Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 13

Alles über Speicheraufbau

Joysticks ohne Mechanik

Funktionale Arbeitsplätze

Spannende Spiele

Der Sharp MZ-711

Computer und Astronomie

Ein wöchentliches Sammelwerk

computer kurs

Heft 13

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

Computer Kurs ist ein wöchentlich erscheinendes Sammelwerk. Die Gesamtzahl der Hefte ergibt ein vollständiges Computer-Nachschlagewerk. Damit Sie jede Woche Ihr Heft erhalten, bitten Sie Ihren Zeitschriftenhändler, Computer Kurs für Sie zu reservieren.

Zurückliegende Hefte
Ihr Zeitschriftenhändler besorgt Ihnen gerne zurückliegende Hefte. Sie können sie aber auch direkt beim Verlag bestellen.

Deutschland: Das einzelne Heft kostet DM 3,80. Bitte füllen Sie eine Postzahlkarte aus an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Computer Kurs

Österreich: Das einzelne Heft kostet öS 30. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs, Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Computer Kurs.

Schweiz: Das einzelne Heft kostet sfr 3,80. Bitte überweisen Sie den Betrag durch die Post (grüner Einzahlungsschein) auf das Konto: Schmidt Agence AG, Kontonummer Basel 40-879, Kennwort: Computer Kurs, und notieren Sie ihre Bestellung auf der Rückseite des Giroabschnittes (rechter Abschnitt).

Abonnement

Sie können Computer Kurs auch alle 2 Wochen (je 2 Ausgaben) per Post zum gleichen Preis im Abonnement beziehen. Der Abopreis für 12 Ausgaben beträgt DM 45,60 inkl. Mwst., den wir Ihnen nach Eingang der Bestellung berechnen. Bitte senden Sie Ihre Bestellung an: Marshall Cavendish Int. Ltd. (MCI), Sammelwerk Service, Postgiroamt Hamburg 86853-201, Postfach 105703, 2000 Hamburg 1, Kennwort: Abo Computer Kurs. Bitte geben Sie an, ab welcher Nummer das Abo beginnen soll und ob Sie regelmäßig für jeweils 12 Folgen einen Sammelordner wünschen. Bei Bestellungen aus Österreich oder Schweiz senden Sie Ihren Auftