen.

Schleifen innerhalb von Schleifen

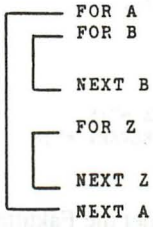
Sie können auch innerhalb von Schleifen noch einmal Schleifen einrichten. Diese werden dann verschachtelte Schleifen genannt. Wie verschachtelt liegen die Schleifen ineinander. Die innere Schleife wird jedesmal, wenn die äußere Schleife durchlaufen wird, gleichfalls so oft durchlaufen, wie Sie wünschen. Ein Beispiel mag das verdeutlichen.

Tippen Sie NEW sowie das folgende Programm ein:

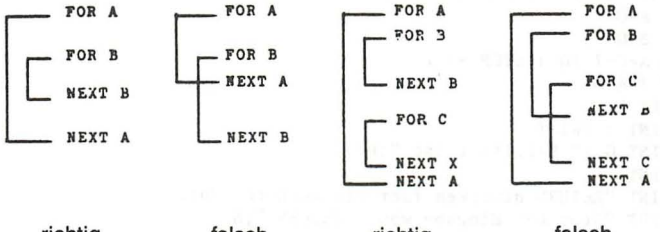
```
10 FOR A=1 TO 3
20 PRINT "AEUSSERSTE SCHLEIFE NR. ";A
30 PRINT
40 FOR B=1 TO 4
50 PRINT "INNERSTE SCHLEIFE NR. ";B
60 NEXT B
70 PRINT
80 FOR Z=1 TO 2000 : NEXT Z
90 NEXT A
```

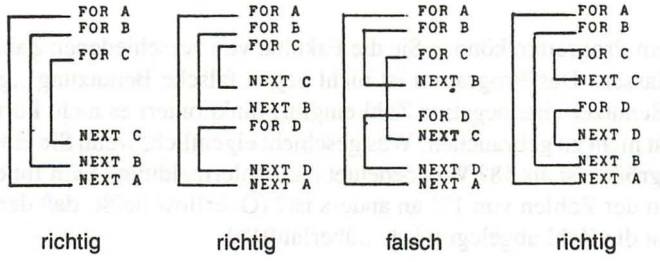
Dieses Programm hat drei Schleifen, deren erste die spezielle Pausenschleife ist.

Die Schleife FOR B=1 TO 4...NEXT B liegt vollständig innerhalb der Schleife FOR A=1 TO 3...NEXT A. Neben der B-Schleife liegt noch die Schleife FOR Z=1 TO 2000...NEXT Z. Auch diese Schleife befindet sich innerhalb der A-Schleife. Schematisch dargestellt sieht das so aus:



Achten Sie bei der Einrichtung verschachtelter Schleifen besonders darauf, daß die innere Schleife auch tatsächlich innerhalb der äußeren Schleife liegt. Die Schleifen dürfen einander nicht überschneiden!





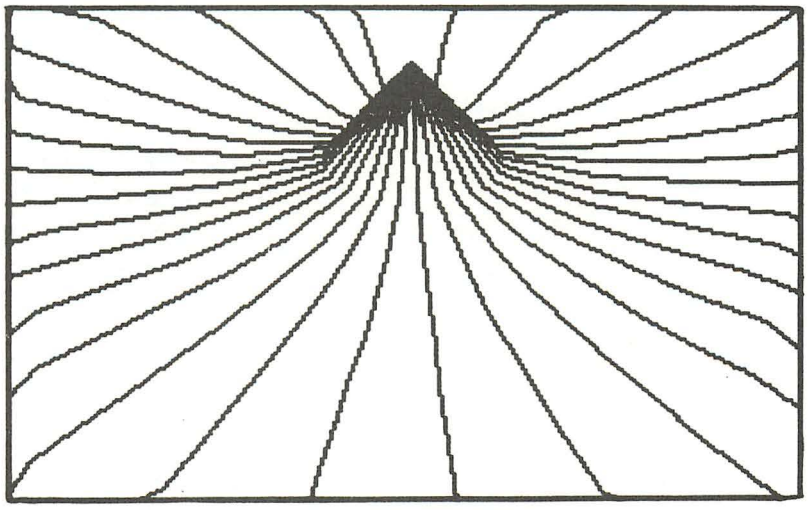
Zum Abschluß dieses Kapitels nun noch zwei kurze Rechenprogramme.
Das erste Programm errechnet für Sie den Mittelwert einer Zahlenreihe.

```
10 SCREEN 0
20 PRINT : PRINT "Ich errechne fuer Sie den Mittelwert "
30 PRINT "aus einer Liste von Ihnen genannter"
40 PRINT "Zahlen."
50 PRINT : INPUT "WIEVIELE ZAHLEN IN DER LISTE";N
60 PRINT : PRINT "Druecken Sie nach jeder Zahl RETURN."
70 PRINT
80 S=0
90 FOR L=1 TO N
100 PRINT "Zahl Nr.";L;" = ?";
110 INPUT X
120 S=S+X
130 NEXT L
140 M=S/N
150 PRINT : PRINT "Die SUMME =";S
160 PRINT : PRINT "Der MITTELWERT =";M
```

Das nächste Programm berechnet die Fakultät einer Zahl. Die Fakultät der Zahl N wird mit N! angegeben (sprich: N Fakultät). Das steht für N*(N-1) * (N-2) * (N-3).....3*2*1. Normalerweise wird das nur bei positiven ganzen Zahlen gebraucht. Das Programm sieht dann so aus:

```
10 SCREEN 0
20 PRINT "Fakultaetsberechnung"
30 PRINT
50 INPUT "BITTE EINE ZAHL.";G
60 LET A=G
70 LET B=G
80 FOR A=G-1 TO 1 STEP -1
90 LET B=A*B
100 NEXT A
120 PRINT : PRINT
130 PRINT G ;" Fakultaet ist ";B
140 PRINT
150 PRINT "RETURN druecken fuer ein weiteres Mal."
160 INPUT "Stop bei Eingabe von 1 RETURN.";N
170 PRINT : IF N<>1 THEN GOTO 30
180 PRINT "Ende"
```

Mit diesem Programm können Sie die Fakultät von verschiedenen Zahlen schnell errechnen lassen. Das Programm ist nicht gegen falsche Benutzung „gesichert". Wenn der Benutzer eine negative Zahl eingibt, funktioniert es nicht korrekt. Auch ein Bruch ist nicht zu gebrauchen. Was geschieht eigentlich, wenn Sie eine Zahl eingeben, die größer ist als 48? Was bedeutet die Fehlermeldung? Fällt Ihnen auf, daß die Notation der Zahlen von 17! an anders ist? (Overflow heißt, daß der Speicherplatz, in den die Zahl abgelegt wird, „überläuft".)



9. LOGIK UND IF...THEN...

Vergleichen

Ihr MSX ist zwar ein ausgezeichneter Computer, aber selbständiges Denken kann man von ihm nicht verlangen; auch Entscheidungen trifft der MSX nicht. Wohl aber kann der Computer zwei Zahlen miteinander vergleichen. Das Verhältnis zwischen zwei Zahlen kann der MSX unter folgende Nenner bringen:

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

Achten Sie auf die Notation, die von der Ihnen vielleicht noch aus der Mathematik bekannten Notation abweicht:

< > oder > <	wird normalerweise angegeben mit $\neq$
> = oder = >	wird normalerweise angegeben mit $\geqslant$
< = oder = <	wird normalerweise angegeben mit $\leqslant$

Wenn der Computer zwei Zahlen miteinander verglichen hat, können wir ihn anhand des Ergebnisses eine Aufgabe ausführen, einer Variablen einen Wert zuweisen oder in eine Programmzeile gehen lassen. Der Befehl hierzu lautet:

IF...THEN...	(wenn...dann...)

Die ersten Pünktchen von IF...THEN... stehen für das Vergleichen von zwei Werten. Wenn sich die Werte so verhalten, wie es das zwischen ihnen stehende Zeichen (aus der Liste oben) ausdrückt, das heißt, wenn die Behauptung wahr ist, dann (THEN) führt der Computer den Befehl oder die Befehle auf den Pünktchen hinter THEN aus.

Wir können auch sagen: IF die Behauptung wahr ist, THEN wird der Befehl ausgeführt.

In der Praxis zeigt sich, daß dies eine sehr sinnvolle Erleichterung ist, denn auf diese Weise kann der Computer doch bestimmte „Entscheidungen" treffen. Hier ein paar Beispiele:

```
IF X<10 THEN GOTO 300
```

Wenn der Wert von X kleiner ist als 10, oder anders: wenn die Behauptung X < 10wahr ist, geht der Computer in Zeile 300 (THEN GOTO 300). Ist die Behauptung unwahr (X ist nicht kleiner als 10), setzt der Computer das Programm in der nächstfolgenden Programmzeile fort, ohne den Befehl hinter THEN auszuführen. In ein Miniprogramm gebracht, sieht das so aus:

```
10 INPUT X
20 IF X<10 THEN GOTO 300
30 PRINT "X ist groesser als oder gleich 10"
40 END
300 PRINT "X ist kleiner als 10"
```

Beachten Sie, daß eine Variable nur dann als Wert in einem Vergleich dienen kann, wenn der Variablen bereits vorher im Programm ein Wert zugewiesen worden ist. Andernfalls betrachtet der MSX die Variable als 0, was unerwünschte Folgen haben kann.

IF A>B THEN GOTO 300

Wenn der Wert von A größer ist als der Wert von B (A > B ist wahr), geht der Computer in Zeile 300. Gilt das nicht (A > B ist unwahr), wird das Programm mit der nächsten Zeile fortgesetzt.

IF Y=0 THEN LET 7=6

Wenn der Wert von Y gleich 0 ist (Y=0 ist wahr), weist der Computer der Variablen Z den Wert 6 zu und setzt anschließend das Programm mit der nächsten Zeile fort. Ist die Behauptung unwahr, geht der Computer gleich zur nächsten Zeile über.

IF K< >L THEN PRINT ''K ist nicht gleich L''

Der Text zwischen Anführungszeichen wird nur dann gedruckt, wenn K nicht gleich L ist (K< >L ist wahr).

IF P>=Q THEN CLS

Der Schirm wird gelöscht, wenn die Behauptung P>=Q wahr ist. Andernfalls geht der Computer gleich zur nächsten Programmzeile über.

IF T<=S THEN BEEP

Es ertönt ein Bliep, wenn T kleiner oder gleich S ist. Nehmen Sie als Beispiel dieses Programm:

```
10 LET S=11
20 FOR T=20 TO 6 STEP -1
```

```
30 PRINT "T=";T,"S=";S
40 IF T<=S THEN BEEP
50 FOR X=1 TO 500 : NEXT X
60 NEXT T
70 END
```

ELSE

In dem Befehl IF ..1.. THEN ..2.. kann bei ..2.. jeder BASIC-Befehl (oder jede Befehlskombination) stehen. Wenn die Behauptung ..1.. wahr ist, führt der Computer den Befehl oder die Befehle aus. Ist die Behauptung nicht wahr, negiert der Computer die Befehle bei ..2.. und geht zur nächsten Programmzeile über. Neben diesem allgemeinen BASIC-Befehl kennt MSX-BASIC noch eine erweiterte Form:

IF...THEN...ELSE... (wenn...dann...sonst...)

Dieser Befehl bewirkt, daß der Computer nicht sofort zur nächsten Zeile übergeht, wenn die Behauptung hinter IF unwahr ist, sondern zunächst einmal den Befehl hinter ELSE ausführt. Etwa:

IF A>Z THEN GOTO 200 ELSE PRINT '' ''

Wenn die Behauptung A>Z wahr ist, geht der Computer in Zeile 200, sonst (das heißt A>Z ist nicht wahr) gibt der Computer ein Leerzeichen aus.

IF Q=10 THEN A=B ELSE A=C

Wenn die Behauptung Q=10 wahr ist, erhält A den Wert von B, sonst erhält A den Wert von C. Anschließend setzt der Computer das Programm in der folgenden Programmzeile fort. Zur Verdeutlichung nehmen Sie dieses Programm:

```
10 LET B=5
20 LET C=7
30 INPUT "Q:";Q
40 IF Q=10 THEN A=B ELSE A=C
50 PRINT A
60 GOTO 30
```

Folgendes ist zur Benutzung von IF...THEN...-Verweisen noch festzuhalten:
Wenn hinter THEN der Befehl GOTO Zeilennummer steht, kann GOTO fortgelassen werden; stattdessen kann man aber auch THEN unterschlagen.

IF X=Z THEN GOTO 200
IF X=Z THEN 200
IF X=Z GOTO 200
} Diese Zeilen sind gleichbedeutend

Wenn hinter einem ELSE der Befehl GOTO Zeilennummer folgt, kann GOTO fortgelassen werden.

Nach THEN oder ELSE dürfen einer oder mehrere Befehle folgen; die Befehle müssen durch Doppelpunkte (:) voneinander getrennt werden. Die Befehle hinter THEN werden entweder alle ausgeführt (Behauptung wahr) oder keiner von ihnen wird ausgeführt (Behauptung nicht wahr); das gleiche gilt umgekehrt auch für ELSE.

Logische Operatoren

Beim Programmieren in BASIC werden Sie AND, OR und NOT vor allem als logische Operatoren (Mittel) kennenlernen. Sie funktionieren folgendermaßen:

ccc AND ddd	ist nur wahr, wenn ccc wahr ist UND ddd auch wahr ist.
ccc OR ddd	ist nur wahr, wenn ccc wahr ist ODER ddd wahr ist ODER beide wahr sind.
NOT ccc	ist nur wahr, wenn ccc NICHT wahr ist.

Und hier ein paar Beispiele:

```
IF X>0 AND X<1 THEN ...
```

Wenn der Wert von X größer ist als 0 und zugleich kleiner als 1, dann ist die Gesamtbehauptung wahr, und der Computer wird den Befehl hinter THEN ausführen.

```
IF X>0 OR Y<1 THEN ...
```

Wenn der Wert von X größer ist als 0 oder der Wert von Y kleiner als 1 oder beides gilt, dann ist die Gesamtbehauptung wahr. Der Computer führt den Befehl hinter THEN aus.

```
IF 2+3=5 AND 4+7=11 ist wahr
IF 2+3=5 OR 4+3=11 ist ebenfalls wahr

IF NOT X=Y*A THEN...
```

Wenn der Wert von X nicht gleich Y*A ist, dann ist die Behauptung X=Y*A NICHT wahr. Der Befehl hinter THEN wird ausgeführt.

Bei jedem dieser IF...THEN...-Befehle ist die Verwendung von ELSE möglich.

Die logischen Operatoren können auch miteinander kombiniert werden:

```
IF NOT (X=1 AND Y=9) THEN ...
IF (X=1 AND Y=9) OR (X=-1 AND Y=-9) THEN ...
IF (X=1 OR X=9) AND (A=-1 OR A=-9) THEN ...
```

In der ersten Zeile ist die gesamte Behauptung dann wahr, wenn X nicht gleich 1 oder Y nicht gleich 9 ist.

In der zweiten Zeile ist die gesamte Behauptung dann wahr, wenn (X=1 und Y=9) oder (X=-1 und Y=-9) gilt.

In der dritten Zeile ist die gesamte Behauptung bei folgenden Kombinationen wahr:

X=1 und A=-1
X=1 und A=-9
X=9 und A=-1
X=9 und A=-9

Sie sehen, daß die logischen Operatoren zuweilen Klammern benötigen wie auch die mathematischen Operatoren. Finden sich in einem Vergleich sowohl logische als auch mathematische Operatoren, werden zunächst die mathematischen Operatoren bearbeitet. Sie können diese Reihenfolge durch die Verwendung von Klammern ändern.

Anwendung

Die Anwendung logischer Operatoren muß mit der nötigen Sorgfalt erfolgen. Beim Gebrauch dieser Logik kommt es schnell zu Fehlern, die nur sehr schwer zu erkennen sind, und infolgedessen ist es auch gar nicht so einfach, sie aus dem Programm zu eliminieren.

```
IF X>0 OR X<1 THEN...
```

Diese Behauptung ist sinnlos. Wenn der Wert von X größer ist als 0, oder der Wert von X kleiner ist als 1, oder beides stimmt, ist die Behauptung wahr. Diese Behauptung ist also immer wahr. Es gibt für X keinen Wert, der kleiner als 0 oder größer als 1 wäre.

```
IF P>0 AND P<0 THEN PRINT ''das Perpetuum mobile''
```

Es gibt keinen einzigen Wert für P, der sowohl größer als auch kleiner als 0 wäre. Diese Behauptung ist immer unwahr, und der Befehl hinter THEN wird niemals zur Ausführung gelangen.

Selbstverständlich kennt der MSX für derlei Fehler keine Fehlermeldung. Bei einer unwahren Behauptung wird der Befehl nach THEN einfach negiert, und der Computer geht zum Befehl hinter ELSE oder zur nächsten Programmzeile über. Der Computer kann nicht nachprüfen, ob Sie ihn mit unsinnigen Behauptungen arbeiten lassen.

Ein sehr einfaches Programm, in dem IF...THEN...vorkommen, ist das Zahlenrate-Programm.

```
10 REM rate die Zahl
20 SCREEN 1
30 LET X%=INT(RND(1)*100)
40 PRINT "Sie muessen eine Zahl"
50 PRINT "zwischen 1 und 100 erraten."
60 PRINT "Wenn Sie die von mir gewaehlte"
70 PRINT "Zahl erraten, haben"
80 PRINT "SIE GEWONNEN!"
90 PRINT
100 INPUT "Auf welche Zahl tippen Sie";G
110 IF G=X% THEN GOTO 200
120 IF G<X% THEN PRINT G;" ist zu niedrig"
130 IF G>X% THEN PRINT G;" ist zu hoch"
140 GOTO 100
200 FOR S=1 TO 15 : BEEP : NEXT S
210 COLOR 15,6,5
220 PRINT : PRINT "Ausgezeichnet, ";G;" hatte ich gewaehlt."
```

Neu in diesem Programm ist der Befehl

RND() (RaNDom = zufällig)

Dieser Befehl bewirkt, daß der Computer für uns eine zufällige Zahl heraussucht. Der Befehl kann auf unterschiedliche Arten gebraucht werden. Wenn Sie in den Klammern eine Zahl größer als 0 einsetzen, werden sogenannte pseudo-zufällige Zahlen kleiner als 1 und größer oder gleich 0 negiert. Die Zahlen scheinen alle zufällig zu sein, aber dem ist nicht so. Wenn Sie das Programm oben mehrmals laufen lassen, werden Sie feststellen, daß der Computer immer die gleichen Zahlen auswählt. Es ist dabei gleichgültig, welche Zahl größer als 0 Sie in die Klammern einsetzen. Jedesmal wird die gleiche Zahlenreihe durchlaufen.

Wenn Sie in die Klammern eine 0 einsetzen, gibt der RND-Befehl die letzte negierte zufällige Zahl noch einmal an. Dies kann bei der Überprüfung eines Programms nützlich sein oder wenn ein Programm es erforderlich macht, daß zweimal genau die gleiche zufällige Zahl genommen wird.

Wenn Sie in die Klammern eine negative Zahl einsetzen, geht der Computer anhand dieser negativen Zahl zu einer Position innerhalb der bereits erwähnten Reihe von pseudo-zufälligen Zahlen. Wird anschließend ein positiver Wert eingesetzt, kann man die Reihe von diesem Punkt an ablaufen lassen. Eine negative Zahl bewirkt nämlich jedesmal einen Sprung zum gleichen Punkt innerhalb der Reihe. Wenn Sie dreimal ?RND(-1) eingeben, erhalten Sie dreimal die Zahl .04389820420821.

Auf diese Weise erhalten Sie jedesmal, wenn Sie das Programm laufen lassen, die gleichen Zahlen. Es gibt jedoch eine Variable, die ganz und gar nicht zufällig ist, aber die jedesmal, wenn das Programm durchlaufen wird, einen anderen Wert hat. Das ist die Variable TIME, die ja von dem Augenblick an, da Sie MSX-BASIC einschalten, jede fünfzigstel Sekunde um 1 erhöht wird. Mit dieser Variablen als negativem Wert in den Klammern geht der Computer jedesmal, wenn Sie das Programm laufen lassen, zu einem anderen Punkt innerhalb der Reihe, von dem aus die Reihe dann durchlaufen werden kann. Dafür fügen Sie in das Programm oben die Zeile 25 ein:

```
25 R=RND (-TIME)
```

Selbstverständlich können Sie diesen Trick auch bei Ihren eigenen Programmen anwenden.

Dieses Programm ist allerdings ein bißchen lang für den Zweck, den es erfüllen soll. Nur eine Zahl erraten lassen, und dafür dann 22 Programmzeilen benötigen! Da kann man einiges sparen. Erst mal ist es ein wenig übertrieben, zunächst zu testen, ob G gleich X ist, anschließend ob G kleiner als X ist und dann noch, ob G größer ist als X. Wenn die ersten beiden Behauptungen nicht wahr sind, ist die dritte automatisch wahr. Deshalb kann Zeile 130 fortfallen, und Zeile 120 muß dann lauten:

```
120 IF G<X% THEN PRINT G; ''ist zu niedrig'' ELSE PRINT G; ''ist
zu hoch''
```

Um es noch kürzer zu machen, können sogar die Zeilen 120 und 110 zusammengezogen werden. Der Übersicht halber fassen wir die Mitteilungen „zu niedrig" und „zu hoch" ein bißchen kürzer.

```
110 IF G=X% THEN 200 ELSE IF G<X% THEN PRINT
''zu niedrig'' ELSE PRINT ''zu hoch''
```

Sie stellen fest, daß hier verschiedene IF...THEN...-Befehle hintereinander gesetzt werden. Weitere Kürzungen lassen sich in diesem Programm nur durch Zusammenstreichen der Erläuterungen und Mitteilungen auf dem Schirm erreichen.

Das Programm oben ist ganz einfach zu einem Spiel zu erweitern, das wie Mastermind aussieht.

Versuchen Sie doch selber einmal, ein Flußdiagramm und ein Programm zu schreiben, das:

- vier zufällige Zahlen auswählt;
- den Benutzer um vier Zahlen bittet;
- die erratenen Zahlen mit den zufällig gewählten Zahlen vergleicht;
- dem Benutzer Punkte für die Anzahl richtiger Zahlen an den richtigen Stellen gutschreibt;
- das Spiel dann beendet, wenn alle Zahlen erraten worden sind;
- dies alles mit ausreichenden Erläuterungen und Bildwiedergaben versieht.

Wenn Sie selber damit nicht zurecht kommen (aber versuchen Sie es zuerst!), finden Sie unten eine mögliche Lösung.

Sie können damit auch Ihr selbsterarbeitetes Programm vergleichen. Wenn Sie den gleichen Anforderungen mit einem anderen, gut funktionierenden Programm genügen, heißt das ganz gewiß nicht, daß Ihr Programm schlechter ist. Ein bestimmtes Problem läßt sich oft mit unterschiedlichen Programmen lösen.

```
10 CLS
20 SCREEN 1 : COLOR 15,4,4
30 A=RND(-TIME)
```

```
40 W%=INT(RND(1)*10-1)
50 X%=INT(RND(1)*10-1)
60 Y%=INT(RND(1)*10-1)
70 Z%=INT(RND(1)*10-1)
80 PRINT "ERRATE DIE 4 ZAHLEN."
90 PRINT
100 P=0
110 INPUT "Ihre Zahlen (4 X)";A,B,C,D
120 IF A=W% AND B=X% AND C=Y% AND D=Z% THEN GOTO 500
130 IF A=W% THEN P=P+1
140 IF B=X% THEN P=P+1
150 IF C=Y% THEN P=P+1
160 IF D=Z% THEN P=P+1
170 PRINT P;" gut"
180 GOTO 100
190 END
500 COLOR 15,6,7
510 PRINT : PRINT : PRINT"!!!!!! RICHTIG  !!!!!!"
520 END
```

10. STRINGS: DER COMPUTER ALS TEXTVERARBEITUNGSGERÄT

Was ist eigentlich ein String?

In diesem Kapitel werden wir die Möglichkeiten des Computers erweitern. Bis jetzt wurden nur Ziffern und Zahlen verarbeitet, und Text war lediglich dazu da, etwas auf dem Schirm wiederzugeben. Nun sehen wir uns einmal an, wie Text vom MSX bearbeitet und verarbeitet werden kann.

Der MSX bedient sich sogenannter STRINGS (=Reihen), um Text verarbeiten zu können. Ein String ist eine Aneinanderreihung von Zeichen. Jedes Zeichen ist dazu geeignet, und der String kann bis zu maximal 254 Zeichen lang werden.

Sie dürfen lediglich nicht vergessen, dem Computer mitzuteilen, daß Sie ihn mit Strings arbeiten lassen wollen. Der Computer soll dann ein Zeichen (beispielsweise a oder C oder %) nicht als Variable ansehen und auch nicht (etwa 1 oder 6 oder 34627.8) als Zahl. Aus diesem Grunde müssen Sie den gesamten String immer in Anführungszeichen ('') setzen und ihn mit einem Namen versehen, dem ein Dollarzeichen ($) folgt.

```
LET A$ = ''string''
```

Das heißt: betrachte die Zeichen s-t-r-i-n-g als einen String mit dem Namen A$.

Nun, ist das nicht bereits bekannt? Wo ist uns das schon einmal begegnet? Richtig, im Zusammenhang mit dem PRINT-Befehl. Dort erhält der String keinen Namen, und der Computer hat ihn lediglich darzustellen. Aber auch beim PRINT-Befehl darf der Computer die Zeichen nicht als Variablen oder Zahlen ansehen und sie auch nicht bearbeiten. Deshalb stehen die darzustellenden Zeichen immer zwischen Anführungszeichen.

Wir gehen nun noch einen Schritt weiter. Es ist möglich, den MSX eine Serie eingegebener Zeichen als String bearbeiten zu lassen. Dazu bedienen Sie sich des gleichen INPUT-Befehls wie schon bei der Eingabe von Zahlen. Sie fügen nur noch hinzu, daß es sich hier um einen String handelt und sorgen dafür, daß der String in einer Variablen gespeichert wird, die einen String beinhalten kann ($).

```
INPUT ''tippen Sie etwas'';A$
```

Wenn Sie etwas eingeben, nachdem die Mitteilung auf dem Schirm erschienen ist, und anschließend auf RETURN drücken, betrachtet der Computer den INPUT als einen String. Sie haben ja durch das Dollarzeichen hinter der Variablen A zu erkennen gegeben, daß die Eingabe als String gelesen werden soll.

Wir benutzen nun unser bisher erworbenes Wissen für ein ziemlich langes, aber einfaches Programm. Lesen Sie es sich zunächst einmal durch. Wenn Sie es nicht vollständig verstehen, tippen Sie es am besten erst ein und sehen sich dann an, was

es macht. Wenn Sie dagegen keinerlei Verständnisschwierigkeiten haben, können Sie die einzelnen Zeilen nach Ihren eigenen Wünschen verändern. Dieses Programm ist der Anfang einer Textverarbeitung auf dem MSX.

```
10 SCREEN 0
20 WIDTH 40 : KEY OFF
30 INPUT "Name des Geburtstagskindes: ";N$
40 INPUT "Strasse: ";A$
50 INPUT "Wohnort: ";P$
60 INPUT "Geburtstag: ";V$
70 INPUT "Telefonnummer: ";T$
80 CLS
90 PRINT : PRINT
100 PRINT N$
110 PRINT A$
120 PRINT P$
130 PRINT : PRINT
140 PRINT "Mein lieber ";N$;","
150 PRINT
160 PRINT "an diesem Festtag, am ";V$;","
170 PRINT "gratuliere ich Dir, ";N$;","
180 PRINT "herzlich zu Deinem Geburtstag."
190 PRINT "Ich hoffe, dass dies heute, dem ";V$;","
200 PRINT "ein schoener Tag fuer Dich wird."
210 PRINT "Obwohl ich viel zu tun habe, ver-"
220 PRINT "suche ich, in der ";A$;" in"
230 PRINT P$;" vorbeizuschauen."
240 PRINT "Sollte es nicht klappen, rufe ich ";T$
250 PRINT "an, um Dir noch kurz zu gratulieren."
260 PRINT "Bis dahin, ";N$;"!"
270 PRINT "Alles Gute, "
280 PRINT : PRINT
290 PRINT "        XXXXXXXXXXXXXXXXXXX"
300 GOTO 300
```

Setzen Sie anstelle der XX Ihren Namen ein, und Sie haben einen persönlichen Geburtstagsbrief an einen Ihrer Freunde. Wenn Sie über einen Drucker verfügen, können Sie das Schreiben sogar auf Papier bringen lassen und es so verschicken. Aber dies hier ist nicht nur ein persönlicher Brief an einen einzigen Freund oder eine einzige Freundin, sondern Sie können bei nochmaligem Durchlauf des Programms auch einen anderen Namen, eine andere Anschrift und einen anderen Geburtstag eingeben, und schon haben Sie einen persönlichen Geburtstagsbrief an alle Ihre Freunde.

Selbstverständlich bleibt es hier bei einem lustigen Spiel auf Ihrem Bildschirm; aber stellen Sie sich einmal vor, Sie verfügten über solch ein Programm mit einem ordentlichen und richtigen Text, außerdem hätten Sie einen ausgezeichneten Drucker, dessen Schriftbild nicht von dem einer Schreibmaschine zu unterscheiden wäre, sowie die Möglichkeit, die Namen und Adressen vom Computer einsetzen zu lassen. Sehen Sie, schon ist Ihnen klar, wie all die netten, freundlichen und höchst „persönlichen" Briefe von Bücherclubs, Börseninformationsdiensten und Werbeagenturen zustande kommen.

Zur richtigen Textverarbeitung gehört allerdings noch einiges mehr. Ein Textverarbeitungsprogramm muß allen eingegebenen Text bearbeiten, verarbeiten und speichern können. Zu den Möglichkeiten eines solchen Programmes gehören:

- Textkorrektur an einer beliebigen Stelle;
- alphabetisches Sortieren;
- Suchen bestimmter Wörter oder Zeichenkombinationen;
- Layout einer Seite;
- Zählen der eingegebenen Zeichen oder Wörter.

Zur Zeit gibt es nur ganz wenige Textverarbeitungsprogramme für den MSX. Es ist allerdings zu erwarten, daß es rasch mehr werden, und daß auch Programme auf Disketten entwickelt werden, wodurch sich die Schnelligkeit solcher Programme erheblich erhöht.

Stringvariablen

Bevor wir mit den Strings weiter fortfahren, erinnern Sie sich noch einmal eines Abschnitts über die Variablen. Im Kapitel „Variablen" haben Sie gelernt, mit Buchstaben anstelle von Ziffern zu rechnen.

Darüber hinaus haben Sie gelernt, daß eine Variable nicht nur eine Zahl beinhalten kann, sondern auch eine Buchstabenreihe. Solch eine Buchstabenreihe wird String genannt. Vorsicht! In manchen Übersetzungen englischer Bücher bezeichnet man das, was bei uns ein String ist, als Reihe (der englische Begriff). Eine Stringvariable ist in einigen Übersetzungen also eine Reihenvariable.

Damit dem Computer klar ist, wann ein bestimmter Buchstabe eine bestimmte Variable meint als Name für eine Zahl und wann eine Variable als Bezeichnung für einen String steht, setzten wir zur Unterscheidung ein Dollarzeichen hinter den Namen eines Strings.

Rechnen mit Strings und Zahlen

Der Computer sieht das, was Sie ihm als String eingeben, nicht als Variable oder zu verarbeitende Zahl an. Die maximale Länge eines Strings beträgt 254 Buchstaben (eine Programmzeile darf auch nicht mehr als 254 Zeichen umfassen). Die minimale Länge eines Strings ist null Zeichen. Es handelt sich dann um einen Nullstring oder einen leeren String. Sie geben das so an:

```
LET A$ = "" ""
```

Die Anführungszeichen werden nicht als zum String gehörig mitgezählt. Sie bezeichnen lediglich Anfang und Ende. Der MSX ordnet jeder Stringvariablen, die bislang ohne Inhalt geblieben ist, den Inhalt Null oder leer zu. Wenn Sie eingeben:

```
RESET          (die spezielle RESET-Taste drücken, nach
               BASIC)
PRINT Q$
```

Der Schirm bleibt leer und der Computer meldet „Ok". Der Inhalt von Q$ ist leer, auch dann, wenn Sie Q$ noch überhaupt nicht deklariert (mit einem Wert versehen) haben.

Da jeder Buchstabe in einem String vorkommen kann, ist es gut, darauf einzugehen, daß auch eine Zahl ein String sein kann. Nehmen Sie etwa das Zeichen 7. Sie können es auf zwei Arten benutzen:

```
als eine Zahl:      7
als einen String:   "7"
```

Sie können von der einen Art zur anderen wechseln. Wir nehmen einen String mit dem Namen A, und der Inhalt dieses String ist eben dieses Zeichen 7.

```
10 LET A$="7"
```

Die beiden Betrachtungsweisen der 7 können nicht so ohne weiteres zusammen gebraucht werden. Es wird nicht gelingen, dem MSX folgenden Befehl zu geben:

```
20 PRINT A$+8
```

Zunächst haben Sie dem Computer gesagt, daß 7 ein String sei (mit der Bezeichnung A), und der Computer hat gelernt, Strings nicht als Variablen oder Zahlen anzusehen. Also läßt er die Finger davon und macht eine Fehlermeldung:

```
Type mismatch in 20 (Art in Zeile 20 stimmt nicht)
```

Sie haben die Arten String und Zahl durcheinander gebracht. Und das geht nicht.

Es ist jedoch sehr wohl möglich, Strings in Zahlen umzuwandeln und umgekehrt. Zuallererst wird jedes Zeichen im Computer als eine Zahl gespeichert; diese Zahlen stammen aus dem ASCII-Set. Hier hat jedes Zeichen einen bestimmten Wert. Sie finden die Werte der Zeichen im Anhang dieses Buches. Dafür können Sie auch ein kurzes Programm erstellen. Der MSX kennt einen Befehl, mit dem der ASCII-Wert eines Zeichens angezeigt wird. Das ist

```
ASC(..)
```

In die Klammer setzen Sie ein Zeichen oder den Namen eines Strings (eine Stringvariable) ein. Ein Programm, das den Anwender immer ein Zeichen eingeben läßt, um anschließend den ASCII-Wert zu nennen, sieht so aus:

```
10 CLS
20 INPUT "Zeichen ";A$
30 PRINT "ASCII-Werte von ";A$;" = ";ASC(A$)
40 GOTO 20
```

Sie sehen, daß beim ASC-Befehl in Zeile 30 die Anführungszeichen fehlen. Das hängt damit zusammen, daß der Computer nicht den ASCII-Wert des Strings „A$" angeben soll, sondern den ASCII-Wert des Strings mit dem Namen A$. Beachten Sie auch, daß der ASC-Befehl nur den ASCII-Wert des ersten Schriftzeichens eines Strings angibt. Auch wenn Sie verschiedene Zeichen hintereinander eingeben, erhalten Sie lediglich den ASCII-Wert des ersten Zeichens.

Andersherum geht es natürlich auch. Sie geben den ASCII-Wert an, und der Computer sagt Ihnen, welches Zeichen dazu gehört.

CHR$(...) (CHaRacter=Zeichen)

Mit dem folgenden Programm können Sie sich schnell einen großen Teil des Zeichensatzes des MSX zeigen lassen.

```
5 SCREEN 1 : KEY OFF
10 FOR X=33 TO 255
20 PRINT CHR$(X);
30 NEXT X
```

Lassen Sie dieses Programm sowohl mit Screen 0 als auch mit Screen 1 laufen (dazu müssen Sie Zeile 5 verändern). Sehen Sie den Unterschied? Bei diesem kleinen Programm gibt es noch einige erwähnenswerte Dinge. Das Alphabet hat 26 Buchstaben. Für alle Buchstaben (Groß- und Kleinbuchstaben) hat der MSX also 52 Zeichen nötig. Daneben braucht er noch eine ganze Reihe von Werten für verschiedene Zeichen und für die Ziffern. Die ersten 32 Werte benötigt der Computer für allerlei Befehle an den Bildschirm und an die Peripheriegeräte. Um die in der Anlage aufgeführten Zeichen dennoch sehen zu können, müssen diese Werte mit einem doppelten Code aufgerufen werden: zunächst CHR$(1) und dann ein CHR$ mit dem Code, erhöht um 64. Das Programm für den gesamten Zeichensatz sieht so aus:

```
10 SCREEN 1 : KEY OFF
20 FOR X=1 TO 31
30 PRINT CHR$(1);CHR$(X+64);
40 NEXT X
50 FOR X=32 TO 255
60 PRINT CHR$(X);
70 NEXT X
```

Für unser Problem, die Addition von A$ (worunter sich „7" befindet) und 8, stellt dies alles jedoch keine Lösung dar. Es gibt noch eine Reihe von Befehlen, die es ermöglichen, Strings, in denen sich nur Zahlen befinden, in die Werte dieser Zahlen umzusetzen und umgekehrt, Zahlen in Strings mit diesen Zahlen als Inhalt. Um den Wert eines Strings zu erhalten, nehmen Sie:

VAL(..) (VALue = Wert)

Wenn wir das nun auf den String A$ („7") anwenden, geben wir ein:

```
LET X=VAL(A$)
```

Der Inhalt von A$ ist „7", also ergibt VAL(A$) den Wert 7. Die Variable X hat den Wert 7 erhalten.

Der VAL-Befehl funktioniert nur, wenn in der Klammer ein String (oder der Name eines Strings) steht, in dem Zahlen vorkommen. Das erste Zeichen muß ein Pluszeichen (+), ein Minuszeichen (−) oder eine Ziffer sein. Ist das nicht der Fall, wird der durch VAL angegebene Wert Null. Einige Beispiele dazu:

```
LET A$="88796"
LET X=VAL(A$)
PRINT X                          Ergebnis: −88796

LET X=VAL("5456")
PRINT X                          Ergebnis: 5456

PRINT VAL("Zahl")                Ergebnis: 0

PRINT VAL ("3 Zahlen)            Ergebnis: 3
PRINT VAL ("3*3=9)               Ergebnis: 3
```

Umgekehrt funktioniert es natürlich auch. Dazu brauchen Sie:

```
STR$(..)
```

Jeder Wert, der hier in Klammern gesetzt ist, wird zu einem String gemacht. Natürlich kann auch ein Variablenname in der Klammer stehen. In diesem Falle wird die Variable zum String. Etwa:

```
LET X=654
LET A$=STR$(X)
PRINT A$                         Ergebnis: 654
```

Beachten Sie, daß am Ergebnis zuweilen zu sehen ist, ob eine Zahl nun ein String oder eine Zahl ist. Einer Zahl geht schließlich immer eine Leerstelle oder ein Minuszeichen voraus. Der Wert oder der Variablenname in Klammern muß numerisch sein.

Die Addition, die wir aus A$ (Inhalt „7") und 8 vornehmen wollen, können wir so lösen:

```
LET A$="7"
PRINT VAL(A$)+8                  Ergebnis 15
```

Neben den hier erwähnten Möglichkeiten, Strings in Zahlen umzusetzen und um-

gekehrt, gibt es noch andere Stringfunktionen für die Umsetzung von Strings in hexadezimale, oktale oder binäre Zahlen. Um hierüber ein wenig mehr sagen zu können, ist zunächst ein kleiner Ausflug in die Zahlensysteme nötig.

Zahlensysteme

Der Mensch rechnet meist in einem zehnstelligen System. Das heißt, daß er von 0 bis 9 einschließlich zählt und eventuell wieder neu bei 0 anfängt, wobei er sich immer merkt, wie oft er bereits bis neun gezählt hat.

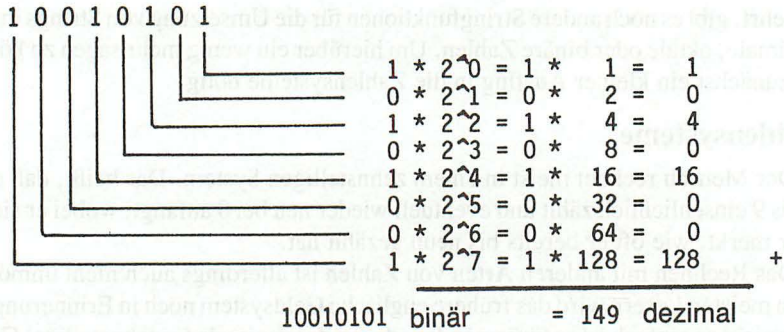
Das Rechnen mit anderen Arten von Zahlen ist allerdings auch nicht unmöglich. Den meisten Lesern wird das frühere englische Geldsystem noch in Erinnerung sein. Gar nicht so einfach ist es für jemanden, der an das dezimale (=zehnstellige) Geldsystem gewöhnt war; ein Engländer allerdings hatte mit seinem merkwürdigen System keine Probleme. Das englische oder amerikanische System für die Wiedergabe von Längenmaßen (yards, feet oder inches) ist gleichfalls nicht dezimal; aber im täglichen Umgang damit haben Engländer und Amerikaner keinerlei Schwierigkeiten.

Ein Computer arbeitet ebenfalls nicht dezimal. Er setzt sich aus zahlreichen kleinen Schalterchen zusammen, und jeder Schalter kann ein oder aus sein. Aus diesem Grunde kann ein Computer nur bis eins zählen: 0 und 1. Danach beginnt er wieder bei 0, wobei er sich merkt, wie oft er bereits bis 1 gezählt hat. Dieses System nennt man ein zweistelliges oder binäres Zahlensystem.

Zunächst scheint es sinnlos zu sein, sich mit der internen Zählweise des Computers eingehender zu beschäftigen, wenn das Ergebnis ja doch in der für den Menschen leicht zu verstehenden zehnstelligen Form auf dem Bildschirm erscheint. In dem Kapitel über Grafik und Farbe werden Sie allerdings schon bald erkennen, daß es sehr nützlich ist zu wissen, wie der Computer intern verschiedene Informationen speichert. Und wenn Sie später mal in Maschinensprache programmieren wollen, ist dieses Wissen sogar unumgänglich.

Ein MSX-Computer ist ein 8-Bit-Computer. Das heißt, daß die Speicherbits zu Gruppen von je 8 Bits zusammengefaßt sind. Solch eine Gruppe nennt man Byte. Innerhalb einer solchen Gruppe von 8 Bits kann bis zur Zahl 11111111 (binär) gezählt werden. Das entspricht 255 dezimal. Da jede binäre Zahl für eine Zählung von 0 bis 1 einschließlich steht, können binäre Zahlen mittels 2er-Potenzen in Dezimalzahlen umgewandelt werden.

Binär	Dezimal	Binär	Dezimal
00000000	0	00001011	11
00000001	1	00001100	12
00000010	2	00001101	13
00000011	3	00001110	14
00000100	4	00001111	15
00000101	5	00010000	16
00000110	6	00010001	17
00000111	7	00010010	18
00001000	8	00010011	19
00001001	9	00010100	20
00001010	10	00010101	21



Achtung! Das Zählen der 2er-Potenzen beginnt rechts. Die kleinste Potenz steht ganz rechts. Das gilt schließlich auch für Dezimalzahlen: Die kleinste Zahl steht ganz rechts (die Einer, links daneben die Zehner, noch weiter links dann die Hunderter, Tausender usw.).

Sie sehen, das Umwandeln von 2er-Potenzen funktioniert. Andersherum müssen Sie so große 2er-Potenzen wie möglich von der Zahl abziehen. Um 95 dezimal in eine binäre Zahl umzusetzen, gehen Sie wie folgt vor:

95	abziehen 2^7 ?	geht nicht	Bit 8 = 0
95	abziehen 2^6 ?	geht (Rest 31)	Bit 7 = 1
31	abziehen 2^5 ?	geht nicht	Bit 6 = 0
31	abziehen 2^4 ?	geht (Rest 15)	Bit 5 = 1
15	abziehen 2^3 ?	geht (Rest 7)	Bit 4 = 1
7	abziehen 2^2 ?	geht (Rest 3)	Bit 3 = 1
3	abziehen 2^1 ?	geht (Rest 1)	Bit 2 = 1
1	abziehen 2^0 ?	geht (Rest 0)	Bit 1 = 2

95 dezimal = 01011111 binär.

Achten Sie darauf, daß Bit 8 der ganz links stehende und Bit 1 der ganz rechts stehende ist.

Um dem Computer klar zu machen, daß eine bestimmte Zahl eine binäre Zahl und keine Dezimalzahl ist, geben Sie vor der Zahl ein &B ein.

Die Umwandlung einer binären Zahl (von acht Bits) in eine Dezimalzahl kann auch durch den Computer erfolgen:

```
PRINT &B00110100          Ergebnis: 52
```

Die Umwandlung einer Dezimalzahl in eine binäre Zahl geht mit Hilfe einer Stringfunktion vor sich:

```
BIN$(..)
```

In die Klammer setzen Sie eine Dezimalzahl ein, und der Computer zeigt Ihnen einen String, dessen Inhalt das binäre Äquivalent ist.

```
PRINT BIN$(237)          Ergebnis: 11101101
```

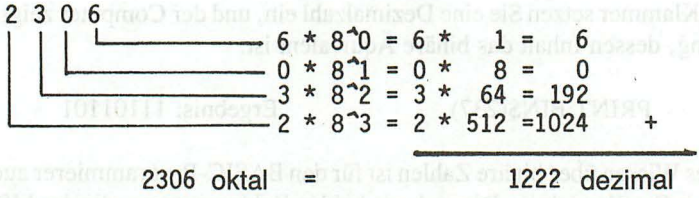
Einiges Wissen über binäre Zahlen ist für den BASIC-Programmierer auch für die Arbeit mit Grafik wichtig. Die anderen beiden Zahlensysteme, die der MSX kennt, sind vor allem für den Programmierer in Maschinensprache und die Erzielung spezieller Effekte von Bedeutung. Der Vollständigkeit halber gehen wir noch kurz auf das achtstellige und das sechzehnstellige System ein.

Das achtstellige oder Oktalsystem ist ein Zahlensystem, dessen Grundzahl 8 ist. Jede Ziffer ist eine Potenz von 8. Vergleichen Sie dieses System einmal mit dem zuvor behandelten binären System, bei dem jede Ziffer eine Potenz von 10 darstellt (1, 10, 100, 1000 usw.). Im Oktalsystem wird von 0 bis 7 einschließlich gezählt; anschließend geht es mit 0 wieder weiter, wobei gespeichert wird, wie oft bereits bis 7 gezählt wurde.

dezimal	oktal	dezimal	oktal
0	0	16	20
1	1	17	21
2	2	18	22
3	3	19	23
4	4	20	24
5	5	21	25
6	6	22	26
7	7	23	27
8	10	24	30
9	11	25	31
10	12	26	32
11	13	27	33
12	14	28	34
13	15	29	35
14	16	30	36
15	17	31	37
		32	40

Die Umwandlung von Zahlen aus dem einen Zahlensystem in ein anderes geht ebenso vor sich wie schon bei den binären Zahlen. Die Ziffer ganz rechts steht für 8^0=1, die Ziffer links davon steht für 8^1=8, die nächste Ziffer links für 8^2=64 usw.

Die Zahl 2306 oktal wird wie folgt in eine Dezimalzahl verwandelt:



Die Umwandlung einer Dezimalzahl in eine Oktalzahl sieht so aus:

967 (dezimal)	abziehen 8^4=4096 ?	geht nicht	0
967	abziehen 8^3= 512 ?	geht (Rest 455)	1
455	abziehen 8^2= 64 ?	geht 7X (Rest 7)	7
7	abziehen 8^1= 8 ?	geht nicht	0
7	abziehen 8^0= 1 ?	geht 7X (Rest 0=	7

967 dezimal = 1707 oktal

Auch für oktale Zahlen muß der Computer über ein besonderes Zeichen verfügen, um sie nicht mit Dezimalzahlen zu verwechseln. Dieses Zeichen ist: &0. Der Computer kann die Umwandlung auch für Sie erledigen. Vom Oktalsystem zum Dezimalsystem geht das so:

```
PRINT &076531
```

Ergebnis: 32089

Vom dezimalen System zu einem String, der eine oktale Zahl beinhaltet:

```
PRINT OCT$(5546)
```

Ergebnis: 126521

Als letztes Zahlensystem erwähnen wir hier noch das sechzehnstellige oder hexadezimale System. Dieses System hat die Grundzahl 16. Jede Ziffer steht für eine Potenz von 16. Beim Hexadezimalsystem gibt es ein zusätzliches Problem. Es gibt nur zehn normale Ziffern, während das hexadezimale System sechzehn benötigt. Das Problem wird gelöst, indem für die fehlenden Ziffern die Buchstaben A bis F einschließlich eingesetzt werden.

dezimal	hexadezimal	dezimal	hexadezimal
0	0	16	10
1	1	17	11
2	2	18	12
3	3	19	13
4	4	20	14
5	5	21	15
6	6	22	16
7	7	23	17

8	8	24	18
9	9	25	19
10	A	26	1A
11	B	27	1B
12	C	28	1C
13	D	29	1D
14	E	30	1E
15	F	31	1F
		32	20

Wollen Sie die Umwandlung von Dezimalzahlen in hexadezimale Zahlen und umgekehrt von Hand vornehmen, seien Sie auf die diesbezüglichen Erläuterungen bei den binären und oktalen Zahlen verwiesen. Mit den hexadezimalen Zahlen geht das ganz genauso; nur müssen Sie jetzt mit Potenzen von 16 arbeiten (1, 16, 256, 4096 usw.). Die Ziffern 10 bis 15 werden im Hexadezimalsystem mit den Buchstaben A bis F bezeichnet.

Der Computer benutzt das Symbol &H, um hexadezimale von dezimalen Zahlen zu unterscheiden.

Doch auch diese Umwandlung kann man dem Computer überlassen:

```
PRINT &H7FB4          Ergebnis: 32692
```

Für die Umwandlung einer Dezimalzahl in einen String mit einer Hexadezimalzahl als Inhalt geben Sie ein:

```
PRINT HEX$(6797)      Ergebnis: 1A8D
```

Rechnen mit Strings

Neben der Arbeit mit Zahlen und Strings und dem eventuellen Umwandeln vom einen in das andere ist es dem MSX auch möglich, nur Strings zu bearbeiten. Zu diesem Zweck kennt der MSX eine Reihe eigener Stringbefehle. Neben den Stringbefehlen, denen Sie in diesem Kapitel bereits begegnet sind, lauten diese:

```
STRING$(..,..)      (string = String)
SPACE$(..)          (space = Leerstelle)
LEFT$(..,..)        (left = links)
MID$(..,..,..)      (middle = Mitte)
RIGHT$(..,..)       (right = rechts)
LEN(..)             (lenght = Länge)
INSTR(..,..,..)     (in string = innerhalb des Strings)
```

Die beiden ersten Befehle sind vor allem da, um schöne Strings zu erzielen. Mit STRING$ können Sie ein bestimmtes Schriftzeichen verschiedene Male wiederholen:

```
PRINT STRING$(10,36)          Ergebnis: $$$$$$$$$$
```

Aufgrund dieses Befehls wird das Zeichen, das zum ASCII-Code 36 (die zweite Zahl hinter STRING$) gehört, 10 mal wiederholt. Das Ergebnis kann natürlich wiederum in einem String gespeichert werden.

Es ist gleichfalls möglich, das erste Zeichen eines angegebenen Strings einige Male wiederholen zu lassen:

```
LET A$=''MSX''
LET B$=STRING$(14,A$)
PRINT B$                      Ergebnis:
                              MMMMMMMMMMMMMM
```

Der zweite Befehl gibt die Anzahl der Leerstellen an. Die Anzahl der gewünschten Leerstellen wird in die Klammer gesetzt:

```
PRINT ''*'';SPACE$(8);''**''  Ergebnis: *        **
```

Der Befehl LEFT$(..,..) gibt die Zeichen wieder, die links im angegebenen String stehen. Etwa:

```
A$=''Buch''
PRINT LEFT$(A$,3)             Ergebnis: Buc
```

In die Klammer setzen Sie zunächst den Namen des Strings ein, um den es hier geht, dann folgt die Anzahl der Zeichen, die wiedergegeben werden sollen. In unserem Beispiel hat der Computer die drei Zeichen ganz links genommen.

Einen ebensolchen Befehl gibt es für die rechten Buchstaben:

```
A$=''Buch''
PRINT RIGHT (A$,2)            Ergebnis: ch
```

Um die Möglichkeiten komplett zu machen, verfügt MSX-BASIC auch über den Befehl MID. Damit kann man Zeichen aus dem String herausgreifen.

```
A$=''Buch''
PRINT MID$(A$2,2)             Ergebnis: uc
```

Bei MID$ setzen Sie zuerst den Namen des Strings ein, der untersucht werden soll, dann die Stelle des Zeichens, mit dem zu beginnen ist, und schließlich die Anzahl der

Zeichen, nach denen die Wiedergabe beendet werden soll. Die Stellen werden von links nach rechts gezählt. Wenn Sie keine zweite Zahl hinter MID$ eingeben, wird der Rest des Strings nach dem angegebenen ersten Zeichen wiedergegeben. Setzen Sie als letzte Zahl eine Zahl ein, die größer ist als die Zahl der verfügbaren Buchstaben, wird der Rest des Strings bis zum Ende wiedergegeben. Setzen Sie als letzte Zahl eine 0 ein, erhalten Sie einen leeren oder Nullstring. Setzen Sie als erste Zahl (die Zeichenposition, an der mit der Wiedergabe begonnen werden soll) eine Zahl ein, die größer ist als die Länge des Strings, erhalten Sie gleichfalls einen leeren String. Beispiele hierzu:

```
A$=''Buch, Stift''
PRINT MID$(A$,5)        Ergebnis: ,Stift
PRINT MID$(A$,5,99)     Ergebnis: ,Stift
PRINT MID$(A$,5,0)      Ergebnis:          (leerer String)
PRINT MID$(A$,10,3)     Ergebnis:          (leerer String)
```

Mit dem gleichen MID$-Befehl ist es möglich, Teile von Strings in andere Strings umzuwandeln. Die allgemeine Form dafür lautet:

```
MID$(..,..,..)=q$
```

Die Stringvariable und die Zahlen in der Klammer von MID$ geben an, welche Buchstaben von welchem String zu verändern sind. Die Buchstaben, die an die Stelle davor kommen, stammen aus dem String, der hinter dem Gleichheitszeichen steht:

```
A$=''Buch, Stift''
B$=''an der gleichen Stelle''
PRINT A$                    Ergebnis: Buch, Stift
PRINT MID$(A$,3,4)          Ergebnis: ch,S
MID$(A$,3,4)=B$
PRINT MID$(A$,3,4)          Ergebnis: an d
PRINT A$                    Ergebnis: Buan dtift
```

Es ist nicht nötig, den gesamten String hinter dem Gleichheitszeichen anzugeben. Auch dann kann MID$ gebraucht werden:

```
A$=''Buch, Stift''
B$=''an der gleichen Stelle''
MID$(A$,3,4)=MID$(B$,8,4)
PRINT A$                    Ergebnis: Bugleitift
```

Genauso kann anstelle von MID$ auch RIGHT$ hinter dem Gleichheitszeichen stehen:

```
A$=''Buch, Stift''
```

```
B$=''an der gleichen Stelle''
MID$(A$,6,4)=RIGHT$(B$,4)   Ergebnis: Buch, ellet
PRINT A$
```

Die Länge des ursprünglichen Strings (in den Beispielen immer A$) kann durch das Gleichheitszeichen nicht verändert werden.

Neben dem Angleichen von (Teilen von) Strings ist es auch möglich, die Aufgabe + auf die Strings anzuwenden. Dadurch werden die Strings einfach addiert. Das heißt, daß die Zeichen des einen Strings hinter die Zeichen des anderen Strings gestellt werden:

```
A$=''3''
B$=''7''
PRINT A$+B$                 Ergebnis: 37 (und nicht 10!)

A$=''abcdefg''
B$=''uvwxyz''
PRINT A$+B$                 Ergebnis: abcdefguvwxyz
PRINT B$+A$                 Ergebnis: uvwxyzabsdefg
```

Die Reihenfolge, in die die Zeichen der beiden Strings gebracht werden, ist abhängig von der Reihenfolge, in der die Strings miteinander addiert werden.

Das folgende Scherzprogramm bedient sich der Befehle LEFT$, RIGHT$, und LEN sowie der Addition von Strings um einen Namen in verballhornter Form auf dem Bildschirm zu bringen. Setzen Sie Ihren Namen ein, nachdem das Programm läuft, und schauen Sie sich an, was mit Ihrem Namen passiert.

```
10 SCREEN 1 : KEY OFF
20 INPUT "Ihren Namen bitte ";X$
30 LET NA$="******** "+X$+" ********"
40 COLOR 15,INT(RND(1)*15),INT(RND(1)*15)
50 FOR X=1 TO LEN(NA$)
60 PRINT LEFT$(NA$,X)
70 NEXT X
80 FOR X=1 TO LEN(NA$)
90 PRINT TAB(X) RIGHT$(NA$,LEN(NA$)-X)
100 NEXT X
110 FOR X=1 TO LEN(NA$)
120 PRINT TAB(LEN(NA$)-X) RIGHT$(NA$,X)
130 NEXT X
140 FOR X=1 TO LEN(NA$)
150 PRINT LEFT$(NA$,LEN(NA$)-X)
160 NEXT X
170 GOTO 40
```

Um die Länge eines Strings bestimmen zu können, nehmen Sie den Befehl LEN.

```
A$=''Buch, Stift''
PRINT LEN(A$)               Ergebnis: 10
```

Bei diesem Befehl müssen Sie lediglich den Namen des zu zählenden Strings eingeben.

Zum Schluß behandeln wir noch den INSTR-Befehl. Mit seiner Hilfe läßt sich feststellen, ob ein bestimmter String in einem anderen String vorkommt.

```
A$="Buch, Stift"
B$="h"
PRINT INSTR(A$,B$)          Ergebnis: 4
```

Hinter INSTR setzen Sie zunächst den Namen des zu untersuchenden Strings ein und dann den Buchstaben oder den Stringnamen, dessen Vorhandensein im anderen String kontrolliert werden soll. Zuweilen ist es sinnvoll, daß nicht gleich am Stringanfang mit der Suche begonnen wird, sondern ein Stückchen weiter hinten. Dafür können Sie noch eine dritte Angabe hinter INSTR machen:

```
A$="Buch, Stift, Druckbuchstabe"
B$="h"
PRINT INSTR(5,A$,B$)          Ergebnis: 20
```

Normalerweise gibt der INSTR-Befehl die Stelle an, an der der gesuchte String zum ersten Male auftaucht. Anschließend wird die Suche eingestellt. Aber die Möglichkeit, die Suche an einer anderen Stelle als am Anfang des Strings aufzunehmen sowie die Tatsache, daß INSTR den Wert 0 angibt, falls der gesuchte String überhaupt nicht im untersuchten Teil vorkommt, erlauben es uns, den gesamten String danach absuchen zu lassen, wie oft ein bestimmter String darin vorkommt.

```
10 SCREEN 1 : KEY OFF
20 INPUT "Gib einen String (max. 254) ein. ";A$
30 PRINT
40 INPUT "Buchstabe(n) zu suchen ";B$
50 X=0
60 T=0
70 PRINT
80 X=INSTR(X+1,A$,B$)
90 T=T+1
100 IF X<> 0 THEN GOTO 80
110 PRINT "Buchstabe(n) '";B$;"' kommt/kommen ";T-1;" mal vor."
```

In Zeile 20 ermöglicht es Ihnen dieses Programm, einen String mit einer Länge von maximal 255 Zeichen einzugeben. In Zeile 40 können Sie dann einen String nennen, nach dem gesucht werden soll. Der letztere String darf länger als nur ein Zeichen sein. Jedesmal, wenn die Programmschleife durchlaufen wird, rückt der Computer von der Stelle aus, an der er beim letzten Mal begonnen hat, um eine Position weiter. Beim ersten Mal beginnt er am Anfang. Immer wenn der genannte String gefunden wird, zählt der Computer zu T (Zähler) 1 hinzu. Dies macht er so lange, bis der String nicht mehr auftaucht. Dann ist X gleich 0, und der Computer verläßt die Schleife. Das Ergebnis erscheint in Zeile 110. Von T muß 1 abgezogen werden, da auch dann, wenn X gleich 0 ist (und also kein String mehr gefunden wurde), zu T 1 hinzugezählt wurde.

Vergleichen von Strings

Neben allen oben genannten Möglichkeiten können Strings auch miteinander verglichen werden. Hierzu nehmen Sie die gleichen Symbole wie in der Mathematik:

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

Bei der Benutzung zusammen mit Strings erhalten diese Vergleiche allerdings eine eigene Bedeutung. Zwei Strings sind nur dann gleich (=), wenn sie gleich lang sind und genau die gleichen Zeichen enthalten. In allen anderen Fällen sind zwei Strings nicht gleich (<>). Strings werden Zeichen für Zeichen miteinander verglichen. Ein Zeichen ist größer als das Zeichen, mit dem es verglichen wird, wenn der ASCII des ersten Zeichens höher ist. Ein String ist also größer als ein anderer String, wenn das erste Zeichen größer ist, das heißt, wenn es später im Alphabet oder der ASCII-Codeliste vorkommt (B>A, Z>A, F>B) und der String mindestens ebenso lang ist wie der String, mit dem verglichen wird. Sind die ersten Zeichen gleich, werden die zweiten Zeichen kontrolliert. Sind auch sie gleich, geht der Computer zu den dritten Zeichen über usw; genau wie auch im Wörterbuch die Wörter angeordnet sind. Sind alle Zeichen gleich, wird nachgeschaut, ob ein bestimmter String länger ist. Der längste ist dann zugleich der größte (ABDC>ABC, XYZ>XY).

Eine besondere Schwierigkeit stellt der Unterschied zwischen Groß- und Kleinbuchstaben dar. Obgleich zu erwarten wäre, daß Kleinbuchstaben kleiner sind als Großbuchstaben, ist dem nicht so. A<a, AB<ab. Sie können dies den ASCII-Werten entnehmen. Die ASCII-Werte der Kleinbuchstaben sind größer als die der Großbuchstaben.

Die übrigen Zeichen zum Vergleichen sprechen für sich.

Ein hübsches Programm, das Strings manipuliert und miteinander vergleicht, ist der sogenannte Palindromtester. Ein Palindrom ist ein Wort, das von links nach rechts gelesen ebenso lautet wie von rechts nach links, etwa „rar" oder der Name „Anna".

Dieses Programm läßt Sie ein Wort eingeben, dreht es um und kontrolliert, ob das Ergebnis genauso aussieht.

```
10 SCREEN 0 : KEY OFF
20 INPUT "Bitte ein Wort. ";A$
30 Z$=""
40 FOR X=LEN(A$) TO 1 STEP -1
50 Z$=Z$+MID$(A$,X,1)
60 NEXT X
70 PRINT "umgekehrt erhalten Sie: ";Z$
80 PRINT
90 IF Z$=A$ THEN PRINT "Das ist ein Palindrom!"
100 PRINT
110 GOTO 20
```

Noch besser sind die Palindrome, die aus mehreren Worten bestehen, beispielsweise „Ein Neger mit Gazelle zagt im Regen nie". Falls eine solche Wortfolge kontrolliert werden soll, stellen die Leerstellen ein besonderes Problem dar. Im folgenden Programm werden zwei zusätzliche Strings eingeschoben. In den ersten String gehören die umgekehrten Worte ohne Leerstellen. Im zweiten steht das Ergebnis der Umkehrung. Sind beide gleich, kann man von einem Palindrom sprechen.

```
10 SCREEN 0 : KEY OFF
20 INPUT "Bitte ein Wort. ";A$
30 Z$=""
40 G$=""
50 H$=""
60 FOR X=LEN(A$) TO 1 STEP -1
70 Z$=Z$+MID$(A$,X,1)
80 IF MID$(A$,X,1)<>" " THEN G$=G$+MID$(A$,X,1)
90 NEXT X
100 FOR P=LEN(G$) TO 1 STEP -1
110 H$=H$+MID$(G$,P,1)
120 NEXT P
130 PRINT "umgekehrt erhalten Sie: ";Z$
140 PRINT
150 IF G$=H$ THEN PRINT "Das ist ein Palindrom!"
160 PRINT
170 GOTO 20
```

Ein schönes Beispiel ist: a man a plan a canal panama. Das Problem bei diesem Programm bleibt, daß der Computer nicht wissen kann, wo im Deutschen Worte getrennt werden müssen (auch mit der englischen Sprache hat der Computer diesbezüglich seine Schwierigkeiten). Wenn die falsch plazierten Leerstellen Sie sehr stören, können Sie in Zeile 140 anstelle von Z$ auch G$ darstellen lassen. Sie erhalten dann das umgedrehte Wort ohne Leerstellen.

Sie können das Programm auch soweit verbessern, daß Satzreihen gleichfalls unberücksichtigt bleiben. Hierbei gibt es eine besondere Komplikation: wenn Sie bei der Stringeingabe ein Komma setzen, nimmt der Computer an, Sie wollten damit sagen, daß der erste String abgeschlossen ist und der zweite beginnt. Geben Sie im Programm oben mal ein: rar, rar.

Der Computer meldet:

```
?Extra ignored                    (zuviel nicht beachtet)
```

Nur das erste Wort wird umgekehrt. Wegen des Kommas hat der Computer angenommen, daß Ihre Eingabe aus zwei verschiedenen Worten bestand, während der INPUT-Befehl nur ein einziges verlangte (A$).

Dies können Sie auf folgende Art vermeiden:

LINE INPUT

Durch diesen Befehl wird eine ganze Zeile mit Schriftzeichen eingegeben. Der Computer achtet dabei nicht auf die Kommata. Das Drücken von RETURN bezeich-

net das Ende der Stringeingabe. Verändern Sie im Programm oben in Zeile 20 den INPUT-Befehl in LINE INPUT, und versuchen Sie es mit „rar, rar" noch einmal. Ändern Sie Zeile 90 mit Hilfe des AND-Operators so, daß auch Satzzeichen herausgefiltert werden. Wenn Sie das Programm so korrigiert haben, können Sie Palindrome aus wesentlich längeren Wortreihen eingeben.

Speicherraum bei Strings

Alle Strings werden in einem besonderen Speicherabschnitt (string area) abgelegt. Dieser Abschnitt ist beim Einschalten des Computers umfangmäßig begrenzt. Solange Sie als Programmierer dagegen nichts unternehmen, ändert sich daran auch nichts. Versuchen Sie, im Palindromprogramm einmal ein ganz langes Wort einzugeben (tippen Sie zwei Zeilen mit Satzzeichen). Sie erhalten dann folgende Fehlermeldung:

Out of string space (zuwenig Stringplatz)

Im Stringabschnitt können maximal 200 Schriftzeichen untergebracht werden. Da das letzte Palindromprogramm vier Strings einrichtet, können in jedem String nur etwa 50 Schriftzeichen untergebracht werden.

Um den für die Strings zur Verfügung stehenden Speicherplatz zu vergrößern, benutzen Sie:

CLEAR .. (freimachen)

Hinter CLEAR setzen Sie die Anzahl der Schriftzeichen ein, die der Stringabschnitt umfassen muß. Seien Sie hier nicht zu großzügig, denn allen Speicherplatz, den Sie für Strings gebrauchen, können Sie für nichts anderes mehr nehmen.

Lassen Sie auch nicht außer Acht, daß jeder String maximal 254 Schriftzeichen lang werden kann. Gibt es im Programm vier Strings, ist es überflüssig, mehr als 1016 Speicherstellen zu reservieren (CLEAR 1016).

Verwechseln Sie CLEAR nicht mit dem CLS-Befehl (CLEAR SCREEN). Der letztere Befehl macht nur den Schirm frei (durch Löschen des Schirmspeichers.)

11. LISTEN MIT DATEN

Einführung

Die bisher in diesem Buch behandelten Programme arbeiteten meist mit nur wenigen Angaben. Zuweilen aber muß der Computer selber Angaben machen oder bestimmte Daten variieren.

Neben ihrer Fähigkeit zu rechnen sind die Computer vor allem dafür bekannt, daß sie große Datenmengen in ziemlich kurzer Zeit verarbeiten können. Dieses Kapitel zeigt, wie solche Daten eingegeben werden können, und wie der Computer diese Daten speichern kann.

Bislang wird hierzu der INPUT- oder LINE INPUT-Befehl benutzt. Dieser Befehl sorgt dafür, daß der Computer im Verlaufe des Programms Angaben erbittet. Diese Daten werden einer Variablen zugeordnet und damit macht sich der Computer dann an die Arbeit. Eine andere Möglichkeit ist der LET-Befehl. Dieser Befehl bewirkt, daß einer Variablen ein Wert zugeordnet wird. Beide Methoden haben den Nachteil, daß es sich hier immer nur um begrenzte Datenmengen handeln kann. Hinzu kommt noch die Tatsache, daß die Daten, wie sie bisher über den INPUT-Befehl eingegeben werden, lediglich einen Programmlauf lang erhalten bleiben. Die einfachste Art, eine Reihe von Daten einzugeben, gibt es über eine DATA-(Angaben)Liste.

DATA-Listen

Mit dem BASIC-Befehl

DATA

können Sie dem Computer eine Datenliste übermitteln. Diese Daten werden voneinander durch Kommata getrennt. Sowohl Zahlen als auch Strings sind möglich. Im DATA-Befehl müssen die Strings nicht mit Hilfe von Anführungszeichen geöffnet und geschlossen werden. Wenn jedoch Kommata, Doppelpunkte oder Leerstellen im String auftauchen, müssen diese in Anführungszeichen stehen. Strings mit und Strings ohne Anführungszeichen können in beliebiger Weise nebeneinander verwandt werden.

Es ist notwendig, daß der zu einem String gehörige Lesebefehl (siehe unten) dem Computer verständlich macht, daß es sich hier tatsächlich um einen String handelt. Eine Liste kann mehr als nur eine Programmzeile umfassen. In diesem Falle muß zu Beginn einer jeden Zeile der DATA-Befehl wiederholt werden. Ungeachtet ihrer Verteilung über mehrere Zeilen erkennt der Computer die Liste als ein Ganzes.

Der Computer liest die Daten der Liste mittels des Befehles:

READ

Die Daten werden in der Reihenfolge gelesen, in der sie in der Datenliste von links nach rechts und von niedriger zu hoher Zeilennummer aufgeführt sind. Hinter READ wird der Name der Variablen angegeben, der die Angabe zugeordnet werden muß. Beispielsweise:

```
10 READ A
10 PRINT A
30 DATA 318, 645, 75
```

oder

```
10 READ A$
20 PRINT A
30 DATA Beispielkette, Text, Text
```

Anhand dieses Beispiels wird deutlich, daß die Befehle DATA und READ nicht beieinanderstehen müssen. Es ist üblich, die Datenliste an das Programmende zu setzen. So wird die Lesbarkeit des Programms erhöht.

So wie oben angewandt, nützt diese Vorgehensweise nicht so viel. Aber es ist möglich, verschiedenen Variablen Werte mittels der Datenliste zuzuweisen:

```
10 READ A, B, C,
20 PRINT A, A*B, C
30 DATA 10, 6, 80, 90, 120
```

Der Computer behält von alleine, wo er sich gerade in der Datenliste befand. Auf diese Weise genügt eine Datenliste, auf die an verschiedenen Stellen im Programm verwiesen wird:

```
10 READ A, B, C
20 PRINT A, A *B, C
:
:
70 READ A, E
80 PRINT A*E, D, E*D
:
:
200 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
```

In diesem Falle sind alle Angaben in eine Programmzeile gebracht. Programme, in denen verschiedene Male auf eine lange Datenliste verwiesen wird, sind besser lesbar, wenn Sie zu jeder zusammengehörigen Datengruppe eine eigene Datenzeile setzen. Dies erleichtert das Wiederauffinden.

Da der Computer so freundlich ist, die fortlaufende Zählung in der Datenliste zu übernehmen, wird das Programmieren für uns einfacher. Nehmen Sie einmal an, Sie

wollten eine einfache Liste mit Telefonnummern anlegen. Wenn ein Name eingegeben wird, soll der Computer die zugehörige Telefonnummer heraussuchen. In dem folgenden Programm sind alle Daten Strings.

```
10 SCREEN 0 : KEY OFF
20 INPUT "Name ";NA$
30 PRINT
40 READ DA$,TE$
50 IF NA$<>DA$ THEN GOTO 40
60 PRINT TE$
100 DATA Jan,434423,Wilhelm,0453-8876,Hans,765453,Marie,078-987876,
200 DATA Rosi,020-565354,Dieter,766545,Barbara,0931-766754,Renate,798548
```

In diesem Programm liest der Computer jedesmal einen Namen aus der Liste (DA$) und eine dazu gehörende Telefonnummer (TE$). Er macht das so lange, bis der eingegebene String übereinstimmt mit einem in der Liste vorkommenden Namen (IF NA$ < > DA$ THEN GOTO 40). Dann verläßt er die Schleife und zeigt die Telefonnummer an. Der Vorteil eines solchen Programms ist der, daß Sie eine fast endlose Liste mit Namen und zugehörigen Telefonnummern eingeben können, um anschließend den Computers zu einem bestimmten Namen die entsprechende Rufnummer suchen zu lassen. Achten Sie bei diesem Programm darauf, daß ein Unterschied zwischen Groß- und Kleinbuchstaben besteht. Für den MSX ist Jan ein anderer als JAN oder jan.

Es ist auch möglich, eine Liste mit Daten verschiedene Male zu benutzen. Hierfür gibt es den Befehl:

RESTORE (stelle wieder her)

Dieser Befehl wirkt auf die Stelle, an der im Speicher des MSX bei einer Angabe aus der Liste ein kleiner Pfeil steht. Dieser Pfeil, ein sogenannter Datenzeiger, sorgt dafür, daß der Computer weiß, wo er sich in der Liste befunden hat. RESTORE ohne jeden weiteren Zusatz bewirkt, daß der Datenzeiger zur ersten Angabe hinter dem DATA-Befehl in einem Programm springt.

```
10 READ A, B, C
20 PRINT A, B, C
:
:
40 RESTORE
50 READ D, E, F,
60 PRINT D, E, F
:
:
100 DATA 1, 2, 3, 4, 5, 6
```

Die Werte von A, B und C sollen 1, 2 und 3 sein.

Die Werte von D, E und F sind ebenfalls 1, 2 und 3. Der Datenzeiger ist aufgrund des Befehles in Zeile 40 zur ersten Angabe zurückgegangen.

Hinter RESTORE kann gleichfalls eine Zeilennummer eingegeben werden. In diesem Falle geht der Datenzeiger zur ersten Angabe hinter dem DATA-Befehl in der genannten Zeile zurück.

RESTORE dient nicht nur dazu, den Computer zu Daten zurückgehen zu lassen, mit denen er bereits gearbeitet hat. Mittels der Eingabe einer Zeilennummer ist es auch möglich, eine Reihe von Daten (auch vorübergehend) zu überspringen.

Mit RESTORE können Sie darüber hinaus unter verschiedenen Datenlisten wählen. Mit mehreren Datenlisten lassen sich etwa Ihre Fremdsprachenkenntnisse auffrischen. Der Benutzer kann nun wählen, welche Datenliste er gerne hätte.

```
10 SCREEN 1 : KEY OFF
20 PRINT "     ****************     "
30 PRINT "        SPRACHTEST        "
40 PRINT "         ********         "
50 PRINT : PRINT "Sie koennen waehlen zwischen: ":PRINT
60 PRINT "1- NIEDERL. >> DEUTSCH"
70 PRINT "2- DEUTSCH  >> NIEDERL."
80 PRINT "3- NIEDERL. >> FRANZ."
90 PRINT "4- FRANZ.   >> NIEDERL."
100 PRINT "5- NIEDERL. >> ENGL."
110 PRINT "6- ENGL.    >> NIEDERL."
120 PRINT "7- NIEDERL. >> SCHWED."
130 PRINT "8- SCHWED.  >> NIEDERL."
140 PRINT
150 INPUT "Ihre Wahl";K
160 IF K=1 OR K=2 THEN RESTORE 320
170 IF K=3 OR K=4 THEN RESTORE 330
180 IF K=5 OR K=6 THEN RESTORE 340
190 IF K=7 OR K=8 THEN RESTORE 350
200 CLS : T=0
210 FOR X=1 TO 4
220 READ N$,V$
230 IF INT(K/2)=K/2 THEN 270 ELSE PRINT:PRINT N$
240 INPUT "Ihre Uebersetzung: ";VA$
250 IF A$=V$ THEN T=T+1 : PRINT "AUSGEZEICHNET!" ELSE PRINT "falsch"
260 NEXT X : GOTO 310
270 PRINT:PRINT V$
280 INPUT "Ihre Uebersetzung: ";A$
290 IF A$=N$ THEN T=T+1 : PRINT "AUSGEZEICHNET!" ELSE PRINT "falsch"
300 NEXT X
305 PRINT
310 PRINT T ; " richtige Antworten!"
320 DATA hebben,haben,nieuw,neu,schaduw,schatten,arend,adler
330 DATA brand,feu,huis,maison,gebit,denture,tafel,table
340 DATA hoog,high,boek,book,huis,house,brief,letter
350 DATA vader,far,gevaar,fara,auto,bil,diep,djup
```

Nun ist dies hier ein Computerbuch und kein Sprachkursus; die Datenliste ist infolgedessen sehr beschränkt. Sie können sie allerdings so umfangreich gestalten, wie Sie wollen. Wenn Sie wie im Beispielprogramm mehr als nur eine einzige Sprache in die Datenliste aufnehmen möchten, können Sie sie noch weiter verkürzen, indem

Sie jedesmal ein deutsches Wort eingeben und direkt dahinter die entsprechenden Begriffe in den anderen von Ihnen gewünschten Sprachen. Versuchen Sie einmal herauszufinden, wie Sie dann das richtige Wort aus der Liste holen und anschließend wieder für das Lesen eines deutschen Wortes sorgen können.

Arrays

Mit den Befehlen READ und DATA ist es möglich, dem Computer lange Listen mit Daten oder Werten einzugeben und diese einer Reihe von Variablen bzw. jedesmal der gleichen Variablen zuzuweisen. Zuweilen müssen Sie eine Reihe von Daten gleichzeitig verarbeiten und sie nicht nur – wie in unserem Beispielprogramm – kurz durchlaufen, um die richtigen Angaben herauszusuchen. Für solch eine lange Liste mit Daten können wir einfache Variablen wählen, denen wir die Daten zuweisen. Aber wenn es sich um sehr viele Daten handelt, wird es umständlich, die Variablen auseinanderzuhalten und die Bedeutung jeder einzelnen im Kopf zu behalten.

Der MSX hat daher die Fähigkeit, Listen mit numerierten Variablen anzulegen. Diese Listen nennt man Arrays (Felder). Variablen, die sich in einem Array befinden, sind einfach zu handhaben, und man kann sehr leicht auf sie verweisen. Einzige Vorbedingung für die Arbeit mit einem Array ist, daß Sie dem Computer vor Beginn mitteilen, wieviel Speicherplatz für die Liste freigehalten werden soll. Dazu dient der Befehl:

DIM (DIMension = Größe)

Eindimensionale Arrays

Gesetzt den Fall, Sie wollen ein Feld mit 40 Angaben erstellen. Sie stehen dann vor den folgenden Problemen:

- Wie kann diese Liste von einer anderen, gleichartigen Liste unterschieden werden?
- Wie lassen sich Daten in der Liste untereinander auseinanderhalten?
- Wie verweist man auf eine bestimmte Angabe in der Liste?
- Wieviel Speicherplatz muß freigehalten werden?
- Wie werden die Daten in die Liste gebracht?

Die erste Frage ist am einfachsten zu beantworten. Jede Liste erhält einen eigenen Namen. Als ersten Namen wählen wir beispielsweise L (von Liste).

Die beste Art, eine bestimmte Angabe in der Liste zu bezeichnen, ist die Numerierung der Daten. Es kann dann auf die erste, die zweite, die dritte, die zwölfte oder die xte Angabe in Liste L verwiesen werden.

Die entsprechende Notation sieht so aus:

L(1) – erste Angabe in Liste L

L(2) – zweite Angabe in Liste L
L(X) – xte Angabe in Liste L (falls X nicht bereits zuvor im Programm einen Wert erhalten hat, ist X gleich 0).

Auf diese Art sind alle Daten in jeder Liste genau zu bezeichnen.

Der DIM-Befehl dient zur Reservierung von ausreichendem Speicherplatz. Sie müssen nicht selber ausrechnen, wieviel Platz nötig ist. Sie geben hinter DIM einfach in Anführungszeichen die Länge der Liste an. Etwa:

```
DIM L(40)
```

bei einer Liste mit 40 Angaben. Da der Computer bei 0 anfängt zu zählen, haben Sie mit DIM L(40) genaugenommen Platz für 41 Angaben geschaffen. Solange Sie über ausreichend Speicherplatz verfügen, können Sie das ruhig so belassen (wieder aus Gründen der Lesbarkeit).

Um die Liste mit Daten zu füllen, nehmen Sie den LET-Befehl:

```
10 DIM L(40)
20 LET L(1)=2
30 LET L(2)=4
40 LET L(3)=5
:
:
410 LET L(40)=650
```

Aber das kostet natürlich viel Eingabezeit. Genau so einfach ist es, auf eine bestimmte Angabe in der Liste zu verweisen. Sie können sich doch bestimmt noch an die Datenliste im vorigen Abschnitt erinnern. Viel besser ist also:

```
10 DIM L(40)
20 FOR X=1 TO 40
30 READ L(X)
40 NEXT X
50 DATA 2, 4, 5,..,..,..,.....,650
```

So sparen Sie einige hundert Programmzeilen.

Zweidimensionale Arrays

Zuweilen ist es notwendig, eine ganze Seite mit Zahlen zu verarbeiten. Die Zahlen stehen dann nicht nur horizontal in Beziehung zueinander, sondern auch vertikal. Die Zahlen stehen in Reihen und Spalten, die miteinander etwas zu tun haben. Für eine normale Liste mit Daten, die hintereinander stehen, nimmt man einen sogenannten

eindimensionalen Array. Die erste Dimension wird als Linie dargestellt. Einen solchen eindimensionalen Array nennt man auch Vektor.

Für eine ganze Seite mit Zahlen brauchen wir einen zweidimensionalen Array. Die zweite Dimension wird als ebene Fläche dargestellt. Eine andere Bezeichnung für solch einen Array ist „Tabelle". In einem zweidimensionalen Array sind die Daten in Reihen und Kolonnen geordnet.

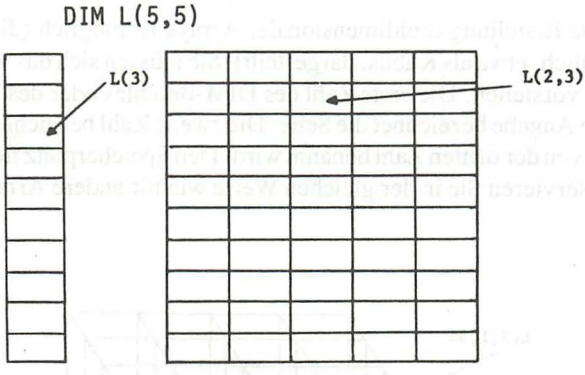
Ein zweidimensionaler Array mit 25 Daten kann beispielsweise so aussehen:

2	4	6	8	10
12	14	16	18	20
22	24	1	3	5
7	9	11	13	15
17	19	21	23	25

Die Verweisung auf eine bestimmte Angabe ist in diesem Array etwas umständlicher. Sie nennen jetzt nicht mehr einfach die soundsovielste Angabe, sondern Sie verweisen auf eine bestimmte Zeile und eine bestimmte Spalte:

L(3, 2) ist die zweite Angabe in der dritten Zeile (24)
L(2, 4) ist die vierte Angabe in der zweiten Zeile (18)
L(Y, Y) ist die yste Angabe in der xten Zeile, oder Zeile X, Spalte Y

Die Reservierung ausreichenden Speicherplatzes für einen zweidimensionalen Array erfolgt mit der Eingabe von zwei Zahlen:



Ein- und zweidimensionale Arrays

Der Array muß nicht unbedingt rechteckig sein. Wenn Sie beispielsweise 100 Daten in einem Array aus 4 Spalten mit je 25 Daten oder aus 25 Reihen mit je 4 Daten unterbringen wollen, erstellen Sie das folgende Programm:

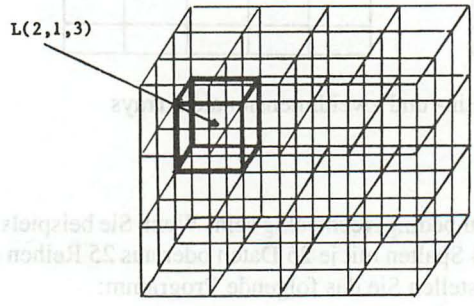
```
10 DIM K(25,4)
20 FOR Y=1 TO 4
30 FOR X=1 TO 25
40 READ K(X,Y)
50 NEXT X : NEXT Y
60 DATA ga1, ga2, ga3, ga4, .., .., ....., ga25
70 DATA gb1, gb2, gb3, gb4, .., .., ....., gb25
80 DATA gc1, ..........................., gc25
90 DATA ..............................., gd25
```

Beachten Sie hierbei folgende Punkte:

- Sie können selber wählen, ob Sie die Daten per Spalte oder per Zeile lesen lassen wollen.
- Achten Sie bei der Eingabe der Datenliste gut darauf, in welcher Reihenfolge eingelesen wird.
- Sie erhöhen die Lesbarkeit und verringern die Fehlermöglichkeiten, wenn Sie die Daten in der Datenliste per Datenzeile logisch anordnen. In unserem Beispiel wurden alle Daten einer Spalte in einer Datenzeile nebeneinander gesetzt. Natürlich können die Daten auch reihenweise eingegeben werden.
- Eine Unterteilung der Datenliste in verschiedene Zeilen ist gleichfalls sinnvoll, wenn bestimmte Teile aus der Liste verschiedene Male gebraucht werden müssen. Der RESTORE-Befehl kann nur auf eine bestimmte Zeile verweisen.

Dreidimensionale Arrays

Auch die Erstellung dreidimensionaler Arrays ist möglich (die dritte Dimension wird räumlich, etwa als Kubus, dargestellt). Sie müssen sich das wie ein ganzes Buch mit Daten vorstellen. Die erste Zahl des DIM-Befehles oder des Verweises auf eine bestimmte Angabe bezeichnet die Seite. Die zweite Zahl bezeichnet eine Zeile, in der die Spalte von der dritten Zahl benannt wird. Den Speicherplatz für dreidimensionale Arrays reservieren Sie in der gleichen Weise wie für andere Arrays auch: DIM (W, X, Y).



Der MSX kann noch mit wesentlich mehr Dimensionen bei der Erstellung von Arrays arbeiten (bis zu 255 Dimensionen). Räumlich ist das allerdings nur äußerst schwer darstellbar.

Problematisch hierbei, ist die Tatsache, daß Arrays unglaublich viel Speicherplatz benötigen. Überlegen Sie genau, ob ein Array überhaupt notwendig ist, und wenn das der Fall ist, machen Sie ihn keinesfalls umfangreicher als unbedingt nötig.

Stringarrays

Bisher haben Sie in diesem Kapitel immer von „Daten" gelesen, während es sich in den Beispielen nur um Zahlen handelte. Der Computer akzeptiert auch Strings als Daten.

Es ist möglich, einen Array zu erstellen, der Strings beinhaltet. Dazu muß der Arrayname mit einem Dollarzeichen ($) abgeschlossen werden.

Stringarrays können, wie numerische Arrays auch, in verschiedenen Dimensionen mit Hilfe des DIM-Befehles angelegt werden. Stringarrays erfordern allerdings noch mehr Speicherplatz.

Arbeiten mit Arrays

Die Variablen in einem Array werden nicht anders behandelt als andere Variable auch. Das heißt, daß hier alle logischen oder mathematischen Vorgänge und Vergleiche in Frage kommen können. Bei Stringarrays sind alle Stringmanipulationen möglich.

Ein ausführliches Beispiel für den Umgang mit Arrays finden Sie im Kapitel „Sortieren". Dort werden Listen mit Zahlen, aber auch Listen mit Strings in eine Reihenfolge gebracht.

Die Energieverbrauchsrechnung

Das folgende Programm zeigt Ihnen, wie eine Datenliste in eine hübsche Grafik umgesetzt werden kann. Über ein Jahr hinweg wurde wöchentlich der Gasverbrauch abgelesen. Für jede Woche konnte also der Verbrauch errechnet werden. Durch Untereinanderreihung aller Verbrauchszahlen würden wir eine lange Liste mit Zahlen erhalten, aus der man auch nicht sehr viel ersehen könnte. Wenn die Verbrauchszahlen nun in eine Grafik gebracht werden, die nach wöchentlichem Verbrauch und Zeit (die Wochen) angelegt ist, kann man auf einen Blick erkennen, wie sich der Energieverbrauch über das Jahr gesehen entwickelt hat. Die Verbrauchszahlen sind in einer Reihe von Datenzeilen gespeichert.

```
10 SCREEN 2 : KEY OFF
15 DRAW "bm10,0"
20 FOR Y%=0 TO 180 STEP 10
30 DRAW "m10,=y%;"
40 DRAW "nm8,=y%;"
50 NEXT Y%
60 DRAW "bm10,180"
70 FOR X%=10 TO 218 STEP 4
80 DRAW "M=x%;,180"
90 DRAW "nm=x%;,183"
100 NEXT X%
110 READ G%
120 H%=180-G%
130 DRAW "bm10,=h%;"
140 FOR A=1 TO 48
150 READ G%
160 H%=180-G%
170 R%=10+A*4
180 DRAW "m=r%;,=h%;"
190 NEXT A
200 GOTO 200
500 DATA 107
510 DATA 77,93,88,76,102,136,149,119,88,100,77,120
520 DATA 120,120,64,53,58,63,50,44
530 DATA 55,34,18,14,14,14,13,7,7,7,7,7,9,9,8,14,15,16
540 DATA 20,15,28,23,25,37,32,70,113
```

Die Zeilen 15-100 bewirken, daß entlang der vertikalen Linie links auf dem Schirm eine Skala für die verbrauchte Gasmenge pro Woche (Kubikmeter) entsteht. Entlang der horizontalen Linie unten auf dem Schirm erscheinen Striche für die einzelnen Wochen. Jeder Strich steht für eine Woche.

Einige Befehle in diesem Programm sind noch nicht behandelt worden. Sie können dazu die Kapitel über Grafik und Farbe lesen. Besser ist es, Sie schauen sich das Programm später noch einmal an, wenn wir auch diese Abschnitte behandelt haben.

Die Zeilen 110-130 lesen die ersten Angaben aus der Datenliste und bringen den grafischen Cursor (den Stift) auf der Linie ganz links in eine Höhe, die von der ersten Angabe (H% = 180-G%) bestimmt wird. Die Zeilen 140-190 bewirken genau das gleiche für alle anderen Angaben. Die Stelle, zu der die Linie der Grafik hingezogen werden muß, ist in der Höhe abhängig von den Verbrauchszahlen aus der Datenliste, und in der Breite ist sie abhängig von der Größe von A (R% = 10 + A*4). Zeile 200 bewirkt, daß die Grafik nicht unmittelbar nach Vollendung gleich wieder verschwindet. Versuchen Sie einmal, diese Zeile zu löschen, und achten Sie auf das Resultat.

Die Zeilen 500 - 520 einschließlich umfassen die Verbrauchszahlen pro Woche. In diesem Programm war von vornherein festgelegt, daß die Datenmenge für genau ein Jahr ausreichend war. Aber es kann ja sein, daß Sie das Programm regelmäßig verändern wollen, so daß jede Woche eine neue, ergänzte Grafik erstellt wird. Dazu müssen Sie A in Zeile 140 nicht von 1 bis 51 laufen lassen, sondern von 1 bis Z. Der Wert von Z hängt von der Datenmenge in der Liste ab. Diese können Sie angeben, indem Sie als erste Angabe in der Datenliste die Datenmenge in der Liste eingeben.

Beispielsweise mittels Erstellung einer neuen Zeile 505:

```
505 DATA 51 : REM Datenmenge in Datenliste
```

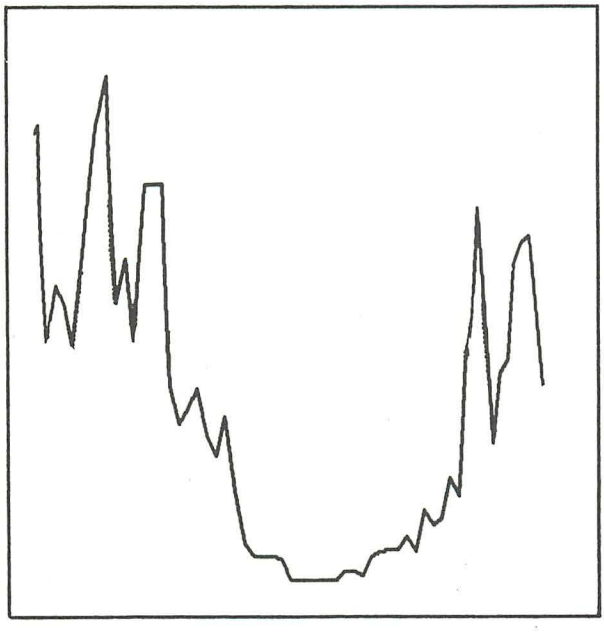
Z muß nun der genaue Wert mit einem Lesebefehl (READ) zugewiesen werden. Fügen Sie eine Zeile 135 ein:

```
135 READ Z : REM Datenmenge in Datenliste
```

Schließlich muß nur noch Zeile 140 verändert werden:

```
140 FOR A=1 TO Z
```

Sie können dieses Programm beliebig variieren, indem Sie die Datenliste entsprechend anpassen. Sehen Sie sich auf diese Weise einmal Ihren eigenen Energieverbrauch in einem Jahr an, oder bringen Sie ganz andere Daten in die Grafik.



12. SORTIEREN

Beim Sortieren von Listen mit Daten ist es unentbehrlich, Daten zeitweise in einem Array ablegen zu können. Die Daten werden der Reihenfolge nach in dem Array untergebracht. Zum Sortieren bedient sich der Computer in einem Array der Numerierung von Variablen.

Ein Sortierprogramm kann in vielerlei Hinsicht sinnvoll sein. Sie können damit lange Zahlenlisten in die richtige Reihenfolge bringen, Sie können Adressenbestände, Ihre Kassetten- und Schallplattensammlung, Ihre Disketten, Computerprogramme und noch manches andere damit in alphabetische Reihenfolge bringen.

Was passiert nun eigentlich beim Sortieren einer Zahlenliste?

Zunächst einmal wird die Liste auf die kleinste Zahl hin untersucht. Diese Zahl nennen wir einmal K. Im Array von Zahlen steht diese Zahl an einer Stelle, der wir den Namen X geben wollen. Die kleinste Zahl finden wir auf L(X). Die niedrigste Zahl wird mit der ersten Zahl in der Liste abgewechselt. L(X) wechselt mit L(1) ab. Nun werden die Zahlen von L(2) an bis hin zum Schluß L(E) untersucht. Der kleinste nun gefundene Wert wird in L(2) untergebracht. Anschließend wird von L(3) an bis hin zum Schluß gesucht. Das setzt der Computer solange fort, bis die Liste nur noch von der vorletzen Zahl L(E-1) bis hin zur letzten Zahl L(E) untersucht werden muß. Diese beiden Zahlen werden miteinander verglichen und eventuell vertauscht. Dann ist die Liste fertig.

Zum Tauschen von zwei Zahlen wird von den meisten Computern eine Hilfsvariable benutzt. Damit Sie diese Konstruktion in Programmen wiedererkennen können, die nicht für den MSX geschrieben worden sind, folgt hier eine kurze Erläuterung:

```
LET Q = L(B)
LET L(B) = L(A)
LET L(A) = Q
```

Mit diesen drei Zeilen wird der Wert der Variablen L(B) zeitweise in Q gespeichert. Anschließend kopiert der Computer den Wert der Variablen L(A) in L(B). Auf diese Weise ist L(A) frei (der alte Wert befindet sich zwar noch darin, aber er wird nun nicht mehr gebraucht). Der Variablen L(A) wird jetzt der Wert von Q zugewiesen. Darin befindet sich der alte Wert von L(B), und so ist der Wert von L(B) in L(A) gelangt und der Wert von L(A) in L(B).

Die MSX-Computer machen es genauso, aber bei ihnen geschieht es intern. Der Benutzer hat damit nichts zu tun. Er muß dem Computer lediglich den Befehl geben, die beiden Variablen zu tauschen:

```
SWAP                    (tausche aus)
```

Dieser Befehl bewirkt den Austausch des Inhaltes von zwei Variablen. Er kann sowohl bei Arrayvariablen als auch bei allen anderen Arten von Variablen angewandt werden. Es ist nicht möglich, den Inhalt von ungleichartigen Variablen auszutauschen.

Um L(A) mit L(B) zu tauschen, nehmen Sie also: SWAP (LA), L(B). Die Reihenfolge der beiden Variablen ist ohne Bedeutung.

Die eben beschriebene Methode, Daten zu ordnen, scheint umständlicher zu sein als einer so „einfachen" Aufgabe wie Sortieren angemessen ist. Der Computer fängt mit der Zahl L(1) an und vergleicht diese mit allen anderen Zahlen. Jedesmal, wenn eine Zahl kleiner ist, werden die beiden Zahlen ausgetauscht, und der Computer sucht in der Liste weiter, jetzt mit der folgenden Zahl beginnend. Ein Programm dazu könnte so aussehen:

```
10 REM Sortierprogamm
20 SCREEN 0 : KEY OFF
30 READ N
40 DIM L(N)
50 PRINT : PRINT "Die Datenliste: "
60 PRINT
70 FOR A=1 TO N
80 READ L(A)
90 PRINT L(A);
100 NEXT A
110 FOR X=1 TO N-1
120 FOR Y=X+1 TO N
130 IF L(Y) >= L(X) THEN GOTO 150
140 SWAP L(X),L(Y)
150 NEXT Y
160 NEXT X
170 PRINT : PRINT
180 PRINT : PRINT "Die sortierte Liste: "
190 PRINT
200 FOR A=1 TO N
210 PRINT L(A);
220 NEXT A
230 DATA 9
240 DATA 12,3,109,16,7,88,8,-1,43,65
```

In Zeile 30 wird die erste Angabe gelesen. Diese Angabe gehört nicht zur zu sortierenden Liste, sondern sie dient dazu, die Anzahl Daten in der Datenliste anzugeben. Die Zeilen 70, 80 und 100 lesen die Daten ein und bringen sie im Array L unter. Die Daten werden zur Kontrolle mittels Zeile 90 auf dem Bildschirm angezeigt.

Das eigentliche Sortieren erfolgt in den Zeilen 110-160. Jedesmal werden zwei Zahlen miteinander verglichen. Ist die Zahl an einer höheren Stelle im Array größer als die hiermit verglichene Zahl, werden sie nicht untereinander ausgetauscht. In einem solchen Fall geht der Computer zu Zeile 150, wo der erste der beiden NEXT-Befehle dafür sorgt, daß das folgende Zahlenpaar an die Reihe kommt.

Das Resultat wird durch die Zeilen 170-220 angezeigt. In Zeile 230 steht die Angabe, die die Länge der Datenliste festlegt. In Zeile 240 steht die Datenliste.

Wenn Sie schrittweise sehen möchten, wie der Computer die Zahlen jedesmal austauscht, können Sie die folgenden Zeilen einfügen:

```
122 FOR A=1 TO N
124 LOCATE 20, A*2+5
```

```
126 PRINT L(A);
128 FOR Z=1 TO 50 : NEXT Z
129 NEXT A
```

Sogar mit nur ganz geringer Verzögerung (Zeile 128) zur Sichtbarmachung des Zahlentausches kostet es sehr viel Zeit, so eine kurze Datenliste zu ordnen.

Das Programm oben ist ein schönes Sortierprogramm, und das kleine Stück, in dem sich das Sortieren dann tatsächlich abspielt, kann gut als Subroutine in einem anderen Programm untergebracht werden. Das Programm ist allerdings nicht wirklich effizient. Der Computer muß schließlich jedes Mal neu die gesamte Datenliste durcharbeiten, um festzustellen, ob die beiden Zahlen ausgetauscht werden müssen oder nicht. Es gibt noch eine wesentlich raschere Art, Daten zu sortieren, Man nennt sie „bubble-sort". Hierbei werden jedesmal die Zahlen, die direkt nebeneinander stehen (oder bei einer Liste mit untereinander stehenden Zahlen solche, die untereinander stehen), miteinander verglichen. Die Zahlen werden dann ausgetauscht, wenn die Reihenfolge der Größe nach nicht stimmt. Der Computer vergleicht dann die größere der beiden Zahlen mit derjenigen Zahl, die wiederum neben der größeren steht, und tauscht diese eventuell miteinander aus, und so fort.

Beim ersten Listendurchlauf wird L(1) mit L(2) verglichen. Wenn L(2) kleiner ist als L(1), werden die beiden ausgetauscht. Anschließend wird L(2), worin sich zu diesem Zeitpunkt die größere der beiden Zahlen L(1) und L(2) befindet, mit L(3) verglichen. Auch jetzt wird ausgewechselt, wenn L(3) kleiner ist. Dieser Vorgang wird so lange wiederholt, bis L(E-1) mit L(E) verglichen worden ist und eventuell auch hier ein Austausch stattgefunden hat.

Beim zweiten Durchlauf werden dann die Zahlen von L(1) bis L(E-1) in gleicher Weise miteinander verglichen. In L(E) befindet sich jetzt die größte Zahl aus allen Vergleichen des ersten Durchlaufs. Das macht der Computer so lange, bis am Ende nur noch L(1) und L(2) miteinander verglichen werden. Auf diese Weise werden die größten Zahlen nach unten gebracht, und die kleinsten „sprudeln" (von engl. „to bubble", daher auch die Bezeichnung) allmählich nach oben.

Wenn man will, daß der Computer nicht jedesmal die Liste durchläuft, während sie eigentlich schon soweit in Ordnung ist, kann man noch eine besondere Kontrolle einbauen. Immer wenn der Computer mit einer neuen Runde beginnt, wird eine Speicherstelle auf „fertig" gesetzt. Wenn der Computer nun beim Durchlaufen der Liste einen Austausch vornimmt, wird diese Speicherstelle „nicht-fertig". Bevor er mit einer neuen Runde anfängt, sieht der Computer an dieser Speicherstelle nach, ob sie „fertig" oder „nicht-fertig" ist. Ist sie „fertig", wurde beim letzten Mal nichts ausgetauscht, und alle Zahlen stehen in der richtigen Reihenfolge. Zeigt die Speicherstelle „nicht-fertig" an, ist in der vorigen Runde noch wenigstens ein Austausch vorgenommen worden und der Computer muß noch einen weiteren Durchlauf machen. Das Programm dazu sieht so aus:

```
10 REM bubblesort
20 SCREEN 0 : KEY OFF
30 READ N
40 DIM L(N)
```

```
50 PRINT : PRINT "Die Datenliste: "
60 PRINT
70 FOR A=1 TO N
80 READ L(A)
90 PRINT L(A);
100 NEXT A
110 REM das Sortieren
120 LET E=N-1
130 LET K$="nicht-fertig"
140 REM Anfang Schleife
150 LET K$="fertig"
160 FOR G=1 TO E
170 IF L(G)<=L(G+1) THEN GOTO 210
180 REM tauschen
190 SWAP L(G),L(G+1)
200 LET K$="nicht-fertig"
210 NEXT G
220 IF K$<>"fertig" THEN GOTO 150
230 PRINT : PRINT
240 PRINT : PRINT "Die sortierte Liste: "
250 PRINT
260 FOR A=1 TO N
270 PRINT L(A);
280 NEXT A
290 DATA 20
300 DATA 13,21,99,76,56,74,39,-1,111,-44
310 DATA 38,92,37,52,17,-1,-999,67,778,2
```

Natürlich ist es kürzer, anstelle der Stringvariablen K$ mit dem Wert „nicht-fertig" oder „fertig" eine integere Variable mit den Werten 0 oder 1 zu nehmen. Im Programm oben wurden lediglich die Worte eingesetzt, um Ihnen den Gebrauch sogenannter Flags zu verdeutlichen. Ein Flag ist ein Zeichen, das anzeigt, ob eine bestimmte Situation eingetreten ist. Nur wenn der Flag auf „fertig" steht, soll das Durchlaufen der Liste gestoppt werden.

Das Sortieren mit Hilfe von „bubble-sort" ist nicht immer die rascheste Methode. In der Hobby-Anwendung ist jedoch die weitaus am häufigsten auftretende Situation die, daß einige Daten in eine bereits bestehende und sortierte Datenliste eingefügt werden müssen. In diesen Fällen ist „bubble-sort" allerdings die schnellste Methode.

Neben Zahlen können Sie auch Worte in alphabetische Reihenfolge bringen lassen. Nach dem vorangegangenen Programm sollte das nun folgende eigentlich keine Schwierigkeiten mehr bereiten. Eventuell lesen Sie noch einmal im Kapitel über Strings nach. Um das Programm möglichst vielseitig anwenden zu können, ist es nicht nur imstande, nach dem Namen eines bestimmten Programms zu sortieren, sondern es vermag auch nach Kassetten (alle Programme einer bestimmten Kassette zusammen) oder nach Jahren (chronologische Reihenfolge der Programme) zu sortieren.

In den Zeilen 230, 330 und 430 finden Sie Befehle, die bislang noch nicht behandelt worden sind. Es handelt sich dabei um Subroutinen. Das sind kleine Unterprogramme, die verschiedene Male wiederholt werden. Von verschiedenen Stellen im Programm aus kann auf diese Subroutinen mittels des Befehles GOSUB verwiesen werden, dem die Nummer der Zeile mit dem Beginn der Subroutine folgt. Am Ende der

Subroutine steht der Befehl RETURN, der den Computer wieder zu der Zeile zurückkehren läßt, von der aus er die Subroutine begonnen hat. Mehr hierüber erfahren Sie im Kapitel „Kleine Abstecher".

```
10 REM Allessortierer
20 SCREEN 0 : KEY OFF
30 READ N
40 DIM N$(N):DIM C(N):DIM P(N):DIM J(N)
50 FOR A=1 TO N
60 READ N$(A):READ C(A):READ P(A):READ J(A)
70 NEXT A
80 REM Wahl treffen
90 PRINT "Sie koennen aus diesen Sortiermoeglichkeiten waehlen: "
100 PRINT "-PROGRAMMNAME (N)"
110 PRINT "-KASSETTENNUMMER (K)"
120 PRINT "- JAHR (J)" : PRINT
130 INPUT "Ihre Wahl ";A$
140 IF A$="N" OR A$="n" THEN GOTO 280
150 IF A$="K" OR A$="k" THEN GOTO 380
160 IF A$="j" OR A$="J" THEN GOTO 180
170 GOTO 130
180 E=N-1 : K=0
190 REM Jahr
200 K=1
210 FOR G=1 TO E
220 IF J(G)<=J(G+1) THEN GOTO 240
230 GOSUB 710
240 NEXT G
250 E=E-1
260 IF K<>1 THEN GOTO 200
270 GOTO 480
280 E=N-1 : K=0
290 REM Programmname
300 K=1
310 FOR G=1 TO E
320 IF N$(G)<=N$(G+1) THEN GOTO 340
330 GOSUB 710
340 NEXT G
350 E=E-1
360 IF K<>1 THEN GOTO 300
370 GOTO 480
380 E=N-1 : K=0
390 REM Kassettennummer
400 K=1
410 FOR G=1 TO E
420 IF C(G)<=C(G+1) THEN GOTO 440
430 GOSUB 710
440 NEXT G
450 E=E-1
460 IF K<>1 THEN GOTO 400
470 REM drucken
480 CLS
490 PRINT "die sortierte Liste lautet:"
500 LOCATE 0,2 : PRINT "PROGRAMMNAME     KAS.  NR.  JAHR"
510 FOR A=1 TO N
520 LOCATE 0,A+4 : PRINT N$(A)
530 LOCATE 20,A+4 : PRINT C(A)
540 LOCATE 25,A+4 : PRINT P(A)
550 LOCATE 30,A+4 : PRINT J(A)
```

```
560 NEXT A
570 DATA 11
580 DATA "Sortiere",1,000,1984
590 DATA "Flipper",10,050,1985
600 DATA "Flaechen",7,125,1984
610 DATA "Kubus",6,150,1984
620 DATA "Textverarb.",12,75,1985
630 DATA "Rakete",3,000,1983
640 DATA "Helikopter",3,050,1983
650 DATA "Palindrom",1,100,1984
660 DATA "Eisscholle",9,50,1985
670 DATA "Mittelwerte",3,150,1983
680 DATA "Test",0,0,0
690 END
710 SWAP N$(G),N$(G+1)
720 SWAP C(G),C(G+1)
730 SWAP P(G),P(G+1)
740 SWAP J(G),J(G+1)
750 K=0
760 RETURN
```

13. DATEIEN BEARBEITEN

Der Sinn von Dateien

In den vorigen Kapiteln bereits hat sich als praktisch erwiesen, Datenlisten nicht jedesmal in ein Programm aufnehmen zu müssen. Die effizienteste Methode ist es, eigene Programme und eigene Datenlisten zu haben. Nehmen wir einmal an, Sie verfügen über eine ganze Serie von Datenverarbeitungsprogrammen sowie über eine Reihe von Datenlisten. Diese Datenlisten sind vollkommen getrennt von den Programmen. Auf diese Weise können Sie immer eine Datenliste von einem bestimmten Programm bearbeiten lassen. Einige Standardprogramme wären dann beispielsweise: ein Programm zur grafischen Darstellung, ein Sortierprogramm, ein Printprogramm (mit oder ohne Tabellen), ein Programm zur Eingabe neuer Daten und Korrektur oder Löschung bereits vorhandener Daten, ein statistisches Programm, ein Spielprogramm oder ein Lehrprogramm.

Nicht jedes Programm kann mit den gleichen Daten arbeiten, aber es gibt doch eine ganze Reihe von Überschneidungen. Darüber hinaus ist es möglich, verschiedene Funktionen in ein Programm aufzunehmen, doch die Nachteile davon liegen auf der Hand: ein langes Programm, geringere Übersicht aufgrund einer komplizierten Programmstruktur und weniger Speicherplatz für die Datenliste.

Als denkbare Datensammlungen können genannt werden: Energieverbrauch, Programmothek, Diskothek, Kassettensammlung, Bibliothek, Untersuchungsergebnisse, Wörterbücher, schulische Daten, Spiele oder die Erstellung von Zeichnungen.

Der MSX verfügt über ausreichende Kapazitäten zur Erstellung gesonderter Dateien. Diese Dateien werden im Englischen „files" genannt. Diesem Begriff werden Sie in der Computerliteratur noch häufig begegnen. Obgleich ein Teil der Befehle beim MSX für den Umgang mit Dateien unabhängig vom Speichermedium (Kassette oder Diskette) ist, gibt es doch einige Unterschiede beim Öffnen und Schließen eines Bestandes.

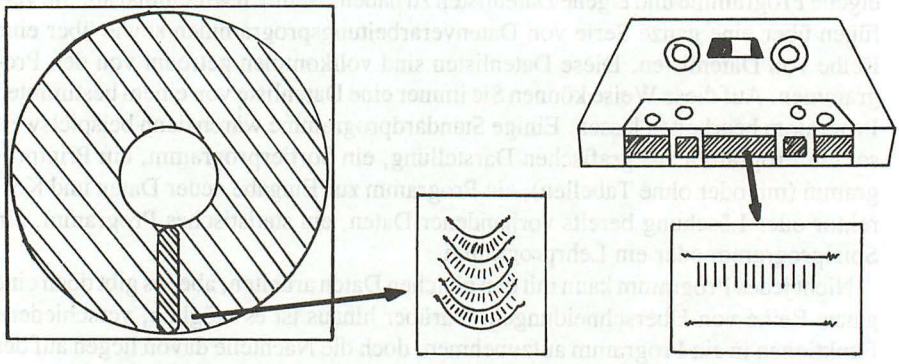
Der Unterschied zwischen Kassetten- und Diskettendateien

Um Ihnen eine Vorstellung vom Umgang mit Dateien zu vermitteln, sollten Sie etwas über die Art und Weise zu wissen, wie Dateien auf Kassette oder Diskette gespeichert werden.

Jede Sammlung zusammengehöriger Daten wird Datei genannt. Die Dateien werden als magnetische Signale auf magnetische Träger gebracht. Im Hobbybereich werden das in erster Linie Kassetten oder Disketten sein. Der große Unterschied zwischen Diskette und Kassette ist der, daß die letzte eine lange Datenliste trägt, die von Anfang bis Ende gelesen werden muß. Dies ist eine Folge der Übertragung der Signale auf eine Kassette. Das Band läuft langsam an den Tonköpfen des Kassettenrekorders vorbei, und während dieses Vorbeilaufens werden die Signale auf die Kassette

gebracht. Zum Lesen muß das Band wieder zum Beginn zurückgespult werden, und der Lesevorgang erfolgt wiederum vom Bandanfang bis zum Bandende.

Eine Diskette setzt sich – vereinfacht dargestellt – aus einer ganzen Menge von Kassettenbändern zusammen, wobei die Bänder kreisförmig ineinander angeordnet sind. Ans „Ende" eines solchen kreisförmigen Bandes (eines Track = Spur) angelangt, haben wir gleich wieder einen „Anfang".



Schematische Darstellung eines Kassettenbandes und einer Diskette

Der Kopf eines Diskdrives (des Gerätes, mit dem die Disketten gelesen und beschrieben werden können) kann sehr rasch von einer Spur auf die nächste wechseln. Aufgrund der Kreisform, sowie der Möglichkeit, von einer Spur in eine andere zu gehen, kann bei Gebrauch von Disketten gleich eine bestimmte Angabe aus einer bestimmten Datei gelesen werden. Die Diskette umfaßt neben den Daten immer auch eine Auflistung aller vorhandenen Dateien, wodurch der Computer weiß, wohin er zu wechseln hat.

Dadurch, daß sich eine Diskette wesentlich schneller bewegt als eine Kassette, ist die Geschwindigkeit, mit der hier gearbeitet werden kann, um etliches höher als bei Kassetten.

Der große Nachteil von Disketten und Diskettenlaufwerken ist allerdings die Tatsache, daß sie recht kostspielig sind. Solange bei der Bearbeitung von Dateien nicht unbedingt Wert auf große Schnelligkeit gelegt wird, stellt der Kassettenrekorder immer noch eine gute und preiswerte Lösung dar.

In diesem Buche wird Ihnen erklärt, wie Sie mit Kassettendateien umgehen müssen. Falls nötig, wird auf die Unterschiede zur Diskettendatei kurz eingegangen.

Das Anlegen einer Datei

Eine Datei ist mit den Datenlisten vergleichbar, wie sie hiervor immer in DATA-Statements untergebracht worden sind. Es gibt allerdings zwei bedeutende Unterschiede:

- Daten in DATA-Statements können nur in dem Programm benutzt werden, in dem die Daten auch stehen. Daten aus einer Datei kann man in verschiedenen Programmen gebrauchen.
- Daten in DATA-Statements müssen vom Benutzer eingegeben werden, während die Daten in Dateien vom Computer angelegt sein können.

Es gibt zahlreiche Methoden, die Benutzung von Dateien anschaulich zu machen. Ungeachtet der Darstellungsweise müssen folgende Punkte berücksichtigt werden:

- Jede Datei hat einen eigenen Namen.
- Jede Datei besteht aus einer Reihe geordneter Daten, wobei jede Angabe einen festen Platz innerhalb der Datei hat.
- Dateien müssen vor der Benutzung geöffnet und nach Benutzung (das ist sehr wichtig!) wieder geschlossen werden.
- Daten werden nacheinander und Stück für Stück in eine Datei eingefügt oder aus der Datei gelesen (eigentlich kopiert, denn das Original verbleibt in der Datei).

Das folgende Programm eröffnet dem Benutzer die Möglichkeit, immer neu eine Zahl einzugeben. Der Computer schreibt diese Zahl dann auf eine Kassette.

```
10 CLS
20 PRINT "Lege die Kassette in den Rekorder ein."
30 PRINT "Spule zur richtigen Stelle vor."
40 PRINT "Stelle auf AUFNAHME."
50 PRINT "Druecke RETURN, wenn fertig."
60 INPUT X
70 PRINT : PRINT "Bitte warten."
80 OPEN "cas:xxx"FOR OUTPUT AS #1
90 PRINT : PRINT
100 INPUT "Nenne eine Zahl:";GE
110 PRINT #1,GE
120 PRINT
130 INPUT "Fertig? (j/n)";A$
140 IF A$<>"j" THEN PRINT : GOTO 100
150 PRINT : PRINT "Bitte warten"
160 CLOSE #1
170 PRINT "ENDE" : END
```

Die ersten sieben Programmzeilen sagen dem Benutzer des Programms, was alles zu geschehen hat. Im Kassettenrekorder muß sich eine Kassette befinden und der Rekorder muß aufnahmebereit sein. Vorsicht! Wenn Sie nicht gut aufpassen, kann es geschehen, daß Sie unbeabsichtigt eine bereits bestehende Datei löschen, indem Sie sie einfach überschreiben. Der MSX kann nicht zuerst kontrollieren, ob sich bereits etwas auf dem Band befindet.

In Zeile 80 wird eine Datei mit dem Namen xxxx eröffnet. Dazu brauchen Sie den Befehl

OPEN ''CAS:.....'' FOR OUTPUT AS #..

Mit diesem Befehl teilen Sie mit, daß Sie eine Datei eröffnen wollen, um Daten darin abzuspeichern. Hinter dem Wort OPEN steht zwischen Anführungszeichen zunächst die Angabe des Speichermediums, für das Sie sich entschieden haben. Sie können wählen zwischen:

CAS:	Kassettenrekorder
CRT:	Textschirm
GRP:	Grafikschirm
LPT:	Drucker

Neben diesen Beschreibungen der Speichermedien, die für jeden MSX-Computer gelten, gibt es auch noch andere Erläuterungen, die bei ROM-Erweiterungen in Betracht kommen. Lesen Sie hierüber mehr im Handbuch zu der betreffenden Erweiterung.

Sie stellen fest, daß das Diskettenlaufwerk nicht erwähnt wird. Um eine Datei auf Diskette zu schreiben, befehlen Sie:

```
OPEN ''A:.....'' FOR OUTPUT AS #..
```

An der Stelle der ersten Auslassungspünktchen nennen Sie sowohl bei einer Kassette als auch bei einem Diskettenlaufwerk den Namen der Datei. Sie können dazu einen Namen wählen, der aus maximal sechs Schriftzeichen besteht. Das erste Schriftzeichen muß ein Buchstabe sein.

Die Möglichkeiten CRT: und LPT: sind nicht wirklich sinnvoll. Sie brauchen sie, um Texte auf einen Bildschirm oder Drucker zu bringen. Beide Anwendungsmöglichkeiten lassen sich über einfachere Befehle realisieren (PRINT und LPRINT). Da es nicht möglich ist, so ohne weiteres einen Text auf den Grafikschirm zu bringen, ist GRP: sehr wichtig. Wie Sie damit Texte auf dem grafischen Schirm darstellen lassen können, sehen Sie in den Kapiteln über Grafik.

Nach der Angabe des Speichermediums und des Namens der zu eröffnenden Datei (dies alles als ein String in Anführungszeichen) teilen Sie mit, ob die Datei eröffnet wird, um daraus zu lesen oder um Daten hineinzuschreiben. Im obigen Programmbeispiel wurde eine Datei eröffnet, um Daten darin zu speichern. Es handelte sich schließlich um eine neue Datei. Es besteht die Wahl zwischen:

FOR OUTPUT	für Ausgabe
FOR INPUT	für Eingabe
FOR APPEND	für Ergänzung

Von der letzteren Möglichkeit werden Sie nicht so bald Gebrauch machen. Die Möglichkeiten, Daten aus einer Datei zu lesen oder in eine Datei zu schreiben, sind zunächst ausreichend – auch hinsichtlich der Erweiterung einer bereits bestehenden Datei. FOR APPEND funktioniert außerdem bei Kassettensystemen nicht.

Nach der Angabe der Verbindung zwischen Computer und Speichermedium muß noch mitgeteilt werden, unter welcher Nummer wir in dem Programm auf die Datei verweisen wollen:

```
AS #..
```

Das Zeichen # ist das amerikanische Symbol für Nummer. Hinter # geben Sie eine Zahl zwischen 0 und 15 ein. Bei der Benutzung der Nummern 0 und 2 bis 15 gibt es noch Probleme.

Der MSX geht davon aus, daß Sie immer nur eine Datei aufrufen wollen. Wenn Sie mit einem Kassettensystem arbeiten, ist das auch der Normalfall. Wollten Sie mehr als eine Datei gleichzeitig eröffnen, müßten Sie auch mehr als eine Kassette zum Schreiben in eine Datei in Ihrem Kassettengerät bereithalten. Das heißt, Sie müßten häufig die Kassetten wechseln. Bei einem Diskettensystem ist es aufgrund der höheren Flexibilität wesentlich einfacher, mehr als nur eine Datei gleichzeitig zu eröffnen. Der Computer sucht gemeinsam mit dem Diskettenlaufwerk für Sie die Stelle, an der sich die gerade gewünschte Datei befindet und in der die Daten abgespeichert werden sollen. Diese Flexibilität ist bei einer Kassette wegen der Methode der Bandvorbeiführung am Tonkopf nicht gegeben.

Solange Sie nur eine Datei benutzen, sollten Sie dem Computer keine anderslautenden Angaben machen; auch als Dateinummer sollten Sie nur die Nummer 1 gebrauchen: AS # 1. Wenn Sie mehrere Dateien aufrufen wollen, müssen Sie folgenden Befehl eingeben:

```
MAXFILES =..
```

Anstelle der Pünktchen können Sie eine Zahl zwischen 0 und 15 eingeben. Der MSX entscheidet sich von selber für die 1. Sie brauchen diesen Befehl also eigentlich nur dann, wenn Sie ein Diskettensystem besitzen.

Wenn Sie MAXFILES 0 wählen, beschränken Sie die Möglichkeiten der Einrichtung einer Datei und des Lesens von einer Datei beträchtlich. Nachdem MAXFILES auf 0 gesetzt wurde, können Sie nur noch CSAVE, CLOAD, CLOAD?, SAVE und LOAD benutzen; das heißt, daß Sie Programme speichern und lesen, aber keine Dateien mehr einrichten können. Die Erweiterung von MAXFILES erfordert zusätzlichen Speicherplatz.

Nach der Eröffnung der Datei können wir Daten in die Datei einlesen. Zeile 100 des letzten Programms bittet den Benutzer um die Eingabe einer Zahl.

Zeile 110 gibt diese Zahl in die Datei. Dazu wird der PRINT-Befehl benutzt, dem Sie bereits mehrmals begegnet sind. In diesem Falle soll die Zahl allerdings nicht auf dem Bildschirm erscheinen (PRINT) oder auf dem Papier (LPRINT), sondern sie soll in eine Datei gesetzt werden:

```
PRINT #1, ..
```

Hinter PRINT geben Sie mit # und einer Nummer an, in welcher Datei die Eingabe hinter dem Komma untergebracht werden soll. Passen Sie auf, daß die Nummer hinter PRINT # gleich der Nummer ist, die Sie der Datei mit dem OPEN-Befehl gegeben haben. Bei Kassettensystemen wird dies immer Nummer 1 sein. Bei einem Diskettensystem können Sie verschiedene Dateien zugleich eröffnet haben, und mittels der Nummer hinter PRINT # läßt sich bestimmen, in welche Datei eine bestimmte Eingabe gehört.

Mit einem PRINT #-Befehl können Sie verschiedene Daten wegschreiben lassen. Jede Eingabe muß von der vorhergehenden durch Komma getrennt sein. Sie können sowohl den Inhalt numerischer Variablen als auch von Stringvariablen wegschreiben lassen.

PRINT #1, GE	numerische Variable
PRINT #1, A$	Stringvariable
PRINT #1, GE, A$	Kombination
PRINT #1, ''string'', 1234	String und Zahl
PRINT #1, ''string'', 1234, A$, GE	Kombination

Sie stellen fest, daß auf alle möglichen Arten Daten weggeschrieben werden können. Das einzige, worauf Sie achten müssen, ist, daß beim Lesen der Daten aus der Datei die richtigen Angaben den richtigen Variablen zugewiesen werden. Es ist nicht möglich, einen String einer numerischen Variablen zuzuweisen! Lesen Sie hierzu auch den folgenden Paragraphen über die Eingabe.

Mit der Schleife in Zeile 140 wird die Eingabe der Daten jedesmal wiederholt, bis der Benutzer auf die Frage in Zeile 130 „j" antwortet.

In Zeile 160 folgt ein besonders wichtiger Befehl. Nachdem der Benutzer die Arbeit mit einer Datei beendet, muß sie geschlossen werden. Nur auf diese Weise stellen die in einer Datei zusammengefaßten Informationen ein geschlossenes Ganzes dar. Dies ist noch wesentlich wichtiger bei der Arbeit mit Disketten!

```
CLOSE #1
```

Achten Sie darauf, daß die Nummer der abzuschließenden Datei mit der Nummer der zuvor geöffneten Datei übereinstimmt. Bei Kassettensystemen ist das immer 1.

Besitzen Sie einen Kassettenrekorder mit Fernbedienung, wird der Computer dafür sorgen, daß der Kassettenmotor im richtigen Augenblick zu laufen beginnt. Folgenden Befehl dürfen Sie zuvor nicht vergessen:

```
MOTOR ON
```

Auf diese Weise teilen Sie dem MSX mit, daß er auf einen Kassettenmotor zu achten hat. Wollen Sie das Gegenteil erreichen (etwa beim schnellen Vorlauf), befehlen Sie:

```
MOTOR OFF
```

Das Lesen in eine Datei

Um die Daten, die Sie in die Datei gebracht haben, wieder hervorzuholen, bedienen Sie sich eines kurzen Programms, das dem vorigen Programm sehr ähnlich ist:

```
10 CLS
20 PRINT "Lege die Kassette in den Rekorder ein."
30 PRINT "Spule zur richtigen Stelle vor."
40 PRINT "Stelle auf ABSPIELEN."
50 PRINT "Druecke RETURN, wenn fertig."
60 INPUT X
70 PRINT: PRINT "Bitte warten."
80 OPEN "cas:xxx"FOR INPUT AS #1
90 PRINT :PRINT
100 INPUT#1,GE
110 PRINT GE
120 PRINT
130 GOTO 100
160 CLOSE #1
170 END
```

Wenn Sie dieses Programm laufen lassen wollen, müssen Sie bei Bedarf die Kassette mit den Daten, die Sie bei dem vorigen Programm eingegeben haben, wieder in den Kassettenrekorder einlegen und an die richtige Position vorspulen. Stellen Sie den Rekorder auf Wiedergabe und drücken Sie RETURN.

In Zeile 80 wird die Datei wieder geöffnet:

OPEN ''CAS:xxx'' FOR INPUT AS #1

Hinter der Angabe des Gerätes, von dem die Daten stammen (CAS:) steht der Name der zu lesenden Datei. Bei Kassettensystemen können Sie den Namen beim Lesen oder Schreiben fortlassen. Wenn Sie den Namen beim Lesen nicht nennen, wird die erste Datei gelesen, die der Computer auf dem Band findet. Lassen Sie einen Namen beim Schreiben weg, wird die Datei ohne Bezeichnung auf die Kassette gebracht. Bei Diskettensystemen geht es ohne Dateinamen nicht.

Haben Sie in diesem Programm den gleichen Namen gewählt wie schon beim Wegschreiben im vorigen Programm, oder haben Sie in diesem Programm keinen Namen angegeben, wird der Computer für einen Augenblick nur „Moment bitte" anzeigen. Anschließend erscheinen die beim ersten Programm eingegebenen Zahlen.

Um die Daten aus einer Datei einzulesen, befehlen Sie:

INPUT #..

Dieser Befehl ist mit dem nomalen INPUT-Befehl vergleichbar. Hinter INPUT # geben Sie die Nummer ein, die Sie der Datei mit dem OPEN-Befehl zugeteilt haben. Bei Kassettensystemen ist das eine 1. Anschließend machen Sie hinter dem Befehl die zu lesenden Eingaben, durch Kommata voneinander getrennt. Achten Sie bei

der Angabe der zu lesenden Daten darauf, daß ein String nur einer Stringvariablen zugewiesen werden kann.

Sehen Sie sich noch einmal die Beispiele an, die wir zum PRINT #-Befehl gegeben haben. Hierzu passen die folgenden INPUT #-Befehle:

INPUT #1, GE	numerische Angaben
INPUT #1, A$	Stringangabe
INPUT #1, GE, A$	Kombination
INPUT #1, X$, Y	String und numerische Angabe
INPUT #1, X$, Y, A$, GE	Kombination

Achten Sie beim Wegschreiben und Lesen von Daten auf die Reihenfolge. Der Computer verändert diese Reihenfolge nie. Sie müssen die Daten in der Reihenfolge zurücklesen, in der sie weggeschrieben worden sind; andernfalls bekommen Sie Schwierigkeiten mit der Zuweisung der Variablen.

Im Gegensatz zum normalen INPUT-Befehl braucht hinter INPUT# kein RETURN zu folgen.

Zeile 110 des Programms stellt die eben der Datei entnommenen Daten dar. Leerstellen, die zu Beginn einer numerischen Angabe stehen, bleiben unberücksichtigt. Die Anführungszeichen der Strings werden ebenfalls nicht dargestellt.

Zeile 130 bewirkt eine Schleife. Dadurch liest der Computer jedesmal neu Daten aus der Datei. Der Befehl CLOSE #1 wird deshalb nicht ausgeführt. Wenn der Computer in der Datei keine Daten mehr finden kann, während dennoch Daten gelesen werden müssen, macht er eine Fehlermeldung:

INPUT PAST END IN ... (Eingabe nach dem Ende in ...)

Das Programm ist also nicht in Ordnung. Es gibt einen besonderen Befehl, mit dem sich nachsehen läßt, ob das Ende einer Datei bereits erreicht ist. Mit diesem Befehl in der Schleife können wir den Computer veranlassen, den Weg hin zur Datei am Ende der Datei wieder zu schließen und das Programm zu stoppen.

EOF End of File (Dateiende)

Anstelle der Pünktchen geben Sie die Nummer ein, die der Datei mit dem OPEN-Befehl zugewiesen worden ist.

EOF ist −1, wenn das Ende der Datei erreicht ist, und 0, solange dies nicht der Fall ist. Die einfachste Art, eine Fehlermeldung wie oben zu vermeiden, ist die Änderung von Zeile 130 im Programm:

```
130 IF EOF (1)=0 THEN GOTO 100
```

Da EOF im Computer als −1 oder 0 gespeichert ist und die Überprüfung von IF...THEN...-Behauptungen ebenfalls mit 0 und −1 vor sich geht, ist es nicht nötig,

hinter EOF =1 einzugeben; weil darüber hinaus GOTO in einer IF...THEN-Behauptung gleichfalls überflüssig ist, kann diese Zeile verkürzt werden:

```
130 IF NOT EOF (1) THEN 100
```

Wollen Sie den Weg lediglich nach Erreichen von EOF schließen, geben Sie ein:

```
IF EOF (1) THEN CLOSE #1
```

Um mit diesem Programm arbeiten zu können, war es notwendig, die Kassette wieder zu der Stelle zurückzuspulen, an der die Datei beginnt. Damit Irrtümer ausgeschlossen sind, ist es ratsam, zu diesem Zweck immer einen „runden" Zählerstand im Zählwerk des Kassettenrekorders zu haben. Etwa:

```
000 = Bücher
100 = Plattensammlung
200 = Programmsammlung
300 = Karte
usw.
```

Haben Sie den richtigen Zählerstand eingestellt, geben Sie den Befehl RUN. Sicherheitshalber gibt Ihnen das Programm noch Gelegenheit, den richtigen Zählerstand zu finden. Danach muß der Rekorder auf Wiedergabe gestellt und RETURN gedrückt werden. Ein Kassettenrekorder mit Fernbedienung wird nun vom Computer gesteuert vorwärtslaufen und anhalten, wobei die Daten gelesen und auf dem Schirm dargestellt werden. Ein normaler Kassettenrekorder wird weiterlaufen und jedesmal auf ein Dateifragment treffen, das gelesen und dargestellt wird (bis EOF).

Hier zeigt sich der Nachteil eines Kassettenrekorders ohne Fernbedienung. Beim Aufnehmen der Daten in eine Datei dreht sich der Rekorder immer weiter, während der Computer Daten solange sammelt, bis ein Daten-„Block" voll ist. Immer wenn ein „Block" voll ist, bringt der Computer ihn auf die Kassette. Zwischen den verschiedenen „Blöcken" können beim langsamen Eingeben von Hand (wie beim ersten Beispielprogramm) große leere Stellen zwischen den einzelnen „Blöcken" entstehen. Beim Abspielen der Kassette zur Datenentnahme ergeben sich so zuweilen beträchtliche Verzögerungen. Ein Kassettenrekorder mit Fernbedienung vermeidet solche Probleme, da er vom Computer nach jedem „Block" angehalten wird, bis der folgende „Block" fertig ist. Sie können allerdings auch alle Daten zunächst in eine Datenliste setzen, um sie anschließend rasch hintereinander in eine Datei zu schreiben. Auf diese Weise wird der Unterschied zwischen einer Datei und einer Datenliste in einem Programm zwar geringer, aber der Vorteil einer Datei, daß eine Liste mit Daten nur ein einziges Mal eingegeben werden muß, um für zahlreiche Zwecke und Programme verfügbar zu sein, bleibt bestehen.

Das Programm unten zeigt Ihnen, wie Sie Daten zunächst in eine Datenliste setzen und diese dann vom Computer in eine eng beschriebene Datei bringen lassen. Bei den benutzten Daten handelt es sich um Zeichenkoordinaten für ein geographisches Programm. Die ersten Daten im gesonderten DATA-Befehl bezeichnen lediglich die Anzahl der benutzten Daten.

```
10 CLS
20 PRINT "Lege die Kassette in den Rekorder ein."
30 PRINT "Spule zur richtigen Stelle vor."
40 PRINT "Stelle auf AUFNAHME."
50 PRINT "Druecke RETURN, wenn fertig."
60 INPUT X
70 PRINT : PRINT "Bitte warten "
80 OPEN "cas:Karte" FOR OUTPUT AS #1
90 READ A
100 FOR C=1 TO A
110 READ X,Y
120 PRINT #1,X,Y
130 NEXT C
140 CLOSE #1
150 END
200 DATA 260
210 DATA 75,37,70,63,64,83,57,97,53,103,49,104,50,106,52,107,54,110,59,114
220 DATA 59,114,62,115,65,116,68,115,72,117,72,119,67,119,63,123,56,123,56,127,59,136
230 DATA 56,137,50,135,44,126,40,125,32,127,26,130,26,134,31,135,32,137,35,137
240 DATA 38,141,42,141,45,139,46,138,49,138,59,141,62,144,66,144,67,143,66,137
250 DATA 70,134,70,139,74,138,76,135,81,135,82,136,79,139,80,141,87,139,89,134
260 DATA 92,133,94,135,93,143,100,148,108,148,112,147,114,152,123,152,124,155,120,163
270 DATA 120,171,117,177,117,179,120,181,120,183,123,183,126,182,131,183,132,181,132,177
280 DATA 136,175,136,171,132,167,130,167,128,161,132,161,132,157,135,153,134,151,132,151
290 DATA 132,147,139,139,139,131,134,122,128,115,125,114,125,110,135,109,135,107,140,106
300 DATA 147,111,158,103,160,104,163,97,161,94,156,95,156,92,160,91,171,83,172,77
310 DATA 170,71,168,67,164,68,160,67,155,62,157,59,156,56,158,53,167,52,168,39
320 DATA 175,31,176,19,177,17,176,15,168,13,167,11,163,10,161,3,158,1,146,2
330 DATA 138,8,136,5,128,6,114,11,107,15,101,23,101,27,104,31,100,41,100,43
340 DATA 110,44,116,42,117,44,112,47,111,51,114,54,116,55,126,53,128,54,129,56
350 DATA 12,59,125,67,122,71,116,76,115,81,110,83,106,83,91,80,92,77,86,76
360 DATA 91,73,93,69,93,65,94,63,94,64,93,65,90,63,89,57,91,56,94,57
370 DATA 96,54,99,53,100,51,94,49,91,51,90,49,91,41,88,37,85,38,84,39
380 DATA 83,41,81,40,78,37,78,35,76,35,75,37,0,0,64,119,62,121,57,119
390 DATA 54,120,50,117,50,114,48,113,44,114,43,112,44,111,49,111,51,113,64,119
400 DATA 0,0,51,122,50,124,48,123,45,121,4,120,38,121,36,118,42,115,50,119
410 DATA 51,122,0,0,54,125,55,129,54,132,49,129,48,127,54,125,0,0,58,144
420 DATA 44,155,40,156,39,151,34,149,26,153,22,151,20,146,23,141,30,141,40,145
430 DATA 46,143,47,141,50,140,58,144,0,0,96,73,95,77,102,78,104,81,110,81
440 DATA 111,76,121,69,124,65,123,59,120,59,115,57,113,58,96,73,0,0,76,27
450 DATA 75,33,77,34,80,31,82,30,83,25,82,23,81,22,76,27,0,0,88,11
460 DATA 83,16,82,21,85,19,86,17,90,13,88,11,0,0,111,2,96,5,92,8
470 DATA 92,11,95,12,97,9,100,8,102,6,110,4,111,2
```

In Zeile 80 wird eine Datei eröffnet und ein Kanal gewählt (#1). Zeile 90 liest, wie lang die Liste ist (DATA in Zeile 200). Zeile 100 bewirkt eine Wiederholungsschleife. Zeile 110 liest gleichzeitig eine X-Koordinate und eine Y-Koordinate. Zeile 120 bringt diese Koordinaten in die Datei. Es ist nicht nötig, beide Angaben jedesmal mit einigen PRINT #-Befehlen wegschreiben zu lassen. Wenn Ihnen das übersichtlicher erscheint, können Sie auch so vorgehen:

```
PRINT #1, X
PRINT #2, X
```

So werden die Daten in genau der gleichen Reihenfolge weggeschrieben.

Die Karte der Niederlande

Um aus diesem Datenberg etwas Sinnvolles zu machen, müssen Sie das Programm, das Daten liest, ein wenig erweitern. Es ist nicht beabsichtigt, die Daten lediglich als Zahlen auf dem Bildschirm erscheinen zu lassen. Da es sich um eine Liste mit Zeichenkoordinaten handelt, muß der Leseteil mit einem Zeichenteil versehen werden. Die Erklärung der verschiedenen Zeichenbefehle erfolgt später. Sie können das Programm aber jetzt schon dazu benutzen, um eine hübsche Karte von den Niederlanden darstellen zu lassen:

```
10 CLS
20 PRINT "Lege die Kassette in den Rekorder ein."
30 PRINT "Spule zur richtigen Stelle vor."
40 PRINT "Stelle auf WIEDERGABE."
50 PRINT "Druecke RETURN, wenn fertig."
60 INPUT Z
70 SCREEN 2
80 COLOR 9,15,4
90 CLS
95 LINE-(75,37),15
100 OPEN "cas:Karte" FOR INPUT AS#1
110 IF EOF(1) THEN 150
120 INPUT #1,X,Y
130 IF X>0 AND Y>0 THEN LINE-(X,Y),9 ELSE INPUT#1,X,Y:LINE-(X,Y),15
140 GOTO 110
150 CLOSE#1
160 GOTO 160
```

Die Zeilen 10-60 kennen Sie bereits aus den vorigen Programmen.

Zeile 60 bewirkt, daß der Computer solange wartet, bis der Benutzer durch Eingabe von RETURN mitteilt, daß der Kassettenrekorder läuft.

Zeile 70 stellt den grafischen Schirm her, auf dem erst Zeichnungen möglich werden.

Zeile 80 dient der Farbgebung. Rote Linien, ein weißer Hintergrund sowie ein blauer Rand. Durch Löschen des Schirms in Zeile 90 werden diese Farbgebungsbefehle wirksam.

Zeile 100 zieht eine weiße (und und infolgedessen unsichtbare) Linie zum Anfangspunkt der Karte der Niederlande.

Zeile 110 öffnet einen Weg, um eine Datei „Karte" auf dem Kassettenrekorder durch Eingabe von Nummer 1 benutzen zu können.

Zeile 120 liest jedesmal wieder zwei Daten. Das sind die X- und die Y-Koordinaten des folgenden Punktes, zu dem hin eine Linie gezogen werden muß.

Zeile 130 kontrolliert, ob zu einem Punkt eine rote Linie gezogen werden muß, oder ob dort vielleicht gerade eine Insel anfängt, so daß die Linie in weiß (=unsichtbar) gezogen werden soll. Der Anfang einer Insel wird immer dadurch angegeben, daß sowohl die X- als auch die Y-Koordinate 0 ist. Zwei Nullen hintereinander bedeuten also: überschlage diese Koordinaten, lese neue Koordinaten, aber ziehe die Linie jetzt nicht in Rot, sondern in Weiß.

Zeile 140 läßt den Computer in 160 wechseln, wenn das Ende der Datei erreicht ist.
Zeile 150 bewirkt eine Wiederholungsschleife.
Zeile 160 schließt den Kanal.
Zeile 170 sorgt dafür, daß das Programm nicht endet. Wenn das Programm fertig ist, kehrt der MSX immer wieder zum Textschirm 0 zurück. In diesem Falle verschwindet die Zeichnung; solange das Programm allerdings weiterläuft, bleibt die Zeichnung sichtbar. Um das Programm zu stoppen, geben Sie CTRL und STOP ein.

Das Zeichnen der Karte dauert einige Zeit. Der Hemmfaktor ist die Geschwindigkeit, mit der Dateien auf einer Kassette verarbeitet werden können. Sie können eine raschere Wiedergabe erreichen, indem Sie nicht immer ein kleines Stückchen der Karte zeichnen lassen, sondern zunächst einmal alle Daten wieder in einem Array speichern. Sobald die gesamte Datei gelesen ist, stellt der Computer die Karte dar. In das Programm, das Sie unten sehen, wurden ein kurzer Text sowie einige Geräusche eingefügt, um dem Benutzer die Wartezeit zu verkürzen. Wenn Sie die Kapitel über Grafik und Ton gelesen haben, können Sie sich während der Wartezeit mit einer kleinen Zeichnung und ein wenig Musik unterhalten lassen.

Das nachfolgende, etwas umfangreichere Programm weist eine weitere Verbesserung auf. Neben den Umrissen der Niederlande sind auch einige große Städte eingezeichnet. Benutzen Sie das folgende kurze Programm, um alle Daten von Städten in eine gesonderte Datei zu schreiben. Bringen Sie diese Datei direkt hinter diejenige mit den Koordinaten der Landkarte.

```
10 CLS
20 PRINT "Kassette auf AUFNAHME"
30 PRINT "RETURN wenn fertig"
40 INPUT Z
50 OPEN "cas:stadt" FOR OUTPUT AS #1
60 READ A
70 FOR C=1 TO A
80 READ X,Y,ST$
90 PRINT #1,X,Y,ST$
100 PRINT X;"";Y,ST$
110 NEXT C
120 CLOSE #1
190 DATA 19
200 DATA 119,174,"maast",84,76,"amste",72,74,"haarl",62,90,"den h"
210 DATA 68,105,"rotte",78,124,"breda",101,116,"den b",98,90,"utrec"
220 DATA 128,99,"arnhe",158,79,"henge",117,22,"leeuw",147,15,"groni"
230 DATA 106,64,"lelys",150,34,"assen",68,85,"leide",77,36,"den h"
240 DATA 122,111,"nijme",109,135,"eindh",130,172,"heerl"
```

Den größten Teil dieses Programmes kennen Sie bereits. Neu ist, daß in Zeile 80 nicht nur zwei Zahlen gelesen werden, sondern zwei Zahlen und ein String. Alle drei Daten werden in der Datei in Zeile 90 untergebracht. Zur Kontrolle werden die Zahlen und der String auf dem Schirm dargestellt.

Das nachfolgende Programm beinhaltet die Strings. Diese wurden hinzugefügt, um es Ihnen zu ermöglichen, eine Kombination aus Zeichnung und Frage- und Antwortspiel zu erstellen. Probieren Sie einmal, ob es möglich ist, dem Benutzer gleich-

zeitig die Karte mit einer eingezeichneten Stadt vorzuführen. Der Benutzer soll dann den Namen der Stadt angeben. Die ersten fünf Zeichen der Antwort müssen mit dem String verglichen werden, der zu den Koordiaten der eingezeichneten Stadt gehören.

Das folgende Programm liefert Ihnen den Zeichenteil.

```
10 CLS
20 DIM X(300)
30 DIM Y(300)
40 DIM PX(20)
50 DIM PY(20)
60 DIM P$(20)
70 A=0
80 PRINT "Kassette in richtige Position bringen."
90 PRINT "Auf WIEDERGABE stellen."
100 PRINT "RETURN druecken, wenn fertig."
110 INPUT Z
120 PRINT:PRINT:PRINT"          Die Angaben laden !"
130 PRINT:PRINT
140 PRINT"    ***************************"
150 OPEN "cas:Karte" FOR INPUT AS #1
160 A=A+1
170 INPUT #1,X(A),Y(A)
180 IF A\3=INT(A/3) THEN BEEP
190 IF EOF(1) THEN GOTO 210
200 GOTO 160
210 CLOSE #1
220 CLS : A=0
230 PRINT:PRINT:PRINT"          Die Staedte laden !"
240 PRINT:PRINT
250 PRINT
260 OPEN "cas:Stadt" FOR INPUT AS #1
270 A=A+1
280 INPUT #1,PX(A),PY(A),P$(A)
290 IF A/2=INT(A/2) THEN BEEP
300 IF EOF(1) THEN GOTO 320
310 GOTO 270
320 CLOSE #1
330 CLS
340 SCREEN 2
350 COLOR 9,15,4
360 CLS
370 LINE -(75,37),15
380 FOR A=1 TO 260
390 IF X(A)>0 AND Y(A)>0 THEN LINE -(X(A),Y(A)),9 ELSE PSET (X(A+1),Y(A+1)),15:A=A+1
400 NEXT A
410 FOR A=1 TO 19
420 PSET (PX(),PY(A))
430 LINE (PX(A)PY(A))-(PX(A)+3,PY(A)-3),9,B
440 BEEP
450 LINE (PX(A),PY(A))-(PX(A)+3,PY(A)-3),9,BF
460 NEXT A
470 GOTO 470
```

Vergleichen Sie die Zeilen 150, 180, 190, 200 mit 250, 280 und 290. Die erste Zeilenreihe sorgt für den Abschluß der Datei, nachdem das Dateiende vom Computer festgestellt wurde. Die zweite Zeilenreihe bewirkt die Schließung des Dateikanals

nach einer zuvor bestimmten Anzahl von Daten. Diese letztere Methode ist nur dann geeignet, wenn Sie genau wissen, wie umfangreich die Datei ist.

Das Programm geht davon aus, daß die Datei „Stadt" direkt hinter der Datei „Karte" auf der Kassette steht. Wollen Sie die Möglichkeit haben, während des Lesens von „Karte" und „Stadt" die Kassette zu wechseln oder die Kassette weiterzuspulen, müssen Sie zwischen die Zeilen 200 und 210 eine Pause einfügen wie bereits in den Zeilen 80-110.

14. ABSTECHER

Einleitung

Bis jetzt waren die vorgestellten Programme nicht besonders lang oder kompliziert. Aber einmal kommt der Augenblick, wo ein bestimmtes Problem mit einem kurzen Programm nicht zu lösen ist. Der Computer muß so viele Befehle erhalten, daß dafür lange Programme nötig sind. Ist solch ein Programm dann auch noch mehrfach verschachtelt, wird es immer schlechter lesbar. Davon einmal abgesehen, ist auch das Schreiben eines langen Programmes nicht einfach, da der Programmierer schnell durcheinander kommen kann und schließlich nicht mehr weiß, wo er eigentlich gerade ist.

Um so ein Problem angehen zu können, greift man zur „Differenzierung" und „Spezialisierung". Oder: Die Arbeit wird verteilt (Differenzierung) und die einzelnen Bearbeiter übernehmen einen bestimmten Bereich (Spezialisierung).

Das kann man also auch beim Programmieren machen. Bestimmten Teilen des Programms werden spezielle Aufgaben zugewiesen, und das Hauptprogramm wird kürzer und besser lesbar. Im Idealfall ist das Hauptprogramm nicht mehr als eine Art Koordinator, der die einzelnen Arbeiten an die entsprechenden Programmteile verteilt.

Subroutinen

In BASIC gibt es zur Ausführung einer bestimmten Aufgabe durch einen bestimmten Programmteil die Subroutine. Das Wort erklärt bereits einiges. „Sub" weist auf eine andere Ebene von Teilaufgaben hin, und „Routine" meint die mehrfache Ausführung einer Aufgabe durch den entsprechenden Programmteil.

Als Beispiel nehmen wir einmal die Errechnung des größten gemeinsamen Teilers zweier Zahlen durch den Computer. Die Aufgaben, die der Computer hintereinander zu bewältigen hat, sind:

- Darstellung der Instruktionen
- Eingabe der Zahlen
- Zahlen der Größe nach ordnen
- Rechnen
- Wiedergabe des Ergebnisses

Alle diese Teilaufgaben scheinen fast zu begrenzt zu sein zur Einrichtung besonderer Verschachtelungen in einem Programm. Aber nur anhand eines kleinen und übersichtlichen Aufgabenpaketes läßt sich die Subroutine überhaupt erläutern.

Wenn die einzelnen Aufgaben in strikter Aufgabentrennung von besonderen Subroutinen erledigt werden, sieht das Programm folgendermaßen aus:

```
10 REM groesster gemeinsamer Teiler
20 GOSUB 200
30 INPUT "Ihre erste Zahl";X
40 INPUT "Ihre zweite Zahl";Y
50 GOSUB 300
60 GOSUB 400
70 GOSUB 500
80 END
200 REM Instruktionen
210 CLS:KEY OFF
220 PRINT "Dieses Programm benutzt die"
230 PRINT "euklidische Methode, um den"
240 PRINT "groessten gemeinsamen Teiler"
250 PRINT "zweier Zahlen zu finden."
260 RETURN
300 REM groesste Zahl in A
310 IF X<Y THEN SWAP X,Y
320 RETURN
400 REM Rechenvorgang
410 A=X:B=Y
420 RE=A MOD B
430 A=B
440 B=RE
450 IF RE<>0 THEN GOTO 420
460 RETURN
500 REM Ausdruck des Ergebnisses
510 CLS : PRINT : PRINT
520 PRINT "    ********************************"
530 PRINT : PRINT "    Der groesste gemeinsame Teiler"
540 PRINT : PRINT "    von";X;"und";Y;"ist";A
550 PRINT "    ********************************"
560 RETURN
```

Der übermäßige Einsatz von Subroutinen kann allerdings auch zu unerwünschten Resultaten führen. Obgleich schon das Hauptprogramm recht kurz ist (Zeilen 10-80), kann man es kaum lesen. Der Leser des Programms wird immer wieder auf die Subroutinen verwiesen, will er erkennen, was das Programm macht. Es ist deshalb ratsam, die REM-Befehle, die die Arbeit der Subroutine verdeutlichen, direkt dem Verweis auf die Subroutine (GOSUB) folgen zu lassen.

Achten Sie auf die besondere Teilung in Zeile 420:

MOD

Es handelt sich hier um eine Teilung ganzer Zahlen mit einem Rest. So ist beispielsweise 17/4 gleich 4.25. 17 MOD 4 jedoch ist gleich 4 (mit Rest 1).

Subroutinen werden mit folgendem Befehl aufgerufen:

GOSUB (gehe zu Subroutine)

Diesem Befehl folgt die Zeilennummer, auf der die Subroutine beginnt. Günstig ist es, wenn dazu hohe Zeilennummern gewählt werden, so daß immer genügend Platz für ein ununterbrochenes Hauptprogramm bleibt. Sorgen Sie dafür, daß die Zeile, auf die verwiesen wird, auch tatsächlich existiert! Der Computer gibt sonst eine Fehlermeldung aus:

Undefined line number (nicht definierte Zeile)

Vorsicht! Sie dürfen bei GOSUB nur Zeilennummern eingeben. Mit einer Berechnung oder einer Variablen können Sie hier nichts anfangen.

Um von der Subroutine wieder ins Hauptprogramm zurückzukommen, befehlen Sie:

RETURN (zurückgehen)

Dieser Befehl bewirkt, daß der Computer zu der Zeile nach dem GOSUB-Befehl zurückkehrt. In Zeile 50 heißt es: GOSUB 300. Der RETURN-Befehl in Zeile 320 bewirkt, daß der Computer in die Zeile zurückkehrt, die auf Zeile 50 folgt, in diesem Fall Zeile 60.

Der Befehl

END (Ende)

in Zeile 80 sorgt dafür, daß der Programmlauf beendet wird. Im anderen Fall begänne der Computer wieder mit Zeile 200. Aber diese Zeilen waren als Subroutine gedacht und nicht als Hauptprogramm. Gelangt der Computer auf diese Weise nun zu Zeile 260, findet er dort ein RETURN, dem kein zugehöriges GOSUB vorausging. Dies führt zu der Fehlermeldung:

RETURN without GOSUB (RETURN ohne GOSUB)

Sorgen Sie also immer dafür, daß die Subroutinen hinter dem Hauptprogramm stehen, und daß der Computer nicht irrtümlich nach Durchlaufen des Hauptprogramms (mit Verschachtelungen) eine Subroutine beginnen kann. Dies erreichen Sie mit der Eingabe von END.

Es ist möglich, eine bestimmte Subroutine verschiedene Male von unterschiedlichen Programmzeilen aus aufzurufen. Sinnvoll ist das vor allem dann, wenn eine bestimmte Aufgabe mehrmals erledigt werden soll. Der Computer merkt sich selbständig, von welcher Zeile aus er zur Subroutine überging und wo er infolgedessen nach Durchlaufen der Subroutine wieder fortfahren muß.

Die wichtigsten Merkmale von Subroutinen sind:

- es handelt sich um besondere Programmteile;
- sie sind mehrmals von unterschiedlichen Stellen aus aufrufbar;
- sie werden mit GOSUB aufgerufen;
- sie werden mit RETURN wieder verlassen;
- Variablen aus dem Hauptprogramm können in der Subroutine untergebracht werden.

Wählen zwischen verschiedenen Subroutinen

Neben der zuvor genannten Verwendung des GOSUB-Befehles kennt der MSX auch den folgenden Befehl:

ON...GOSUB (bei...gehe zu Subroutine...)

Durch diesen Befehl wird der Computer dazu veranlaßt, mit einer Subroutine zu beginnen, die mit einer Zahl hinter ON bezeichnet wird.

Angenommen, Sie haben eine ganze Reihe von Werten. Sie möchten, daß der Computer anhand dieser Werte mit einer bestimmten Subroutine beginnt. Etwa:

```
5 CLS : KEY OFF
10 INPUT "Ganze Zahl (>0, <8)";A
20 PRINT
30 ON A GOSUB 100,150,200,300,400,450,525
40 PRINT
50 GOTO 10
99 REM erste Subroutine
100 PRINT "Diese Subroutine beginnt bei 100"
110 PRINT "Sie haben also die Zahl 1 gewaehlt."
120 RETURN
149 REM zweite Subroutine
150 PRINT "Dies ist Zeile 150"
160 PRINT "Sie haben die Zahl 2 gewaehlt."
170 RETURN
199 dritte Subroutine
200 PRINT "Sie haben 3 gewaehlt."
210 RETURN
299 REM vierte Subroutine
300 PRINT "Sie haben 4 gedrueckt."
310 RETURN
399 REM fuenfte Subroutine
400 PRINT "Die Zahl war 5."
410 RETURN
449 REM sechste Routine
450 PRINT "Diesmal war es eine 6."
460 RETURN
524 REM siebte Subroutine
525 PRINT "7, eine Glueckszahl!"
530 RETURN
```

In Zeile 10 wird um eine ganze Zahl zwischen 1 und 7 gebeten. Zeile 30 bewirkt, daß abhängig vom eingegebenen Wert zu einer Subroutine übergegangen wird, die bei einer der hinter GOSUB genannten Zeilennummer beginnt. Dazu können Sie selber Zeilennummern auswählen. Sie müssen nur beachten, daß die Zeile, auf die verwiesen wird, tatsächlich vorhanden ist, und daß die angesprochene Subroutine dort auch beginnt. Es ist nicht notwendig, die Zeilennummern der Höhe nach zu ordnen. Sie können also auch folgenden Befehl eingeben:

ON A GOSUB 100, 1000, 500, 3450, 200

Wenn Sie als Wert eine 0 eingeben oder eine Zahl, die größer ist als die Anzahl der Zeilennummern hinter GOSUB, geht der Computer einfach zur nächsten Zeile über, und der gesamte ON...GOSUB-Befehl wird weiter nicht beachtet.

Geben Sie dagegen einen negativen Wert ein oder eine Zahl, die höher als 255 ist, erhalten Sie eine Fehlermeldung:

Illegal function call	(unerlaubter Funktionsaufruf)

Der Wert hinter ON darf eine Variable, eine Zahl oder eine Berechnung sein. Wenn das Ergebnis der Berechnung oder der Wert der Variablen keine ganze Zahl ist, werden die Ziffern hinter dem Komma nicht berücksichtigt. Für die Werte kleiner als 0 und größer als 255 gilt das zuvor schon Gesagte.

Die Rückkehr aus einer Subroutine erfolgt mittels eines RETURN-Befehles. Dieser Befehl veranlaßt den Computer, mit der Zeile fortzufahren, die der Zeile mit dem ON...GOSUB-Befehl folgt.

Achten Sie bei diesem Beispielprogramm auf die Verwendung der REM-Befehle. Sie befinden sich alle unmittelbar vor Beginn der Subroutine. Manche Programmierer sind der Ansicht, alle REMs seien aus einem Programm zu entfernen, da sie es nur unnötig länger machen und auch noch Speicherplatz benötigten. Dadurch, daß die REMs direkt vor der Subroutine stehen, wird das Programm durch Entfernen aller REMs nicht beeinflußt. Normalerweise allerdings ist es besser, wenn die REM-Befehle stehen bleiben, da sie den Programmaufbau klarer machen.

Wählen zwischen verschiedenen Verschachtelungen

Den oben beschriebenen Befehl ON finden Sie in den meisten BASIC-Dialekten. Im MSX-BASIC wurde die Anwendbarkeit von ON jedoch weiterentwickelt. Auch bei dem Befehl GOTO kann der ON-Befehl gebraucht werden:

```
10 INPUT ''Tippe 1, 2 oder 3'';A
20 BEEP : ON A GOTO 10, 20, 40
30 PRINT
40 PRINT ''Gott sei Dank, Sie haben 3 gewählt''
```

Dieser Befehl unterscheidet sich nicht wesentlich vom ON...GOSUB-Befehl. Allerdings geht der Computer jetzt nicht zu einer bestimmten Subroutine über, sondern verschachtelt das Programm entsprechend dem eingegebenen Wert. Wenn Sie im Programm oben eine 1 eingeben, geht der Computer zurück nach Zeile 10. Geben Sie eine 3 ein, geht er nach Zeile 40, gibt einen kurzen Text wieder und ist am Ende des Programms. Bei der Eingabe einer 2, geht er nach 20, um dort in eine unendliche Schleife zu geraten. Jedesmal ist A dann 2, so daß der Computer immer wieder nach Zeile 20 geht. Sie können das noch besonders mit einem Ton unterstreichen, der aus einer Reihe rasch hintereinander erklingender Piepstöne besteht.

Mit dem ON...-Befehl kann man den Computer auch kontrollieren lassen, ob eine bestimmte Sache bereits geschehen ist und wenn ja, ihn in eine bestimmte Subroutine schicken.

Beginnen wir mit dem Befehl

ON KEY GOSUB

Mit diesem Befehl können Sie eine Subroutine an die Betätigung einer oder mehrerer Funktionstasten koppeln. Wird eine dieser Tasten während des Programmdurchlaufs gedrückt, geht der Computer zur entsprechenden Subroutine, führt diese aus und kehrt anschließend wieder ins Hauptprogramm zurück. Während der Ausführung der Subroutine, die zu einer bestimmten Funktionstaste gehört, beachtet der Computer vorübergehend die Betätigung der Funktionstaste nicht. Dies dient der Vermeidung ungewollter Wiederholungen. Nur wenn Sie die Taste in der Subroutine noch einmal drücken, wird der Computer diese Subroutine sofort nach dem Durchlauf gleich noch einmal beginnen. Sie können das verhindern, indem Sie den Befehl KEY OFF in die Subroutine aufnehmen, so daß der Computer nicht mehr auf Drücken der Taste reagiert. Wünschen Sie, daß der Computer nach Durchlaufen der Subroutine die Taste nicht länger ignoriert, erteilen Sie hierzu den Befehl KEY ON.

Ein Beispiel für den Einsatz dieses Befehles:

```
10 CLS
20 ON KEY GOSUB 100,200,300
30 KEY (1) ON:KEY (2) ON
40 KEY (3) ON
50 FOR A=1 TO 9999
60 PRINT A
70 NEXT A
80 END
99 REM erste Subroutine
100 BEEP
110 PRINT "Funktionstaste 1"
120 PRINT
130 RETURN
199 REM
200 BEEP:BEEP
210 PRINT "Funktionstaste 2"
220 PRINT
230 RETURN
299 REM
300 BEEP:BEEP:BEEP
310 PRINT "Funktionstaste 3"
320 PRINT
330 RETURN
```

In Zeile 20 wird dem Computer aufgetragen, nach Drücken einer der Funktionstasten 1, 2 oder 3 zu einer Subroutine in Zeile 100, 200 oder 300 zu gehen.

In den Zeilen 30 und 40 werden die Funktionstasten „eingeschaltet". Von diesen Zeilen an beachtet der Computer fortwährend die drei Funktionstasten.

Die Zeilen 50-70 stellen ein normales Programm dar. Die Zahlen 1 bis 9999 werden dargestellt. Aber während des Programmdurchlaufs achtet der Computer die ganze Zeit über auf die Funktionstasten. Drücken Sie eine von ihnen, wird die zugehörige Subroutine ausgeführt, wonach der Computer wieder in das normale Programm zurückkehrt. Beachten Sie, daß die Durchführung der Subroutinen wesentlich langsamer vonstatten geht, wenn Sie die Funktionstasten mehrmals kurz hintereinander drücken, oder wenn Sie die Subroutine zu lang machen.

In vergleichbarer Weise funktioniert

ON STRIG GOSUB

STRIG steht in diesem Befehl für SPACEBAR (Leertaste) und TRIGGER (Auslöseknopf am Joystick). Mit diesem Befehl veranlassen Sie den Computer, das Drücken der Leertaste oder des Auslöseknopfes eines angeschlossenen Joystick zu beachten. Es gibt fünf Schalter, die beachtet werden:

0 = Leertaste
1 = Auslöseknopf 1 von Joystick 1
2 = Auslöseknopf 1 von Joystick 2
3 = Auslöseknopf 2 von Joystick 1
4 = Auslöseknopf 2 von Joystick 2

Wie schon beim ON KEY GOSUB-Befehl werden unerwünschte Wiederholungen dadurch vermieden, daß der Computer während des Durchlaufens der Subroutine nur noch auf die erneute Betätigung der Taste oder des Auslöseknopfes achtet.

Das folgende Programm ist das gleiche wie vorhin, nur ist diesmal die Leertaste aktiviert.

```
10 CLS
20 ON STRIG GOSUB 100
30 STRIG(0) ON
50 FOR A=1 TO 9999
60 PRINT A
70 NEXT A
80 END
99 REM Subroutine
100 BEEP
110 PRINT "Zwischenraum"
120 PRINT
130 RETURN
```

In Zeile 20 wird der Computer darauf hingewiesen, daß er auf die Leertaste zu achten hat. In Zeile 30 wird die Leertaste „eingeschaltet". Beachten Sie, daß sich zwischen dem G und der ersten Klammer keine Leerstelle befinden darf.

Wenn Sie nur den zweiten Auslöseknopf des ersten Joystick aktivieren wollen,

brauchen Sie keine Zeilennummern für die vorangehenden (nicht gebrauchten) Auslöseknöpfe einzusetzen. An deren Stelle setzen Sie lediglich Kommata. Also:

```
ON STRIG GOSUB 100, , 300
```

Auf diese Weise geht der Computer über zu Zeile 100, wenn die Leertaste gedrückt wird, und zu Zeile 300, wenn Sie den Auslöseknopf 2 von Joystick 1 drücken. Achten Sie darauf, daß Sie diesen Auslöseknopf auch aktivieren:

```
STRIG(3) ON
```

Bei der Erstellung von Zeichen in Form von Sprites können Sie einen Befehl verwenden wie:

```
ON SPRITE GOSUB
```

Dieser Befehl wird im Kapitel über Sprites noch behandelt werden.

Bei der Fehlersuche und deren Korrektur hilft Ihnen einen Befehl wie:

```
ON ERROR GOSUB
```

Auf diesen Befehl wird im Anhang über Fehlermeldungen eingegangen.

Ein Befehl, mit dem Sie den Computer dazu veranlassen, nach einer bestimmten Zeit eine Aufgabe auszuführen, lautet:

```
ON INTERVAL GOSUB
```

Hinter dem Wort INTERVAL geben Sie ein Gleichheitszeichen und ein Zeitintervall ein. Dieses Zeitintervall geben Sie in 50stel Sekunden an. Nachdem der Computer den Befehl

```
INTERVAL ON
```

erhalten hat, achtet er auf die Zeit. Ist der angegebene Zeitraum verstrichen, geht er über zu einer Subroutine.

Das folgende Programm kombiniert einen ON KEY- mit einem ON INTERVAL-Befehl.

```
10 CLS
20 KEY OFF
30 ON KEY GOSUB 100
40 KEY (1) ON
50 FOR A=1 TO 9999
60 PRINT A
```

```
70 NEXT A
80 END
99 REM
100 PRINT
110 BEEP
120 PRINT "Funktionstaste gedrueckt"
130 ON INTERVAL=100 GOSUB 500
140 INTERVAL ON
150 RETURN
500 PRINT
510 BEEP:BEEP:BEEP:BEEP
520 PRINT "Zwei Sekunden verstreichen"
530 PRINT
540 INTERVAL OFF
550 RETURN
```

In der Subroutine, die aufgerufen wird, sobald Sie Funktionstaste 1 drücken, wird der Computer dazu aufgefordert, die Zeit weiter zu messen (Zeilen 130 und 140). Unmittelbar darauf kehrt der Computer ins Hauptprogramm zurück, doch die Zeitmessung setzt er fort. Nach 100 50stel Sekunden (zwei Sekunden) beginnt die zweite Subroutine. Diese stellt einen Text dar, läßt einige Töne erklingen und schaltet die Zeitmessung wieder ab.

Der letzte Befehl aus dieser Reihe dient zur Sicherung gegen Eindringen in ein Programm. Jedes Programm ist mit der STOP-Taste und eventuell zusammen mit der CTRL-Taste anzuhalten. Indem der Computer den Befehl erhält, nach Drücken der STOP-Taste in eine Subroutine zu gehen anstatt zu stoppen, ist es dem Benutzer unmöglich, das Programm anzuhalten. Nur mit dem RESET-Knopf kann das geschehen, aber in diesem Falle ist dann auch der Speicher gelöscht. Das folgende Programm müssen Sie zunächst auf Kassette bringen, bevor Sie es laufen lassen, andernfalls müßten Sie es noch einmal eingeben. Das Programm kann nicht gestoppt werden, ohne daß man es vernichtet.

```
10 CLS
20 ON STOP GOSUB 100
30 STOP ON
40 PRINT "Und ich komme NICHT!"
50 GOTO 40
99 REM
100 PRINT
110 BEEP
120 PRINT "Bestimmt nicht, auch nicht bei STOP"
130 PRINT :BEEP
140 RETURN
```

Funktionen

Neben den Subroutinen gibt es die Funktionen. Der wesentliche Unterschied zur Subroutine besteht darin, daß eine Funktion immer ein Ergebnis liefert: eine Zahl oder einen String. Eine Subroutine liefert als Antwort nicht immer ein Ergebnis, son-

dern führt zuweilen nur etwas aus. Eine Subroutine kann einfach eine Reihe von Befehlen hintereinander ausführen. MSX kennt verschiedene Funktionen. So ist

SQR (SQuare Root = eQuadratwurzel)

eine mathematische Funktion, die die Wurzel einer Zahl oder des Wertes einer Variablen oder einer Berechnung errechnet.

Sie können jedoch auch ihre eigenen Funktionen definieren. Nachdem Sie in einem Programm eine Funktion definiert haben, und der Computer die entsprechenden Zeilen hinter sich gebracht hat, kann eine eigene Funktion ebenso benutzt werden wie jede andere Funktion, die bereits in MSX-BASIC vorhanden ist. Das heißt, überall im Programm können Sie diese Funktion einsetzen, um ein bestimmtes Resultat zu erlangen. Etwa:

```
10 DEF FNQUADRAT (A)=A*A
```

Diese Funktion liefert das Quadrat einer Zahl. Ungeachtet der Zahl, die Sie der Funktion angeben, errechnet diese Funktion das Quadrat der genannten Zahl. Versuchen Sie es einmal mit folgendem Programm:

```
10 CLS : KEY OFF
20 DEF FNQUADRAT(A)=A*A
30 INPUT "Bitte eine Zahl:";X
40 PRINT X;"im Quadrat ist";FNQUADRAT(X)
50 GOTO 30
```

In Zeile 20 wird die Funktion definiert. Das geschieht durch Eingabe des Befehls

DEF FN

und anschließender Angabe des Namens der Funktion. Hinter dem Funktionsnamen folgen in Klammern die in der Funktionsdefinition verwandten Variablen. Danach setzen Sie ein Gleichheitszeichen und schließlich die Berechnung ein, die von der Funktion durchgeführt werden soll.

Der Aufruf einer Funktion erfolgt in der gleichen Weise wie schon der Aufruf einer Variablen oder einer Berechnung. Sie geben FN ein sowie den Namen der Funktion. In Klammern setzen Sie die Werte ein, mit denen der Computer zu rechnen hat. Als Wert können Sie sowohl Zahlen als auch Variablen verwenden. Nachdem FNQUADRAT definiert ist, kann sie so aufgerufen werden:

```
PRINT FNQUADRAT(13)
oder
LET A=10:LET B=5
PRINT FNQUADRAT(A)*FNQUADRAT(B)
```

Die Definition einer Funktion darf nicht mehr als eine einzige Programmzeile umfassen.

Beim Aufruf einer Funktion müssen Sie darauf achten, daß Sie so viele Werte mit angeben, wie in der Berechnung gebraucht werden. In dem Beispiel oben wurde immer ein einziger Wert gebraucht und also auch nur ein einziger Wert beim Aufruf der Funktion angegeben. Es ist aber möglich, mehr als nur einen Wert zu nehmen:

```
DEF FNUNTERSCHIED(X, Y)=(X-Y)
```

Diese Funktionsdefinition gibt immer den absoluten Unterschied zwischen zwei Werten an (das heißt, wenn dieser Unterschied negativ ist, wird das Minuszeichen nicht beachtet).

Diese Funktion wird so aufgerufen:

```
PRINT FNUNTERSCHIED(10, 7)
PRINT FNUNTERSCHIED(A, B)
Z=FNUNTERSCHIED(A, B)*FNUNTERSCHIED(P, 6)
R=FNUNTERSCHIED(A, FNUNTERSCHIED(P, Q))
```

Sie sehen, verschiedene Kombinationen sind möglich, und darüber hinaus können Sie sogar innerhalb eines Funktionsaufrufes eine Funktion aufrufen.

Die Variablen, die innerhalb einer Definition verwandt werden, sind sogenannte lokale Variablen. Das heißt, daß der Wert der Variablen innerhalb dieses Vorganges unabhängig ist vom Wert einer eventuell gleichlautenden Variablen im Hauptprogramm.

15. EINFACHE GRAFIK

Die verschiedenen Schirme

Der MSX hat verschiedene Schirme, mit denen gearbeitet werden kann. Sie werden mittels eines einzigen grafischen Chips angesteuert. Dieser Chip kann sogar 16K Speichervermögen steuern, so daß die Zentraleinheit keinen Speicherplatz mit dem Aufbau des Bildschirmes verliert. Dem Programmierer steht somit mehr freier Speicherplatz zur Verfügung

Das sind die einzelnen Schirme:

Modus	Grafik	Text	Farbe	Auflösungsvermögen	Sprites
0	nein	ja	2	40 mal 24 Zeichen	nein
1	nein/ja	ja	2	32 mal 24 Zeichen	ja
2	ja	nein/ja	16	256 mal 192 Punkten	ja
3	ja	nein/ja	16	64 mal 48 Blöcke	ja

Da dieses Schema einiger Erläuterungen bedarf, folgen an dieser Stelle einige Worte zu den Möglichkeiten eines jeden Schirmes.

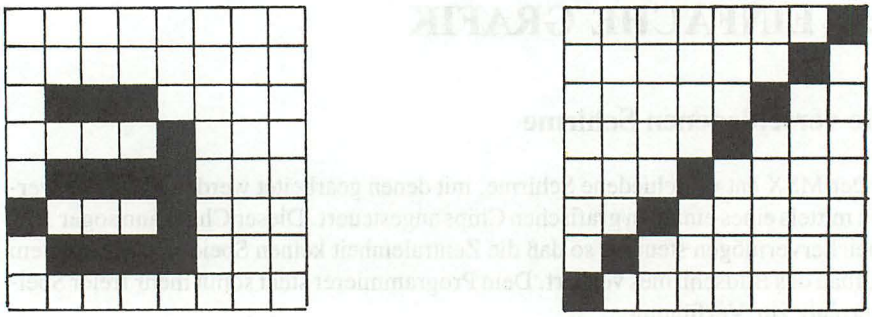
Schirm 0

Schirm 0 ist derjenige Schirm, den Sie vorfinden, wenn Sie den Computer einschalten oder die RESET-Taste gedrückt haben. Es handelt sich hier um den sogenannten „default"-Modus oder Standardmodus. Dieser Bildschirm stellt vierzig Zeichen je Zeile dar. Die Standardeinstellung umfaßt jedoch nur 37 Zeichen je Zeile. Dies hat man so festgelegt, da nicht jedes Fernsehgerät das gesamte vom MSX produzierte Bild wiedergeben kann. Manchmal fallen die Ränder einfach fort. Wollen Sie alle vierzig Zeichen ausnutzen, befehlen Sie

WIDTH (Breite)

Die Listings Ihres Programmes sind in diesem Schirmmodus am besten lesbar. Vor allem für Texte ist dieser Schirm ausgezeichnet geeignet.

Jedes Zeichen wird in einem Raster aus 8 mal 8 Punkten dargestellt. Hier sehen Sie das „a" sowie einen grafischen Schrägstrich in starker Vergrößerung:



Das „a" und der „/" stark vergrößert

Beachten Sie die Ausnutzung des Rasters. Der grafische Schrägstrich verläuft von ganz links nach ganz rechts und von ganz unten nach ganz oben. Beim „a" dagegen ist nach allen Seiten noch viel Platz. Dafür gibt es zwei Gründe:

- Sowohl unter als auch über dem „a" muß noch Spielraum für Buchstaben mit Ober- oder Unterlängen (b, d, f, g, h, j, k, l, p, q, t und y) bleiben, die ja mit dem gleichen Raster dargestellt werden.

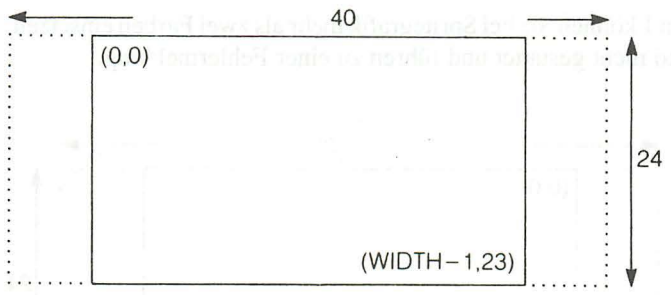
- Im Schirmmodus 0 stehen in der Breite für jedes Zeichen nur 6 der 8 Rasterpunkte zur Verfügung. Da sich die einzelnen Zeichen nicht berühren dürfen, bleibt die dritte Spalte von rechts leer. Auf diese Weise können Raster aus jeweils 6 Punkten nebeneinander stehen, ohne sich zu berühren.

Die grafischen Zeichen, die sowohl im Schirmmodus 0 als auch im Modus 1 gebraucht werden können, beanspruchen ein Raster aus 8 mal 8 Punkten meist völlig. Diese Zeichen werden im Schirmmodus 0 nicht vollständig wiedergegeben.

Im Schirmmodus 0 können Sie keine Sprites verwenden (siehe Kapitel über Sprites). Lediglich zwei der sechzehn verfügbaren Farben können eingesetzt werden; der Benutzer kann diese beiden Farben allerdings selbst bestimmen. Die Standardeinstellung sieht weiße Buchstaben auf blauem Hintergrund vor (da sich so bei einem Farbfernsehgerät das schärfste Bild erzielen läßt). Die Farbe des Randes ist nicht wählbar, sie ist immer identisch mit der des Hintergrundes.

Solange Sie nicht KEY OFF befehlen, werden die Funktionsdefinitionen in der 23. Bildschirmzeile dargestellt. Alle grafischen Befehle sind in diesem Modus zu unterlassen. Versuchen Sie es dennoch, erhalten Sie folgende Fehlermeldung:

Illegal function call (unerlaubter Funktionsaufruf)



Schirm 1

Mit diesem Schirm lassen sich 32 mal 24 Zeichen darstellen. Wie Sie oben bereits gesehen haben, stehen diesen Zeichen alle 8 mal 8 Rasterpunkte zur Verfügung. Die Standardeinstellung des Schirmes sieht eine Breite von 29 Zeichen vor. Sie können dies mit Hilfe des WIDTH-Befehls ändern.

Sie können nur zwei der sechzehn Farben einsetzen, der Rand allerdings darf eine abweichende dritte Farbe erhalten. Dazu befehlen Sie

COLOR

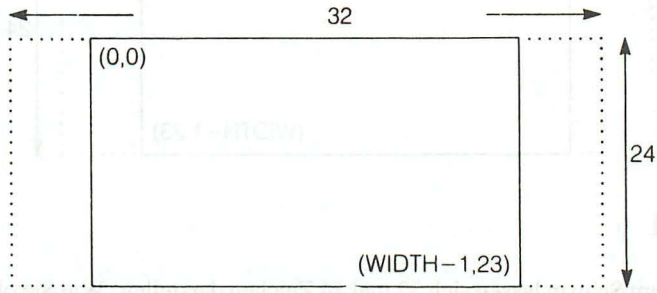
Nach COLOR können Sie drei Zahlen angeben. Die erste Zahl bezeichnet den Vordergrund, die zweite den Hintergrund und die dritte Zahl steht für den Rand. Wenn Sie nur eine oder zwei dieser drei Möglichkeiten auch tatsächlich verwenden wollen, setzen Sie anstelle der anderen Zahlen ein Komma ein.

COLOR 1,7,15	schwarzer Vordergrund, blauer Hintergrund, weißer Rand
COLOR 11,4,2	gelber Vordergrund, blauer Hintergrund, grüner Rand
COLOR ,,6	dunkelroter Rand
COLOR 15	weißer Vordergrund
COLOR 5,,13	hellblauer Vordergrund, magentafarbener Rand
COLOR ,1	schwarzer Hintergrund

Unter den folgenden Farben können Sie wählen:

Code	Farbe	Code	Farbe
0	transparent	8	rot
1	schwarz	9	hellrot
2	grün	10	dunkelrot
3	hellgrün	11	hellgelb
4	dunkelblau	12	dunkelgrün
5	hellblau	13	magenta
6	dunkelrot	14	grau
7	blau	15	weiß

In Schirm 1 können Sie bei Spritegrafik mehr als zwei Farben einsetzen. Grafische Befehle sind nicht gestattet und führen zu einer Fehlermeldung.



Schirm 2

Schirmmodus 2 ist für die Erstellung von Grafiken mit hoher Auflösung bestimmt.

In diesem Schirmmodus ist das Bild nicht in Zeichenpositionen unterteilt, wobei sich an jeder Position ein bestimmtes Zeichen befinden kann, sondern der Schirm setzt sich aus 256 mal 192 Punkten zusammen. Bei jedem Punkt (Pixel oder Dot genannt) können Sie festlegen, ob er ein- oder abgeschaltet ist. Die Farbe eines jeden Punktes ist nicht besonders bestimmbar. Die Farbverteilung ist beim MSX weniger fein als die Verteilung der Punkte. Die Farben setzen sich aus Blöcken mit einer Breite von 8 Punkten und einer Höhe von 1 Punkt zusammen. Auch Zeichenpositionen haben eine Breite von 8 Punkten.

Bei der Herstellung von Zeichnungen in Schirmmodus 2 müssen Sie also gut auf die Farben achten. Pro Block aus 8 mal 1 Punkten können Sie nur eine Vordergrund- und eine Hintergrundfarbe festlegen. Rufen Sie eine zweite Vordergrund- oder eine zweite Hintergrundfarbe auf, nimmt der gesamte Block die neue Farbe an und nicht nur der zuletzt gesetzte Punkt.

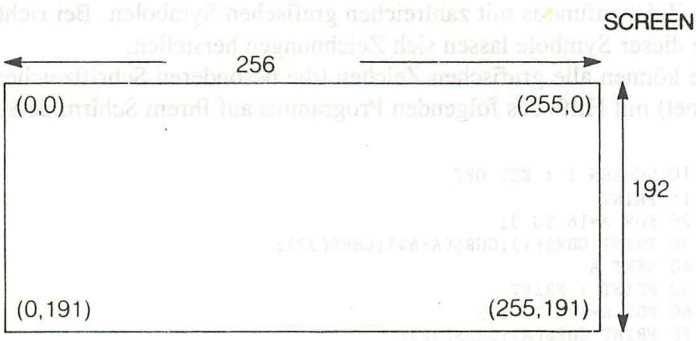
In Kapitel 17 wird genauer erklärt, wie Grafiken in diesem Schirmmodus funktionieren.

Schirm 2 ist ein grafischer Schirm. Das heißt, die Hauptaufgabe ist die Darstellung von Grafiken. Die Textwiedergabe ist nicht so leicht möglich. Die Tastendefinitionen können nicht wiedergegeben werden, und Sie sind auf einen besonderen Trick angewiesen, um auf einem grafischen Schirm zu PRINTen (siehe Kapitel 17).

Der COLOR-Befehl führt im Grafikmodus im Gegensatz zu den Textschirmen nicht gleich zu einer Reaktion. Um den Hintergrund farblich verändern zu können, müssen Sie zunächst einen COLOR-Befehl und anschließend einen CLS-Befehl geben. Einzelne Zeichnungen lassen sich farblich zwar verändern, aber dazu sind grafische Befehle nötig.

INPUT dürfen Sie nicht benutzen. Damit würden Sie den Computer zur Rückkehr zum Standardschirm (Schirm 0) veranlassen und die Zeichnung wäre verschwunden.

Schirm 2 hat wie auch Schirm 3 die Eigenschaft, daß das Bild vom Computer nicht festgehalten werden kann. Am Programmende kehrt der Computer automatisch zum Standardschirm 0 zurück. Wenn Sie das verhindern wollen, müssen Sie am Programmende eine unendliche Schleife einfügen, die bewirkt, daß das Programm weiterläuft und der Computer nicht zu Schirm 0 zurückgeht. In Modus 2 können Sie mit Sprites arbeiten.

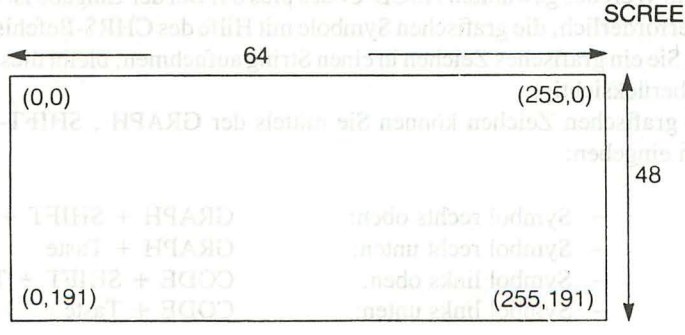


Schirm 3

Dieser grafische Schirm ist weniger fein aufgeteilt als Schirm 2. Ihnen stehen hier 64 mal 48 Blöcke zur Verfügung. Dafür läßt sich bei jedem einzelnen Block die Farbe bestimmen und ob er ein- oder ausgeschaltet ist. Jedem Block können Sie eine eigene Farbe zuweisen. Die Farben werden hier nicht wie in Schirm 2 verlaufen.

Die Bezeichnung der einzelnen Stellen erfolgt bei Schirm 3 in der gleichen Weise wie bei Schirm 2. Sie können so ein und dasselbe Programm auf beiden Schirmen laufen lassen, und es läßt sich rasch herausfinden, wie eine Zeichnung besser darstellbar wird: mit feineren Linien oder mit helleren Farben. Zuweilen kann das sogar zu hübschen „computerisierten" Zeichnungen führen. Stellen Sie die Karte der Niederlande aus dem Kapitel „Dateien bearbeiten" einmal auf Schirm 3 dar.

Schirmmodus 3 funktioniert übrigens in der gleichen Weise wie Schirm 2.



Grafik auf Schirm 1

Obgleich die Schirme 2 und 3 für die Erstellung grafischer Wiedergaben gedacht sind, ist es manchmal sinnvoll, auch auf einem Textschirm Zeichnungen anfertigen zu können. Es kann vorkommen, daß Sie einen Text mit einer Abbildung illustrieren wollen.

Speziell für die Textbildschirme verfügen die MSX-Computer über einen reichhaltigen Zeichenfundus mit zahlreichen grafischen Symbolen. Bei richtiger Verwendung dieser Symbole lassen sich Zeichnungen herstellen.

Sie können alle grafischen Zeichen (die besonderen Schriftzeichen nicht mitgerechnet) mit Hilfe des folgenden Programms auf Ihrem Schirm darstellen lassen:

```
10 SCREEN 1 : KEY OFF
15 PRINT
20 FOR A=16 TO 31
30 PRINT CHR$(1);CHR$(A+64);CHR$(32);
40 NEXT A
50 PRINT : PRINT
60 FOR A=192 TO 215
70 PRINT CHR$(A);CHR$(32);
80 NEXT A
90 PRINT : PRINT
100 FOR A=219 TO 223
110 PRINT CHR$(A);CHR$(32)
120 NEXT A
```

Computerbücher haben den Nachteil, daß sich die Schärfe und Präzision einer Computerzeichnung in den Kapiteln über Grafik nicht auf dem Papier wiedergeben lassen. Um Unklarheiten bei der Eingabe zu vermeiden, werden Sie in diesem Abschnitt in den Programmlistings allen grafischen Zeichen als einem PRINT CHR$(..)-Befehl begegnen. Mit diesem Befehl veranlassen Sie den Computer, dasjenige Symbol darzustellen, das dem in Klammern angegebenen ASCII-Code entspricht. Die ASCII-Codes finden Sie im Anhang. Die grafischen Symbole mit ASCII-Code 1 bis 31 einschließlich verlangen Ihre besondere Aufmerksamkeit. Im offiziellen ASCII-Set sind die normalen Funktionen dieser Codes unter anderem der Steuerung eines Druckers vorbehalten. Um diese Codes dennoch als grafische Zeichen einsetzen zu können, müssen Sie zunächst CHR$(1) eingeben und anschließend CHR$ mit dem Wert des gewählten ASCII-Codes plus 64! Bei der Eingabe ist es nicht unbedingt erforderlich, die grafischen Symbole mit Hilfe des CHR$-Befehls einzutippen. Wenn Sie ein grafisches Zeichen in einen String aufnehmen, bleibt dies vom Computer unberücksichtigt.

Die grafischen Zeichen können Sie mittels der GRAPH-, SHIFT- und CODE-Tasten eingeben:

- Symbol rechts oben: GRAPH + SHIFT + Taste
- Symbol recht unten: GRAPH + Taste
- Symbol links oben: CODE + SHIFT + Taste
- Symbol links unten: CODE + Taste

Bei einigen MSX-Computern sind alle Symbole auf den Tasten dargestellt. Andere haben die Symbole gesondert aufgelistet. Für das Modell SONY HIT BIT ohne Darstellung auf den Tasten gibt es eine spezielle Karte, die in einen Schlitz oberhalb der Tasten gesteckt wird und das rasche Ablesen der Symbole ermöglicht.

Das folgende Programm vermittelt Ihnen eine Vorstellung von einer einfachen Zeichnung auf Schirm 1. Die Zeichnung kann auch auf dem Schirm 0 wiedergegeben werden (versuchen Sie es ruhig einmal). In diesem Fall ist die Zeichnung schmaler, und an der rechten Seite eines jeden grafischen Zeichens fallen zwei Spalten mit Punkten fort.

```
10 SCREEN 1
20 COLOR 4,15,7
30 CLS : KEY OFF
40 B1$="  "+CHR$(206)+"  "
50 B2$="  "+CHR$(1)+CHR$(74)+"  "
60 B3$=" "+CHR$(1)+CHR$(88)+CHR$(219)+CHR$(1)+CHR$(89)+"  "
70 B4$="  "+CHR$(1)+CHR$(74)+"  "
80 B5$="  "+CHR$(219)+"  "
90 B6$=" "+CHR$(1)+CHR$(93)+CHR$(219)+CHR$(1)+CHR$(94)+" "
100 B7$=" "+STRING$(3,CHR$(195))+" "
110 LOCATE 0,13
120 PRINT B1$
130 PRINT B2$
140 PRINT B2$
150 PRINT B3$
160 PRINT B4$
170 PRINT B5$
180 PRINT B6$
190 PRINT B7$
```

Mit den Zeilen 10-30 wird der Schirm freigemacht und die richtigen Farben werden gewählt.

In den Zeilen 40-100 wird die Darstellung der Zeichnung festgelegt. Es handelt sich hier um nichts anderes als um die Summierung von Zeichen und Strings. Jede Reihe der Zeichnung wird in eine besondere Stringvariable gebracht. Es sind dies diejenigen Befehle, die Sie gegebenenfalls bei der Eingabe durch direkt eingegebe Strings ersetzen können, in denen die entsprechenden Symbole enthalten sind. Der ASCII-Liste im Anhang können Sie entnehmen, welche Symbole eingesetzt werden müssen.

Zeile 110 versetzt den Cursor ein Stückchen nach unten.

Die Zeilen 120-190 bewirken, daß die Rakete auf dem Schirm dargestellt wird. Beachten Sie Zeile 140. Hier findet sich der String der zweiten Reihe noch einmal. So wird die Rakete etwas länger. Wollen Sie die Rakete noch länger machen, geben Sie ein:

```
165 PRINT B3$
168 PRINT B4$
```

So können Sie die Länge der Rakete selbst bestimmen.
Der Befehl

LOCATE (plaziere)

sagt dem Computer, daß er den Textkursor an eine bestimmte Stelle auf dem Schirm setzen soll. Die erste Zahl hinter LOCATE legt fest, wie weit der Cursor nach rechts versetzt wird, die zweite Zahl bestimmt, wie weit er nach unten gehen soll.

Lift-off

Um die Rakete abheben zu lassen, brauchen Sie die Darstellung von Bewegung. Hier kann Bewegung auf dem Schirm erreicht werden, indem die Rakete jedesmal neu auf dem Bildschirm dargestellt wird. Die Wiedergabe erfolgt immer ein wenig versetzt zur vorhergehenden Darstellung. Bei geringer Resolution des Bildschirmes heißt das, daß die Rakete immer um ein Zeichen oder um eine Reihe höher dargestellt wird. Versuchen Sie es einmal mit folgendem Programm:

```
10 SCREEN 1
20 COLOR 4,15,7
30 CLS : KEY OFF
40 B1$="  "+CHR$(206)+"  "
50 B2$="  "+CHR$(1)+CHR$(74)+"  "
60 B3$=" "+CHR$(1)+CHR$(88)+CHR$(219)+CHR$(1)+CHR$(89)+"  "
70 B4$="  "+CHR$(1)+CHR$(74)+"  "
80 B5$="  "+CHR$(219)+"  "
90 B6$=" "+CHR$(1)+CHR$(93)+CHR$(219)+CHR$(1)+CHR$(94)+" "
100 B7$=" "+STRING$(3,CHR$(195))+" "
110 LOCATE 0,15
120 PRINT B1$
130 PRINT B2$
140 PRINT B2$
150 PRINT B3$
160 PRINT B4$
170 PRINT B5$
180 PRINT B6$
190 PRINT B7$
200 FOR A=14 TO 1 STEP -1
205 FOR Z=1 TO 50 : NEXT Z
210 LOCATE 0,A
220 PRINT B1$ : PRINT B2$
230 PRINT B2$ : PRINT B3$
240 PRINT B4$ : PRINT B5$
250 PRINT B6$ : PRINT "    "
260 NEXT A
270 GOTO 270
```

Bis Zeile 190 stimmt dieses Programm weitgehend mit dem vorhergehenden überein. Nur die Stelle, an der die Rakete zum erstenmal dargestellt wird, liegt ein wenig tiefer. Auf diese Weise kann sie länger fliegen.

In Zeile 200 wird eine Schleife eröffnet, die die Rakete verschiedene Male zeichnet. Die Schleife in Zeile 205 dient einzig der Verzögerung.

In Zeile 210 wird der Textkursor an eine bestimmte Position auf dem Bildschirm versetzt. Dadurch, daß diese Position von A abhängig ist, wandert sie mit jedem Schleifendurchlauf um eine Stelle nach oben. Die Zeilen 220-250 bewirken, daß die Rakete immer wieder neu dargestellt wird. Beachten Sie den zweiten Teil von Zeile 250. Hier werden einige Leerstellen eingeschoben. Auf diese Weise wird die Rakete von unten her gelöscht. Würden Sie diesen Teil der Zeile auslassen, erhielten Sie eine sehr lange Rakete. Probieren Sie das ruhig einmal aus, indem Sie vorübergehend den zweiten Teil der Zeile 250 entfernen. Die Rakete ähnelt so eher einem Weihnachtsbaum.

In Zeile 270 geht das Programm in eine unendliche Schleife über. Obgleich Sie in den vorhergehenden Abschnitten gelesen haben, daß dies nur bei den grafischen Schirmen notwendig ist, ist dieser Trick auch hier angebracht. Ein Fortlassen dieser Zeile würde bewirken, daß der Cursor und der Text Ok direkt unter der Rakete erscheinen würden.

Um die Rakete noch ein bißchen echter aussehen zu lassen, wurde sie im nächsten Programm mit einem Feuerschweif und Geräuschen ausgestattet. Die Rakete bleibt so lange am Boden, bis Sie das „S" für Start drücken.

```
10 SCREEN 1
20 COLOR 4,15,7
30 CLS : KEY OFF
40 B1$="  "+CHR$(206)+"  "
50 B2$="  "+CHR$(1)+CHR$(74)+"  "
60 B3$=" "+CHR$(1)+CHR$(88)+CHR$(219)+CHR$(1)+CHR$(89)+"  "
70 B4$="  "+CHR$(1)+CHR$(74)+"  "
80 B5$="  "+CHR$(219)+"  "
90 B6$=" "+CHR$(1)+CHR$(93)+CHR$(219)+CHR$(1)+CHR$(94)+" "
100 B7$=" "+STRING$(3,CHR$(195))+" "
110 LOCATE 0,15
120 PRINT B1$
130 PRINT B2$
140 PRINT B2$
150 PRINT B3$
160 PRINT B4$
170 PRINT B5$
180 PRINT B6$
190 PRINT B7$
200 T$=INKEY$ : IF T$<>"s" THEN 200
205 FOR Z=1 TO 50 : NEXT Z
210 LOCATE 2,21: PRINT CHR$(1);CHR$(79)
220 FOR A=14 TO 2 STEP -1
230 GOSUB 290
240 NEXT A
250 LOCATE 0,22 : PRINT"    "
260 GOSUB 290
270 GOTO 260
280 REM Subroutine
290 FOR Z=1 TO 50 : NEXT Z
300 LOCATE 2,A+7 : PRINT CHR$(1);CHR$(79)
310 LOCATE 0,A
320 PRINT B1$ : PRINT B2$
330 PRINT B2$ : PRINT B3$
```

```
340 PRINT B4$ : PRINT B5$
350 PRINT B6$ : PRINT "    "
360 RETURN
```

Achten Sie auf die Befehle in Zeile 200. Mit dem Befehl

INKEY$

veranlassen Sie den Computer, auf die Tastatur zu achten. Immer wenn der Computer diesem Befehl begegnet, kontrolliert er, welche Taste gerade gedrückt ist. Wenn im zweiten Teil der Zeile eine Schleife hin zum Zeilenanfang folgt, wartet der Computer in dieser Zeile, bis ein bestimmtes Zeichen eingegeben wird. In diesem Falle führt der Computer Zeile 200 solange aus, bis „S" eingegeben wird.

Streichhölzer

Das nun folgende Programm zeigt, daß auch mit geringer Resolution hübsche Grafikprogramme möglich sind. Das Spiel selber ist ganz einfach. In der richtigen „Verpackung" allerdings wird es für den Spieler interessant.

```
10 REM Streichholzspiel
20 SCREEN 1 : KEY OFF
30 COLOR 1,5,5
40 PRINT : PRINT : PRINT
50 PRINT STRING$(25,210)
60 PRINT : PRINT
70 PRINT "Das 23-Streichhoelzer-Spiel"
80 PRINT : PRINT "Sie duerfen jedesmal 1,2 "
90 PRINT "oder 3 Streichhoelzer weg-"
100 PRINT "nehmen. Ich mache das gleiche."
110 PRINT "Der Spieler, der das letzte Streich-"
120 PRINT "holz wegnehmen muss, "
130 PRINT "hat verloren."
140 PRINT : PRINT
150 PRINT STRING$(25,210)
160 PRINT : PRINT
170 PRINT "Willkuerlich gewaehlt, darf/duerfen....."
180 LU 23
190 FOR A=1 TO 10 : BEEP
200 FOR B=1 TO 300 : NEXT B : NEXT A
210 IF RND(1)<.6 THEN GOTO 300:REM Spieler beginnt
220 SCREEN 0 : COLOR 6,11
230 PRINT : PRINT
240 PRINT "ich anfangen."
250 PRINT "Ich nehme 2 Streichhoelzer."
260 LU=21 : GOSUB 1000 : REM Wiedergabe Streichh.
270 GOTO 500 : REM Spiel-Schleife
280 REM Spieler
300 SCREEN 0 : COLOR 6,11
310 PRINT : PRINT
320 PRINT "Sie anfangen."
330 LU=23 : GOSUB 1000 : REM Wiedergabe Streichh.
340 REM Spiel-Schleife
```

```
500 PRINT : PRINT : PRINT "Wieviele Streichhoelzer nehmen Sie?"
510 INPUT "1,2 oder 3";SP
520 IF SP=1 OR SP=2 OR SP=3 THEN LU=LU-SP ELSE PRINT "Sie
    spielen falsch!!":GOTO 500
530 CLS : GOSUB 1000 : REM Wiedergabe Streichh.
540 PRINT : PRINT : PRINT "Ich denke ..."
550 FOR Z=1 TO 20 : BEEP : FOR Q=1 TO 20 NEXT Q : NEXT Z
560 IF LU=4 THEN MS=3 : GOSUB 2000 : GOTO 500
570 IF LU=3 THEN MS=1 : GOSUB 2000 : GOTO 500
580 IF LU=2 THEN MS=1 : GOSUB 2000 : GOTO 500
590 IF LU=1 THEN GOTO 700 : REM gewonnen
600 IF LU<=0 THEN GOTO 900 : REM verloren
610 IF LU>4 THEN MS=4-SP : GOSUB 2000 : GOTO 500
620 PRINT "Fehler im Programmlauf"
630 END
699 REM Spieler gewonnen
700 SCREEN 1 : COLOR 11,6,7
710 LOCATE 2,2
720 FOR LR=1 TO 26
730 PRINT CHR$(1);CHR$(79);
740 NEXT LR
750 FOR BB=3 TO 23
760 LOCATE 2,BB
770 FOR LR=1 TO 26
780 PRINT CHR$(1);CHR$(86);
790 NEXT LR
800 COLOR INT(RND(1)*15),INT(RND(1)*15),INT(RND(1)*15)
810 BEEP:NEXT BB
820 COLOR 12,15,3
830 LOCATE 6,12 : PRINT "SIE HABEN GEWONNEN"
840 END
899 REM Spieler verloren
900 SCREEN 1 : COLOR 15,1,1
910 LOCATE 6,12 : PRINT "SIE HABEN VERLOREN"
920 END
999 REM Wiedergabe Streichh.
1000 PRINT
1010 PRINT "Noch ";LU;"Streichholz/hoelzer vorhanden."
1020 LOCATE 6,6
1030 FOR LR=1 TO LU
1040 PRINT CHR$(1);CHR$(79);
1050 NEXT LR
1060 FOR BB=7 TO 14
1070 LOCATE 6,BB
1080 FOR LR=1 TO LU
1090 PRINT CHR$(1);CHR$(86);
1100 NEXT LR : NEXT BB
1110 RETURN
2000 CLS
2010 PRINT "Ich nehme ";MS;" Streichhoelzer weg."
2020 LU=LU-MS
2030 GOSUB 1000:REM Wiedergabe Streichh.
2040 GOTO 500:SPIEL-SCHLEIFE
2050 RETURN
```

Die Zeilen 10-160 erstellen eine Einführungsseite. Dem Spieler wird gezeigt, um was für ein Spiel es sich handelt, und wie die Spielregeln aussehen. Achten Sie auf die Verwendung des STRING$-Befehls in den Zeilen 50 und 150. Sie bewirken die Wiederholung eines bestimmten Symboles, so daß Zierränder dargestellt werden.

Die Zeilen 170-200 simulieren das Nachdenken des Computers. In Wahrheit „denkt" der Computer natürlich wesentlich schneller. Durch die absichtliche Verzögerung erhält der Spieler die Möglichkeit, die Spielregeln zu lesen. Eine Methode, das zu erreichen, haben Sie im letzten Programm gesehen: Der Spieler muß eine Taste drücken, wenn er alles gelesen hat. Zeile 190 bewirkt zehn Piepser, und durch Zeile 200 wird eine Pause zwischen den einzelnen Tönen gemacht.

Mit Zeile 210 wird erreicht, daß nicht immer derselbe anfängt. Durch Eingabe einer beliebigen Zahl mit dem RND-Befehl erhält der Computer einen Wert zwischen 0 und 0.99999999999999 (wenn der Wert hinter RND positiv ist). Durch Vergleich dieses Wertes mit .5 wird der Computer den Vergleich durchschnittlich jedes zweite Mal als wahr ansehen und in Zeile 300 gehen. In diesem Falle beginnt der Spieler. Geht der Computer jedoch in Zeile 220, fängt der Computer an.

Die Zeilen 220-250 zeigen an, daß der Computer beginnt, und zwar indem er 2 Streichhölzer entfernt.

Zeile 260 gibt die neue Zahl der Streichhölzer an (23-2=21) und führt zur Subroutine in Zeile 2000. Diese Subroutine bewirkt jedesmal die Angabe der korrekten Anzahl der Streichhölzer.

Zeile 270 läßt den Computer den Programmteil überspringen, in dem es darum geht, daß der Spieler anfängt. In Zeile 500 beginnt diejenige Schleife, die den Rest des Spieles steuert.

Die Zeilen 300-330 stimmen mit den Zeilen 220-250 überein, doch fängt in diesem Fall der Spieler an.

Auf Zeile 500 beginnt die Schleife. Hierhin wird der Computer so lange zurückkehren, bis das Spiel beendet ist.

Die Zeilen 500-520 bitten den Spieler um die Angabe, wieviele Streichhölzer er fortnehmen möchte. Beachten Sie Zeile 520, die Falschspiel verhindert.

Zeile 530 bewirkt wieder die Darstellung der Streichhölzer (Subroutine in Zeile 1000).

Die Zeilen 540-550 bewirken eine Verlangsamung und erneut die Simulation eines „denkenden" Computers.

Mit den Zeilen 560-610 wird festgelegt, wieviele Streichhölzer der Computer jedesmal wegnimmt.

Um die Zeilen nicht zu lang werden zu lassen, ist der Teil, der immer der gleiche bleibt, in einer Subroutine untergebracht, die in Zeile 2000 anfängt. Wenn Zeile 610 erreicht wird, befindet sich das Spiel im letzten Stadium und der Computer bedient sich einer einfachen Subtraktion, um auszurechnen, wieviele Streichhölzer entfernt werden müssen.

Zeile 620 stellt eine besondere Zeile dar, die nur im Teststadium eines Programms gebraucht wird. Wenn das Programm gut läuft, können Sie diese Zeile löschen. Sie sagt, daß der Computer die Zeile niemals erreichen darf. Erreicht er sie dennoch, erscheint auf dem Bildschirm die Meldung, daß sich irgendwo im Programm ein Fehler befinden muß. Haben Sie ein Programm erstellt, das nicht so recht laufen will, fügen Sie an verschiedenen Stellen, von denen Sie sicher wissen, daß der Computer sie nicht erreichen darf, diese Zeile ein. Darüber hinaus können Sie jeden PRINT-Befehl mit einer Zeilenangabe versehen. Das vereinfacht die Fehlersuche erheblich.

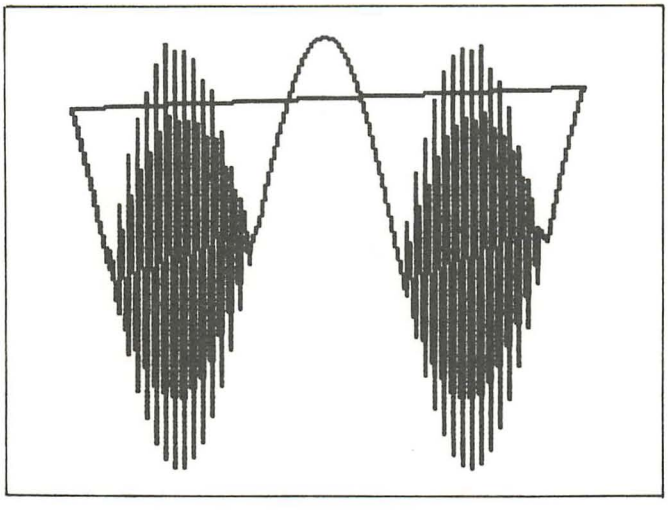
Die Zeilen 700-840 bewirken einen blinkenden und bunten Bildschirm sowie die Mitteilung, daß der Spieler gewonnen hat.

Durch die Zeilen 900-920 erscheint die einfache Feststellung, daß der Spieler der Verlierer ist.

Wenn Sie es dem Spieler ermöglichen wollen, noch einmal mit dem Spiel anzufangen, ohne RUN eingeben zu müssen, können Sie zwischen den Zeilen 920 und 999 einen INPUT-Befehl unterbringen, der vom Benutzer wissen will, ob er noch einmal spielen möchte. Der Antwort entsprechend wird das Programm entweder beendet, oder es fängt noch einmal von vorne an. Wünschen Sie ein Programm, dessen Listing dem Benutzer nicht so ohne weiteres zugänglich ist (schließlich läßt sich daran die „Taktik" des Computers erkennen), bedienen Sie sich des ON STOP-Befehls aus dem vorigen Kapitel. Will der Spieler das Spiel beenden, muß das Programm nicht richtig gestoppt werden, sondern in eine unendliche Schleife führen. Andernfalls kann der Benutzer das Listing doch noch sehen.

Die Zeilen 1000-1110 stellen die Subroutine dar, die die Wiedergabe der Streichhölzer bewirkt. Zunächst wird die Anzahl der Streichhölzer in Buchstaben und Ziffern mitgeteilt (Zeile 1010), und anschließend kommt noch eine Zeichnung der korrekten Anzahl der Streichholzköpfe (Zeilen 1030-1050). Zum Schluß werden einige Male so viele Striche gezogen, wie es Streichhölzer gibt. Die Streichhölzer werden gezeichnet, indem man die Striche untereinander stellt. (Zeile 1060-1100).

Die Zeilen 2000-2050 bewirken die Berechnungen des Computers sowie die Wiedergabe seiner Entscheidung.



16. GRAFIK UND FARBE

Einleitung

Im vorhergehenden Kapitel haben Sie gelesen, daß Schirm 3 vor allem zur Wiedergabe von Farben geeignet ist. Darüber hinaus hat Schirm 3 noch den „Vorteil", daß es bei ihm nur Blöcke gibt und keine Punkte. Daher sieht eine auf Schirm 3 gezogene Linie aus wie eine Reihe ziemlich grober Blöcke. Einerseits ist dies ein Nachteil, da so auf Schirm 3 keine feinen Striche gezogen werden können; andererseits lassen sich damit hübsche Effekte erreichen.

Um ein Männchen zum zeichnen, müssen Sie nur eine Reihe von Strichen in der richtigen Farbe ziehen. Der Computer füllt die Blöcke aus, wodurch das Männchen seine Form erhält. Versuchen Sie es einmal mit dem folgenden Programm:

```
10 SCREEN 3 : COLOR5,15,7
20 CLS
30 LINE (48,4)-(44,18),9
40 LINE (44,4)-(44,18),9
50 LINE (44,18)-(36,46),12
60 LINE (36,46)-(28,58),4
70 LINE (28,58)-(10,70),4
80 LINE (10,70)-(10,76),1
90 LINE (36,46)-(52,44),4
100 LINE (52,44)-(60,62),4
110 LINE (60,62)-(66,62),1
120 LINE (44,18)-(26,26),11
130 LINE (26,26)-(26,34),11
140 LINE (44,18)-(50,34),11
150 LINE (50,34)-(60,34),11
160 LINE (60,34)-(64,34),11
170 GOTO 170
```

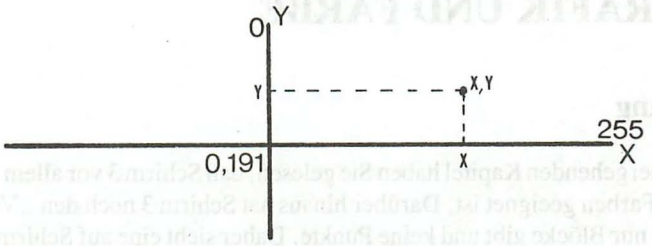
In den Zeilen 30-160 taucht immer der Befehl

LINE

auf. Dieser Befehl bewirkt, daß der Computer von einem Punkt zum anderen einen Strich zieht.

Bestimmung eines Punktes

Die Punkte werden in einem rechtwinkligen Koordinatensystem angegeben. Von links nach rechts verläuft die X-Achse, sie reicht von 0 bis 255. Die Y-Achse verläuft senkrecht dazu von 0 bis 191.



Am Beispiel dieses Achsensystems sehen Sie, wie ein bestimmter Punkt mit Hilfe von zwei Zahlen genau bestimmt werden kann. Die erste Zahl gibt immer einen Punkt auf der X-Achse (von links nach rechts) an, und die zweite Zahl einen Punkt auf der Y-Achse (von oben nach unten). Mathematisch vorgebildete Leser werden hier noch einmal besonders gewarnt: die Y-Achse verläuft VON OBEN NACH UNTEN!

Für den LINE-Befehl benötigen Sie zwei Zahlenpaare: ein Zahlenpaar für den Anfangspunkt der zu ziehenden Linie und ein Zahlenpaar für den Endpunkt dieser Linie. Beide Zahlenpaare stehen in Klammern und zwischen beiden Zahlenpaaren wird ein Gedankenstrich eingegeben.

Man kann das erste Zahlenpaar fortlassen (aber nicht den Gedankenstrich vergessen!). In diesem Falle wird die Linie nun von der augenblicklichen Position des grafischen Cursors hin zum zweiten Punkt verlaufen.

Der grafische Cursor ist mit dem Textcursor vergleichbar, mit dem Unterschied, daß der grafische Cursor nicht zu sehen ist. Sie können allerdings die Position des Cursors beeinflussen. Der Cursor befindet sich immer an der Stelle, an die ihn die letzte grafische Anweisung bewegt hat. Es ist möglich, den grafischen Cursor durch Setzen eines einzigen Punktes an jede beliebige Stelle zu bewegen:

PSET (,) (Point SETt = gebe Punkt an)

Hinter PSET setzen Sie in der Klammer zwei Zahlen ein; diese Zahlen geben die Koordinaten der darzustellenden Zahl an. Als dritte Zahl können Sie hinter der Klammer noch eine Angabe für die Farbwahl machen. Ohne diese dritte Zahlenangabe wird der Punkt in der augenblicklichen Vordergrundfarbe dargestellt. Bei Angabe eines Farbcodes erscheint der Punkt in der von Ihnen gewünschten Farbe.

Ein ähnlicher Befehl sieht so aus:

PRESET (,) (Point RESET = gebe Punkt neu an)

Mit diesem Befehl nimmt ein Punkt die Farbe des Hintergrundes an mit dem Ergebnis, daß der Punkt nicht sichtbar ist. Wenn als dritte Zahl ein Farbcode eingegeben wird, erscheint der Punkt nicht in der Hintergrundfarbe, sondern in der dem Code entsprechenden Farbe. In diesem Falle ist der Befehl der gleiche wie PSET mit Farbcode.

Es gibt keinen Unterschied zwischen den Koordinatensystemen von Schirm 2 und Schirm 3. Das folgende Programm können Sie sowohl auf Schirm 2 als auch auf Schirm 3 anwenden. Versuchen Sie es ruhig mit beiden Schirmen.

```
10 SCREEN 2 : COLOR ,15,14
20 FOR A=1 TO 300
30 X=INT(RND(1)*255)
40 Y=INT(RND(1)*192)
50 C=INT(RND(1)*15)
60 PSET (X,Y),C
70 NEXT A
80 GOTO 80
```

Dieses Programm setzt dreihundertmal einen Punkt, und zwar immer an einer willkürlichen Stelle. Die Zeilen 30 und 40 bewirken eine willkürliche X-Koordinate und eine willkürliche Y-Koordinate. Mit Zeile 50 wird eine willkürliche Farbe erreicht.

Wenn Sie das Programm, wie es hier steht, laufen lassen, erhalten Sie eine Vielzahl kleiner farbiger Punkte. Das sind die Lichtpunkte des Schirms 2 mit hohem Auflösungsvermögen.

Lassen Sie das Programm dagegen auf Schirm 3 laufen, erhalten Sie farbige Blöcke. Jeder dieser Blöcke setzt sich aus 4 mal 4 = 16 Punkten zusammen.

Probieren Sie einmal, auf Schirm 2 wesentlich mehr als 300 Punkte setzen zu lassen. Verändern Sie in Zeile 20 die Maximumanzahl der Schleife in 19000. Das Programm dauert nun ein wenig länger, aber das Resultat ist sehr aufschlußreich: Die Farbwiedergabe im Schirmmodus 2 ist nicht perfekt.

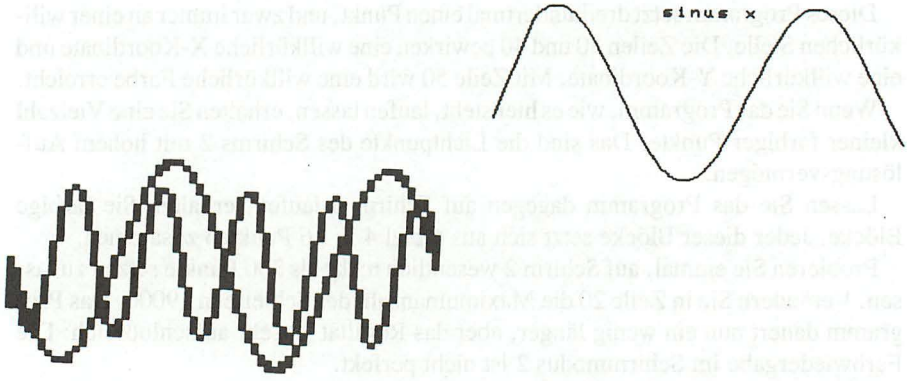
Das nächste Programm zeigt Ihnen, daß auch im Schirmmodus 3 sehr schöne und manchmal auch nützliche Grafiken angefertigt werden können. Dieses Programm zeichnet eine Grafik aus zwei Sinusfunktionen und einer Kosinusfunktion, die übereinander liegen. Da jede Grafik ihre eigene Farbe hat, sind die einzelnen Funktionsgrafiken gut zu erkennen.

```
10 SCREEN 2 : COLOR ,15,15
20 PSET (0,100),6
30 FOR X=0 TO 12.5 STEP .1
40 LINE -(X*20,SIN(X)*70+100),6
50 NEXT X
60 PSET (0,100),5
70 FOR X=0 TO 12.5 STEP.1
80 LINE -(X*20,SIN(X*2.5)*50+100),5
90 NEXT X
100 PSET (0,100),10
110 FOR X=0 TO 12.5 STEP .1
120 LINE -(X*20,COS(X)*70+100),10
130 NEXT X
140 GOTO 140
```

Die Grafiken werden mit den Zeilen 30-50, 70-90 und 110-130 gezeichnet. Die PSET-Befehle im Programm dienen zur Versetzung des grafischen Cursors immer wieder hin zum Anfang der Grafik. Der Computer zeichnet aufgrund des Befehles:

LINE (,) – (,),

Den Anfangspunkt der Linie setzen Sie in der ersten Klammer ein. Die Koordinaten des Endpunktes gehören in die zweite Klammer. Hinter beiden Klammern können Sie gegebenenfalls noch die Farbe der Linie angeben. Nennen Sie keine Farbe, zeichnet der Computer diese Linie in der augenblicklichen Vordergrundfarbe. Wenn Sie die Klammer mit dem ersten Koordinatenpaar fortlassen, wird die Linie von der augenblicklichen Cursorposition hin zum angegebenen Punkt gezogen. Das kommt in diesem Programm mehrmals vor, weil auf diese Weise viel Rechenarbeit entfällt und das Programm schneller läuft.



Wollen Sie das Programm rascher laufen lassen, können Sie in den Zeilen 30, 70 und 110 die Schrittgröße der Schleife auf .2 oder .3 erhöhen. Mit Anwachsen der Schrittgröße nimmt auch die Ungenauigkeit der Grafik zu; das ist bei Schrittgröße .2 aber sicher noch nicht zu erkennen.

Beachten Sie, auf welche Weise die normale Sinusberechnung angeglichen wird, damit eine ordentliche Wiedergabe der Grafik gewährleistet ist. Bei einer unbehandelten Grafik hätte die Y-Koordinate immer den Wert von SINus (X). Der Wert dieser Sinusfunktion liegt jedoch immer zwischen –1 und 1. Das ist ein Unterschied von zwei. Wenn wir versuchen, das direkt auf den Schirm zu übertragen, erhalten wir eine gerade Linie. Die Blöcke auf Schirm 3 umfassen 4 Koordinatenpunkte von links nach rechts und gleichfalls 4 von oben nach unten. Die Sinusfunktion bewegt sich also weniger als einen Block auf und ab. Deshalb vergrößern wir den Abstand zwischen Minimum und Maximum des Y-Wertes mit Hilfe eines Vergrößerungsfaktors. Im Programm oben wurde daraus: SIN(X)*70, SIN(X*2.5)*50 und COS(X)*70. Die SIN(X)- und COS(X)-Funktionen wurden gleich viel vergrößert; die SIN(X*2.5)-Funktion dagegen wurde weniger stark vergrößert. In gleicher Weise erfolgte die Vergrößerung der Grafik in der X-Richtung. Bei allen drei Grafiken war das X*20. Am Ende ist also die Grafik in der Y-Richtung mehr vergrößert als in der X-Richtung. Die Grafiken sind also ein wenig deformiert.

Mit einer letzten Angleichung wird die gesamte Grafik ein Stückchen nach unten verschoben; schließlich haben ja sowohl die Sinus- als auch die Kosinusgrafik auch negative Ergebnisse. Diese würden andernfalls oben am Schirm wegfallen. Passen Sie auf! Die Y-Achse verläuft von oben nach unten und nicht von unten nach oben. Die Grafiken stehen auf dem Kopf! Die Verschiebung erfolgt jeweils durch Addieren von 100 zur Y-Koordinate. Sehen Sie sich das in den Zeilen 40, 80 und 120 an.

Das nun folgende Programm bedient sich gleichfalls des Verschiebens und Vergrößerns von Zeichnungen. In diesem Programm wird das Männchen aus dem ersten Programm in diesem Kapitel noch einmal gezeichnet. Zahlreiche kleine Koordinaten lassen es allerdings diesmal realistischer erscheinen. Durch Multiplikation dieser Koordinaten mit einem bestimmten Faktor kann das Männchen größer oder kleiner gemacht werden, und durch Einfügen eines Verschiebungsfaktors ist es möglich, das Männchen auf dem Bildschirm hin und her zu bewegen. Geben Sie zunächst einmal das Programm ein:

```
10 SCREEN 3 : COLOR 5,15,7
20 CLS
30 DIM PA(13) : DIM XA(13) : DIM YA(13) : DIM KA(13)
40 DIM PB(13) : DIM XB(13) : DIM YB(13) : DIM KB(13)
50 DIM PC(13) : DIM XC(13) : DIM YC(13) : DIM KC(13)
60 DIM PD(13) : DIM XD(13) : DIM YD(13) : DIM KD(13)
70 FOR Z=1 TO 13
80 READ PA(Z),XA(Z),YA(Z),KA(Z)
90 NEXT Z
100 FOR Z=1 TO 13
110 READ PB(Z),XB(Z),YB(Z),KB(Z)
120 NEXT Z
130 FOR Z=1 TO 13
140 READ PC(Z),XC(Z),YC(Z),KC(Z)
150 NEXT Z
160 FOR Z=1 TO 13
170 READ PD(Z),XD(Z),YD(Z),KD(Z)
175 NEXT Z
180 A=4 :B=4 : Y=90
185 FOR X=20 TO 230 STEP 30
190 GOSUB 1190
240 FOR L=1 TO 13
260 IF PA(L)=0 THEN PSET(XA(L)*A+X,YA(L)*B+Y),KA(L)
270 LINE -(XA(L)*A+X,YA(L)*B+Y),KA(L)
280 NEXT L
285 FOR W=1 TO 100 : NEXT W : CLS
290 GOSUB 1190
340 FOR L=1 TO 13
360 IF PB(L)=0 THEN PSET(XB(L)*A+X,YB(L)*B+Y),KB(L)
370 LINE -(XB(L)*A+X,YB(L)*B+Y),KB(L)
380 NEXT L
385 FOR W=1 TO 100 : NEXT W : CLS
390 GOSUB 1190
440 FOR L=1 TO 13
460 IF PC(L)=0 THEN PSET(XC(L)*A+X,YC(L)*B+Y),KC(L)
470 LINE -(XC(L)*A+X,YC(L)*B+Y),KC(L)
480 NEXT L
485 FOR W=1 TO 100 : NEXT W : CLS
490 GOSUB 1190
```

```
540 FOR L=1 TO 13
560 IF PD(L)=0 THEN PSET(XD(L)*A+X,YD(L)*B+Y),KD(L)
570 LINE -(XD(L)*A+X,YD(L)*B+Y),KD(L)
580 NEXT L
585 FOR W=1 TO 100 : NEXT W : CLS
590 NEXT X
890 END
900 REM
910 DATA 0,-3,-3,11,1,-3,-5,11,1,2,-5,11,1,3,-3,11,1,6,-3,11,1,7,-4,11
920 DATA 0,-7,7,1,1,-7,5,1,1,-2,3,4,1,0,0,4,1,4,0,4,1,5,5,4,1,7,5,1
930 DATA 0,2,-4,11,1,-1,-5,11,,2,-5,11,1,3,-2,11,1,6,-2,1,1,6,-3,11
940 DATA 0,-7,1,1,1,-5,1,1,1,-2,4,4,1,0,0,4,1,4,2,4,1,4,7,4,1,6,7,1
950 DATA 0,1,-1,11,1,-2,-2,11,1,2,-5,11,1,1,-2,11,1,3,-2,11,1,4,-2,11
960 DATA 0,-5,3,1,1,-4,0,1,1,0,3,4,1,0,0,4,1,4,3,4,1,-1,6,4,1,1,7,1
970 DATA 0,0,-1,11,1,-2,-4,11,1,2,-5,11,1,1,-3,11,1,3,-2,11,1,4,-2,11
980 DATA 0,-1,7,1,1,-3,7,1,1,1,4,4,1,0,0,4,1,7,1,4,1,2,4,,1,4,5,1
1180 REM
1190 PSET (0+X,0+Y),12
1200 LINE -(2*A+X,-6*B+Y),12
1210 LINE -(3*A+X,-9*B+Y),9
1220 LINE (2*A+X,-6*B+Y)-(2*A+X,-9*B+Y),9
1230 RETURN
```

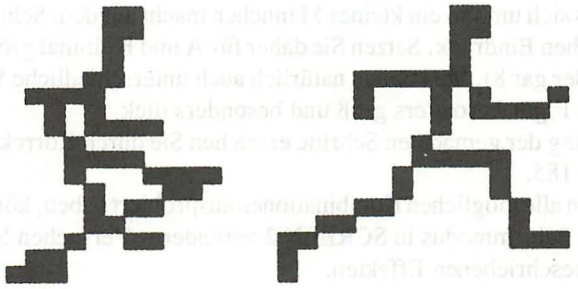
Die Zeilen 10 und 20 dienen der üblichen Gestaltung des Schirmes.

In den Zeilen 30-60 handelt es sich um die Dimensionierung einer Reihe von Arrays, in denen die Daten aus der Datenliste untergebracht werden. Der Computer kann sich Daten aus einem Array am raschesten holen, und da bei Zeichnungen Schnelligkeit recht wichtig ist, werden die Daten zunächst in einem Array untergebracht.

Es werden vier verschiedene Bewegungsphasen eines Männchens gezeichnet, wobei für jede Zeichnung dreizehn Zeichenpunkte benötigt werden. Zu jedem Punkt muß angegeben werden, ob eine Linie dorthin gezogen werden muß (erste Angabe = 1) oder ob der grafische Cursor zu einem Punkte hinbewegt werden soll, ohne eine Linie zu ziehen (erste Angabe =0). Weiterhin müssen sowohl X- als auch Y-Koordinate eines jeden Punktes bekannt sein (zweite und dritte Angabe) und schließlich ist noch die Farbe anzugeben (vierte Angabe). Es werden also 4 mal 4 Arrays erstellt mit je 13 Daten. Sie können natürlich auch weniger Arrays erstellen, die dann mehrdimensional sind, wenn Ihnen das lieber ist. Am Endergebnis ändert sich dadurch nichts.

Die Zeilen 70-175 lesen alle Daten in den Arrays.

Zeile 180 ist sehr wichtig. Diese Zeile weist dem Vergrößerungs- und dem Verschiebungsfaktor einer jeden Zeichnung einen Wert zu. Sie werden gleich sehen, daß vor allem durch Änderung dieser Zeile die unterschiedlichsten Effekte erzielt werden. Zeile 185 bewirkt eine Schleife, die den größten Teil des Programmes umfaßt. Diese Schleife besorgt nichts anderes als die Vergrößerung des Verschiebungsfaktors von links nach rechts immer dann, wenn alle vier Zeichnungen fertig sind. Auf diese Weise „läuft" das Männchen von links nach rechts über den Bildschirm.



Zeile 190 verweist auf eine Subroutine, die in Zeile 1190 beginnt. Diese Subroutine bewirkt das Zeichnen von Kopf und Rumpf des Männchens; diese bleiben nämlich immer an der gleichen Stelle, ungeachtet der Haltung von Armen und Beinen. Nur mittels des Vergrößerungs- und des Verschiebungsfaktors läßt sich die Position des Körpers ändern, doch sind diese Faktoren durch das gesamte Programm hindurch gleich.

Mit den Zeilen 240-280 wird die erste Bewegungsphase des Männchens gezeichnet. Zeile 260 kontrolliert, ob ein Punkt gesetzt werden muß. Ist die erste Angabe eine 0, muß zu diesem Punkt (dem Ende eines Armes oder Beines) keine Linie gezogen, sondern der Cursor soll versetzt werden. Bei diesem Befehl können Sie die Vergrößerungs- und Verschiebungsfaktoren bereits erkennen. Die X-Koordinate XA(L) wird immer mit A multipliziert und mit X verschoben. Die Y-Koordinate YA(L) dagegen multipliziert man mit B und verschiebt sie mit Y. Muß kein Punkt gesetzt werden, zeichnet Zeile 270 eine Linie zum nächsten Punkt.

Zeile 285 bewirkt eine Verlangsamung, so daß das gezeichnete Männchen ein wenig länger sichtbar bleibt, bevor es mit dem CLS-Befehl in derselben Zeile gelöscht wird, um der nächsten Zeichnung zu weichen.

Die Zeilen 290-385, 390-485 und 540-585 bewirken genau das gleiche, nur eben für die Zeichnungen B, C und D.

Zeile 590 stellt den zweiten Befehl der Schleife aus Zeile 185 dar.

Die Zeilen 910-980 bilden eine lange Datenliste. Die Daten für die Arme des Männchens sind in einer Zeile zusammengefaßt, desgleichen die Daten für die Beine, und das gleich viermal: für jede Bewegungsphase des Männchens also zwei Datenzeilen.

Die Zeilen 1190-1230 stellen eine Subroutine zum Zeichnen von Kopf und Rumpf dar. Auch hier begegnen uns wieder die Vergrößerungs- und Verschiebungsfaktoren. In diesem Falle sind in den Zeichenbefehlen zugleich die Koordinaten verarbeitet, da sie immer konstant bleiben.

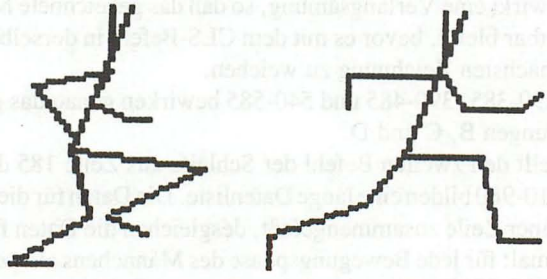
Dadurch, daß im gesamten Programm konsequent Variablen zur Bestimmung der jeweiligen Vergrößerung und zum Verschieben eingesetzt wurden, können Sie mit einfachen Veränderungen das Aussehen des Männchens wesentlich variieren. Wenn in Zeile 180 A und B jeweils auf 2 verringert werden, erscheint die Zeichnung halb so groß wie zuvor. Eine Kombination aus Koordinatensystem und Schrittgröße der X-Schleife in Zeile 185 läßt das Männchen ängstlich um sich blicken; nach jedem

Schritt dreht es sich um. So ein kleines Männchen macht auf dem Schirm einen ziemlich jämmerlichen Eindruck. Setzen Sie daher für A und B einmal größere Werte ein (vielleicht 6 oder gar 8). Sie können natürlich auch unterschiedliche Werte nehmen. Dann wird die Figur besonders groß und besonders dick.

Eine Änderung der gemachten Schritte erreichen Sie durch Korrektur der Schrittgröße in Zeile 185.

Wenn Sie nun alle möglichen Kombinationen ausprobiert haben, können Sie in Zeile 10 noch den Schirmmodus in SCREEN 2 verändern. Versuchen Sie es auch hier mit den eben beschriebenen Effekten.

Der Nachteil dieser Effekte ist der, daß Sie kein sich wirklich lebensecht bewegendes Männchen zeichnen können. Der Computer benötigt einfach viel zu viel Zeit, um die Figur immer wieder neu auf den Bildschirm zu bringen. So kommt es zu einem Flackern, das eine gute Bewegungssimulation verhindert. Es gibt zwei Möglichkeiten, hieran etwas zu ändern: Zunächst einmal ist das Programm in Maschinensprache neu zu schreiben; Maschinensprache ist wesentlich schneller als BASIC. Dafür sind aber besondere Kenntnisse erforderlich. Aus Platzgründen wurde in diesem Buch auf eine Einführung in die Maschinensprache verzichtet. Die zweite Möglichkeit führt allerdings auch in BASIC zu recht ansehnlichen Resultaten. Sie müssen sich hierzu der Sprites (Musterzeichnungen) bedienen. Lesen Sie hierüber mehr in den Kapiteln 17 und 18.



17. GRAFIK: HOHE AUFLÖSUNG

Die Zeichnungen aus dem vorigen Kapitel waren zwar farbig, aber die Linien sahen noch ein bißchen kräftig aus. Aus diesem Grund gibt es beim MSX den Schirmmodus 2. Dieser Modus eröffnet Ihnen die Möglichkeit, Zeichnungen mit feinen Linien anzufertigen. Der Bildschirm ist in ein Raster aus 255 Linien von links nach rechts und 191 Linien von oben nach unten unterteilt. Anhand des folgenden Programms können Sie die Arbeit mit einem hochauflösenden grafischen Schirm üben. Mit den vier Pfeiltasten läßt sich eine Linie auf dem Schirm ziehen und eine komplette Zeichnung kann so allmählich aufgebaut werden.

```
10 SCREEN 0 : KEY OFF
20 PRINT "     ZEICHENTISCH"
30 PRINT : PRINT : PRINT
40 PRINT "Sie koennen mit den Pfeilen zeichnen."
50 PRINT "Das Druecken der Leertaste "
60 PRINT "beendet das Zeichnen"
70 PRINT "oder schaltet es wieder ein."
80 PRINT :INPUT"Wie ist Ihre Anfangsposition (x,y)";X,Y
90 SCREEN 2
100 COLOR 4,15,14 : CLS
110 Q=1 : KL=4
120 ON STRIG GOSUB 250
130 STRIG(0) ON
140 PSET(X,Y),KL
150 REM
160 P=STICK(0)
170 IF P=0 THEN GOTO 220
180 IF P=3 THEN X=X+1 : GOTO 220
190 IF P=5 THEN Y=Y+1 : GOTO 220
200 IF P=7 THEN X=X-1 : GOTO 220
210 IF P=1 THEN Y=Y-1 : GOTO 220
220 LINE -(X,Y),KL
230 GOTO 160
240 REM Subroutine
250 IF Q=1 THEN KL=4 : Q=0 : RETURN
260 KL=15 : Q=1 : RETURN
```

Die Zeilen 10-80 enthalten einen Einführungstext, der dem Benutzer erklärt, was anschließend vor sich gehen wird. Darüber hinaus wird um Eingabe der Anfangsposition gebeten. In Zeile 90 wird der Schirm so verändert, daß er nun ein grafischer Schirm ist.

Zeile 110 macht aus 0 eine 1, und Q ist die Variable, die mitteilt, ob etwas gezeichnet werden soll oder nicht. Immer wenn Sie die Leertaste drücken, macht Q aus 0 eine 1 oder aus 1 eine 0. Die Variable KL steht für die Farbe, die der zu ziehende Strich haben soll. Wenn KL gleich 4 ist, wird dieser Strich blau; ist KL gleich 15, ist die Farbe des Striches weiß und er bleibt somit unsichtbar.

In Zeile 120 wird STRIG startbereit gemacht. Der Computer achtet von nun an darauf, ob die Leertaste gedrückt wird. Wenn Sie sie drücken, geht er zu einer Subroutine in Zeile 250 über. Zeile 130 macht Zeile 120 wirksam.

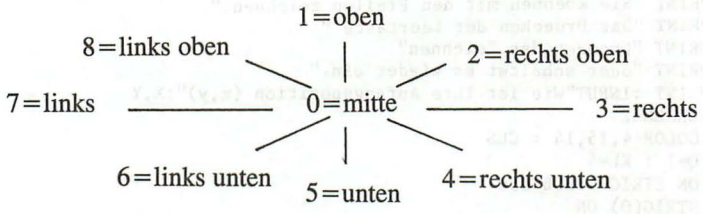
Zeile 140 bewegt den Cursor hin zu dem Punkt, den der Benutzer angegeben hat; und dieser Punkt erscheint in blauer Farbe.

Die Zeilen 160 bis 220 beinhalten die Schleife, die den eigentlichen Zeichenvorgang steuert.

Zeile 160 leitet den Wert der Pfeiltasten weiter an die Variable P. Hierzu geben Sie ein:

```
STICK( )
```

In die Klammer setzen Sie eine 0, 1 oder 2 ein. Eine 0 sagt dem Computer, daß er auf die Pfeiltasten zu achten hat; drücken Sie eine 1, beachtet er stattdessen den über Eingang 1 angeschlossenen Joystick. Eine 2 teilt dem Computer mit, daß er den Joystick in Eingang 2 zu beachten hat. Die Werte dieses Befehles sind von der jeweiligen Position des Joysticks bei Betätigung der Pfeiltasten abhängig:



In den Zeilen 170-210 werden die Werte von X und Y mittels des Ergebnisses des Befehls in Zeile 160 erhöht oder verringert. Drückt der Benutzer auf die Pfeiltaste, die nach oben führt, ist P gleich 1. Entsprechend der Zeile 210 wird dann der Y-Wert um 1 verringert. Dadurch bewegt sich der Cursor auf dem Bildschirm um eine Position nach oben (Zeile 220). Wird die Pfeiltaste gedrückt, die nach unten führt, ist P gleich 5, und durch Zeile 190 erhöht sich der Y-Wert um 1, so daß auf dem Bildschirm ein kleiner Strich einen Schritt weiter nach unten gezogen wird. Gleiches gilt für die Pfeiltasten, die nach links und nach rechts führen, wodurch der X-Wert verringert bzw. erhöht und P 7 bzw. 3 wird.

Zeile 220 sorgt dafür, daß der zu ziehende Strich die Farbe KL annimmt. Mit Zeile 230 wird die Schleife schließlich geschlossen.

Die Zeilen 250 und 260 stellen eine Subroutine dar, die durch Drücken der Leertaste aufgerufen wird. Zunächst einmal wird festgestellt, welchen Wert Q hat. Ist dieser Wert 1, erhält KL den Wert 4; Q ändert sich anschließend in 0 und die Subroutine wird verlassen.

Ist der Wert von Q ungleich 0, nimmt KL den Wert 15 an und wird gleich 1, wonach die Subroutine beendet ist. Mittels der Leertaste kann also entschieden werden, ob ein Zeichenvorgang stattfinden soll oder nicht.

Bei abgeschaltetem Zeichenmodus kann der Benutzer durch zweimaliges Drücken der Leertaste kontrollieren, wo sich der grafische Cursor gerade befindet. Die Position des grafischen Cursors wird kurz durch einen blauen Punkt angezeigt.

Wollen Sie zweifarbig zeichnen, können Sie KL anstelle von 15 den Wert 14 zuweisen. In diesem Falle würden Sie in den Farben Blau und Grau zeichnen.

Besser noch geht es, wenn Sie die Subroutine folgendermaßen verändern:

```
250 IF KL=15 THEN KL=0 ELSE KL=KL+1
260 RETURN
```

Sie können in jeder gewünschten Farbe zeichnen. Achten Sie auf das Ende des Striches, und Sie sehen, welche Farbe als nächste kommt. Wenn Sie möchten, daß eine Bewegung nicht als Strich erscheint, drücken Sie so oft auf die Leertaste, bis die Farbe weiß erscheint; dann bewegen Sie den Cursor in die gewünschte Position und drücken anschließend wieder auf die Leertaste, bis die Farbe erscheint, die Sie sich ausgesucht haben. Nun setzen Sie den Zeichenvorgang fort.

Die Leertaste läßt sich auch dann betätigen, wenn Sie die Pfeiltasten niedergedrückt halten.

Punkte

Im vorigen Kapitel haben Sie den Befehl PSET kennengelernt. Dieser Befehl läßt sich auch auf Schirm 2 anwenden. Die Möglichkeit, eine Vielzahl von Punkten an willkürlich gewählten Stellen zu zeichnen, ist Ihnen bereits bekannt.

Dadurch, daß auf dem Schirm ein willkürlicher Punkt in einer willkürlichen Farbe gesetzt wird, können ausdruckstarke Abbildungen erzielt werden. Der Benutzer kann seiner Phantasie freien Lauf lassen. Das nun folgende Programm bringt an willkürlichen Stellen Punkte an. Der Anfang wird allerdings in der Schirmmitte gemacht, und alle anderen Punkte gruppieren sich um die Bildmitte.

```
10 SCREEN 2 : COLOR 4,15,15
20 PSET (126,95)
25 X=0 : Y=0 : DX=0 : DY=0
30 X=INT(RND(1)*10-4.5)
40 Y=INT(RND(1)*10-4.5)
50 DX=DX+X
60 DY=DY+Y
70 PSET (126+DX,95+DY)
80 GOTO 30
```

In diesem Programm gibt es keine unbekannten Befehle mehr. Es dürfte also nicht sonderlich schwierig sein, die Arbeitsweise dieses Programms zu erkennen. Ein Aspekt ist besonders bemerkenswert. Wenn das Programm eine Weile gelaufen ist, kommen keine neuen Punkte mehr hinzu, in einer unendlichen Schleife allerdings setzt das Programm weiterhin seine Punkte. Lassen Sie das Programm zum zweiten Male laufen, geschieht genau das gleiche; es entsteht sogar die gleiche „zufällige" Figur. Der Grund dafür ist die Tatsache, daß der MSX keine wirklich zufälligen Zahlen generiert, sondern nur eine Liste mit pseudo-zufälligen Zahlen durchgeht, und auch beim zweiten Durchgang ändert sich nichts an dieser Liste. Der Zufallseffekt dieser Liste läßt sich mit der Variablen TIME erhöhen.

Striche

Mit Hilfe des PSET-Befehles ist auch das Ziehen von Strichen möglich. Versuchen Sie es einmal mit diesem Programm.

```
10 SCREEN 2
20 FOR X=1 TO 250
30 PSET (X,100),4
40 NEXT X
50 GOTO 50
```

In diesem Programm wird X von 1 bis 250 immer um 1 erhöht. An der Position (X, 100) erscheint ein Punkt, und auf diese Weise erhalten Sie eine ganze Reihe von Punkten hintereinander, die schließlich einen geraden Strich ergeben. Genau so läßt sich natürlich auch ein senkrechter Strich ziehen. Dazu ist es lediglich notwendig, mit einer FOR..NEXT..-Schleife die Y-Koordinate zu verändern; soll es sich um einen Schrägstrich handeln, werden beide Koordinaten verändert. Zum Beipiel:

```
10 SCREEN 2
20 FOR X=1 TO 250
30 PSET (X,X/2),4
40 NEXT X
50 GOTO 50
```

In dem Programm werden sowohl die X- als auch die Y-Koordinate geändert. Dazu erfolgt eine Erhöhung der X-Koordinate in einer FOR..NEXT..-Schleife von 1 auf 250; die Y-Koordinate nimmt jedesmal einen Wert an, der dem halben Wert der X-Koordinate entspricht. Der Schrägstrich, den Sie jetzt sehen, ist eine Darstellung der Funktion Y=X/2. (Unten ist hier oben aufgrund der Drehung der Y-Achse!)

Es gibt noch einen anderen Befehl, mit dem sich Striche ziehen lassen, und auch ihm sind Sie schon einmal begegnet.

LINE (,) – (,)

In die Klammern setzen Sie die Koordinaten der Anfangs- und Endpunkte ein. So würde sich unser Programm oben nun ändern:

```
10 SCREEN 2
20 LINE (1, 1) – (250-125), 4
30 GOTO 30
```

Sie sehen, das Programm kann auch um einiges kürzer ausfallen. Es kommt noch hinzu, daß der Zeichenvorgang wesentlich rascher vonstatten geht.

Beim LINE-Befehl gibt es noch zwei besondere Effekte: die Möglichkeiten, ein Rechteck bzw. ein gefülltes Rechteck zu zeichnen.

LINE (100, 90) – (130, 160), 4, B

Wenn Sie diesen Befehl in Zeile 20 des vorigen Programmes setzen, wird der Computer Ihnen ein Rechteck zeichnen, dessen linke obere Ecke mit dem ersten Koordinatenpaar und dessen rechte untere Ecke mit dem zweiten Koordinatenpaar bezeichnet wird. Der Buchstabe B hinter dem Befehl teilt dem Computer mit, daß er eine Box (Schachtel, Rechteck) zeichnen soll. Denken Sie also daran: In einem LINE-Befehl stehen bei dem Buchstaben B die Koordinatenpaare für die linke obere und die rechte untere Ecke.

Auf die gleiche Weise lassen Sie auch ein gefülltes Rechteck zeichnen. Sie müssen dem Computer dazu lediglich hinter dem B noch ein F angeben (für Fill = füllen). Probieren Sie das einmal in Zeile 20 des letzten Programmes aus.

Wenn Sie sehen wollen, wie rasch das alles vor sich gehen kann, versuchen Sie es doch mal mit dem nächsten Programm.

```
10 SCREEN 2 : COLOR ,1,1 : CLS
20 KL=INT(RND(1)*15)
30 X1=INT(RND(1)*255)
40 X2=INT(RND(1)*255)
50 Y1=INT(RND(1)*191)
60 Y2=INT(RND(1)*191)
70 LINE (X1,Y1)-(X2,Y2),KL,BF
80 GOTO 20
```

Dieses Programm zeichnet willkürliche Rechtecke, die alle gefüllt sind. Wenn Sie offene Rechtecke haben wollen, müssen Sie in Zeile 70 aus BF B machen. Selbst die Entscheidung, ob BF oder B gesetzt werden soll, kann willkürlich erfolgen. Ändern Sie dazu Zeile 70

```
70 IF Q=1 THEN LINE (X1, Y1) – (X2, Y2), KL, BF ELSE
LINE(X1, Y1) – (X2-Y2), KL, B
```

und fügen hinzu:

```
65 Q=INT(RND(1)*)
```

Kurven

Mit geraden Strichen lassen sich zwar schöne Zeichnungen anfertigen, aber Zeichnungen mit gebogenen Linien sind auch nicht schlecht. Bei den Sinusgrafiken im letzten Kapitel haben Sie das bereits feststellen können.

Eine gebogene Linie kann man als eine Reihe von Punkten oder als eine Reihe von sehr kurzen geraden Strichen ansehen, die alle im gleichen Winkel zueinander stehen. Die einfachste Art, eine Kurve zu erhalten, ist die Darstellung einer Reihe von Punkten.

```
10 SCREEN 2
20 FOR X=0 TO 250
30 Y=INT(X*X/200)
40 PSET (X,Y)
50 NEXT X
60 GOTO 60
```

Dieses Programm zeichnet die Grafik der Funktion Y=X*X. Eigentlich handelt es sich um die Funktion Y=X*X/200, wobei die Division durch 200 die Grafik ordentlich auf den Schirm bringt. Fehlte diese Division, würde die Grafik als fast gerader Strich nach unten verlaufen. Nun steht die Grafik auf dem Kopf, und wenn Sie das ändern wollen (das gilt für alle Programme in diesem Kapitel), müssen Sie für die Y-Koordinate nicht den Wert von Y eingeben, sondern 191-Y.

Das Problem bei dieser Grafik ist, daß Sie zwar eine prächtige Kurve erhalten, die aber unten (oder oben) ein bißchen unsauber wird, da die Punkte zu weit auseinanderstehen. Dies läßt sich korrigieren, indem Striche gezogen anstatt Punkte gesetzt werden. Ändern Sie Zeile 40 folgendermaßen:

```
40 LINE - (X,190-X)
```

und setzen Sie hinzu:

```
15 PSET (0,191)
```

Um den Unterschied zwischen den beiden Möglichkeiten noch deutlicher werden zu lassen, können Sie beide Programmversionen mit einem Teilungsfaktor von 100 anstelle von 200 versehen. Zeile 30 sieht dann so aus:

```
30 LET Y=INT(X*X/100)
```

Soll die komplette Grafik auf den Schirm passen (negative Werte von X fehlen jetzt), können Sie den Maßstab noch weiter vergrößern, verkleinern oder verschieben. Das Resultat ist:

```
10 SCREEN 2
15 PSET (0,INT(-125*-125/90))
20 FOR X=-125 TO 125
30 LET Y=INT(X*X/90)
40 LINE -(X+125,190-Y)
50 NEXT X
60 GOTO 60
```

Eine hübsche Anwendung finden diese Techniken im folgenden Programm, das die Bahn eines Spielballes simuliert. Ein Ball wird nach vorne geworfen.

```
10 SCREEN 0
20 BX=25
30 BY=50
40 VS=0 : N=0
50 PRINT : PRINT : PRINT "Dieses Programm zeigt die Bahn "
60 PRINT "eines rollenden Balles."
70 PRINT
80 INPUT "Anfangsgeschwindigkeit (0-25)";HS
90 SCREEN 2
100 PSET (250,190)
110 LINE -(25,190)
120 LINE -(25,0)
130 PSET (BX,BY)
140 REM Schleife
150 BX=BX+HS/2
160 VS=VS+2
170 BY=BY+VS
180 IF BY>=190 THEN GOSUB 500 : GOTO 210
190 IF BX>=255 THEN GOSUB 1000 : GOTO 210
200 IF BX<=25 THEN GOSUB 1500
210 LINE -(BX,BY)
220 FOR A=1 TO (190-BY)/2 : NEXT A
230 IF N<=20 THEN GOTO 150
240 GOTO 240
250 END
499 REM boden
500 BY=190
510 VS=-.9*VS
520 N=N+1
530 BEEP
540 RETURN
999 REM Mauer rechts
1000 BX=255
1010 HS=-.9*HS
1020 BEEP
1030 RETURN
1499 REM Mauer links
1500 BX=25
1510 HS=-.9*HS
1520 BEEP
1530 RETURN
```

Die ersten Zeilen dienen der Wahl des Schirmmodus sowie dem Einsetzen der verschiedenen Variablen. Die Variablen BX und BY geben die Position des Balles an. VS ist die vertikale und HS die horizontale Geschwindigkeit; N gibt an, wie oft der hopsende Ball den Boden berührt hat. Die horizontale Geschwindigkeit kann vom Benutzer mit Zeile 80 vorgegeben werden.

In den Zeilen 90-130 wird ein anderer Schirm aufgerufen und Rahmen werden abgesteckt. Anschließend bewegt sich der Cursor zum Anfangspunkt.

Zeile 150 bestimmt die X-Koordinate des Balles. Diese entspricht jeweils der alten Position zuzüglich der Hälfte der vom Benutzer eingegebenen horizontalen Geschwindigkeit.

Die vertikale Geschwindigkeit nimmt (aufgrund der Schwerkraft) zu. Sie ist gleich der ursprünglichen Geschwindigkeit plus 2 (Zeile 160). Die vertikale Position (die Y-Koordinate) ist von der vorherigen Position und der vertikalen Geschwindigkeit abhängig.

In den Zeilen 180, 190 und 200 besteht die Möglichkeit, besondere Subroutinen zu beginnen. Zu Anfang ist dies die Position des Balles. Befindet sich der Ball knapp über dem Boden, geht der Computer über zu Zeile 500. Vor einer Berührung der linken oder rechten Begrenzung geht der Computer in die Zeilen 1500 oder 1000.

Zeile 210 bewirkt die Zeichnung der Bahn des Balles.

Zeile 220 sorgt für eine Verlangsamung, die abhängig ist von der Höhe des Balles. Je höher sich der Ball auf dem Schirm befindet, desto langsamer bewegt er sich. Auch das dient der besseren Simulation.

Zeile 230 bewirkt eine begrenzte Anzahl von Hopsern und Zeile 240 erreicht, daß der grafische Schirm nicht gleich wieder verschwindet.

Die Subroutine in Zeile 500 ermöglicht es, daß der Ball genau den Boden berührt (BY=190) und kehrt die Richtung der Geschwindigkeit um (VS=-.9*VS). Dabei wird die Geschwindigkeit ein wenig reduziert, da ein hopsender Ball bei Berührung eines Gegenstandes ja immer etwas an Schnelligkeit verliert. Wollen Sie die Bahn eines schlecht hopsenden Balles sehen, müssen Sie den Faktor .9 in Zeile 510 verringern. Richtig schön bewegt sich der Ball mit einem Verlustfaktor von 1.

Die Subroutine in Zeile 1000 ist vergleichbar mit derjenigen in Zeile 500. Nur geht es diesmal um die Berührung der rechten Begrenzung. Die horizontale Geschwindigkeit wird umgekehrt und ein wenig reduziert.

Die Subroutine in Zeile 1500 bewirkt das gleiche für die linke Begrenzung.

Kreise, Ellipsen und Kuchenstücke

Neben den PSET-, PRESET- und LINE-Befehlen gibt es beim MSX auch spezielle Befehle für das Zeichnen von Kreisen oder Teilen davon. Das kann zuweilen recht nützlich sein. Ein Kreis kann zwar auch auf die Weise gezeichnet werden wie die Grafik, die Sie gerade gesehen haben, doch da dies in BASIC geschieht, dauert das wesentlich länger als die Routine, die in Maschinensprache bereits im MSX integriert ist.

Um einen Kreis zu zeichnen, geben Sie den Mittelpunkt sowie den Radius des Kreises an:

```
CIRCLE ( , ),
```

In die Klammer setzen Sie die Koordinaten für den Mittelpunkt; hinter der Klammer folgt der Radius. Darüber hinaus können Sie natürlich noch einen Farbcode angeben.

```
10 SCREEN 2
20 CIRCLE (100,90),50
30 CIRCLE (40,60),30,3
40 GOTO 40
```

Dieses kleine Programm zeichnet zunächst einen Kreis mit dem Radius 50 und dem Mittelpunkt (100, 90) in der augenblicklichen Vordergrundfarbe. Anschließend wird ein Kreis mit dem Radius 30 und (40, 160) als Mittelpunkt in der Farbe Schwarz gezeichnet.

Hier sehen Sie auch bereits, wie Ellipsen dargestellt werden können. Sie geben einen Kreis an und erhalten eine Ellipse. Dies hat mit dem Verhältnis links-rechts und unten-oben beim Fernsehgerät zu tun. Um schöne Kreise zu erhalten, müssen Sie die Kreise in der Y-Richtung mit 1.4 multiplizieren:

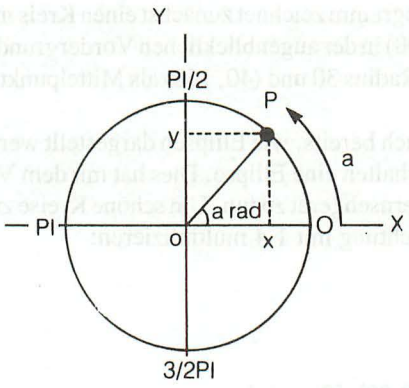
```
10 SCREEN 2
20 CIRCLE (100,90),50,,,,1.4
30 CIRCLE (40,160),30,1,,,1.4
40 GOTO 40
```

Das Verhältnis zwischen Y- und X-Achse eines Kreises ist die letzte Angabe, die Sie bei einem CIRCLE-Befehl machen können. Zahlen größer als 1.4 bewirken Ellipsen, die hoch und schmal sind; Zahlen kleiner als 1.4 solche, die breit und niedrig sind. Um Ihnen das anschaulicher zu machen, folgt hier ein schon bekanntes Programm, mit dem an willkürlichen Stellen eine Zeichnung angefertigt wird. In diesem Falle handelt es sich nicht um Striche oder Rechtecke, sondern um Ellipsen.

```
10 SCREEN 2 : COLOR ,1,1 : CLS
20 KL=INT(RND(1)*15)
30 X=INT(RND(1)*255)
40 Y=INT(RND(1)*255)
50 EL=RND(1)*3
60 ST=INT(RND(1)*70)
70 CIRCLE (X,Y),ST,KL,,,EL
80 GOTO 20
```

Wie Sie in den beiden vorhergehenden Programmen bereits gesehen haben, passen zwischen die Angabe der Farbe und das Verhältnis zwischen X- und Y-Achse noch zwei Zahlen. Bisher wurden diese Zahlen nicht gebraucht, und an ihrer Stelle haben Kommas gestanden.

Mit Hilfe dieser beiden Zahlen können Sie dem Computer befehlen, den Teil eines Kreises zu zeichnen. Hierzu geben Sie den Anfangswinkel und den Endwinkel an. Diese Winkel, die zwischen 0 und 2*PI liegen müssen, drücken Sie in Halbmessern aus. Ein vollständiger Kreis besteht aus 2 PI Halbmessern. Sehen Sie sich hierzu die folgende Zeichnung an.



Die Winkel werden von der X-Achse ab gemessen und entgegen dem Uhrzeigersinn.

Der Wert PI ist nicht im MSX integriert. Sie können ihn aber errechnen, indem Sie 4 mal den Arctangens von 1 nehmen. Im Programm unten finden Sie ein Beispiel hierzu.

```
10 SCREEN 2
20 COLOR 4,1,1 : CLS
30 PI=4*ATN(1)
40 CIRCLE (100,90),60,,0,PI,1.4
50 GOTO 50
```

Dieses Programm zeichnet einen Halbkreis. Der Mittelpunkt befindet sich in (100, 90), der Radius beträgt 60, und es wird der Teil zwischen 0 und PI gezeichnet. Das heißt, gezeichnet wird die obere Hälfte des Kreises. Probieren Sie anhand dieses Programms die verschiedenen Möglichkeiten einmal aus. Verändern Sie beliebig oft diejenigen Zahlen, die jetzt 0 und PI sind; und sie brauchen nicht nur Zahlen einzugeben, die in PI ausgedrückt sind, es können auch ganz gewöhnliche Zahlen sein. Es geht einfach darum, daß Sie eine Reihe von Halbmessern eingeben. Sind es jedoch mehr als etwa 6, 28 (2*PI), erhalten Sie eine Fehlermeldung.

Durch Veränderung des Verhältnisses 1.4 können Sie auch Teile von Ellipen zeichnen; hierbei müssen Sie Ellipsen als verformte Kreise ansehen.

Dadurch, daß die Zahlen für die Angabe des Anfangs- und Endwinkels ein Minuszeichens (–) erhalten, wird nicht nur ein Teil des Kreises gezeichnet, sondern darüber hinaus werden auch die Endpunkte des Kreisbogens mit dem Kreismittelpunkt verbunden. Auf diese Weise erhalten Sie ein „Kuchenstück". Verändern Sie einmal Zeile 40 im Programm oben:

CIRCLE (100,90), 60,, – .5, – 5.5,1.4

Diese Anweisungen können Sie zu einem hübschen Programm zusammenfassen, mit dem sich schöne Zeichnungen erstellen lassen. Der Computer dient hier als eine Art Spirograph. Ansehnliche Resultate erzielen Sie durch Wiederholen einer Zeichnung mit immer einer kleinen Abweichung im Vergleich zur jeweils vorhergehenden Zeichnung. Versuchen Sie selber, an verschiedenen Punkten im Programm Änderungen vorzunehmen, und sehen Sie sich das Ergebnis genau an.

```
10 SCREEN 2
20 COLOR 4,1,1 : CLS
30 PI=4*ATN(1)
40 FOR A=1 TO 30
50 CIRCLE (100+A,60+A),10+A*2,,-.5,-6,1.4
60 NEXT A
70 GOTO 70
```

Färben

Neben den Möglichkeiten, die der LINE-Befehl beim Füllen eines Rechteckes bietet, gibt es beim MSX auch einen Befehl zum Ausfüllen einer willkürlichen Figur. Er lautet:

PAINT (,),

In die Klammer setzen Sie die Koordination eines Punktes ein, der sich innerhalb der auszufüllenden Figur befinden muß. Der Wert, der hinter der Klammer folgt, bezeichnet die für die Figur vorgesehene Farbe. In Schirmmodus 2 muß die Farbe des Randes mit der Farbe der Fläche übereinstimmen.

Nach einem PAINT-Befehl beginnt der Computer vom angegebenen Punkt aus die Form abzutasten, und zwar von links nach rechts und von oben nach unten. Die gewünschte Farbe wird so oft eingezeichnet, bis die Form ausgefüllt ist. Sehen Sie sich das nächste Programm einmal an, das zunächst eine Ellipse in der Farbe Rot zeichnet und sie anschließend einfärbt.

```
10 SCREEN 2 : COLOR ,15,15 : CLS
20 CIRCLE (150,100),80,6,,,.5
30 FOR Z=1 TO 200 : NEXT
40 PAINT (150,100),6
50 GOTO 50
```

Was wird wohl passieren, wenn Sie den Computer mit dem PAINT-Befehl in Zeile 40 nach einer anderen Farbe suchen lassen als der, in der die Ellipse gezeichnet ist? Ändern Sie Zeile 40 folgendermaßen:

40 PAINT (150,100),9

Auch weniger regelmäßige Formen sind denkbar. Setzen Sie in den Zeilen 20 und 40 folgendes ein:

```
20 CIRCLE (150,100),80,9,-1,-6,.5
40 PAINT (140,100),9
```

Beachten Sie, daß Zeile 40 deshalb geändert wird, weil der Punkt, von dem aus gePAINTet wird, innerhalb der Form liegen muß.

In Schirmmodus 2 ist es möglich, dem Rand eine andere Farbe zu geben als dem Innern der zu zeichnenden Figur. Versuchen Sie es einmal mit diesem Programm:

```
10 SCREEN 2 : COLOR ,15,15 : CLS
20 CIRCLE (120,100),110,4,-1,-6,.6
30 FOR Z=1 TO 200 : NEXT
40 PAINT (110,100),9,4
50 GOTO 50
```

Dieses Programm zeichnet ein blaues Kuchenstück, und anschließend wird es rot eingefärbt, während der Rand blau bleibt. Sie müssen hierfür hinter den Koordinaten zu PAINT zwei Zahlen eingeben. Mit der ersten ist auch hier die Farbe gemeint, in der die Figur eingefärbt werden soll; die zweite Zahl gibt die Farbe des Randes an, der den eingefärbten Bereich umschließt. Auf diese Weise können Sie selektiv einfärben. Schauen Sie sich mal das nächste Programm an.

```
10 SCREEN 3 : COLOR ,15,15 : CLS
20 CIRCLE (120,100),110,4,-1,-6,.6
30 FOR Z=1 TO 200 : NEXT
40 CIRCLE (195,100),80,1,-.5,-6,2
50 FOR Z=1 TO 200 : NEXT
60 PAINT (190,130),9,1
70 GOTO 70
```

Text auf dem grafischen Schirm

Um Texte auf den grafischen Schirm bringen zu können, ist ein spezieller Trick nötig. Sie können ja mit dem INPUT- und PRINT-Befehl auf dem grafischen Schirm nichts anfangen.

Damit Sie Ihre Grafiken dennoch mit Texten versehen können, müssen Sie einen Dateikanal öffnen und den Text über diesen Kanal hin zu „Datei grafischer Schirm" schicken. Mit dem folgenden Programm wird das schon etwas verständlicher.

```
10 SCREEN 3 : COLOR 1,15,15 : CLS
20 PSET (0,100)
30 FOR X=1 TO 10 STEP .1
40 LINE -(X*24,80-SIN(X)*60),4
```

```
50 NEXT X
60 OPEN "grp:" AS #1
70 PSET (100,20),15
80 PRINT #1,"sinus x"
90 CLOSE #1
100 GOTO 100
```

Die Zeilen 10-50 sind Ihnen bereits bekannt. Sie dienen der Gestaltung des Schirmes, bestimmen die Farben und zeichnen eine Sinusgrafik.

In Zeile 60 wird mit dem Befehl

''GRP:''

ein Kanal geöffnet. Auf diese Weise gelangt der Text auf den grafischen Schirm.

Zeile 70 bewegt den grafischen Cursor. Seien Sie hierbei vorsichtig: Sie arbeiten auf dem grafischen Schirm, und infolgedessen werden auch Texte hin zur Position des grafischen Cursors bewegt. Bei grafischen Schirmen bleibt der LOCATE-Befehl ohne Wirkung.

Zeile 80 gibt den Text in die Datei; in diesem Falle ist das der grafische Schirm.

Mit Zeile 90 wird die Datei wieder geschlossen.

Mit dem COLOR-Befehl kann der Text farbig gestaltet werden. Fügen Sie bei dem Programm oben die Zeilen 65, 68, und 85 hinzu und verändern Sie außerdem Zeile 70:

```
65 FOR KL=1 TO 14
68 COLOR KL
75 PSET (70+3*KL,10+4*KL),15
85 NEXT KL
```

Sie sehen, daß die Buchstaben übereinander wiedergegeben werden. Das hängt damit zusammen, daß der Computer den grafischen Schirm nicht erst löscht, bevor er etwas anderes darauf darstellt. Dies wird vor allem deutlich, wenn Sie das letzte Programm in einer geeigneten Form auf Schirm 3 laufen lassen.

```
10 SCREEN 3 : COLOR 1,15,15 : CLS
20 PSET (0,100)
30 FOR X=1 TO 10 STEP .1
40 LINE -(X*24,80-SIN(X)*60),4
50 NEXT X
60 OPEN "grp:" AS #1
70 FOR KL=1 TO 14
80 COLOR KL
90 X=10+3*KL:Y=5*KL
100 DRAW "bm=x;,=y;"
110 PRINT #1,"sinus x"
120 NEXT KL
130 CLOSE #1
140 GOTO 140
```

Bei diesem Programm können Sie feststellen, daß es sehr gut möglich ist, saubere und deutliche Buchstaben dadurch zu erhalten, daß Sie den Text einige Male übereinander und ein klein wenig versetzt darstellen lassen. Hiermit sollten Sie etwas herumexperimentieren; die Ergebnisse können recht verblüffend sein, insbesondere dann, wenn Sie den Text in unterschiedliche Richtungen versetzen.

Verschiedenes

Mit allen bisher behandelten Techniken lassen sich allerlei hübsche Zeichnungen erstellen. Jede dieser Zeichnungen entsteht dadurch, daß der Computer eine einfache Darstellung viel Male wiederholt; bei dem letzten Programm etwa war das so.

Das nun folgende Programm zeichnet eine Brille.

```
10 SCREEN 2
20 COLOR 4,15,9 : CLS
30 FOR X=-7 TO 4 STEP .2
40 Z=12*SIN(X)
50 FOR Y=-Z TO Z
60 LINE -(X*15+125,90-Y*7)
70 NEXT Y
80 NEXT X
90 GOTO 90
100 DRAW "bm=x;,=y;"
110 PRINT #1,"sinus x"
120 NEXT KL
130 CLOSE #1
140 GOTO 140
```

Probieren Sie unterschiedliche Effekte durch Veränderung der Schrittgröße in den Zeilen 30, 40 und 50 aus. Sie können auch den Bereich immer wieder ändern. Sie sehen, mit diesem einen Programm lassen sich zahlreiche, sehr unterschiedliche Zeichnungen anfertigen.

Der DRAW-Befehl

Neben denjenigen Zeichenbefehlen, von denen jeder eine besondere Funktion hat, gibt es beim MSX auch einen allgemeinen Zeichenbefehl. Er lautet:

DRAW ” ” (zeichne)

Dieser allgemeine Zeichenbefehl simuliert das Zeichnen mit einem Bleistift und Papier. Der Befehl selber teilt dem Computer nicht viel mehr mit, als daß alle Angaben, die sich im String hinter dem Befehl befinden, Zeichenbefehle sind. Von diesen „Teilbefehlen“ gibt es eine ganze Reihe. Jeder Teilbefehl setzt sich aus mehreren

Buchstaben zusammen, denen eine oder mehrere Zahlen folgen.

Die vier einfachsten Teilbefehle sind:

U	(Up = nach oben)
D	(Down = nach unten)
R	(Right = nach rechts)
L	(Left = nach links)

Hinter diesen Teilbefehlen müssen Sie die Länge des zu ziehenden Striches angeben; er wird als eine Reihe von Punkten eingegeben. Etwa:

```
10 SCREEN 2
20 DRAW "D55R87"
30 GOTO 30
```

So wird ein Strich gezogen, und zwar – ausgehend von der augenblicklichen Position des grafischen Cursors – 55 Lichtpunkte nach unten und 87 Lichtpunkte nach rechts. Beachten Sie, daß der Strich an einer anderen Stelle auf dem Schirm gezogen wird, wenn Sie das Programm zum zweitenmal laufen lassen; der grafische Cursor befindet sich dann nämlich in einer anderen Position.

Um auch Striche ziehen zu können, die nicht nur parallel zur X- oder Y-Achse verlaufen, geben Sie einen oder mehrere der folgenden Teilbefehle ein, mit denen diagonal gezeichnet werden kann.

E	(nach rechts oben)
F	(nach rechts unten)
G	(nach links oben)
H	(nach links unten)

Diese Befehle sind in der gleichen Weise zu verwenden wie die Teilbefehle U, D, R und L. Diesmal allerdings geben Sie hinter dem Teilbefehl nicht die absolute Länge des Striches an, sondern die Versetzung in X- und Y-Richtung. Die tatsächliche Länge des gezogenen Striches entspricht also der Wurzel aus dem doppelten Quadrat der eingegebenen Zahl (Satz des Pythagoras).

Versuchen Sie es einmal mit diesem Programm:

```
10 SCREEN 2
20 PSET (150,50)
30 DRAW "G100"
40 GOTO 40
```

Auf dem Bildschirm sehen Sie nun einen diagonalen Strich.

Die Wirksamkeit des DRAW-Befehles liegt in der Möglichkeit, alle Teilbefehle hintereinander anzuordnen. Ersetzen Sie Zeile 30 durch

```
30 DRAW "G100U60R110D30F40U100G100"
```

Sie können die Teilbefehle auch in einer Stringvariablen unterbringen:

```
10 SCREEN 2
20 PSET (150,50)
30 A$="G100u60r110d30f40u100g100"
40 B$="u150150f100"
50 DRAW A$+B$
60 GOTO 60
```

Aufgrund des DRAW-Befehles werden in diesem Programm zwei hintereinanderliegende Strings als Teilbefehle angesehen. Ebenfalls denkbar ist die Verwendung eines Strings innerhalb eines anderen Strings. Ändern Sie Zeile 40 in

```
40 B$= ''U150XA$;L50F100
```

Auf diese Weise wird B$ mit A$ erweitert. Nachdem der Strich um 150 Lichtpunkte hochgezogen wurde, werden die Teilbefehle von A$ ausgeführt. Anschließend geht es dann mit B$ weiter. Um so eine Stringvariable als Teilbefehl nutzen zu können, müssen Sie vor dem Variablennamen X angeben und dahinter ein Semikolon (;).

Der nächste wichtige Teilbefehl lautet:

M (Move = bewegen)

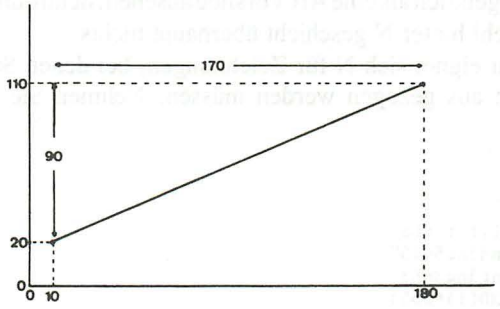
Mit diesem Teilbefehl läßt sich ein Strich hin zu einem Punkt ziehen, der durch zwei Zahlen hinter dem M bezeichnet wird. Bei den beiden Zahlen handelt es sich um die X- und die Y-Koordinate. Beide Koordinaten sind absolut, das heißt, daß vom Anfangspunkt an gerechnet wird (dem Punkt 0,0).

```
10 SCREEN 2
20 DRAW ''M195, 53''
30 GOTO 30
```

Wenn Sie dieses Programm zum erstenmal laufen lassen, erhalten Sie einen Strich. Der genaue Verlauf dieses Striches ist von der augenblicklichen Position des grafischen Cursors abhängig, und der wiederum hängt vom zuletzt gelaufenen Programm ab. Lassen Sie dieses Programm als erstes nach Einschalten des Computers laufen, erhalten Sie einen Strich, der bei dem Punkt (0,0) beginnt. Gleichgültig, welcher Strich auch gezogen wird, immer endet er bei (195, 53). Nach dem zweiten Durchlauf dieses Programmes sehen Sie an Position (195, 53) lediglich einen Punkt; der grafische Cursor befindet sich nämlich bereits an der Stelle, zu der hin der Strich gezogen werden soll.

Werden die Koordinaten des Punktes mit einem Minus- oder Pluszeichen versehen, zeichnet der Computer den Strich nicht hin zur absoluten Position (X, Y), sondern zur relativen. Das heißt, der Strich wird hin zu einem Punkt gezogen, der um X entfernt ist und um Y höher oder tiefer liegt.

Bei Verwendung eines Plus- oder Minuszeichens geben die Koordinaten eine Positionsveränderung an; absolute Koordinaten dagegen bezeichnen einen Punkt. Anhand der nächsten Zeichnungen wird der Unterschied zwischen Positionsveränderung und Punktbezeichnung deutlich.



In Programmform sieht das folgendermaßen aus. Können Sie auf dem Bildschirm erkennen, was die Positionsveränderung und was die Punktbezeichnung ist?

```
10 SCREEN 2
20 PSET (50,50)
30 DRAW "nl50nu50"
40 DRAW "c1ml00,130"
45 DRAW "c15nl100nul30"
50 PSET (150,50)
60 DRAW "nr50nd80"
70 DRAW "c6m+50,+80"
80 GOTO 80
```

Das Programm zeichnet zwei Striche. Der eine dieser beiden Striche wird durch die Koordinationen von Anfangs- und Endpunkt (beide durch weiße Striche hin zu X- und Y-Achse dargestellt) bestimmt. Der zweite Strich wird durch zwei Striche bezeichnet, die die Versetzung in X- und Y-Richtung angeben.

Neben dem Teilbefehl M können Sie in diesem Programm noch zwei weitere neue Teilbefehle entdecken.

C (Color = Farbe)

Der Teilbefehl C bestimmt die Farbe zu den sich anschließenden Teilbefehlen. Beim C-Befehl lassen sich die gleichen Farbcodes wie bereits beim COLOR-Befehl verwenden. Im Gegensatz zum COLOR-Befehl bewirkt der Teilbefehl C jedoch keine Änderung der Vordergrundfarbe.

Der Teilbefehl

N (Not = nicht)

veranlaßt den Computer zwar zu zeichnen, aber er bewegt den grafischen Cursor nicht von der Stelle. Diesen Befehl kann man nur vor einem anderen grafischen Teilbefehl verwenden. Er ist nur bei demjenigen Teilbefehl wirksam, vor dem er unmittelbar steht. Probieren Sie einmal aus, was passiert, wenn Sie im Programm oben bei jedem DRAW-Befehl den zweiten Teilbefehl N entfernen.

N muß man eigentlich als eine Art Vorsilbe ansehen, denn ohne einen richtigen grafischen Teilbefehl hinter N geschieht überhaupt nichts.

Besonders gut eignet sich N für Zeichnungen, bei denen Striche von einem bestimmten Punkt aus gezogen werden müssen. Nehmen Sie beispielsweise einen Stern:

```
10 SCREEN 2
20 COLOR ,1,1 : CLS
30 A$="nu5nd5nr5nl5"
40 B$="ne3nf3ng3nh3"
50 X=INT(RND(1)*255)
60 Y=INT(RND(1)*192)
70 KL=INT(RND(1)*15)
80 SC=INT(RND(1)*10)
90 DRAW "bm=x;,=y;s=sc;c=kl;xa$;xb$;"
100 GOTO 50
```

In den Zeilen 30 und 40 wird die Form des Stern vorgegeben. In 30 heißt das: nach oben, unten, rechts und links. In Zeile 40 sind es die verschienen Diagonalen.

Die Zeilen 50-100 stellen eine Schleife dar, die diesen Stern immer wieder neu in einer anderen Größe, anderen Farbe und an einer beliebigen Stelle zeichnet. Die Zeilen 50 und 60 bewirken die Darstellung an beliebiger Stelle; Zeile 70 sorgt für eine beliebige Farbe und Zeile 80 für eine beliebige Größe.

Zeile 90 bedient sich der willkürlichen Zahlen, wie sie in den Zeilen davor den Variablen zugewiesen sind. Die Art, in einen DRAW-String einen anderen String aufzunehmen, ist Ihnen ja bereits bekannt: X davor angeben und ; dahinter. In gleicher Weise wird eine Variable anstatt einer Zahl aufgenommen. Hierzu geben Sie vor der Variablen ein „=" an und hinter dem Variablennamen ein „;".

Ein neuer Teilbefehl in diesem Programm lautet

S (Scale = Skala)

Dieser Teilbefehl bezieht sich auf den Skalenfaktor, mit dem die Teilbefehle dahinter multipliziert werden. Standardmäßig ist der Skalenfaktor 4; das heißt, wenn Sie hinter S eine Zahl kleiner als 4 einsetzen, wird die Zeichnung entsprechend vergrößert. Geben Sie hinter S eine 4 oder ein 0 an, erhalten Sie eine Zeichnung in normaler Größe.

Auch in Zeile 90 gibt es eine neue „Vorsilbe".

B (Blanco = leer)

Diese „Vorsilbe" kann vor jedem Teilbefehl stehen, und sie bewirkt, daß der Teilbefehl dahinter nicht gezeichnet wird; der grafische Cursor allerdings bewegt sich. Zusammen mit dem Teilbefehl M ist B beinahe gleichbedeutend mit dem PSET-Befehl.

Der letzte Teilbefehl heißt

A	(Angle = Winkel)

Mit diesem Befehl können Sie eine Vierteldrehung des Koordinatensystems erreichen. Hinter A lassen sich dann folgende Werte eingeben:

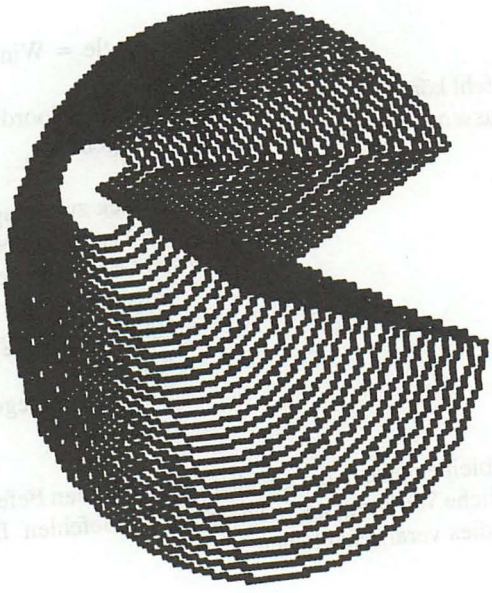
A0	zurück zur Ausgangsposition (0,0 ist links oben)
A1	90 Grad entgegen dem Uhrzeigersinn
A2	180 Grad entgegen dem Uhrzeigersinn
A3	270 Grad entgegen dem Uhrzeigersinn

Die Koordinaten bleiben jedoch die gleichen. Bei absoluten Befehlen bleibt dieser Befehl ohne sonderliche Wirkung, anders bei relativen Befehlen. Das folgende Programm kann Ihnen dies veranschaulichen.

```
10 SCREEN 2
20 COLOR ,1,1 : CLS
30 A$="bm=x;,=y;c=kl;s=sc;u5nglnfl"
40 X=INT(RND(1)*255)
50 Y=INT(RND(1)*192)
60 KL=INT(RND(1)*15)
70 SC=INT(RND(1)*10)
80 DRAW "a2xa$;"
90 GOTO 40
```

Das Ergebnis ist ein wahrer Pfeilregen. Wenn Sie Zeile 80 ändern und Zeile 75 einfügen, erhalten Sie eine Unzahl von Pfeilen, die in alle möglichen Richtungen weisen.

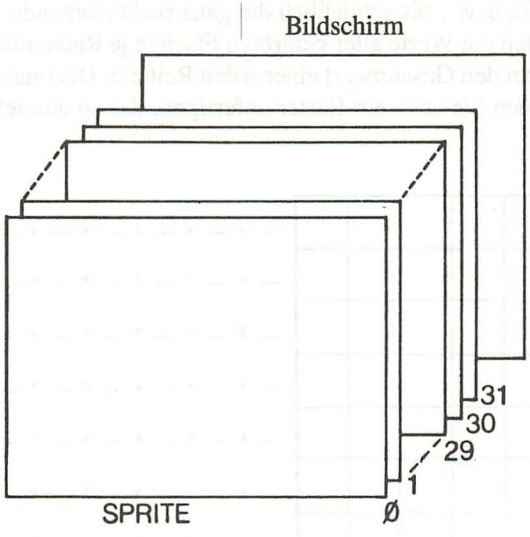
```
75 RI=INT(RND(1)*4)
80 DRAW ''A=RI;XA$;''
```



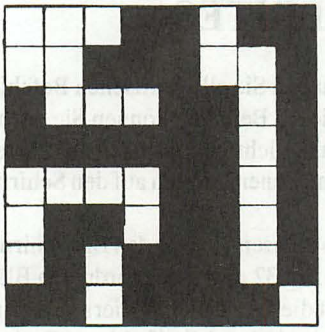
18. GRAFIK: SPRITES

In den vorigen Kapiteln haben Sie alle grafischen Befehle kennengelernt, über die der MSX verfügt. Neben diesen Befehlen können Sie auch mit Sprites arbeiten. Ein Sprite ist ein festes Muster aus Lichtpunkten in einem Raster von 8 mal 8 oder 16 mal 16. Sprites können in verschiedenen Größen auf den Schirm gebracht und als Ganzes bewegt werden.

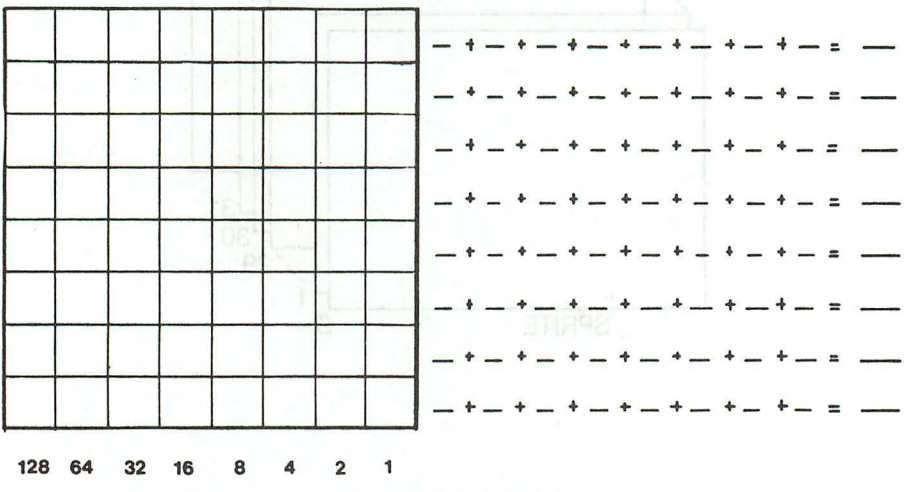
Beim Umgang mit Sprites müssen Sie sich den Bildschirm so vorstellen, als bestünde er aus einem Bildschirm mit 32 davor angeordneten Blättern durchsichtigen Zeichenpapiers. Auf jedes Blatt dieses Zeichenpapiers kann ein Sprite gebracht werden. Da alle 32 Blätter durchsichtig sind und die (farbigen) Sprites nicht, läßt sich mit einer Reihe Sprites eine Zeichnung anfertigen. Sprites, die teilweise hinter anderen Sprites verschwinden, werden nicht gezeichnet. Der Effekt ist der, daß Sprites auf den oberen Blättern über Sprites auf den darunterliegenden Blättern gezeichnet werden. Die Spriteflächen sind durchnumeriert, und zwar von 0 (der vorderen oder oberen) bis 31 (der letzten oder unteren). Hinter all den Spriteflächen liegt dann noch der eigentliche Bildschirm. Die folgende Darstellung soll Ihnen das ein wenig verdeutlichen.



Ein Sprite wird mittels eines Rasters aus 8 mal 8 Fächern angelegt, diejenigen Fächer, die auf dem Bildschirm erscheinen sollen, werden eingefärbt. Nehmen wir zum Beispiel einmal ein hockendes Äffchen.



Um dieses Äffchen herum sehen Sie allerlei Zahlen. Jede Reihe der angefertigten Zeichnung soll besonders betrachtet werden. Computertechnisch muß nun jede eingefärbte Fläche als eine 1 und jede nicht eingefärbte Fläche als eine 0 gesehen werden. Dieser binäre Code für jede Reihe ist die Angabe, mit der der Computer arbeiten kann. Sie selber können es sich aber einfacher machen; Sie müssen nur den „Wert" jeder einzelnen Fläche haben. So hat eine Fläche ganz links den Wert 128; die Fläche unmittelbar rechts davon hat den Wert 64; und die nächste Fläche wiederum rechts davon den Wert 32 usw., bis schließlich die ganz rechts liegende Fläche den Wert 1 hat. Dadurch, daß die Werte aller gefärbten Flächen je Reihe miteinander addiert werden, erhält man den Gesamtwert einer jeden Reihe in Dezimalangaben. Zu diesem Zwecke können Sie auch ein Raster anfertigen, das so aussieht:



Schauen wir uns einmal die oberste Reihe des Äffchens an. Die ganz links liegende Fläche ist nicht gefärbt: keine 128; die nächste Fläche ist ebenfalls nicht gefärbt: keine 64; das gleiche gilt auch für die folgende Fläche: keine 32. Und nun folgt eine einge-

färbte Fläche: 16, die nächste ebenfalls: 8. Die Fläche rechts davon wiederum ist nicht gefärbt: keine 4. Die beiden letzten Flächen sind gefärbt: 2 bzw. 1. Zählen wir diese Werte zusammen, erhalten wir 16+8+2+1=27. In genau der gleichen Weise müssen alle Reihen dieser Zeichnung kontrolliert und berechnet werden.

Nun haben Sie eine Zeichnung angefertigt und deren 8 Werte berechnet (für jede Reihe einen Wert). Wie sieht nun der Weg von den Werten zur Darstellung auf dem Bildschirm aus?

Zunächst einmal teilen Sie dem Computer mit, daß alle Werte zusammen einen Sprite bilden:

SPRITE$( )=CHR$( )+CHR$( )..... +CHR$( )

Hinter Sprite$ folgt die Spritenummer, die Sie diesem Sprite zuweisen wollen; im restlichen Programm wird mittels dieser Nummer auf den entsprechenden Sprite verwiesen. Hinter dem Gleichheitszeichen folgen nun 8 CHR$( )-Funktionen. In die Klammern dieser 8 Funktionen setzen Sie die acht Werte ein, die Sie beim Berechnen der Zeichnung erhalten haben. Und nun sehen Sie sich mal das folgende Programm an

```
10 SCREEN 2
20 SPRITE$(1)=CHR$(27)+CHR$(57)+CHR$(141)+CHR$(253)+CHR$(13)+CHR$(109)+CHR$(93)+CHR$(222)
30 PRESET (100,100)
40 PUT SPRITE 1,(100,100)
50 GOTO 50
```

In Zeile 20 wird der Sprite definiert. Sie finden dort alle Zahlen, die Sie bei der Berechnung der Zeichnung herausgefunden haben.

Zeile 30 bewegt den grafischen Cursor irgendwohin in die Mitte des Schirmes. Diese Zeile ist eigentlich überflüssig. Wenn Sie den Sprite nicht an eine bestimmte Stelle bringen, wird er dort erscheinen, wo sich der grafische Cursor gerade befindet (der grafische Cursor erscheint dann links oben in diesem Sprite).

In Zeile 40 wird allerdings nachträglich exakt die Plazierung des Sprites festgelegt. Dies machen Sie mit folgendem Befehl:

PUT SPRITE , (,), , (setze Sprite)

Dieser Befehl bestimmt, auf welchem durchsichtigen Blatt Zeichenpapier (Spritefläche) dieser Sprite stehen soll, wo der Sprite auf dieser Fläche erscheinen, in welcher Farbe er gezeichnet werden und um welchen Sprite es sich handeln soll.

Es gibt ja 32 Spriteflächen, die von 0 (der vorderen) bis 31 einschließlich (der hinteren) numeriert sind. Sprites, die auf einer der vorderen Flächen gezeichnet werden (das heißt also auf Flächen mit einer niedrigen Nummer), werden über die Sprites auf Spriteflächen mit einer höheren Nummer gezeichnet. Sie können auf jede Spritefläche nur einen einzigen Sprite bringen.

Die Position des Sprites wird mit den in der Klammer genannten Koordinaten bestimmt. Wenn Sie diese Koordinaten fortlassen, erscheint der Sprite dort, wo sich der grafische Cursor augenblicklich befindet.

Mit dem Farbcode, der direkt nach den Koordinaten eingegeben wird, kann die Farbe des Sprites festgelegt werden. Die zu wählenden Farbcodes gleichen denen des COLOR-Befehles. Fehlt die Angabe des Farbcodes, nimmt der Computer die augenblickliche Vordergrundfarbe an.

Zum Schluß müssen Sie dem Computer noch mitteilen, um welchen Sprite es sich handelt; hierzu geben sie als letzte Zahl die Spritenummer ein. Wenn Sie diese Nummer fortlassen, geht der Computer davon aus, daß Sie den Sprite mit der gleichen Nummer wie die Spritefläche meinen.

Besondere Erwähnung verdient die Möglichkeit, anstelle des CHR$( )-Befehles in Zeile 20 den Buchstaben einzugeben, der zu dem gewünschten Code gehört. Zeile 20 läßt sich so ändern:

```
20 SPRITE$(1)=CHR$(27)+CHR$(57)+CHR$(141)+CHR$(253)
+CHR$(13)+"m"+" "+CHR$(222)
```

Welche Buchstaben zu welchen Codes gehören, entnehmen Sie bitte der ASCII-Tabelle in der Anlage.

Spriteflächen

Damit Sie sehen, wie sich mit mehr als nur einer Spritefläche arbeiten läßt, zeichnet das folgende Programm drei verschiedene Sprites in drei Farben. Dadurch, daß die Sprites immer an eine andere Stelle gesetzt werden, werden sie sich irgendwann überschneiden. Derjenige Sprite auf der Fläche mit der niedrigsten Nummer wird dabei über den anderen Sprites erscheinen.

```
10 SCREEN 2,1
20 COLOR 4,10,10 : CLS
30 LINE (0,0)-(255,191),4
40 LINE (0,191)-(255,0),8
50 SPRITE$(1)=CHR$(27)+CHR$(57)+CHR$(141)+CHR$(253)+CHR$(13)+"m"+"O"+CHR$(222)
60 SPRITE$(2)=CHR$(1)+CHR$(3)+CHR$(7)+CHR$(15)+CHR$(27)+CHR$(63)+CHR$(127)+CHR$(255)
70 SPRITE$(3)=CHR$(16)+CHR$(48)+CHR$(112)+CHR$(255)+CHR$(255)+CHR$(112)+CHR$(48)+CHR$(16)
80 PUT SPRITE 1,(120,100),1,1
90 PUT SPRITE 2,(10,100),9,2
100 PUT SPRITE 3,(240,100),15,3
110 FOR Z=1 TO 300 : NEXT Z
120 FOR X=1 TO 650
130 PUT SPRITE 1,(120-X/10,100),1,1
140 PUT SPRITE 2,(10+X*1.5,100),9,2
150 PUT SPRITE 3,(240-X,100),15,3
160 FOR Z=1 TO 25 : NEXT Z
170 NEXT X
180 GOTO 180
```

Aus dem Effekt, daß Sprites übereinander hinweggleiten, bietet dieses Programm noch weitere interessante Dinge.

In den Zeilen 20-40 wird ein Schirm eingefärbt und zwei farbige Striche sind mit Hilfe der üblichen grafischen Befehle gezeichnet worden. Diese Striche bleiben unbeeinflußt von den Sprite-Befehlen, und sie sind die ganze Zeit über zu sehen. Die Spriteflächen sind vollkommen durchsichtig.

In Zeile 10 sehen Sie einen bekannten Befehl in neuer Form. Es handelt sich um

SCREEN

Bisher haben Sie zusammen mit SCREEN immer nur eine Zahl eingegeben, die sich auf den Schirmmodus bezog. Die zweite Zahl hinter SCREEN dient der Gestaltung der Sprites. Hier können Sie zwischen vier Zahlen wählen:

SCREEN , 0	8x8 Sprites, nicht vergrößert
SCREEN , 1	8x8 Sprites, vergrößert
SCREEN , 2	16x16 Sprites, nicht vergrößert
SCREEN , 3	16x16 Sprites, vergrößert

Die Sprites vergrößern heißt, die Sprites werden sowohl in der Höhe als auch in der Breite um einen Faktor 2 vergrößert. Im Programm oben wird mit 8x8 Sprites gearbeitet, die vergrößert sind.

Neben diesen beiden häufig gebrauchten Funktionen des SCREEN-Befehles gibt es noch drei weitere Funktionen, deren man sich allerdings seltener bedient. Der Vollständigkeit halber sollen sie hier auch genannt werden:

SCREEN Modus, Spriteform, Tastensignal, Baud-Geschwindigkeit, Drucker

Die dritte Zahl, die hinter SCREEN eingegeben werden kann, legt fest, ob die Tasten der Tastatur bei Betätigung ein kurzes Tik hören lassen. Wenn Sie an dieser Stelle nichts angeben, ändert sich nichts an der Situation, wie sie von den SCREEN-Befehlen bestand. Unmittelbar nach Einschalten des Computers erzeugt das Drücken einer Taste ein kurzes Tik. Wenn Sie eine 0 eingeben, bleibt die Tastatur stumm, geben Sie allerdings eine Zahl zwischen 1 und 255 ein, ist jede Tastenbetätigung mit einem Geräusch verbunden.

Die vierte mögliche Zahl bezieht sich auf die Geschwindigkeit, mit der Daten auf den Kassettenrekorder geschrieben oder vom Kassettenrekorder eingelesen werden. Diese Geschwindigkeit wird in „Baud" ausgedrückt. 10 Baud ist 1 Zeichen pro Sekunde. Geben Sie hier 1 ein, erfolgen Ein- und Ausgabe der Daten mit 1200 Baud (Standardwert); bei Angabe einer 2 werden die Daten mit 2400 Baud weitergeleitet. Manche kommerziellen Programmkassetten sind mit 2400 Baud aufgenommen. Mit dem SCREEN-Befehl müssen Sie zunächst Baud regeln, bevor Sie laden können.

Mit der letzten Zahl läßt sich der von Ihnen gebrauchte Drucker bestimmen. Wenn Sie mit einem richtigen MSX-Drucker arbeiten, werden alle Symbole des MSX in

der korrekten Weise zu Papier gebracht. Drucker, die nicht dem MSX-Standard entsprechen, können nicht alle diese Symbole wiedergeben. Bei Eingabe einer Zahl von 1 bis 255 druckt der Computer anstelle der unbekannten Symbole lediglich Leerstellen aus; geben Sie eine 0 ein, werden die speziellen MSX-Symbole ausgedruckt.

Zeile 110 in diesem Programm bewirkt eine Verlangsamung.

In Zeile 160 steckt eine lange Schleife.

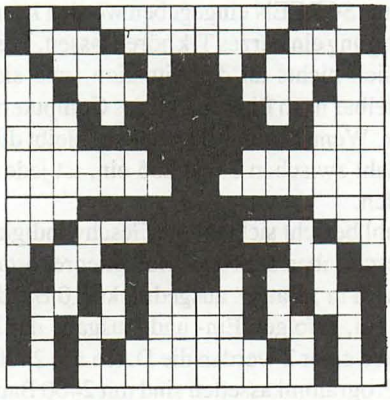
Die Zeilen 130-150 positionieren die Sprites. Jedesmal wenn die Schleife durchlaufen wird, werden die Sprites erneut positioniert. Da es auf jeder Fläche nur einen einzigen Sprite geben kann, braucht der Programmierer nicht darauf zu achten, ob die „alten" Sprites eventuell gelöscht werden.

Zeile 160 bewirkt wiederum eine Verlangsamung. Ohne diese Zeile würden sich die Sprites rascher über den Schirm bewegen.

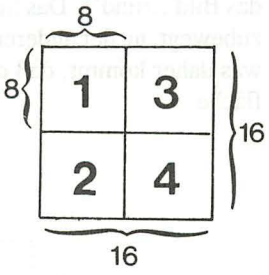
Es gibt 32 Spriteflächen. Sie können jedoch maximal 256 in der Größe 8x8 oder 64 Sprites in der Größe 16x16 herstellen. Der Reihe nach lassen sie sich dann auf eine Spritefläche bringen. Solch eine hohe Zahl von Sprites macht allerdings eine Menge Speicherkapazität erforderlich, das restliche Programm darf also nicht mehr sonderlich lang sein.

Große Sprites

Neben Sprites mit einer Größe von 8x8 kann man auch Sprites in einer Größe von 16x16 Lichtpunkten anfertigen. Im Grunde handelt es sich um nichts anderes als um die Summen vier kleiner Sprites. Die Reihenfolge, in der die Sprites addiert werden müssen, sieht so aus:



1 = links oben
2 = links unten
3 = rechts oben
4 = rechts unten



Eine Vogelspinne:

Das Programm, mit dem Sie diese Vogelspinne über den Bildschirm krabbeln lassen, lautet:

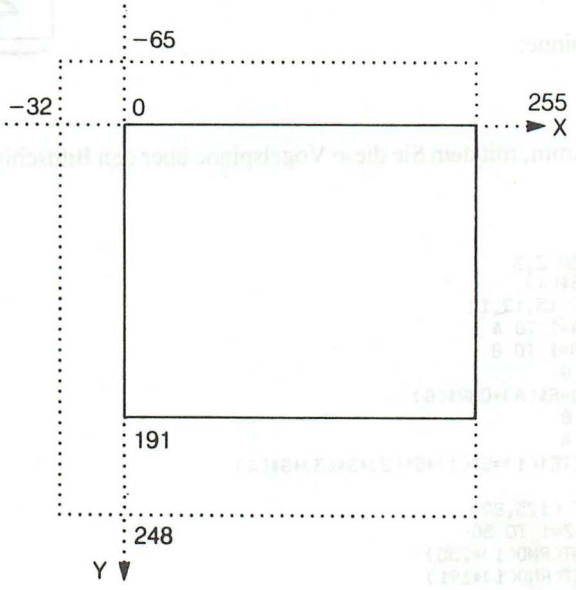
```
10 SCREEN 2,3
20 DIM S$(4)
30 COLOR 15,12,12
40 FOR A=1 TO 4
50 FOR B=1 TO 8
60 READ G
70 S$(A)=S$(A)+CHR$(G)
80 NEXT B
90 NEXT A
100 SPRITE$(1)=S$(1)+S$(2)+S$(3)+S$(4)
110 REM
120 PSET (125,80)
130 FOR Z=1 TO 50
140 X=INT(RND(1)*255)
150 Y=INT(RND(1)*191)
160 DRAW "nm=x;,=y;"
170 NEXT Z
180 REM
190 FOR Y=1 TO 700 STEP .5
200 PUT SPRITE 1,(125,190-Y),1,1
210 NEXT Y
220 GOTO 220
230 DATA 32,168,164,148,213,75,35,31
240 DATA 1,3,31,23,35,69,136,176
250 DATA 4,21,37,41,171,242,196,248
260 DATA 128,192,248,232,196,162,17,13
```

Mit diesem Programm bewegt sich die aus vier Sprites zusammengesetzte Vogelspinne. Beachten Sie, wie die große Datenmenge in den Spritedefinitionen untergebracht wird. In den Zeilen 40-90 werden vier mal acht Daten gelesen. Diese Daten befinden sich zu je acht in einem String mit den notwendigen CLR$-Befehlen. In Zeile 100 werden die so gebildeten Strings aneinandergehängt, um eine lange Spritedefinition zu bilden. Mit den Zeilen 120-170 wird auf dem Hintergrundschirm ein Spinnennetz dargestellt.

Die Zahlen 190-210 ermöglichen es, daß sich die Spinne über den Schirm bewegt.

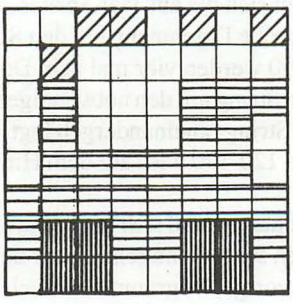
Hier entdecken Sie noch einen sehr wichtigen Aspekt der Sprites beim MSX. Sowohl in horizontaler (siehe das vorige Programm) als auch in vertikaler Richtung ist

das Bild „rund". Das heißt, daß ein Sprite, der sich auf die eine Seite des Schirmes zubewegt, an der anderen Seite wieder auftaucht; dazwischen liegt eine kleine Pause, was daher kommt, daß die Spriteflächen ein wenig größer sind als die Schirmoberfläche:



Sprites mit mehreren Farben

Oben haben Sie gelesen, daß ein Sprite nur eine einzige Farbe besitzen kann. Das stimmt auch. Um allerdings einen Sprite mit mehreren Farben zu erhalten, können Sie einige Sprites erstellen und diese dann übereinander anordnen; der Hintergrund des Sprite ist ja immer durchsichtig. Nehmen wir als Beispiel einmal einen Zug.



Und hier haben Sie auch das Programm, mit dem Sie den Zug fahren lassen können:

```
10 SCREEN 2,1
20 DIM S$(3)
30 COLOR 15,1,1
40 FOR A=1 TO 3
50 FOR B=1 TO 8
60 READ G
70 S$(A)=S$(A)+CHR$(G)
80 NEXT B
90 SPRITE$(A)=S$(A)
100 NEXT A
110 SPRITE$(1)=S$(1)+S$(2)+S$(3)
120 REM
130 LINE (0,80)-(255,190),10,BF
140 LINE (0,0)-(255,79),7,BF
150 FOR X=1 TO 900
160 IF INT(X/5)=X/5 THEN PUT SPRITE 1,(255-X,64),15,1
170 PUT SPRITE 2,(255-X,64),12,2
180 PUT SPRITE 3,(255-X,64),9,3
190 NEXT X
200 GOTO 200
210 DATA 53,64,0,0,0,0,0,0
220 DATA 0,0,71,70,126,254,153,0
230 DATA 0,0,0,0,0,0,102,102
```

Dieses Programm gleicht weitgehend dem vorangegangenen. Beachten Sie Zeile 160. Mit der Bedingung IF INT (X/5)=INT/5 erreichen Sie, daß sich der Zug fünfmal so rasch bewegt wie der Rauch; das macht ihn noch viel echter.

Zusammenstöße und Verstecken von Sprites

Es kann vorkommen, daß zwei Sprites miteinander kollidieren; oder anders ausgedrückt: Es sieht für uns so aus, als würden zwei Sprites miteinander kollidieren. In Wahrheit sind die Sprites ja jeweils an ihre Spritefläche gebunden und können sich niemals berühren. Es ist mitunter recht nützlich, wenn man den Computer bestimmen läßt, wann genau sich zwei Sprites überlagern. Damit lassen sich allerhand Spiele garnieren; beispielsweise dann, wenn ein außerirdisches Raumschiff vernichtet werden soll. Sind sowohl das Geschoß als auch das Raumschiff je ein Sprite, ist mit der folgenden Technik ganz einfach festzustellen, wann ein Treffer erfolgt ist.

Zur Feststellung einer Kollision befehlen Sie:

ON SPRITE GOSUB

Sie kennen diesen Befehl bereits aus der Reihe der ON..-Befehle. Dieser Befehl muß also auch hier zunächst wirksam gemacht werden mit

SPRITE ON

Um dafür zu sorgen, daß die Subroutine nicht fortlaufend aufgerufen wird (die Sprites überlagern sich längere Zeit), wird die Kollisionsbestimmung der Sprites mit dem ersten Befehl der Subroutine (SPRITE OFF) wieder abgeschaltet.

Da der ON SPRITE GOSUB-Befehl nicht bestimmen kann, wo welche Sprites miteinander kollidieren, sondern nur, daß zwei Sprites kollidieren, müssen sich im Rest des Programmes Befehle befinden, die bestimmen, wo die Kollision stattgefunden hat. In unserem nächsten Programm haben wir die Farben durch Überlagerung von drei Sprites.

Das Programm führt Ihnen ein Unglück vor, in das ein Flugzeug und ein Zug verwickelt sind.

```
10 ON SPRITE GOSUB 280
20 SCREEN 2,1
30 DIM S$(3)
40 COLOR 15,1,1
50 FOR A=1 TO 3
60 FOR B=1 TO 8
70 READ G
80 S$(A)=S$(A)+CHR$(G)
90 NEXT B
100 SPRITE$(A)=S$(A)
110 NEXT A
120 SPRITE$(1)=S$(1)+S$(2)+S$(3)
130 REM
140 LINE (0,80)-(255,190),10,BF
150 LINE (0,0)-(255,79),7,BF
160 SPRITE ON
170 FOR X=1 TO 900 STEP 2
180 PUT SPRITE 1,(255-X,64),1,1
190 PUT SPRITE 2,(100-X*5,10+X/10),1,2
200 PUT SPRITE 3,(116-X*5,10+X/10),1,3
210 FOR Z=1 TO 40 : NEXT Z
220 NEXT X
230 GOTO 230
240 DATA 53,64,71,70,126,254,157,102
250 DATA 0,1,3,127,255,7,3,1
260 DATA 2,199,143,255,248,0,128,192
270 REM Kollisionsroutine
280 SPRITE OFF
290 KEY OFF
300 FOR Q=1 TO 14 : BEEP : BEEP : SCREEN 1 : COLOR ,Q,Q+1 : NEXT Q
310 END
```

Das Verstecken von Sprites erfolgt mit der Eingabe eines besonderen Y-Wertes als Koordinate. Geben Sie den Wert 208 als Y-Koordinate an, wird der Sprite auf der betreffenden Fläche gelöscht, desgleichen alle Sprites auf Spriteflächen mit einer höheren Flächennummer.

Wenn Sie als Wert der Y-Koordinate 209 eingeben, wird nur die eine betreffende Spritefläche gelöscht. Und nun verändern sie einmal die Zeilen 300 und 310 im Programm oben.

```
300 PUT SPRITE 1,(255-X,209),1
310 GOTO 310
```

Bei dem Zusammenstoß wird der Zug nun in Rauch aufgehen, und das Flugzeug rührt sich vor Schreck nicht mehr von der Stelle. Ändern Sie Zeile 300 in

```
300 PUT SPRITE 1,(255-X,208),1,1
```

Jetzt verschwinden beide Fahrzeuge gleichzeitig.

Vier Sprites und Bewegung

Es gibt bei der Verwendung von Sprites eine Einschränkung. Auf einem beliebigen horizontalen Strich können Sie niemals mehr als vier Sprites gleichzeitig darstellen lassen. Wenn Sie es dennoch versuchen, wird der Rest ganz einfach nicht gezeichnet.

Es ist sehr gut möglich, mit Sprites sich bewegende Gegenstände zu erstellen. Nehmen wir doch mal das inzwischen bekannte Äffchen und lassen es über den Bildschirm turnen.

```
10 SCREEN 2,1
20 DIM S$(3)
30 COLOR 15,1,1
40 FOR A=1 TO 3
50 FOR B=1 TO 8
60 READ G
70 S$(A)=S$(A)+CHR$(G)
80 NEXT B
90 SPRITE$(A)=S$(A)
100 NEXT A
110 FOR Y=174 TO 0 STEP -2
120 IF INT(Y/12)=Y/12 THEN 125 ELSE 130
125 PUT SPRITE 2,(X,208),4,2:PUT SPRITE 1,(100+Y/2,Y),4,1:FOR Z=1 TO 150:NEXT Z
130 PUT SPRITE 1,(100+Y/2,209),4,1
140 PUT SPRITE 2,(100+Y/2,Y),4,2
150 PUT SPRITE 3,(100+Y/2,Y+16),4,3
160 NEXT Y
170 GOTO 170
180 DATA 27,57,141,253,13,109,93,222
190 DATA 27,57,13,253,141,13,13,14
200 DATA 16,16,48,32,32,96,0,0
```

19. GERÄUSCHE UND MUSIK

So wie eine Schreibmaschine eine Klingel hat, mit der sie anzeigt, wenn eine Zeile zu Ende ist, so haben auch Computer bereits seit langem die Möglichkeit, einen einfachen „Beep" erklingen zu lassen. Dieser Ton hat den Benutzer immer auf etwas aufmerksam machen wollen.

Als dann die ersten Computerspiele auf den Markt kamen, reichte so ein einfacher Ton nicht mehr aus. Heute kann man diesen Ton bei den meisten Mikrocomputern regeln, so daß (einstimmige) Geräusche erzeugt werden können.

Die MSX-Computer verfügen zur Geräuscherzeugung über eine besondere Vorrichtung, den sogenannten PSG (Programmable Sound Generator = programmierbarer Geräuschgenerator) vom Typ AY-3-8912 und entwickelt von General Instruments in den USA. Es handelt sich dabei um den vermutlich populärsten Geräuschchip für Mikrocomputer, der im Augenblick auf dem Markt ist. Dieser Geräuschchip besteht aus drei verschiedenen Tongeneratoren (und er erzeugt sehr schöne Töne). Mit diesen drei äußerst vielseitigen Tongeneratoren ist es möglich, eine beinahe unbegrenzte Zahl von Geräuschen zu erzeugen. Die Anzahl der möglichen verschiedenen Töne ist so groß, daß der MSX nicht nur verschiedene Pieptöne von sich geben, sondern auch dreistimmige Musikstücke vorspielen kann. In diesem Kapitel werden ausschließlich die Programmierprinzipien vorgestellt werden, natürlich zusammen mit Programmbeispielen. Da Musik letztlich eine Frage des Geschmacks ist und der MSX einfach zu viele Möglichkeiten der Musikerzeugung bietet, sollten Sie am besten selber herausfinden, was Ihnen hier besonders gefällt. Besondere Experimentierprogramme werden jedoch eingehend dargestellt.

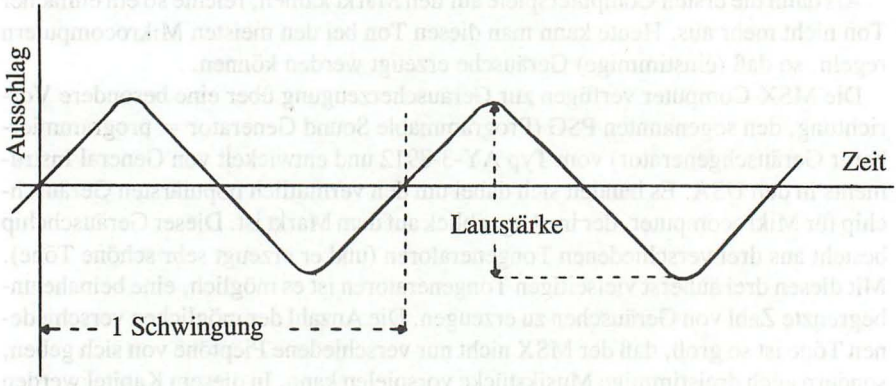
Geräusch

Geräusche sind Luftschwingungen, die beispielsweise durch das Schwingen einer Saite (vielleicht noch durch einen Klangkörper verstärkt), einer Trommelbespannung, eines Röhrchens (bei einem Fagott) u.ä.m. erzeugt werden. Luftschwingungen dürfen nicht als ein Auf- und Niederschwingen verstanden werden, sondern eher als ein Aufeinanderfolgen unterschiedlich starker Luftverdichtungen. Die Luft wird hintereinander kurz verdichtet und dann wieder „auseinandergezogen".

Auf dem Papier werden diese Schwingungen meist als Welle dargestellt; diese Form der Wiedergabe ist das Ergebnis einer Gegenüberstellung der Faktoren Zeit und Amplitude. Mit Amplitude ist hier der Unterschied zwischen Verdichtung und Entspannung gemeint. Bei einem Geräusch gibt es zwei Basisgrößen: Volumen und Frequenz der Schwingung. Der Umfang des Volumens ist hörbar. Auf dem Papier kann man das Volumen am Wellenausschlag ablesen; je höher die Welle ist, um so lauter ist der Ton.

Auch die Frequenz kann man hören; sie sagt etwas über die Tonhöhe aus. Je größer die Frequenz ist, desto höher erklingt der Ton. Die Frequenz gibt an, wie oft Verdichtung und Verdünnung der Luft im Verlaufe einer einzigen Sekunde hintereinander

erfolgen; man spricht auch von der Anzahl der Schwingungen pro Sekunde. Die Maßeinheit hierzu ist Hertz (Hz = Anzahl der Schwingungen pro Sekunde). Das menschliche Ohr vermag Geräusche von 30 Hz (einem sehr tiefen Brummen) bis hin zu ungefähr 15 000 Hz (einem sehr hohen Pfeifen) wahrzunehmen. Menschliche Sprache und Musik spielen sich etwa zwischen 100 und 4000 Hz ab.



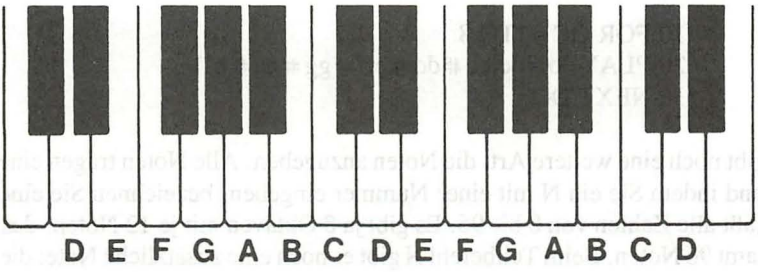
Die BASIC-Steuerung

Die Erzeugung von Geräuschen durch den MSX kann vom Benutzer mit zwei Befehlen bestimmt werden: PLAY und SOUND.
Der PLAY-Befehl dient vornehmlich der Erzeugung von Musik. Zur Grundlage dieses Befehls hat man die Verwendung eines Pianos genommen. Der Befehl funktioniert so ähnlich wie der DRAW-Befehl; das heißt, dem Befehl folgt ein String, in dem sich eine Reihe von Teilbefehlen befinden. Dadurch, daß der Benutzer aus diesen Teilbefehlen irgendwelche auswählt, kann er jede gewünschte Musik erzeugen.
Der SOUND-Befehl dient eher der Ezeugung bestimmter Geräusche. Er wirkt wesentlich direkter auf den Geräuschgenerator, und das Geräusch kann noch individueller eingegrenzt werden. Es ist durchaus möglich, mit dem SOUND-Befehl die gleichen Effekte zu erzielen wie mit dem PLAY-Befehl, aber das erfordert einige Mühe. Aus diesem Grunde ist es sinnvoller, die Grobeinteilung so zu belassen, wie sie ist, und mit dem SOUND-Befehl nur solche Geräusche zu erzeugen, die das Vermögen des PLAY-Befehles überschreiten würden. Sie werden feststellen, daß es hier natürlich Überschneidungen gibt.

Der PLAY-Befehl

Mit dem PLAY-Befehl kann der MSX Musik machen. Sie müssen dabei allerdings bedenken, daß ein Teil der Wiedergabe von Musik von den Möglichkeiten Ihres Fernsehgerätes oder Monitors abhängt. Der MSX verfügt über keinen eigenen Lautspre-

cher, sondern er nimmt den vom Fernsehgerät oder Monitor. Wenn Ihnen ein bestimmtes Geräusch zu laut oder zu leise erscheint, können Sie das durch Korrektur der Befehle ändern; das gleiche läßt sich allerdings auch durch Regulieren der Lautstärke von Fernseher oder Monitor erreichen. Das hängt davon ab, ob Sie Ihr Fernsehgerät so eingestellt haben, daß Sie das Klicken der Tastatur und das Piepsen des Computers hören können.
Die Tastatur eines Pianos diente als Grundlage für den Entwurf des PLAY-Befehles. Beinahe alles, was Sie auf einem Klavier spielen können, läßt sich auf dem MSX erzeugen; darüber hinaus aber kann der MSX noch wesentlich mehr. Es ist ratsam, sich bei der Behandlung der einzelnen Teilbefehle immer wieder ein Klavier zu vergegenwärtigen. Besser noch wäre es natürlich, wenn Sie ein echtes Piano besäßen.



Klaviatur

Die Tasten eines Klaviers bezeichnen verschiedene zu Oktaven zusammengefaßte Noten. Eine Oktave setzt sich aus den acht ganzen Noten C, D, E, F, G, A und H zusammen. Die folgende Note wäre jetzt das C der nächsthöheren Oktave. Daneben gibt es beim Klavier noch schwarze Tasten, die halbe Noten bezeichnen (in der Notenschrift mit # angedeutet). Insgesamt besteht eine Oktave also aus C, C#, D, D#, E, F, F#, G, G#, A, A#, und H.
Beim PLAY-Befehl wird der Tonbereich in genau der gleichen Weise bezeichnet; um eine bestimmte Note zu spielen, sagen Sie dem MSX einfach, welche Note das ist!

```
PLAY "A"
```

Wenn Sie das eingeben und auf die RETURN-Taste drücken, hören Sie über ihren Fernseher einen Ton. Das ist das A der mittleren Oktave auf dem Klavier.
Die schwarzen Tasten auf dem Klavier (die halben Noten) deuten Sie mit einem #-Zeichen, einem +-Zeichen oder einem --Zeichen. # oder + bezeichnen einen Ton der um einen Halbtonschritt über dem angegebenen Ton liegt. Ein − bezeichnet eine um einen Halbtonschritt tiefere Note. Es ist nicht möglich, Halbtonschritte anzugeben, die es auf dem Klavier nicht gibt.

```
PLAY ''C#''
PLAY ''C+''
PLAY ''D-''
```

Diese drei Befehle erzielen alle das gleiche Resultat.
Um alle Noten einer Oktave hören zu können, befehlen Sie:

```
PLAY ''CC#DD#EFF#GG#AA#H''
```

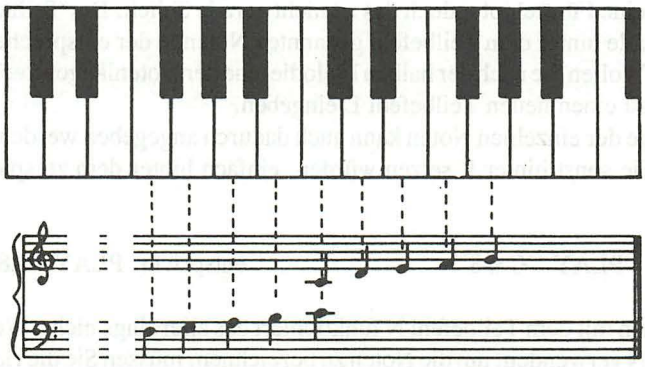
Der MSX umfaßt wie die meisten Klaviere auch acht Oktaven. Diese bezeichnen Sie mit dem Großbuchstaben O (nicht mit einer Null - 0 - zu verwechseln), dem die Nummer der gerade gewünschten Oktave folgt. Wenn Sie nichts weiter eingeben, hören Sie die Oktave Nummer 4.

```
10 FOR OC=1TO 8
20 PLAY''o=oc;cc#dd#eff#gg#aa#h''
30 NEXT OC
```

Es gibt noch eine weitere Art, die Noten anzugeben. Alle Noten tragen eine Nummer, und indem Sie ein N mit einer Nummer eingeben, bezeichnen Sie eine Note. N umfaßt alle Zahlen von 0 bis 96. Es gibt ja 8 Oktaven mit je 12 Noten, das macht insgesamt 96 Noten. Beim Teilbefehl N gibt es noch eine zusätzliche Note: die Stille. Wenn Sie hinter N eine 0 eingeben, erklingt nichts. N1 entspricht dann C # der ersten Oktave (O1). N 96 bezeichnet das C in der neunten Oktave (läßt sich mit Teilbefehl 0 nicht erreichen). O1C (das C der ersten Oktave) ist mit Teilbefehl N nicht zu erzielen.

```
10 FOR NO=0 TO 96
20 PLAY ''N=NO;''
30 NEXT NO
```

Dieses Programm führt zu dem gleichen Ergebnis wie das vorhergehende Programm. Dadurch, daß die PLAY-Befehle alle voneinander getrennt sind, erfolgt die Wiedergabe der verschiedenen Töne ebenfalls mit ein wenig Verzögerung. Sie werden feststellen, daß die ganz hohen Töne nicht mehr so schön klingen. Die Frequenz ist einfach zu hoch.



Klaviatur

Nun, da die verschiedenen Noten festliegen, ist es möglich, zu musizieren. Eine ganz einfache Art ist die Erstellung einer kleinen Liste mit Werten, die der MSX durchgehen soll. Wenn Sie diese Liste sehr sorgfältig zusammenstellen, kann der Computer eine kleine Melodie spielen.

```
10 FOR A=1 TO 16
20 READ NO
30 PLAY ''N=NO;''
40 NEXT A
50 DATA 36,38,40,38,36,40,38,36,36,35,33,35,36,33,35,36
```

Es ist zwar hübsch, wenn der Computer eine Melodie zum besten gibt, aber es klingt nicht besser, als ein Klavierschüler in seiner ersten Unterrichtsstunde.

Jeder, der Musik macht, lernt schon bald, daß nicht jeder Ton gleich lang klingen darf. Die einzelnen Noten werden also unterschieden nach:

L1	Ganze Note
L2	Halbe Note
L4	Viertel Note
L8	Achtel Note
L16	Sechzehntel Note
L32	Zweiunddreißigstel Note

Die Länge der Noten wird mit dem Teilbefehl L angegeben. In der Liste oben sehen Sie, daß L1 ganze Noten bewirkt, L2 halbe Noten usw. Der höchste Wert hinter L ist 64. Sie können auch dazwischenliegende Notenlängen verwenden. So erreichen

Sie mit L5 eine Fünftelnote, doch das ist nicht gerade üblich. Der Teilbefehl L sorgt dafür, daß alle hinter dem Teilbefehl genannten Noten in der entsprechenden Länge erklingen. Wollen Sie nach der halben Melodie eine der Notenlängen verändern, können Sie dazu einen neuen Teilbefehl L eingeben.

Die Länge der einzelnen Noten kann auch dadurch angegeben werden, daß Sie die Zahl, die Sie sonst hinter L setzen würden, einfach hinter dem zu spielenden Ton eingeben.

PLAY "C#8" entspricht: PLAY "L8C#"

Zusammen mit dem Teilbefehl N funktioniert das allerdings nicht. Wollen Sie den Teilbefehl N verwenden, um die Noten zu bezeichnen, müssen Sie die richtige Länge der Töne immer mit dem Teilbefehl L eingeben. Beim nächsten Programm sehen Sie, wie das geht. In diesem Programm diente der Teilbefehl N zur Bezeichnung der verschiedenen Noten. Das hat den Vorteil, daß die unterschiedlichen Noten als Ziffern in einer Datenliste untergebracht werden können.

Die Tonlänge wird bei jeder einzelnen Note neu mit dem Teilbefehl L bezeichnet. Das sieht vielleicht umständlich aus, aber auch hier besteht der Vorteil, daß die Daten als Zahlen abgespeichert werden können.

```
10 PLAY "t250"
20 FOR Z=1 TO 2
30 FOR A=1 TO 43
40 READ NO,LE
50 PLAY "l=le;n=no;"
60 NEXT A
70 LE=LE/2
80 PLAY "l=le;n=no;"
90 RESTORE
100 NEXT Z
110 DATA 36,2,38,4,40,2,38,2,43,2,31,2,31
120 DATA 1,33,2,35,4,36,2,35,2,36,2,40,2
130 DATA 38,1,35,2,36,4,38,2,35,2,38,2,40,2
140 DATA 41,2,38,2,31,2,31,2,36,2,38,2,40,2
150 DATA 43,2,43,2,41,2,40,2,38,2,36,2,31,2,40,2
160 DATA 38,4,36,1,36,2,36,2,45,2,45,2,43,2,35,2,36,1
```

Wie bereits bei den DRAW-Teilbefehlen muß Variablen, die innerhalb des Strings eines PLAY-Befehles verwendet werden, ein Gleichheitszeichen und ein Semikolon vorangehen.

Schon in der ersten Zeile des Programms oben sehen Sie einen neuen Teilbefehl: das T. Dieser Befehl steht für Tempo, und er bestimmt die Geschwindigkeit, mit der die Musik erklingt. T kann jeden ganzen Wert zwischen 32 und 255 erhalten; 32 ist langsam, 255 sehr schnell. Die Schnelligkeit einer Melodie hängt natürlich auch vom Vorhandensein ganzer und halber Noten ab.

Wie die Teilbefehle O und L muß auch der Teilbefehl T nur einmal gegeben werden, wonach das Tempo beibehalten wird, bis ein neuer Teilbefehl T erfolgt.

Nun, wie sieht es aus; erkennen Sie die kleine Melodie?

Ein Teilbefehl, der in einer Melodie für eine Pause sorgt, lautet R. Er steht für Rest (Ruhe). Die Zahlen, die hinter R folgen, gleichen denen, die bei L stehen können, und sie bezeichnen eine Pause, die einer ganzen, einer halben, einer achtel Note usw. entspricht.

Eine zusätzliche Möglichkeit bei der Bestimmung der Tonlänge bietet der Punkt (.). Wenn Sie hinter einer bestimmten Note einen Punkt eingeben, wird diese Note um die Hälfte verlängert; hinter einem R kann dieser Punkt ebenfalls stehen.

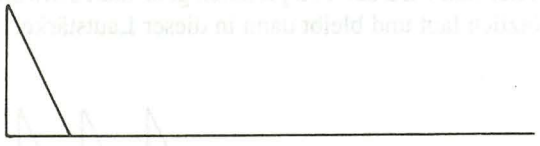
Die Lautstärke der gespielten Melodie kann auf zwei Arten beeinflußt werden: entweder mit dem Lautstärkeregler am Fernsehgerät oder mit dem Teilbefehl V.

Hinter dem V für Volume geben Sie eine Zahl zwischen 0 und 15 ein. Die normale Einstellung zu Beginn ist 8.

Alle bisher genannten Teilbefehle des PLAY-Befehles geben an, welches Geräusch der Computer über das Fernsehgerät erklingen läßt. Aber bei allen diesen Tönen muß doch festgestellt werden, daß sie ziemlich „technisch" klingen. Die beiden letzten Teilbefehle von PLAY ermöglichen es dem Computer, ein bestimmtes Instrument zu imitieren. Natürlich können Sie mit diesen Teilbefehlen auch erreichen, daß die Töne des MSX überhaupt keinem bekannten Instrument mehr ähneln, sondern rein synthetisch klingen.

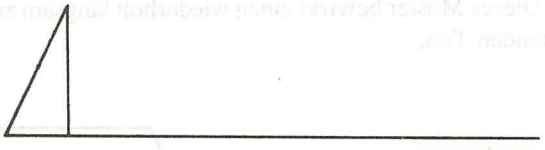
Einer dieser Teilbefehle heiß S, und er steht für Sound (Geräusch). Der Befehl SOUND entspricht ein wenig diesem Teilbefehl. Mit S veranlassen Sie den Computer dazu, einen Ton nach einem bestimmten Wellenmuster erklingen zu lassen, das heißt, die Lautstärke dieses Tones bleibt nicht die ganze Zeit über gleich, sondern sie verändert sich entsprechend einem bestimmten Muster. Der MSX bietet dem Benutzer 8 verschiedene Wellenmuster (englisch = envelope).

```
S = 0, 1, 2, 3 of 9
```



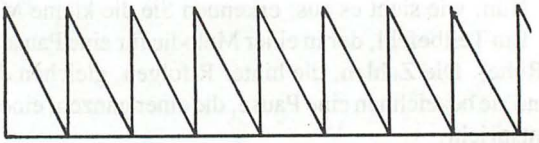
Dieses Muster bewirkt einen rasch zu voller Lautstärke anschwellenden Ton, der danach allmählich wieder leiser wird.

```
S = 4, 5, 6, 7 of 15
```



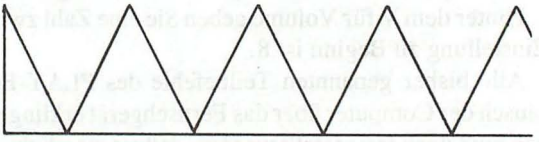
Bei diesem Muster wird der Ton ganz langsam immer lauter und bricht danach abrupt ab.

S = 8



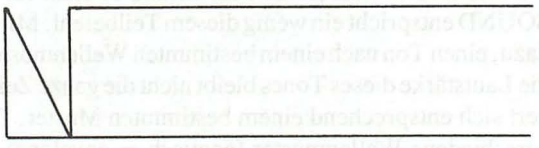
Hier wird der Ton rasch ganz laut und anschließend allmählich immer leiser; dies wird wiederholt.

S = 10



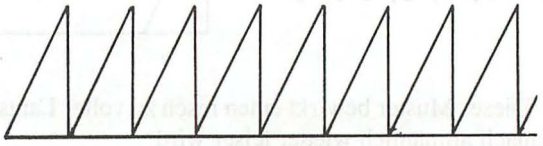
Mit diesem Wellenmuster erreichen Sie einen Ton, der schnell laut ist und dann immer auf- und abschwillt.

S = 11



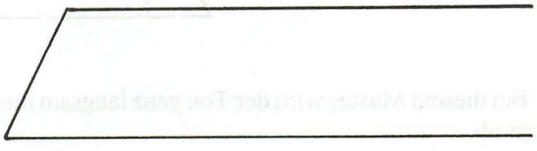
Hier nun wird der Ton plötzlich ganz laut, schwillt allmählich ab, wird wiederum plötzlich laut und bleibt dann in dieser Lautstärke.

S = 12



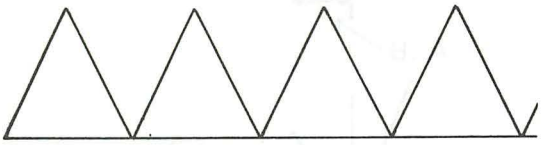
Dieses Muster bewirkt einen wiederholt langsam zunehmenden und abrupt abbrechenden Ton.

S = 13



Der hier erzeugte Ton nimmt in der Lautstärke langsam zu und erklingt dann in dieser Lautstärke fort.

S = 14



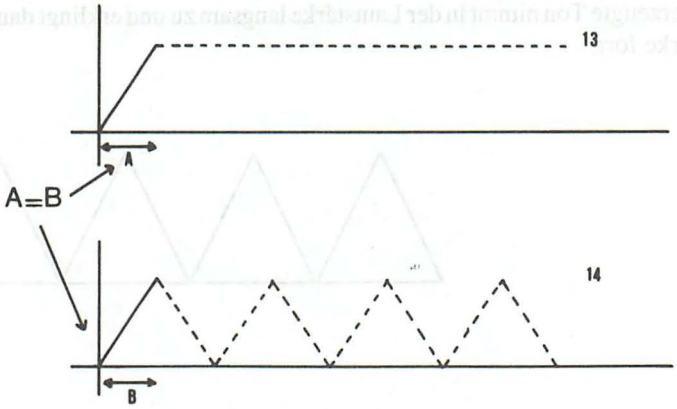
Hiermit wird ein gleichmäßig in der Lautstärke auf- und abschwellender Ton erzeugt.

Das folgende Programm zeigt Ihnen, was die verschiedenen Wellenmuster bei den zwölf Noten einer Oktave bewirken. Beachten Sie, daß viele Wellenmuster gleich sind. Es ist möglich, 16 verschiedene Zahlen einzugeben, aber Sie erhalten „nur" acht verschiedene Wellenmuster. Wenn Sie keinen Wert eingeben, wählt der MSX selbsttätig das Wellermuster 13.

```
10 PLAY "t90m700011"
20 FOR GO=0 TO 10 STEP 2
25 PRINT : PRINT "Wellenmuster:";GO
30 PLAY "s=go;cdef"
40 IF PLAY(1)<>0 THEN GOTO 40
50 NEXT GO
```

In diesem Programm finden Sie in Zeile 10 den letzten neuen PLAY-Teilbefehl.

Es handelt sich hier um den Teilbefehl M für Magnitude. Mit ihm läßt sich bestimmen, wie lange ein bestimmtes Wellenmuster erklingen soll. Die Zahl bezeichnet die Länge des Zyklus, in dem die mit dem Befehl S gewählte Wellenform wiederholt wird. Je höher diese Zahl ist, desto länger ist der Zyklus. Hinter M können Sie eine Zahl zwischen 1 und 65535 eingeben. Langer Zyklus besagt, daß das Wellenmuster lange erklingt. Wenn der Zyklus lang ist, dauert es auch länger, bis das Wellenmuster wiederholt wird. Haben Sie also hinter dem Teilbefehl M sehr hohe Werte eingegeben, wird der Unterschied zwischen den verschiedenen Wellenmustern zu gering, um noch wahrgenommen zu werden. Werden etwa Wellenmuster S13 und S14 sehr gedehnt (M ist sehr groß) und desgleichen die Zeit, die dem Ton zugemessen wird, damit er erklingen kann (hohes Tempo, das heißt, T ist hoch; eine kurze Note bedeutet L ist hoch), so kann nur ein Teil des Wellenmusters erklingen. Dadurch ist es unter Umständen ausgeschlossen, noch irgendeinen Unterschied festzustellen:



Der Wert von M beträgt zu Beginn (vorausgesetzt, Sie geben nichts anderes ein), 255. Die Teilbefehle S und M müssen nur einmal eingegeben werden; sie gelten dann so lange, bis eine neue Eingabe erfolgt. Der Klang einer Melodie hängt natürlich auch vom Tempo (T) und von der Notenlänge (L) ab. Versuchen Sie einmal, im letzten Programm hier da etwas zu ändern, und achten Sie dann auf das Ergebnis. Das Programm, wie es oben aussieht, hat alle Wellenmuster gut gedehnt, so daß Sie die Unterschiede noch klar wahrnehmen können. Zugleich können Sie hören, daß sich auch mit dem PLAY-Befehl ungewohnte Geräusche erzielen lassen.

Ein Befehl aus dem letzten Programm haben wir noch nicht besprochen. Das ist der PLAY-Befehl. Dieser ist nicht mit dem PLAY-Befehl zu verwechseln, der der Erzeugung von Musik dient. Mit dem Befehl PLAY() kann kontrolliert werden, ob der Computer eine Melodie bereits zu Ende gespielt hat. Manchmal kann das ziemlich nützlich sein, denn der Computer erzeugt die einzelnen Geräusche wesentlich schneller, als sie am Ende tatsächlich erklingen. Die Erzeugung eines Geräusches, das nach dem PLAY-Befehl eine Zeit lang erklingen soll, kostet den Computer nur den Bruchteil einer Sekunde. Wenn der Computer schließlich alle Geräusche produziert hat, wird ein Teil dieser für den Lautsprecher bestimmten Geräuschbefehle in den Puffer gegeben. Mit Hilfe des PLAY()-Befehles läßt sich nun überprüfen, ob der Puffer leer ist. Wenn Sie das zuletzt aufgeführte Programm ohne Zeile 40 laufen lassen, werden Sie feststellen, daß der Computer die Schleife bereits mehrmals durchlaufen hat, während Sie erst einige wenige Töne vernehmen konnten. Der Computer ist in der Erzeugung der Geräusche also wesentlich weiter als in deren Vorführung; und alles, was Sie noch nicht gehört haben, befindet sich im Puffer. Der Befehl

```
PLAY(0)
```

kontrolliert den Puffer von Geräuschgenerator 0 (dem einzigen, den wir bislang gebraucht haben). Wenn dieser Puffer leer ist, wird der Wert 0 mitgeteilt, andernfalls wäre der Wert von PLAY(0) gleich −1.

Das tatsächlich vom MSX erzeugte Geräusch ist vornehmlich von einem der Teilbefehle abhängig. Jede Kombination von Teilbefehlen bewirkt immer andere Effekte. Es läßt sich sehr leicht vorhersagen, was die Wahl einer bestimmten Note bewirkt; die Resultate der S- und M-Teilbefehle können allerdings recht überraschend sein.

Wenn Sie das nächste Programm laufen lassen, erhalten Sie eine Geräuschauswahl, während der Bildschirm anzeigt, was gerade passiert. Alle Töne erklingen in C.

```
10 FOR TE=32 TO 250 STEP 50
20 FOR MA=1 TO 6553 STEP TE*10
30 FOR GO=8 TO 15
40 PRINT "Welle:"GO;" Magn.:";MA;" Tempo:";TE
50 PRINT
60 PLAY "s=go;m=ma;t=te;c1"
70 IF PLAY(1)<>0 THEN GOTO 70
80 BEEP
90 NEXT GO
100 NEXT MA
110 NEXT TE
```

Mit dem folgenden Programm wird Ihnen das Ergebnis von Wellenmuster S1 vorgeführt. Dieses Wellenmuster kommt vermutlich dem Klang eines Klaviers noch am nächsten, das ja auch schnell volle Lautstärke erreicht (beim Anschlagen der Tasten) und anschließend allmählich leiser wird (dem Ausschwingen der Saiten). So, wie das Programm hier steht, erklingt das Klavier allerdings in übertriebenem Stakkato. Durch Erhöhung des Wertes für M in Zeile 10 auf 10000 kommen wir der Wirklichkeit schon näher. Experimentieren Sie ruhig ein bißchen damit herum. Finden Sie heraus, was bei Veränderung einzelner Programmschleifen passiert.

```
10 PLAY "t255s0m1000164"
20 A=12 : C=6
30 FOR X=A TO A+4
40 PLAY "n=x;"
50 NEXT X
60 FOR X=A+12 TO A+16
70 PLAY "n=x;"
80 NEXT X
90 IF A>60 THEN C=-C
100 IF A<10 THEN C=-C
110 A=A+C : GOTO 30
```

Drei Geräuschgeneratoren

Bisher sind alle Programme über einen einzigen Kanal erklungen. Der MSX verfügt in ein- und demselben Chip aber über drei verschiedene Geräuschgeneratoren. Den zweiten und dritten programmieren Sie, indem Sie Daten im Anschluß an den PLAY-Befehl für den ersten Geräuschgenerator und durch Kommas voneinander getrennt eingeben.

```
10 A$="o3dfa4df2a2r4"
20 B$="o4dfa2r4"
30 C$="o5dfa4d2r4"
40 PLAY A$,B$,C$
50 PLAY "","",C$
60 PLAY "",B$,""
70 PLAY A$,"",""
80 PLAY A$,B$,""
90 PLAY A$,"",C$
100 PLAY "",B$,C$
110 PLAY "v612xa$;",B$,C$
```

Mit diesem Programm können Sie die verschiedenen Geräuschkanäle unabhängig voneinander erklingen lassen. Für jeden dieser Generatoren müssen Lautstärke, Tempo und Länge der Noten angegeben werden. Für alle drei Generatoren kann nur ein einziges Wellenmuster gelten.

Die Geräuscheffekte

Zur Erzeugung von Geräuscheffekten gibt es beim MSX den Befehl

SOUND , (Geräusch)

Hinter dem Befehl folgen immer ein Registercode, ein Komma sowie der Wert für das Register.

Sie müssen sich den PSG wie eine Schachtel mit 14 Fächern vorstellen. Das, was Sie in eines dieser Fächer hineintun, bestimmt, wie ein Geräusch klingen wird. Der Computer sucht immer alle Fächer ab.

Beim SOUND-Befehl fällt auf, daß Sie dem Computer Daten eingeben, damit ein bestimmtes Geräusch erzeugt werden kann. Eine derjenigen Angaben, die nicht eingegeben werden können, bestimmt die Dauer des Geräusches. Das heißt, mit dem SOUND-Befehl läßt sich ein Geräusch nicht einfach nach einer bestimmten Zeit abschalten. Vorteilhaft ist allerdings, daß das Programm das Geräusch nach Einstellen der Register nicht weiter beachten muß und mit der Abarbeitung des restlichen Programmes einfach fortfahren kann.

Die PSG können im Grunde nur zwei Arten von Geräuschen erzeugen: ein Geräusch, das einer Welle entspricht, und ein Rauschen. Dadurch, daß die Eigenschaften dieser beiden Geräusche festgelegt werden können (etwa mit dem Wellenmuster zur Entwicklung der Lautstärke, siehe PLAY-Befehl), ist es möglich, zahlreiche verschiedenartige Geräusche zu erzeugen.

Um zwei grundlegende Geräusche hören zu können, befehlen Sie

```
SOUND 7,62
SOUND 8,8
```

Sie vernehmen nun einen durchdringenden Ton. Dieser Ton hört sich an wie eine permanent erklingende Hupe. Befehlen Sie jetzt:

```
SOUND 7,55
```

Nun hören Sie Meeresrauschen oder das Rauschen eines Radioempfängers, der nicht sauber abgestimmt ist. Dies ist das sogenannte „weiße Rauschen".

Um diese beiden Geräusche hören zu können, haben wir nichts anderes zu tun, als einen Geräuschkanal zu öffnen, die Lautstärke zu bestimmen und festzulegen, ob ein Wellenton oder ein Rauschen zu erklingen hat. Mit dem Befehl CTRL STOP läßt sich das Rauschen wieder abstellen.

Und das sind die vierzehn Register, denen Sie unterschiedliche Werte zuordnen können:

Register	Funktion
0	Feinabstimmung Kanal A
1	Grobabstimmung Kanal A
2	Grobabstimmung Kanal B
3	Grobabstimmung Kanal B
4	Grobabstimmung Kanal C
5	Grobabstimmung Kanal C
6	Geräuschgenerator
7	weißes Rauschen
8	Volume Kanal A
9	Volume Kanal B
10	Volume Kanal C
11	Golfschläge
12	Golfschläge
13	Golfschläge

Die Register 0, 1, 2, 3, 4 und 5

Diese Register bestimmen die Frequenzen der verschiedenen Kanäle. Für jeden Kanal gibt es zwei Register, eines zur Feinabstimmung und eines zur Grobabstimmung. Die Aufteilung in Minimum- und Maximumwerte sieht so aus:

Register	Funktion	Bereich
o	fein A	0 - 255
1	grob A	1 - 15
2	fein B	0 - 255
3	grob B	1 - 15
4	fein C	0 - 255
5	grob C	1 - 15

Die Frequenz stellen Sie folgendermaßen ein:

Frequenz = Feinregister + (Grobregister * 256)

Auf diese Weise erhalten Sie allerdings keine Frequenzangaben in Hertz, sondern lediglich einen Dezimalwert. Je höher dieser Wert ist, desto tiefer wird der Ton. Den höchsten und tiefsten Ton des MSX können Sie so erklingen lassen:

```
SOUND 0,0
SOUND 1,0
SOUND 7,62
SOUND 8,8
```

Das tiefste Geräusch erklingt, wenn Sie in der Reihe oben die ersten beiden Befehle austauschen:

```
SOUND 0,255
SOUND 1,15
```

Wollen Sie die Frequenzangaben in Herz erhalten (zuweilen brauchen Sie das, um Übereinstimmung zwischen dem MSX und einem bestimmten Musikinstrument zu erreichen), müssen Sie folgende Formel anwenden:

Frequenzwert = 178900 / (16 X Frequenz in Hz)

Die Zahl 178900 entspricht der Uhrenfrequenz des PSG.

Register 6

Mit diesem Register läßt sich die Frequenz des Rauschkanals festlegen. Ihnen stehen hier 32 Werte (0-31) zur Auswahl. Je niedriger ein Wert ist, desto höher wird das Rauschen. Mit dem nächsten Programm können Sie sich das vorführen lassen:

```
10 FOR FR=0 TO 31
20 SOUND 7,55
30 SOUND 8,8
40 SOUND 6,FR
50 FOR Z=1 TO 300 : NEXT Z
60 BEEP
70 NEXT FR
80 STOP
```

Register 7

Hierbei handelt es sich um das vielleicht umständlichste der Register. Mit ihm wird das Geräusch eines jeden der drei Kanäle aktiviert und entschieden, ob ein Kanal ei-

nen Ton oder ein Rauschen erzeugt. Die angegebenen Werte müssen dazu den 8 Bits dieses Registers entsprechen:

Bit	Bitwerte	Funktion	Bit	Bitwerte	Funktion
0	0	Ton auf A	0	1	kein Ton auf A
1	0	Ton auf B	1	1	kein Ton auf B
2	0	Ton auf C	2	1	kein Ton auf C
3	0	Rauschen auf A	3	1	kein Rauschen auf A
4	0	Rauschen auf B	4	1	kein Rauschen auf B
5	0	Rauschen auf C	5	1	kein Rauschen auf C

Den Dezimalwert erhalten Sie, indem Sie die nötigen Bits miteinander addieren. Das geht genau so vor sich, wie die Erstellung einer Reihe eines Spritemusters. Eventuell schauen Sie hierzu noch einmal in dem entsprechenden Kapitel nach. In aller Kürze: Das Bit ganz rechts hat den Wert 1, das nächste Bit links davon den Wert 2, noch eine Stelle nach links erhalten Sie schon den Wert 8; so geht es weiter, bis Sie zum letzten Bit ganz links anlangen, das den Wert 128 hat. Die beiden Bits links werden für dieses Register jedoch nicht gebraucht und müssen daher den Wert 0 erhalten.

Angenommen, Sie möchten ein Rauschen und einen Ton auf A, nichts auf B und ein Rauschen auf C. Die acht Bits lauten dann: 00010110. Die Dezimalwerte sind: 0+2+4+0+16+0+0+0=11. Probieren Sie einmal aus, ob das Programm unten auch funktioniert.

```
10 SOUND 7,22
20 SOUND 2,13
30 SOUND 6,15
40 FOR V0=8 TO 10
50 SOUND V0,8
60 FOR Z=1 TO 600 : NEXT Z
70 SOUND V0,0
80 FOR Z=1 TO 500 : NEXT Z
90 NEXT V0
100 SOUND 9,8
110 SOUND 8,8
```

Wenn Ihnen das Umrechnen zu mühevoll ist, können Sie die Bits auch als Nullen und Einsen darstellen und die Zahl als binäre Zahl an den Computer weitergeben. In unserem Programm läßt sich die Zeile 10 durch

```
10 SOUND 7,&B00010110
```

ersetzen.

Die Register 8, 9 und 10

Mit diesen Registern bestimmen Sie die Lautstärke der Geräusche. Meist haben Sie das nicht nötig, da Ihr Fernsehgerät oder Monitor über einen eigenen Lautstärkeregler verfügt; um allerdings ein gutes Verhältnis zum Klicken der Tastatur herzustellen, sind diese Register keinesfalls überflüssig. Der Minimumwert ist 0, in diesem Falle hören Sie gar nichts; der Maximumwert ist 15. Achten Sie dabei auf die Einstellung des Lautstärkereglers an ihrem Fernsehgerät, womöglich platzt Ihnen sonst noch das Trommelfell.

Über die Werte von 0 bis 15 hinaus können Sie auch den Wert 16 eingeben; so teilen Sie dem Computer mit, daß Sie die Lautstärke mittels eines Wellenmusters bestimmen möchten. Diese Wellenmuster entsprechen denjenigen des PLAY-Befehles. Genaueres lesen Sie bitte im Abschnitt über den PLAY-Befehl nach.

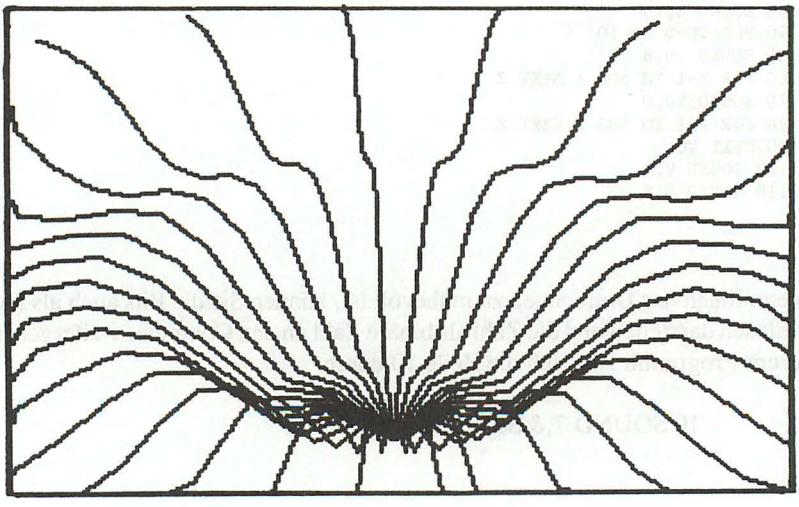
Die Register 11, 12 und 13

Diese drei Register bestimmen die Verwendung des Wellenmusters auf den drei Kanälen. Wenn Sie den Wert 16 eingegeben haben (Register 8, 9 oder 10), entwickelt sich die Lautstärke entsprechend einem der Wellenmuster, wie sie im Zusammenhang mit dem PLAY-Befehl behandelt worden sind.

Die Frequenz des Wellenmusters (siehe nach unter Magnitude beim PLAY-Befehl) wird mittels der Register 11 und 12 bestimmt. Hierzu dient folgende Formel:

Magnitude = Wert von Register 12 + (256 x der Wert von Register 11)

Sie können jeweils nur ein einziges Wellenmuster wählen, es ist allerdings möglich, für jeden Kanal festzulegen, ob die Lautstärke konstant bleibt oder sich entsprechend dem Wellenmuster entwickelt.

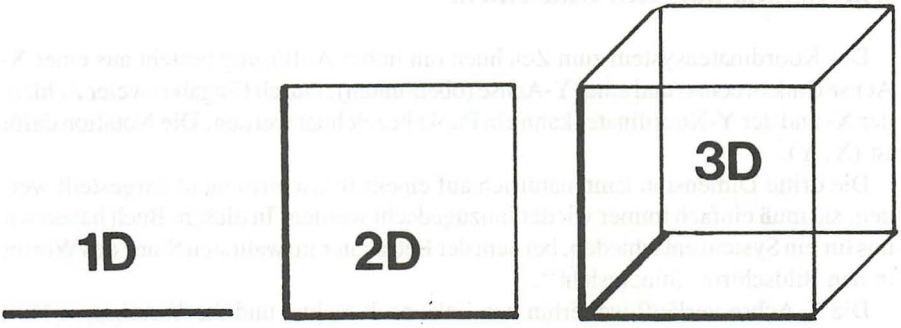


20. DIE DRITTE DIMENSION

Eins, zwei oder drei

Der Bildschirm Ihres Fernsehgerätes oder Monitors ist flach; ebenso das Papier, auf dem Sie diesen Text lesen. Von diesen Dingen sagt man, sie seien zweidimensional.

Die übliche Darstellung der verschiedenen Dimensionen ist die:
Erste Dimension: eine Richtung, ein Strich.
Zweite Dimension: zwei Richtungen, eine Fläche.
Dritte Dimension: drei Richtungen, ein Raum.



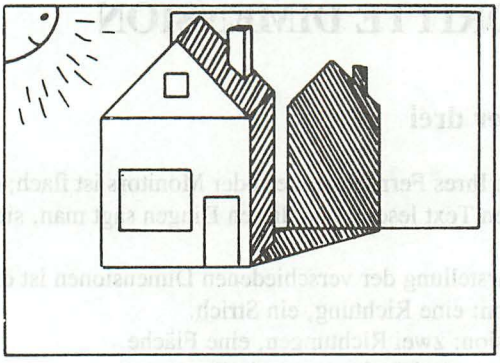
Ein eindimensionales oder zweidimensionales Objekt kann leicht zu Papier gebracht werden, ohne daß an der Form irgend etwas geändert werden muß; immer ist sofort zu erkennen, was für ein ein- oder zweidimensionales Objekt dargestellt werden soll.

Um es uns ein bißchen einfacher zu machen, kürzen wir eindimensional mit 1D, zweidimensional mit 2D und dreidimensional mit 3D ab.

Sobald ein Objekt auf einem Gegenstand mit einer niederen Dimension wiedergegeben werden soll, bekommen wir Schwierigkeiten. In einem solchen Falle muß nämlich projektiert werden. Dieses Projektieren können Sie durchaus wortwörtlich nehmen; Sie gehen dabei wie bei einer Photographie vor. Ein 3D-Objekt, etwa ein Haus, wird auf ein 2D-Objekt, nämlich den Film, in der Kamera projektiert. Ein anderes Beispiel, das sehr häufig vorkommt, ist der Schatten eines 3D-Objektes auf einem 2D-Objekt, etwa auf dem Boden oder der Mauer.

Eines der Probleme bei der Projektion ist die Tatsache, daß nach erfolgter Projektion die ursprüngliche Form nicht mehr zu erkennen ist.

Mit diesen Schwierigkeiten haben Sie auch zu tun, wenn Sie mit dem MSX 3D-Zeichnungen anfertigen wollen. Der Bildschirm, auf dem die Zeichnung ja erscheinen soll, ist schließlich flach. Es muß also projektiert werden. Die Zeichnungen in diesem Kapitel weisen nur die äußeren Umrisse eines Gegenstandes auf.



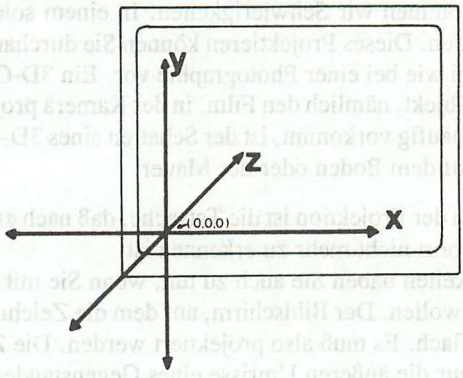
Darstellung auf dem Bildschirm

Das Koordinatensystem zum Zeichnen mit hoher Auflösung besteht aus einer X-Achse (links-rechts) und einer Y-Achse (oben-unten). Durch Eingabe zweier Zahlen, der X- und der Y-Koordinate, kann ein Punkt bezeichnet werden. Die Notation dafür ist (X, Y).

Die dritte Dimension kann natürlich auf einem Bildschirm nicht dargestellt werden, sie muß einfach immer wieder hinzugedacht werden. In diesem Buch haben wir uns für ein System entschieden, bei dem der Betrachter im wahrsten Sinne des Wortes in den Bildschirm „hineinsieht".

Die X-Achse verläuft weiterhin von links nach rechts, und die Y-Achse verläuft von oben nach unten. Für die Bezeichnung der Tiefe nehmen wir eine dritte Achse, nämlich die Z-Achse. Die Z-Achse verläuft von der Oberfläche des Bildschirmes hinein in die Tiefe, das heißt also, vom Betrachter weg, in das Gerät hinein. Die X-Achse steht senkrecht auf der Y-Achse; aber auch auf der Z-Achse steht sie senkrecht. Die Z-Achse ihrerseits steht wiederum senkrecht auf der X- und auf der Y-Achse.

In 3D wird ein Punkt mit Hilfe von drei Koordinaten (X, Y, Z) bezeichnet. Die erste



Zahl gibt die Bewegung links-rechts an; positive Zahlen stehen rechts vom Ausgangspunkt (dem Punkt <0,0,0>), negative Zahlen links vom Ausgangspunkt. Die zweite Zahl bezeichnet die Bewegung oben-unten; positive Zahlen finden sich unter dem Ausgangspunkt, negative darüber. Vorsicht: Bei der üblichen mathematischen Notation sieht das anders aus. Die dritte Zahl bezieht sich auf die Bewegung vor-zurück (in die Tiefe). Positive Zahlen liegen hinter der Schirmfläche, weg vom Betrachter, negative Zahlen vor der Schirmoberfläche.

Matrizen

Dieses Buch will zwar nichts weniger als ein Buch über Mathematik sein, aber beim Projektieren von 3D-Objekten ist ein bißchen Matrix-Mathematik einfach unumgänglich. In groben Zügen finden Sie hier erklärt, wie die in den einzelnen Programmen auftauchenden Formeln zustande gekommen sind, und wenn Ihnen die Matrix-Mathematik nicht unbekannt ist, sind diese Ausführungen nützlich beim Ausarbeiten von Experimenten. Diejenigen Leser, die mit Matrix-Rechnen überhaupt nichts anfangen können, erhalten die Möglichkeit, die Funktionsweise von Programmen zu verstehen.

Führen Sie sich nun noch einmal kurz das soeben behandelte 3D-Koordinatensystem vor Augen. Jeder Punkt kann in diesem System mit Hilfe von drei Zahlenangaben bezeichnet werden: einer X-, einer Y- und einer Z-Koordinate. Man kann auch jeden einzelnen Punkt als das Ergebnis einer Reihe von Bewegungen ansehen, die alle von einem ganz bestimmten Punkt ausgehen. Diese Verlagerungen nennt man Vektoren. Der Punkt (2,4,7) ist also das Resultat einer Verlagerung 2 mal eine Einheit in X-Richtung plus eine Verlagerung 4 mal eine Einheit in Y-Richtung plus eine Verlagerung 7 mal eine Einheit in Z-Richtung.

In einem wichtigen Teilbereich der Mathematik, nämlich im Matrixrechnen, werden geometrische Darstellungen (etwa Rotationen) mit Hilfe einer Matrix wiedergegeben. Eine Matrix ist ein Schema aus einer Reihe von Zahlen (oder Größen), die in Form eines Rechtecks angeordnet sind, und sie setzt sich aus Reihen (horizontal) und Spalten (vertikal) zusammen. Ein Beispiel für eine geometrische Darstellung, die mittels einer Matrix wiedergegen wird, ist folgendes:

$$\begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{pmatrix}$$

In der ersten Spalte solch einer Matrix befinden sich Angaben darüber, was in der geometrischen Darstellung mit dem Punkt (1,0,0) geschieht. In unserem Beispiel wird aus dem Punkt (1,0,0) der Punkt (3,0,0). In der zweiten Spalte finden sich Angaben über den Punkt (0,1,0), aus dem (0,2,0) wird. Die dritte Spalte teilt uns etwas über Punkt (0,0,1) mit; er wird zu (0,0,4).

Was passiert nun mit Punkt (2,4,7), wenn wir auf diese geometrische Darstellung zurückgreifen?

Für (2,4,7) schreiben Sie 2*(1,0,0)+4*(0,1,0)+7*(0,0,1).

Aus (2,4,7) wird hier 2*(3,0,0)+4*(0,2,0)+7(0,0,4).

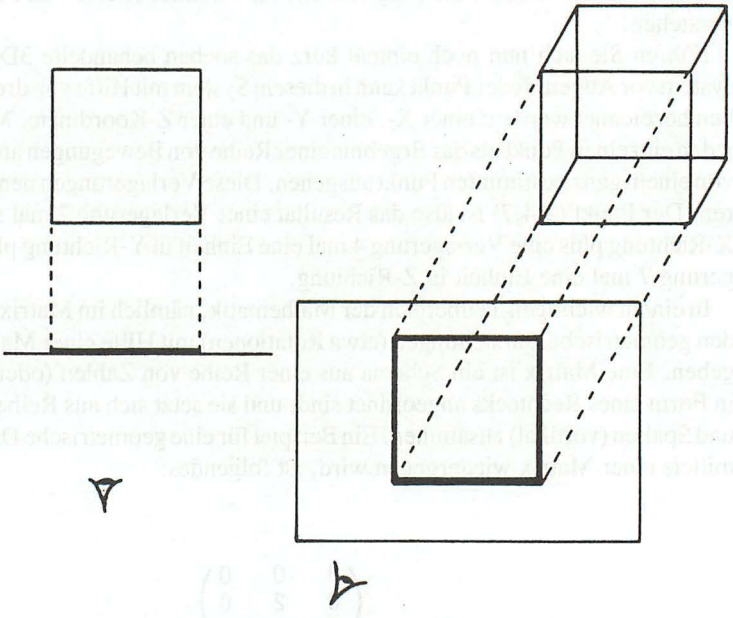
Das ergibt (6,0,0)+(0,8,0)+(0,0,28) und entspricht dem Punkt (6,8,28).

Diese Eigenschaften des Matrixrechnens machen wir uns gleich bei der Rotation in einem 3D-Raum zunutze.

Projektion

Wie bereits gesagt, ist es unmöglich, im Fernsehgerät zu zeichnen. Um nun ein 3D-Objekt auf dem Bildschirm wiedergeben zu können, müssen wir projektieren.

Zunächst einmal ein Beispiel für eine einfache Projektion. Nehmen Sie ein Viereck und betrachten Sie es genau von der Seite an; Sie erkennen nichts als einen Strich. Sie können auch einen Kubus nehmen, den Sie von vorne anschauen; nun sehen Sie nur ein Viereck.



Projektion eines Vierecks und eines Kubus

Aber natürlich wollen wir nicht immer nur die Vorderseite eines Objektes sehen. Bei der hier angewandten Art der Projektion, der orthografischen Projektion, wird jeder Punkt der Z-Koordinate zu Null. Dadurch ist von einem Kubus nach der Projektion lediglich eine flache Abbildung zu erkennen. Die Matrixnotation dafür, daß die Z-Koordinate immer Null wird, sieht so aus:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Der Vektor, der die Verlagerung in Z-Richtung bewirkt, wird null (0,0,0); diejenigen Vektoren, die für eine Verlagerung in X- oder Y-Richtung verantwortlich sind, bleiben unverändert. An den Punkten (1,0,0) und (0,1,0) ändert sich nichts, aus (0,0,1) allerdings wird (0,0,0).

Wenn nun ein bestimmter Punkt entsprechend der orthographischen Projektion projektiert werden soll, kann die Spaltenmatrix, die den Punkt wiedergibt (entsprechend der Notation der Koordinaten, dort jedoch von oben nach unten), mit dem quadratischen 3 mal 3-Matrix multipliziert werden, die für diese Form der Projektion steht. Aus dem Punkt (5,8,9) wird nach der Projektion Punkt (5,8,0). Das ist völlig korrekt, denn alle Punkte müssen schließlich in einer Fläche liegen (die Fläche, auf die projektiert wird).

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 5 \\ 8 \\ 9 \end{pmatrix} = \begin{pmatrix} 5 \\ 8 \\ 0 \end{pmatrix}$$

Um etwas über andere Formen der Projektion (etwa die schiefe Projektion oder die Merkatorprojektion) zu erfahren, können Sie in einem Mathematikbuch oder einem guten Atlas nachlesen.

Eine Zeichnung wird wesentlich besser, wenn wir schräg gegen das Objekt blikken. Das ist auch immer die Art und Weise, in der ein Kubus dargestellt wird. Man zeichnet den Kubus von einem Punkt aus, der ein wenig seitlich und oberhalb des Kubus liegt.

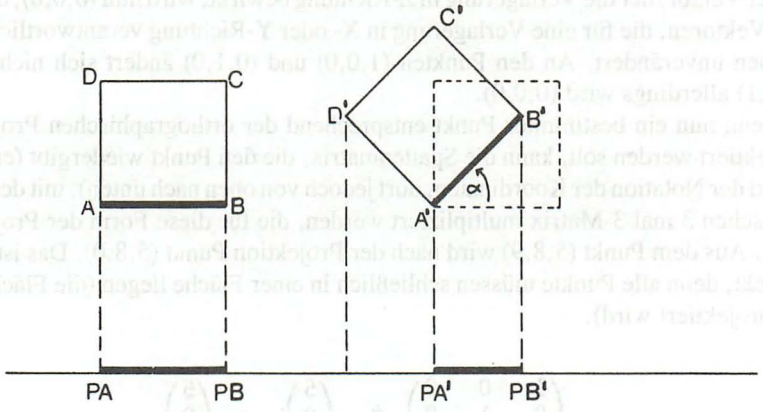
Zu diesem Zweck gibt es zwei Systeme: Entweder der Betrachter verändert seinen Standpunkt, oder aber das Objekt dreht sich.

Zwischen diesen beiden Systemen besteht eigentlich kein wesentlicher Unterschied; die zweite Methode ist lediglich ein wenig einfacher.

Hinsichtlich des mathematischen Hintergrundes ist auf diverse Schulbücher zu verweisen. Hier soll in aller Kürze das Prinzip behandelt werden.

Rotation

Das Viereck aus der vorigen Abbildung wird gedreht und anschließend auf die X-Achse projeziert:



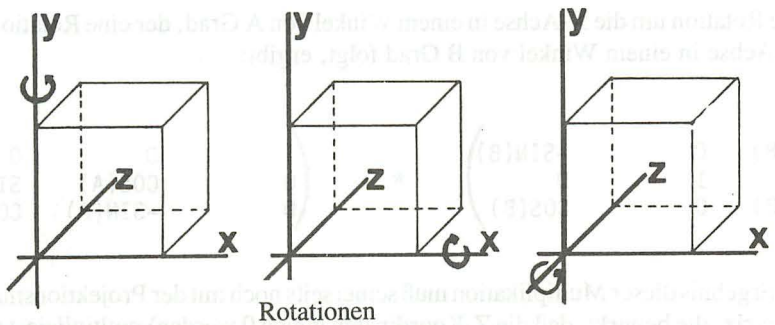
Die Punkte A, B, C und D stellen die Eckpunkte des Viereckes dar, das noch nicht gedreht worden ist. Bei den Punkten A', B', C' und D' handelt es sich um die gleichen Eckpunkte nach einer Drehung. Die Drehung erfolgte in einem Winkel von 45 Grad. Der Abstand AB beträgt 1. Da die Figur ein Quadrat ist, beträgt die Länge aller anderen Seiten gleichfalls 1.

Bei einer Projektion ohne Drehung ist der Abstand PA-PB (der Abstand zwischen den Projektionspunkten von A und B) gleich 1. Der Abstand zwischen PD und PC beläuft sich ebenfalls auf 1; der Abstand zwischen PA und PD und der zwischen PB und PC ist 0.

Nach der Drehung sieht das anders aus. Der Abstand PA' und PB' ist kleiner als 1. Wieviel kleiner, hängt vom Drehungswinkel ab. Bei einer Drehung um 45 Grad beträgt der Abstand zwischen PA' und PB' etwa 0, 7. Für den mathematisch kundigen Leser: Dieser Abstand ist genau 0.5*SQR(2) oder COS(RAD<45>). Je weiter gedreht wird, desto stärker verringert sich der Abstand. Wird um mehr als 90 Grad gedreht, erhalten wir sogar einen negativen Abstand (PB' befindet sich dann links von PA').

Auf diese Weise lassen sich die verschiedenen Abstände auf der Projektionslinie sowie die verschiedenen Projektionspunkte errechnen.

Die Rotation in 3D sieht nicht wesentlich anders aus; Sie können nun allerdings zwischen drei Rotationen wählen: der Rotation um die X-Achse, um die Y-Achse und um die Z-Achse.



Rotationen

Die Matrix einer Rotation um einen Winkel von A Grad um die X-Achse sieht so aus:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \text{COS(A)} & \text{SIN(A)} \\ 0 & \text{-SIN(A)} & \text{COS(A)} \end{pmatrix}$$

Sie sehen, daß der Punkt (1,0,0) bei dieser Rotation unverändert bleibt. Aus dem Punkt (0,1,0) wird (0,COS<A>, -SIN<A>). Wenn wir für A beispielsweise 30 Grad einsetzen, wird der Punkt (0,1,0) zu (0,0.866, – 0,5) (bei 0,866 handelt es sich um eine Annäherung an 0.5*SQR(3), das ist der Kosinus von 30 Grad; 0.5 ist der Sinus von 30 Grad).

Als Matrix für eine Rotation um einen Winkel von B Grad um die Y-Achse erhalten wir:

$$\begin{pmatrix} \text{COS(B)} & 0 & \text{-SIN(B)} \\ 0 & 1 & 0 \\ \text{SIN(B)} & 0 & \text{COS(B)} \end{pmatrix}$$

Bei dieser Rotation bleibt der Punkt (0,1,0) unverändert.

Und das ist die Matrix für eine Rotation um einen Winkel von C Grad um die Z-Achse:

$$\begin{pmatrix} \text{COS(C)} & \text{SIN(C)} & 0 \\ \text{-SIN(C)} & \text{COS(C)} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Hier ändert sich nichts am Punkt (0,0,1).

Das nächste Programm zeichnet einen Kubus, der nicht direkt von vorn betrachtet wird. Zunächst wird er um die X-Achse und dann um die Y-Achse gedreht. Die Matrix zu dieser kombinierten Drehung erhält man durch Multiplikation der Matrix einer Rotation um die X-Achse mit der Matrix einer Rotation um die Y-Achse.

Eine Rotation um die X-Achse in einem Winkel von A Grad, der eine Rotation um die Y-Achse in einem Winkel von B Grad folgt, ergibt:

$$\begin{pmatrix} COS(B) & 0 & -SIN(B) \\ 0 & 1 & 0 \\ SIN(B) & 0 & COS(B) \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & COS(A) & SIN(A) \\ 0 & -SIN(A) & COS(A) \end{pmatrix}$$

Das Ergebnis dieser Multiplikation muß seinerseits noch mit der Projektionsmatrix (der Matrix, die bewirkt, daß die Z-Koordinaten gleich 0 werden) multipliziert werden; anschließend wird mit der Spaltenmatrix des rotierenden Punktes (XYZ) des Kubus multipliziert.

$$\begin{pmatrix} PX \\ PY \\ PZ \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} COS(B) & SIN(A)*SIN(B) & -COS(A)*SIN(B) \\ 0 & COS(A) & SIN(A) \\ SIN(B) & -SIN(A)*COS(B) & COS(A)*COS(B) \end{pmatrix} * \begin{pmatrix} X \\ Y \\ z \end{pmatrix}$$

Das Endergebnis lautet:

$$PX = X*COS(B) + Y*SIN(A)*SIN(B) - Z*COS(A)*SIN(B)$$
$$PY = Y*COS(A) + Z*SIN(A)$$
$$PZ = 0$$

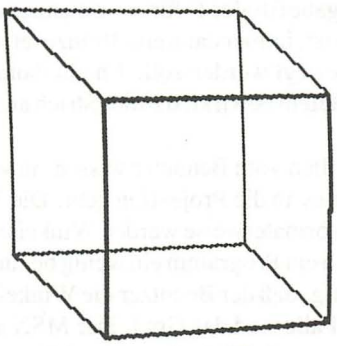
Die PX, PY und PZ stehen für die Werte der X-Koordinate bzw. der Y- und Z-Koordinate des Punktes nach der Projektion. Die Z-Koordinate des Punktes ist gleich 0 und liegt infolgedessen auf der Projektionsfläche (in unserem Falle ist das der Bildschirm). Eine Rotation, die zunächst um die Y-Achse in einem Winkel von B Grad und anschließend um die X-Achse in einem Winkel von A Grad verläuft, mit Projektion ergibt:

$$PX = X*COS(B) - Z*SIN(B)$$
$$PY = X*SIN(A)*SIN(B) + Y*COS(A) + Z*SIN(A)*COS(B)$$
$$PZ = 0$$

Dieses Programm zeigt einen Kubus in unterschiedlichen Lagen.

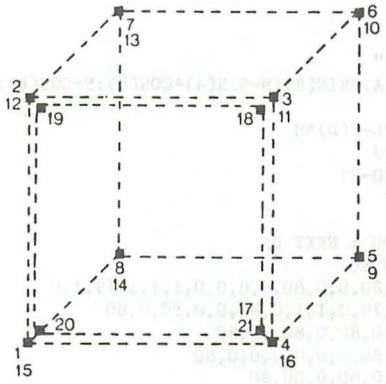
```
10 SCREEN 2
20 DIM X(21)
30 DIM Y(21)
40 DIM Z(21)
50 FOR D=1 TO 21
60 READ X(D),Y(D),Z(D)
70 NEXT D
80 INPUT "Drehung um Y-Achse";A
90 INPUT "Drehung um X-Achse";B
100 A=(A*3.14)/180
```

```
110 B=(B*3.14)/180
120 SCREEN 2
130 DRAW "bm100,130"
140 K=COS(A):L=SIN(A)*SIN(B):M=SIN(A)*COS(B):N=COS(B):J=SIN(B)
150 FOR D=1 TO 21
160 PX=X(D)*K+Y(D)*L-Z(D)*M
170 PY=Y(D)*N+Z(D)*J
180 Q=PX+100 : R=130-PY
190 DRAW "m=q;,=r;"
200 NEXT D
210 FOR PA=1 TO 2000 : NEXT PA
220 SCREEN 1 : GOTO 80
230 DATA 80,0,0,80,80,0,0,80,0,0,0,0,1,1,1,79,1,0
240 DATA 79,79,0,1,79,0,1,1,0,80,0,0,80,0,80
250 DATA 80,80,80,80,80,0,80,80,80
260 DATA 0,80,80,0,80,0,0,0,0,0,0,80
270 DATA 80,0,80,0,0,80,0,80,80
```



In den Zeilen 20-40 wird ausreichend Platz geschaffen für die Zeichnung des Kubus; und in den Zeilen 50-70 befinden sich in einem Array die verschiedenen Daten für den Kubus. Auf diese Weise kann rascher gezeichnet werden.

Die Daten, die in die Datenliste aufgenommen sind, beziehen sich auf die Punkte, zu denen eventuell eine Linie gezogen werden muß (immer eine X-, eine Y- und eine Z-Koordinate). In diesem Beispielprogramm reichte es, jedesmal eine Linie zu ziehen. Für einen Kubus braucht man 16 Linien, und um zu erreichen, daß die vorderen Linien ein wenig kräftiger ausfallen (wodurch sich der 3D-Effekt noch verstärkt), braucht man 21 Linien entsprechend dem folgenden Schema:



Wenn Sie andere Figuren als Kuben zeichnen wollen, werden Sie es zuweilen mit Zeichnungen zu tun haben, bei denen es wesentlich einfacher ist, „den Stift" kurz zu bewegen, ohne eine Linie zu ziehen. Sie müssen dann jedes Koordinatensystem mit einer vierten Zahlenangabe (0 oder 1) für das Zeichnen oder Auslassen einer Linie versehen. Am sinnvollsten ist, immer dann eine 0 einzusetzen, wenn kein Strich gezogen und nur „der Stift" bewegt werden soll. 1 heißt dann Zeichnen eines Striches. Ein simpler IF..THEN..-Befehl bewirkt, daß der Strich auch zum richtigen Punkt hin gezogen wird.

Die Zeilen 80 und 90 wollen vom Benutzer wissen, in welchem Winkel die Form gedreht werden soll, bevor es an die Projektion geht. Die Zeilen 100 und 110 muten ein bißchen kryptisch an. Normalerweise werden Winkel in Grad angegeben, und ein Kreis hat 360 Grad. Um nun ein Programm ein wenig benutzerfreundlicher zu gestalten, ist es auf jeden Fall nötig, daß der Benutzer die Winkel in einer bekannten Größe eingeben kann. In diesem Falle sind das Grad. Der MSX arbeitet allerdings mit Radien, und in einen Kreis passen 2*PI Radien, wobei PI etwa 3, 14 ist. In den Zeilen 100 und 110 rechnet der MSX selbsttätig die Gradzahlen in Radien um.

Zeile 120 schaltet auf den grafischen Schirm um.

Zeile 130 bringt den grafischen Cursor in die Ausgangsstellung.

Zeile 140 müssen Sie zusammen mit den Zeilen 160 und 170 lesen. Die beiden letzten Zeilen werden für jeden Punkt wiederholt, während Zeile 140 bei jeder Zeichnung nur einmal ausgeführt wird. Dadurch, daß so ein Teil der Berechnungen in eine Programmzeile geschrieben wird, wo er nur ein einziges Mal ausgeführt werden muß anstatt viele Male, kann das Zeichnen wesentlich rascher vonstatten gehen. Die Berechnung, die der Computer für jeden zu zeichnenden Punkt anstellen muß, fällt nun kürzer aus. Probieren Sie das aus, indem Sie in die Zeilen 160 und 170 die ursprünglichen Berechnungen einfügen:

```
160 PX=X(D)*COS(A)+Y(D)*SIN(A)*SIN(B)-
Z(D)*SIN(A)*COS(B)
170 PY=Y(D)*COS(B)+Z(D)*SIN(B)
```

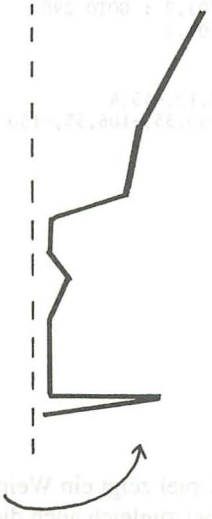
Sie merken, um wieviel langsamer nun gezeichnet wird. Bei der Erstellung Ihres eigenen Zeichenprogramms, in dem viel gerechnet wird, müssen Sie darauf achten, daß soviele Berechnungen wie möglich vorab (das heißt außerhalb der Zeichenschleife) erledigt werden. Eine Wiederholungsschleife sollte so kurz wie möglich sein, damit die Zeichnungen rasch ausgeführt werden können.

Die Zeilen 180 und 190 schließlich bewirken die eigentliche Zeichenarbeit. Zeile 180 sollte ja eigentlich überflüssig sein. Dadurch, daß die Koordination sorgfältig gewählt werden, kann man sich eine Verschiebung ersparen. Andererseits hat ein Verschiebungsfaktor den Vorteil, daß die Form über den Schirm hin und her bewegt werden kann.

Zeile 210 läßt ein wenig Zeit verstreichen, bevor Zeile 220 wieder auf Schirm 1 zurückschaltet und der Benutzer neue Winkeleingaben vornehmen kann.

Ein Weinglas

Durch Erweiterung des letzten Programmes können Sie es sich selber noch ein bißchen einfacher machen. Das Programm unten zeichnet nicht nur einen von Ihnen erdachten Gegenstand, es ‚überlegt' sogar, wie dieser Gegenstand auszusehen hat. Als einziges müssen Sie einen beliebig gekrümmten Strich eingeben. Der Computer dreht diesen Strich immer um ein Grad herum; die Rotation erfolgt dabei um die Y-Achse.



Nach 360 Grad befindet sich der Strich wieder in der Anfangsposition.

Da der Benutzer den sich drehenden Strich direkt von vorn betrachtet, sieht es so aus, als habe der Computer eine 2D-Zeichnung angefertigt. Doch während des Dre-

hens hat der Computer die erhaltenen Daten in einem Array abgespeichert. Wenn Sie nun nach Ausformung der Figur vom Computer verlangen, er solle den Gegenstand um die X-Achse rotieren lassen, können Sie ihn von verschiedenen Seiten anschauen.

```
10 SCREEN 2 : COLOR 4,15,5
20 CLS
30 DIM X(9)
40 DIM Y(9)
50 DIM Z(33,9,2)
60 FOR Q=1 TO 9
70 READ X(Q),Y(Q)
80 NEXT Q
90 FOR D=1 TO 32
100 DRAW "bm125,170"
110 S=SIN(D*.2)
120 C=COS(D*.2)
130 FOR A=1 TO 9
140 Z(D,A,1)=S*X(A)
150 Z(D,A,2)=C*X(A)
160 LINE - (Z(D,A,1)+125,Y(A)+170),1
170 NEXT A : PSET (125,170),15 : NEXT D
180 FOR B=.5 TO 1.8 STEP .3
190 CLS
200 PSET (125,170),15
210 N=COS(B) : J=SIN(B)
220 FOR D=1 TO 32
230 PSET (125,170),15
240 FOR A=1 TO 9
250 PX=Z(D,A,1)
260 PY=Y(A)*N+Z(D,A,2)*J
270 IF Y(A)<-79 THEN 275 ELSE 280
275 LINE -(PX+125,PY+170),9 : GOTO 290
280 LINE -(PX+125,PY+170),4
290 NEXT A : NEXT D
300 NEXT B
310 DATA 40,0,4,6,4,-30,10,-45,4
320 DATA -55,4,-70,30,-80,35,-106,55,-150
```

Dieses Programm zum Beispiel zeigt ein Weinglas. Nachdem das Glas zunächst in 2D gezeichnet wurde, wobei zugleich auch die Daten für die Wiedergabe in 3D berechnet worden sind, wird es schließlich auch noch in 3D dargestellt. Die Winkel, in denen das Glas um die X-Achse gedreht wird, können Sie (in Radien) in Zeile 180 ablesen. Eine Drehung um die Y-Achse ist nicht sonderlich sinnvoll, da der Gegenstand durch Drehung um die Y-Achse entstanden und perfekt symmetrisch ist. Auf-

grund der Drehung um die X-Achse können Sie auch das Innere des Glases sehen.

Die Zeilen 30 und 40 halten Platz für die 2D-Wiedergabe fei.

Zeile 50 bewirkt das gleiche für die Wiedergabe in 3D.

Die Zeilen 60-80 lesen die Daten aus der Datenliste. Diese Daten setzen sich jeweils aus zwei Zahlen zusammen, die einen Punkt des hierzu gezeichneten Striches bezeichnen.

Die Zeilen 90-170 zeichnen 32 mal alle neun Punkte des Striches, und jedesmal wird der Strich ein wenig gedreht. Bei der Wiedergabe in 2D kann man das sehen, weil der Strich sich verschiebt.

Die Zeilen 140 und 150 bringen die 3D-Daten in dem Array unter.

Zeile 160 bewirkt das Zeichnen der 2D-Darstellung.

Zeile 170 beinhaltet eine Schleife und bewirkt die Bewegung des grafischen Cursors.

In den Zeilen ab 180 finden wir Daten für die Wiedergabe in 3D. Wenn Sie lieber selber den Winkel bestimmen wollen, in dem sich das Weinglas dreht, können Sie die Zeilen 180, 190 und 290 folgendermaßen ändern:

```
180 SCREEN 0 : INPUT ''Drehung:'';X
190 B=(X*3.14)/180 : SCREEN 2

290 FOR WA=1 TO 2000 : NEXT WA : GOTO 180
```

Die Zeilen 200-280 sind mit den Zeilen 90-170 vergleichbar. Allerdings muß jetzt nichts berechnet werden, sondern der Computer kann die Daten direkt dem Array entnehmen und zeichnen. In Zeile 210 wird ein Großteil der Berechnungen erledigt; das ist um einiges einfacher als bei dem Kubusprogramm. Grund dafür ist die Tatsache, daß nur um eine einzige Achse gedreht wird.

Zeile 270 bewirkt eine zweifarbige Darstellung; der obere Teil des Glases ist rot, der untere Teil blau.

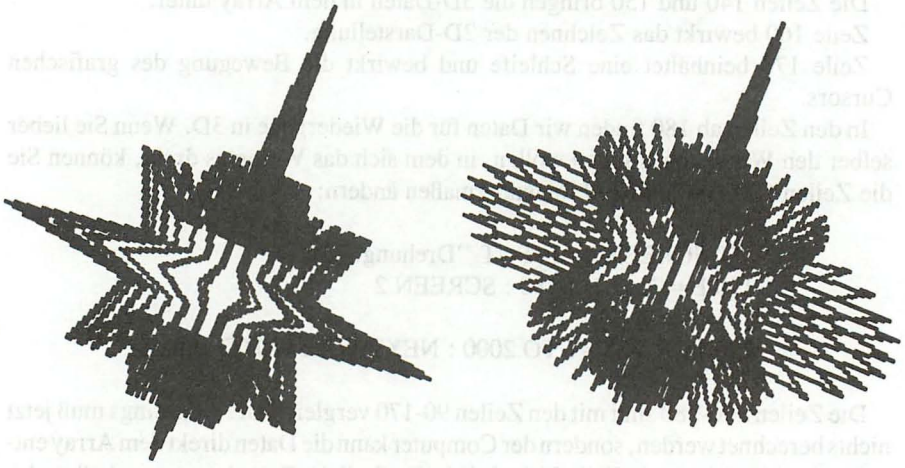
Um nun anstelle eines Weinglases eine andere Figur zu erhalten, müssen Sie lediglich die Daten verändern. Allerdings ist hierbei Vorsicht geboten: Wenn die Anzahl der Daten zunimmt, muß auch die Häufigkeit des Datenlesens korrigiert werden (Zeilen 60, 130, 240); das gleiche gilt unter Umständen auch für die Dimensionierung des Arrays (Zeilen 30, 40 und 50).

Verändern Sie im letzten Programm die folgenden Zeilen:

```
30 DIM X(13)
40 DIM Y(13)
50 DIM Z(33,13,2)
60 FOR Q=1 TO 13
130 FOR A=1 TO 13
240 FOR A=1 TO 13
270 LINE -(PX+125, PY+170), 4
300 DATA 0,10,7,-30,15,0,22-30,40,20,
```

30, – 40,75, – 50,30, – 60,40, – 90,22, – 70,
15, – 100,7, – 70,0, – 150

Die Daten für andere Gegenstände können Sie selber verändern. Jeweils zwei plus zwei Angaben gehören zusammen; immer handelt es sich dabei um eine X-Koordinate und eine Y-Koordinate eines Punktes auf der gekrümmten Linie, die gedreht werden muß. Achten Sie darauf, daß Sie genügend Platz für die verschiedenen Arrays sowie für das mehrmalige Durchlaufen der Schleifen freihalten.



Sie können es noch einfacher haben, wenn Sie den gekrümmten Strich nicht Punkt für Punkt eingeben, sondern den Computer die Punkte anhand einer von Ihnen mitgeteilten Formel bestimmen lassen. So erhalten Sie schöne, gerippte Flächen. Ersetzen Sie Zeile 70 des bereits geänderten Programms durch:

```
70 Y(Q)=-50-70*(SIN<Q-9.9>/):X(Q)=Q*20
```

Die Datenliste brauchen Sie jetzt eigentlich nicht mehr.
Mit folgendem sollten Sie es ebenfalls versuchen:

```
70 -Y(Q)=-90-50*(SIN(((Q/2.5)-.7)/Q-.9)) : Y(Q)=30*Q
```

Sie erhalten hiermit ein Murmelloch.
Auch die folgende Eingabe ist sehr hübsch:

```
70 Y(Q)=-9010*(Q^2)*(SIN(Q)/Q-.9) : X(Q)=30*Q
```

Ein Kubus

Das nächste Programm zeigt Ihnen, wie ein Objekt um alle drei Achsen gedreht werden kann. Auffallend ist die Schnelligkeit des Programmes, und dies ungeachtet der zahlreichen Berechnungen, die für jeden Punkt angestellt werden müssen.

Bei einer Drehung um drei Achsen werden die Berechnungen zur Bestimmung eines Projektionspunktes für jeden einzelnen Punkt um ein ganzes Stück länger. Mit diesem Programm wird immer zunächst um die X-Achse, anschließend um die Y-Achse und zuletzt um die Z-Achse gedreht. Diese Reihenfolge kann natürlich auch anders aussehen, nur hängt davon das letztendliche Ergebnis ab. Durch Festlegung der genauen Drehungswinkel kann der Benutzer allerdings jede Position des Objektes erreichen.

Und so wurde die Projektionsformel gefunden:

Die Matrix einer Projektion auf der X-Y-Fläche (Z=0) nach Rotation um die Z-Achse, nach Rotation um die Y-Achse und nach Rotation um die X-Achse wurde errechnet. Dazu waren zunächst die Matrizen für die Rotation um die Y-Achse und die X-Achse miteinander zu multiplizieren (siehe oben). Das Resultat wurde dann mit einer Matrix der Rotation um die Z-Achse multipliziert (über Winkel C). Und so sieht das Ergebnis aus:

```
/COS(B)*COS(C)    SIN(A)*SIN(B)*COS(C)      -COS(A)*SIN(B)*COS(C)+ \
                  +COS(A)*SIN(C)            SIN(A)*SIN(C)

 -COS(B)*SIN(C)   -SIN(A)*SIN(B)*SIN(C)+    COS(A)*SIN(B)*SIN(C)+
                  COS(A)*COS(C)             SIN(A)*COS(B)

\SIN(B)           -SIN(A)*COS(B)             COS(A)*COS(B)          /
```

Diese Matrix wiederum wird mit der Projektionsmatrix multipliziert (Z=0).

Im folgenden Programm hat der Benutzer zwei Möglichkeiten: Datenliste oder Dateneingabe.

Wenn Sie sich für die Datenliste entscheiden, müssen Sie nichts weiter tun. Das Programm hält eine Reihe von Daten parat, um einen Kubus zu zeichnen, mit dem sich vorführen läßt, was das Programm alles kann. Wünschen Sie einen anderen Gegenstand, entscheiden Sie sich für Dateneingabe. Die dann von Ihnen eingespeisten Daten bleiben allerdings nicht auf Dauer im Programm. Wollen Sie nun zahlreiche verschiedene Gegenstände mit umfangreichen Daten von diesem oder einem der nächsten Programme darstellen lassen, sollten Sie diese Daten in eigenen Dateien abspeichern. Näheres hierzu finden Sie im Kapitel über den Umgang mit Dateien.

```
10 SCREEN 0 : CLS : KEY OFF
20 INPUT "Datenliste(D) oder Eingabe(I)";D$
30 IF LEFT$(D$,1)="I" THEN 140
40 RESTORE 360
50 READ L
60 DIM X(L)
70 DIM Y(L)
80 DIM Z(L)
90 DIM T(L)
100 FOR H=1 TO L
110 READ X(H),Y(H),Z(H),T(H)
120 NEXT H
130 GOTO 250
140 INPUT "Anzahl Punkte";L
150 DIM X(L)
160 DIM Y(L)
170 DIM Z(L)
180 DIM T(L)
190 FOR H=1 TO L
200 PRINT "X-Koordin. Punkt ";H;:INPUT X(H)
210 PRINT "Y-Koordin. Punkt ";H;:INPUT Y(H)
220 PRINT "z-Koordin. Punkt ";H;:INPUT Z(H)
230 INPUT "Strich(1) oder verlagern(0)";T(H)
240 NEXT H
250 CLS
260 PRINT "EINGABE IN GRAD"
270 INPUT "Rotation um X-Achse ";A1
280 INPUT "Rotation um Y-Achse ";B1
290 INPUT "Rotation um Z-Achse ";C1
300 SCREEN 2 : COLOR 1,15,15 : CLS
310 GOSUB 540 : REM Winkelberechnung
320 GOSUB 420 : REM Zeichnen
330 FOR WA=1 TO 3000 : NEXT WA
340 SCREEN 0 : CLS : GOTO 260
350 GOTO 340
360 DATA 21
370 DATA 0,0,0,0,0,50,0,1,50,50,0,1,50,0,0,1,50,0,50,1
375 DATA 50,50,50,1,0,50,50,1,0,0,50,1,50,0,50,1
380 DATA 50,50,50,0,50,50,0,1,0,50,0,0,0,50,50,1
385 DATA 0,0,50,0,0,0,0,1,50,0,0,1,48,2,0,1,48,48,0,1
390 DATA 2,48,0,1,2,2,0,1,48,2,0,1
410 REM Zeichnen
420 PSET (125,125),15
430 FOR H=1 TO L
440 GOSUB 500 : REM Projektion
450 IF T(H)=0 THEN DRAW "bm=px;,=py;"
460 IF T(H)=1 THEN LINE -(PX,PY),1
470 NEXT H
480 RETURN
490 REM Projektion
500 PX=(X(H)*F+Y(H)*G+Z(H)*E)+125
510 PY=(X(H)*I+Y(H)*J+Z(H)*K)+125
520 RETURN
530 REM Winkelberechnung
540 A=(A1*3.14)/180:B=(B1*3.14)/180:C=(C1*3.14)/180
550 F=COS(B)*COS(C)
560 G=SIN(A)*SIN(B)*COS(C)+COS(A)*SIN(C)
570 E=SIN(A)*SIN(C)-COS(A)*SIN(B)*COS(C)
580 I=COS(B)*SIN(C)*-1
590 J=COS(A)*COS(C)-SIN(A)*SIN(B)*SIN(C)
600 K=COS(A)*SIN(B)*SIN(C)+SIN(A)*COS(C)
610 RETURN
```

Mit den ersten Zeilen wird der Schirm geleert, und der Computer fragt den Benutzer, ob dieser die Daten selber eingeben oder sich des mit dem Programm vorgegebenen Kubus bedienen will. Falls sich der Benutzer für die letztere Möglichkeit entschließt, geht der Computer nach Dimensionierung und Einlesen des Arrays über zu Zeile 250 (Zeile 130).

Die Zeilen 140-240 bewirken, daß der Benutzer selber eine Form eingeben kann, mit der der Computer anschließend arbeitet.

Die Zeilen 260-340 beinhalten eine Schleife, die dem Benutzer die Drehungswinkel benennen kann; zugleich bewirken zwei Subroutinen die Berechnungs- und Zeichenvorgänge.

Bei Zeile 350 handelt es sich um eine Falle, in die der Computer nie geraten darf. Sie können diese Zeile durchaus streichen.

Die Zeilen 360-400 stellen die Datenliste dar. Die Angabe hinter dem Databefehl aus Zeile 360 bezeichnet lediglich die Anzahl der zu lesenden Daten in der Datenliste.

Die Zeilen 410-480 bilden die Zeichensubroutine. Der grafische Cursor wird in die richtige Position gebracht (Zeile 420), und in einer Schleife werden nacheinander die Daten abgehandelt. Zunächst erledigt der Computer die Berechnungen; dazu dient eine eigene Subroutine, die in Zeile 490 beginnt. An diesen Berechnungszeilen (500 und 510) können Sie erkennen, daß bereits einiges an Rechnerei erledigt wurde. Wenn nicht von vornherein bereits manches berechnet worden wäre, hätte der Computer für jeden einzelnen Punkt ausrechen müssen:

```
PX  = (H)*COS(B)*COS(C)+Y(H)*(SIN(A)*SIN(B)*COS(C)+COS(A)*
      SIN(C))
      +Z(H)*(SIN(A)*SIN(C)-COS(A)*SIN(B)*COS(C))
PY  = X(H)+-(COS(B)*SIN(C))+Y(H)*(COS(A)*COS(C)-SIN(A)*SIN
      (B)*SIN(C))
      +Z(H)*(COS(A)*SIN(B)*SIN(A)+SIN(A)*COS(C))
```

Diese Berechnungen befinden sich größtenteils in der Subroutine, die außerhalb der Zeichenschleife aufgerufen wird und die Zeilen 503-610 in Beschlag nimmt. Im Anschluß an die Projektionssubroutine wird kontrolliert, ob die letzte der vier Angaben eine 0 oder eine 1 ist. Handelt es sich hierbei um eine 0, wird kein Strich gezogen; bei einer 1 ist das Gegenteil der Fall.

Eine Pyramide

Die Möglichkeiten des Programmierens in 3D sind noch längst nicht erschöpft. Tippen Sie einmal das nächste Programm ab und benutzen Sie das das vorherige Programm als Grundlage, denn es gibt zwischen beiden eine Reihe von Übereinstimmungen.

```
10 SCREEN 0 : CLS : KEY OFF
20 INPUT "Datenliste(D) oder Eingabe(I)";D$
30 IF LEFT$(D$,1)="I" THEN 140
40 RESTORE 360
50 READ L
60 DIM X(L)
70 DIM Y(L)
80 DIM Z(L)
90 DIM T(L)
100 FOR H=1 TO L
110 READ X(H),Y(H),Z(H),T(H)
120 NEXT H
130 GOTO 250
140 INPUT "Anzahl Punkte";L
150 DIM X(L)
160 DIM Y(L)
170 DIM Z(L)
180 DIM T(L)
190 FOR H=1 TO L
200 PRINT "X-Koordin. Punkt ";H;:INPUT X(H)
210 PRINT "Y-Koordin. Punkt ";H;:INPUT Y(H)
220 PRINT "z-Koordin. Punkt ";H;:INPUT Z(H)
230 INPUT "Strich(1) oder verlagern(0)";T(H)
240 NEXT H
250 CLS
260 PRINT "EINGABE IN GRAD"
270 INPUT "Rotation um X-Achse ";A1
280 INPUT "Rotation um Y-Achse ";B1
290 INPUT "Rotation um Z-Achse ";C1
300 INPUT "Skalenfaktor X-Richtung";SX
310 INPUT "Skalenfaktor Y-Richtung";SY
320 INPUT "Skalenfaktor Z-Richtung";SZ
330 INPUT "Verlagerung X-Richtung";VX
340 INPUT "Verlagerung Y-Richtung";VY
350 SCREEN 2 : COLOR 1,15,15 : CLS
360 GOSUB 560 : REM Winkelberechnung
370 GOSUB 440 : REM Zeichnen
380 FOR WA=1 TO 3000 : NEXT WA
390 SCREEN 0 : CLS : GOTO 260
400 DATA 12
410 DATA 0,0,0,0,50,0,0,1,50,0,50,1,0,0,50,1,0,0,0,1
415 DATA 25,-50,25,1,50,0,50,1,0,0,0,1,0,0,50,0
420 DATA25,-50,25,1,50,0,0,1,0,0,50,1
430 REM Zeichnen
440 PSET (125,125),15
450 FOR H=1 TO L
460 GOSUB 520 : REM Projektion
470 IF T(H)=0 THEN DRAW "bm=px;,=py;"
480 IF T(H)=1 THEN LINE -(PX,PY),1
490 NEXT H
500 RETURN
510 REM Projektion
520 PX=(X(H)*F*SX+Y(H)*G*SY+Z(H)*E*SZ)+125+VX
530 PY=(X(H)*I*SX+Y(H)*J*SY+Z(H)*K*SZ)+125+VY
540 RETURN
550 REM Winkelberechnung
560 A=(A1*3.14)/180:B=(B1*3.14)/180:C=(C1*3.14)/180
570 F=COS(B)*COS(C)
580 G=SIN(A)*SIN(B)*COS(C)+COS(A)*SIN(C)
590 E=SIN(A)*SIN(C)-COS(A)*SIN(B)*COS(C)
600 I=COS(B)*SIN(C)*-1
610 J=COS(A)*COS(C)-SIN(A)*SIN(B)*SIN(C)
620 K=COS(A)*SIN(B)*SIN(C)+SIN(A)*COS(C)
630 RETURN
```

Mit Ihrem Wissen aus dem Vorhergehenden müßten Sie dieses Programm eigentlich vollständig nachvollziehen können. Die wenigen Besonderheiten beziehen sich auf die Möglichkeit, einen Gegenstand zu verformen, zu verkleinern oder zu vergrößern. Verkleinern, Vergrößern oder Verformen geschieht auf die gleiche Art und Weise. Die eingegebenen Punkte des Gegenstandes werden mittels eines Multiplikationsfaktors manipuliert. Beläuft sich dieser auf 1, wird das Objekt in der angegebenen Richtung und in wahrer Größe (der Größe, die sich aus der Datenliste ergibt), gezeichnet. Werte größer als 1 vergrößern das Objekt, und Werte kleiner als 1 verkleinern es.

Setzen Sie nun unterschiedliche Werte für die verschiedenen Richtungen (X, Y und Z), so verformt sich der Gegenstand. Bewegung erfolgt mittels der Variablen VX und VY.

Die Werte in der Datenliste bewirken die Umrißzeichnung einer Pyramide. Diese Werte werden in Gruppen von je vier eingegeben. Die ersten drei Werte ergeben eine X-, eine Y- und eine Z-Koordinate. Der vierte Wert bezieht sich die Angabe, ob eine Linie gezogen oder einfach ein anderer Punkt angesteuert werden soll. Auf diese Weise können ganz einfach verschiedene, durchaus auch komplizierte Objekte gezeichnet werden, ohne daß dazu ein ununterbrochener Strich nötig wäre.

Die notwendigen Daten zu finden ist auch bei einer unsymmetrischen Figur kein besonderes Problem. Sie legen dazu einfach den Gegenstand selber (oder ein Modell) auf ein Blatt Rasterpapier. Die Richtung links-rechts stellt die X-Richtung dar. Für jeden einzelnen Punkt müssen Sie zunächst den Abstand von links nach rechts angeben.

Die Richtung vor-zurück heißt Z-Richtung.

Flach vor das Objekt halten Sie ein Stückchen Glas oder durchsichtige Plastikfolie mit einer gleich großen Gitterzeichnung. Die Richtung von oben nach unten ist die Y-Richtung.

Auf diese Weise können Sie alle notwendigen Punkte des Objektes nach Koordinaten bestimmen. Bedenken Sie, daß nach der Eingabe der Koordinaten eines jeden Punktes immer eine 1 folgen muß, wenn ein Strich gezogen werden soll, und eine 0, wenn der Computer lediglich zu einem anderen Punkt übergehen soll.

Handelt es sich um eine recht komplizierte Figur, so können Sie sich eines Kurvenmessers bedienen. Das ist ein Gerät mit einer Reihe eng beieinanderliegender metallener Stäbe. Durch Anlegen der verschiebbaren Stäbe an das Objekt, deren Arretierung und Ablesen der Positionen der einzelnen Stäbe auf dem Rasterpapier lassen sich auch schwierigste Figuren rasch bestimmen. Im einschlägigen Fachhandel ist ein solches Hilfsmittel zu erwerben.

Mit Hilfe dieses Programmes und Ihrer eigenen Phantasie können Sie zahlreiche bekannte Gegenstände auf dem Bildschirm betrachten und untersuchen, ob sie schöner aussähen, wenn sie womöglich höher oder breiter oder länger wären.

Anhang: ASCII-Code und Fehlermeldungen

ASCII-CODE

Alle in dieser Tabelle vorkommenden Zeichen können Sie mittels der Tastatur auf den Bildschirm bringen. Hierzu drücken Sie eine Taste oder eine Tastenkombination.

Sie können diese Zeichen aber auch mit dem CHR$-Befehl aufrufen, hinter dem Sie in Klammern den korrekten Code eingeben:

```
PRINT CHR$(205)
```

Dieser Befehl zeichnet beispielsweise ein kleines Dreieck.

Bei den Codes 1 bis 31 muß man allerdings ein wenig anders vorgehen. Sie sind in der Reihe der offiziellen ASCII-Codes nicht für die Wiedergabe von Symbolen oder Buchstaben vorgesehen. Um sie dennoch handhaben zu können, geben Sie dem MSX für jeden dieser Codes einen doppelten Code an. Für jeden Code zwischen 1 und 31 geben Sie zunächst diesen ein und erhöhen ihn anschließend um 64.

```
PRINT CHR$(1);CHR$(65)
```

läßt ein lachendes Gesicht auf Ihrem Bildschirm erscheinen.

Code	Zeichen	Code	Zeichen	Code	Zeichen	Code	Zeichen
0	(Null)	15	☼	30	↖	45	—
1	☺	16	┼	31	┼	46	.
2	☻	17	┴	32	(Leerzeichen)	47	/
3	♥	18	┬	33	!	48	0
4	♦	19	┤	34	"	49	1
5	♣	20	├	35	#	50	2
6	♠	21	┼	36	$	51	3
7	•	22	│	37	%	52	4
8	◘	23	—	38	&	53	5
9	○	24	┌	39	'	54	6
10	◙	25	┐	40	(	55	7
11	♂	26	└	41	)	56	8
12	♀	27	┘	42	*	57	9
13	♪	28	×	43	+	58	:
14	♫	29	/	44	,	59	;

Code	Zeichen	Code	Zeichen	Code	Zeichen	Code	Zeichen
60	<	100	d	140	î	180	Õ
61	=	101	e	141	ì	181	õ
62	>	102	f	142	Ä	182	Ũ
63	?	103	g	143	Å	183	ũ
64	@	104	h	144	É	184	Ĳ
65	A	105	i	145	æ	185	ĳ
66	B	106	j	146	Æ	186	¾
67	C	107	k	147	ô	187	∽
68	D	108	l	148	ö	188	◇
69	E	109	m	149	ò	189	‰
70	F	110	n	150	û	190	¶
71	G	111	o	151	ù	191	§
72	H	112	p	152	ÿ	192	▬
73	I	113	q	153	Ö	193	▚
74	J	114	r	154	Ü	194	▙
75	K	115	s	155	¢	195	▬
76	L	116	t	156	£	196	▪
77	M	117	u	157	¥	197	▇
78	N	118	v	158	₧	198	▌
79	O	119	w	159	ƒ	199	▟
80	P	120	x	160	á	200	▛
81	Q	121	y	161	í	201	▌
82	R	122	z	162	ó	202	▙
83	S	123	{	163	ú	203	
84	T	124	¦	164	ñ	204	⁄⁄
85	U	125	}	165	Ñ	205	▼
86	V	126	~	166	ª	206	▲
87	W	127		167	º	207	▶
88	X	128	Ç	168	¿	208	◀
89	Y	129	ü	169	⌐	209	⧗
90	Z	130	é	170	¬	210	⋈
91	[	131	â	171	½	211	▮
92	\	132	ä	172	¼	212	▮
93	]	133	à	173	¡	213	▪
94	^	134	å	174	«	214	▪
95	_	135	ç	175	»	215	✠
96	`	136	ê	176	Ã	216	△
97	a	137	ë	177	ã	217	ǂ
98	b	138	è	178	Ĩ	218	ω
99	c	139	ï	179	ĩ	219	▉

Code	Zeichen	Code	Zeichen	Code	Zeichen	Code	Zeichen
220	▄	229	σ	238	∈	247	≈
221	▌	230	μ	239	∩	248	°
222	▐	231	γ	240	≡	249	•
223	▀	232	Φ	241	±	250	·
224	α	233	Θ	242	≥	251	√
225	β	234	Ω	243	≤	252	η
226	Γ	235	δ	244	⌠	253	²
227	Π	236	∞	245	⌡	254	■
228	Σ	237	Ø	246	÷	255	

FEHLERMELDUNGEN

Wenn der MSX in einem Programm einen Fehler entdeckt, stoppt er das Programm, und auf dem Bildschirm erscheint eine Meldung. Anhand dieser Meldung kann der Benutzer rasch erkennen, um was für einen Fehler es sich handelt. Jede Fehlermeldung trägt eine eigene Nummer. Dieser Nummer können Sie sich bedienen, um das Programm bei Entdeckung eines Fehlers in eine bestimmte Subroutine springen zu lassen. Dazu dient der Befehl

ON ERROR GOTO ...

Hinter GOTO geben Sie die Zeilennummer an, in die der Computer springen muß, falls ein Fehler festgestellt wird.

Die Fehlermeldungen sind alphabetisch geordnet.

56 – Bad file name (fehlerhafter Dateiname)

Die Bezeichnung der Datei ist nicht erlaubt.
Die Bezeichnung des Apparates ist bei diesem OPEN-, SAVE- oder LOAD-Befehl nicht erlaubt.

52 – Bad file number (fehlerhafte Dateinummer)

Sie haben einen Befehl oder ein Statement gebraucht, das auf eine nicht existierende Datei verweist oder außerhalb des Bereiches von MAXFILES liegt.
Sie haben PRINT # bei einer Dateinummer gebraucht, die noch nicht eröffnet ist.

17 – Can't CONTINUE (kann nicht fortfahren)

Sie wollen das Programm weiterlaufen lassen, obwohl es wegen eines Fehlers oder mit END gestoppt ist oder aber nach der Unterbrechung korrigiert wurde.
Das von Ihnen aufgerufene Programm existiert nicht.
Das Programm enthält bereits einen CONT-Befehl.

19 – Device I/O error (Fehler bei Ein- oder Ausgabe)

Es ist etwas nicht in Ordnung mit der Eingabe seitens des Kassettenrekorders. Oder der Rekorder arbeitet nicht korrekt, oder die Kassette ist nicht einwandfrei. Das Programm kann nicht geladen werden.
Der Ladevorgang wurde irgendwo abgebrochen.
Sie benutzen eine falsche I/O-Buchse.
Die Lautstärkeregelung des Kassettenrekorders ist nicht in Ordnung.
Sie können diesen Fehler nicht beheben. Sie müssen die Kassette zurückspulen, die Fehlerursache eventuell beseitigen und den Ladevorgang wiederholen.

57 – Direct statement in file (direkter Befehl in Datei)

In der Datei, die gerade geladen wird, findet der Computer einen direkten Befehl (ohne Zeilennummer). Der Ladevorgang wird abgebrochen.
Sie wollen eine Datei als BASIC-Programm laden, die kein BASIC-Programm ist.

11 – Division by zero (Division durch Null)

Die Division durch 0 ist nicht möglich.
Achtung! Nicht definierte Variablen werden vom MSX immer als Null angesehen. Wenn Sie durch eine nicht definierte Variable dividieren, dividieren Sie durch Null, was zu einer Fehlermeldung führt.

54 – File already open (Datei ist bereits eröffnet)

Sie wollen eine Datei eröffnen, die bereits eröffnet ist.

53 – File not found (Datei nicht gefunden)

Ein LOAD- oder OPEN-Befehl verweist auf eine nicht existierende Datei.

59 – File not OPEN (Datei ist nicht eröffnet)

Sie wollen Daten aus einer Datei lesen bzw. in eine Datei schreiben, die noch nicht eröffnet ist.

12 – Illegal direct (direkt nicht erlaubt)

Sie wollen dem Computer einen direkten Befehl übermitteln, obgleich ein solcher Befehl nur in ein Programm aufgenommen werden kann.

5 – Illegal function call (unerlaubter Funktionsaufruf)

Sie teilen einer mathematischen oder einer Stringfunktion einen unkorrekten Parameter zu. Das bedeutet, daß dieser Wert nicht zusammen mit diesem Befehl eingegeben werden darf.

Die Werte einer Funktion liegen außerhalb des zur Verfügung stehenden Bereiches.

55 – Input past end (Eingabe nach Abschluß)

Sie wollen Daten aus einer Datei lesen, aus der bereits alle Daten gelesen worden sind. Diese Fehlermeldung erscheint ebenfalls, wenn die Datei aufgrund eines Fehlers in der Programmierung keinerlei Daten beinhaltet.

51 – Internal error (interner Fehler)

Im Computer ist etwas nicht in Ordnung. Meist handelt es sich dabei um das BASIC-Übersetzungsprogramm.

25 – Line buffer overflow (Zeilenpuffer voll)

Der Zeilenpuffer ist voll. Die eingegebene Zeile umfaßt zu viele Zeichen.

24 – Missing operand (Parameter fehlt)

Sie haben bei einem Befehl zu wenig Parameter verwandt.

1 – NEXT without FOR (NEXT ohne FOR)

Der Computer findet einen NEXT-Befehl, dem kein FOR-Befehl vorausgegangen ist. Dies kann auch passieren, wenn Sie mit einem GOTO oder GOSUB auf eine Zeile innerhalb einer FOR...NEXT...-Schleife verweisen.

Die Variable hinter dem NEXT-Befehl stimmt nicht mit der Variablen hinter dem FOR-Befehl überein.

21 – NO RESUME (kein RESUME)

In der Fehlerkorrekturroutine befindet sich dort, wo der Computer es erwartet, kein RESUME. Sie müssen jede Fehlerkorrekturroutine (siehe zu Beginn dieses Kapitels die Bemerkungen über ON ERROR GOTO) mit END, RESUME oder ON ERROR 0 abschließen.

4 – Out of DATA (kein DATA mehr vorhanden)

Sie wollen eine Angabe mit READ lesen, obgleich alle Daten bereits gelesen sind. Sie haben die DATA-Befehle vergessen.

7 – *2Out of memory (nicht genügend Speicherplatz vorhanden)

Der Computer verfügt zur Ausführung Ihres Befehles über nicht genügend Speicherplatz. Ursache dafür kann ein zu langes Programm sein; es kann aber auch an zu vielen FORs, GOSUBs, Variablen, einem zu langen Array, zu vielen Arrays oder zu vielen Funktionen liegen.

14 – Out of string space (kein Stringplatz mehr vorhanden)

Der für die Arbeit mit Strings vorgesehene Speicherplatz reicht zu dem von Ihnen gedachten Zweck nicht aus. Sie können diesen Speicherplatz mit CLEAR erweitern.

6 – Overflow (Bereichsüberschreitung)

Die Variable, Funktion oder das Statement paßt nicht zu dieser Zahl; sie ist zu groß.

22 – RESUME without error (RESUME ohne Fehler)

Der Computer findet einen RESUME-Befehl ohne den zugehörigen ON ERROR-Befehl.
Diese Meldung kann auch vorkommen, wenn das Hauptprogramm nicht mit dem END-Befehl abgeschlossen wurde oder wenn mit GOTO auf RESUME verwiesen wird.

3 – RETURN without GOSUB (RETURN ohne GOSUB)

Der Computer findet einen RETURN-Befehl, dem kein GOSUB vorausgegangen ist. Diese Meldung erscheint auch, wenn am Ende eines Hauptprogrammes ein END bzw. hinter einer Subroutine ein GOTO fehlt.

10 – Redimensioned array (Array wird wieder dimensioniert)

Sie wollen einen bereits existierenden Array erneut definieren.
Sie benutzen undefinierte Arrayvariablen, die nachträglich dimensioniert werden.

16 – String formula too complex (Stringformel zu komplex)

Eine Stringvariable ist zu kompliziert.

15 – String too long (String zu lang)

Der von Ihnen benutzte String umfaßt mehr als 255 Schriftzeichen

9 – Subscript out of range (Index außerhalb des Bereiches)

Sie geben einer Arrayvariablen eine höhere Nummer, als es entsprechend der Dimensionierung des Arrays möglich wäre.

Sie können ohne Dimensionierung Arrays mit einer Maximallänge von 10 verwenden. Wenn Sie einer Arrayvariablen eine höhere Nummer als 10 zuweisen, müssen Sie den Array zunächst dimensionieren.

2 – Syntax error (Syntaxfehler)

Sie haben innerhalb der BASIC-Grammatik einen Fehler gemacht.

13 – Type mismatch (Typ ist nicht korrekt)

Sie verwenden die falschen Werte. In einem LET-Befehl finden sich zu beiden Seiten des Gleichheitszeichens unterschiedliche Größen (etwa eine numerische Variable und eine Stringvariable).

Sie wollen eine mathematische oder logische Aufgabe mit einem String ausführen.

8 – Undefined line number (nicht definierte Zeilennummer)

In einem GOTO-, GOSUB-, IF...THEN...-, DELETE- oder RESUME-Befehl verwenden Sie eine nicht existierende Zeilennummer. Dies kann zuweilen bei der Neunumerierung eines Programmes (RENUM) vorkommen.

18 – Undefined user function (nicht definierte Benutzerfunktion)

Sie rufen eine FN-Funktion auf, ohne diese zuerst mit DEF FN definiert zu haben.

23 – Unprintabel error (nicht darstellbarer Fehler)

Der Computer hat einen Fehler entdeckt, zu dem es keine Fehlermeldung gibt.

26/49 – Unprintable error (nicht darstellbarer Fehler)

Nachträgliche Erweiterungen vorbehalten.

60/255 – Unprintable error (nicht darstellbarer Fehler)

Diese Fehlermeldungen sind der Definition durch den Benutzer vorbehalten.

20 – Verify error (Kontrollfehler)

Das Programm im Speicher des MSX entspricht nicht dem Programm auf der Kassette.

SACHWORTVERZEICHNIS

A 175
A0 175
A1 175
A2 175
A3 175
Äffchen 178
AKZENT-Taste 11
AND 68
Antennenschalter 8
Arrays 95
Arrays, dreidimensionale 98
Arrays, zweidimensionale 96
ASC 77
ASCII-Code 225
ASCII-Wert 77
Auflösung 157

B 174
Bad file name 227
Bad file number 227
BASIC 2
BASIC-Interpreter 2
Befehle, direkte 21
Bildschirmarten 17
Bildwiedergabe 43
Blanco 174
BS-Taste 13
Buchstabentasten 10

Can't CONTINUE 227
Centronics 6
CHR$ 225
CHR$ 77
CLEAR 89
COLOR 137
Computersprachen 2
CPU 3
CSAVE 34
CTRL-Taste 12
Cursor 27

D 171
DATA 91
Dateien 109
Datenspeicherung 6
default 135
DEFDBL 41
DEF FN 132
DEFINT 41
DEFSNG 41
DEFSTR 41
DEL 23
DEL-Taste 13
dezimal 82
DIM 95
Diskette 110
Diskette, formatiert 35
Disketten 7
Diskettendateien 109
DRAW 171
DRAW 170
dreidimensional 205
Drucker 74

E 171
Eingabe 43
Ellipsen 164
ELSE 67
ENDE 59
Energieverbrauchsrechnung 99
ESC 10
Extra ignored 44

F 171
Fakultät 62
Färben 167
Fehlerkorrektur 22
Fehlermeldungen 227
Feinregister 202
Fernbedienung 117
Fernsehgerät als Bildschirm 8
Flußdiagramm 31
FOR...NEXT 58
Frequenz 202
Funktionen 131
Funktionstasten 14

G 171
Geräuscheffekte 200
Geräuschgenerator 189

Gestaltungstasten 13
GOSUB 14
GOSUB 124
Gradumwandlung 33
Grafik 135
Grafik und Farbe 149
grafische Chips 135
GRAPH-Taste 11
Grobregister 202

H 171
hexadezimal 82
HOME 25
HOME-Taste 13

IF...THEN 65
illegal function call 127
INS 23
INS-Taste 13
INSTR 83

Kassette 110
Kassettendateien 109
Kassettenrekorder 34
KEY 14
KEY OFF 14
KEY ON 14
Kollision 185
Kontrolltasten 10
Koordinaten 173
Kreise 164
Kubus 219
Kuchenstücke 164
Kurven 161

L 171
Lautstärke 197
Layout 75
LEFT$ 83
LEN 83
LEY LIST 14
LINE INPUT 89
Listing 19
LOCATE 53

Matrizen 207
Melodie 193
MID$ 83
Mikroprozessor 3
MOD 124
Monitor 8
MSX BASIC-Version 9
MSX-System 1
Multiplikationszeichen 38
MUSIK 189
NEW 60
NOT 68

oktal 82
ON...GOSUB 126
ON KEY GOSUB 128
ON SPRITE GOSUB 186
ON STRIG GOSUB 129
Operatoren, logische 68
OR 68

PAINT 168
Palindrom 88
PFEIL-Tasten 13
PI 166
PLAY 190
PRESET 150
PRINT 21
PRINT USING 51
Programme herstellen 28
Projektion 208
PSET 160
PSG 189
Puffer 198
Punkte 159
Pyramide 221

QWERTY 10

R 171
Rakete 142
RAM 4
Raster 136
READ 92
Rechnen 75
Register 202
REM 124
RESET-Taste 13

RIGHT$ 83
RND 146
ROM 4
Rotation 210

S 174
Scale 174
Schirm 0 135
Schirm 1 137
Schirm 2 138
Schirm 3 139
Schleifen 55
SCREEN 181
SCREEN 0 47
SHIFT 11
Sortieren 75, 103
SOUND 190
SPACE$ 83
Speicher: RAM 4
Speicher: ROM 4
Spriteflächen 180
Sprites 177
SPRITES 19
Sprites, farbige 184
SQR 132
Sternchen * 38
STICK 158
STOP-Taste 12
Streichhölzer 144
Striche 160
STRIG 129
STRING$ 83
STRINGS 73
Stringvariablen 75
Subroutinen 123
Symbol 50
Syntax error 22

TAB 48
Telefonnummern 93
Text 168
Textverarbeitung 73
TIME 59
Ton 8
Töne 195

U 171
Uhr 58

V 195
Variablen 37
Vergleichsoperatoren 88
Video 6
Vogelspinne 183
Volume 195

Weinglas 215
WIDTH 47
Winkel 175

Z-80A 3
Zahlensysteme 79
Zeilennummer 29

Weitere Bücher zum Thema:

Kenneth P. Goldberg / Robert D. Sherwood

Mikrocomputer: Leitfaden für Eltern

1985. ISBN 3-89028-018-8, 256 Seiten

War es vor ein paar Jahren die Mengenlehre, so sind es heute die Computer, bei denen Eltern ihren Kindern oftmals eine Antwort schuldig bleiben. Mehr und mehr halten Mikrocomputer ihren Einzug in die Schule, und zuhause werden sie zu einem der beliebtesten Hobbies. Dieses Buch soll Eltern helfen, ihren Kindern den Übergang ins Computer-Zeitalter zu erleichtern. Es bietet in leicht verständlicher Sprache einen Überblick über Arbeitsweise und Gebrauch der Mikrocomputer, eine kurze Einführung in die Programmiersprache BASIC und gibt Ratschläge, die Eltern beim Kauf eines Mikrocomputers beachten sollten.

John Heilborn / R. Talbott

Commodore 64 Anwenderhandbuch

1984. ISBN 3-89028-016-1, 466 Seiten

Dieses leicht verständliche, durchgehend illustrierte Anwenderhandbuch vermittelt das nötige Wissen für den Umgang mit dem Commodore 64 und seinen Zusatzgeräten. Dem Anfänger bietet das Buch eine Anleitung für den Aufbau und den Betrieb des C-64 und seiner Peripherie, eine vollständige Einführung in die C-64 BASIC-Programmierung, und eine Darstellung der besonderen Grafik- und Tonfähigkeiten des C-64.
Dem fortgeschrittenen Benutzer dient das Buch als unschätzbares Arbeitsmittel und Nachschlagewerk. Es enthält eine detaillierte Übersicht aller BASIC-Statements und Funktionen, ein ausführliches Verzeichnis der Speicheradressierung zusammen mit einer Erklärung, was jede adressierbare Speicherzelle leistet und wie sie arbeitet, und einen besonderen Abschnitt zur fortgeschrittenen Colorgrafik und Sprite-Grafik.

L. Poole

Praktische BASIC-Programme für den IBM Personal Computer

1983. ISBN 3-89028-000-5, 172 Seiten

Die Anzahl der im Augenblick erhältlichen IBM PC-Software ist noch sehr begrenzt. Mit diesem Buch stehen nun den Benutzern von IBM PCs 34 praktische Programme in deutscher Sprache zur Verfügung, die unmittelbar so eingegeben werden können, wie sie im Buch abgebildet sind. Für wenig Geld erhält man damit nützliche Anwendungen für das Büro, für Übungskurse und für zuhause.

A. Fox / D. Fox

BASIC ganz einfach

1984. ISBN 3-89028-002-1, 246 Seiten

Schritt für Schritt wird der Leser durch zahlreiche Übungen und Illustrationen auf humorvolle Art mit BASIC vertraut gemacht. Man benötigt dazu nicht einmal einen Computer und erst recht keine technischen oder mathematischen Vorkenntnisse. Eine unkomplizierte Einführung in die populäre Computersprache für jung und alt.

T. Hogan

CP/M Anwenderhandbuch

1984. ISBN 3-89028-005-6, 286 Seiten

Dieses Anwenderhandbuch zum bekanntesten Mikrocomputer-Betriebssystem der Welt diskutiert die neuesten CP/M-Entwicklungen einschließlich CP/M 86 und das Betriebssystem für 8086- und 8088-Mikrocomputer, wie zum Beispiel den IBM PC. Der Band stellt auch CP/M-Ableitungen wie MP/M und CP/NET vor.

Consumer Guide (Hrsg.)

BASIC Schnellkurs

1984. ISBN 3-89028-020-X, 88 Seiten

Hier ist Hilfe für Computer-Interessierte, deren Pläne zum Einsatz von Mikrocomputern mangels Kenntnis einer Programmiersprache bisher scheiterten. Diese leicht verständliche, illustrierte Einführung vermittelt auf 88 Seiten Grundkenntnisse der Programmiersprache BASIC. Sehr schnell wird der Anwender BASIC-Befehle eingeben, Figuren zeichnen und eigene Programme für spezielle Zwecke erstellen.

McGraw-Hill Book Co. GmbH · Lademannbogen 136 · D-2000 Hamburg 63 · Tel.: (040) 5382081

Wichert van Engelen

MSX-Computer-Anwenderhandbuch

Die MSX-Computer und der MSX-Standard erobern Europa. Diese kompakten und vielseitigen Computer bauen ganz auf die große Austauschbarkeit von Programmen und Peripheriegeräten. Dieses Buch ist eine umfassende Einführung in die Benutzung der MSX-Computer. Es beginnt mit einem Kurs über MSX-BASIC und demonstriert mit ausführlichen Programmbeispielen sogar das Komponieren von Musikstücken und die Entwicklung dreidimensionaler Zeichnungen.
Eine Einführung in die Arbeit mit Variablen ermöglicht es speziell dem Anfänger, auf MSX-Computern selbst zu programmieren. Der wachsende Schwierigkeitsgrad der Aufgaben fordert sowohl den Anfänger als auch den erfahrenen MSX-Anwender. Durch das ausführliche Stichwortverzeichnis eignet sich das Buch auch als Nachschlagewerk.

ISBN 3-89028-034-X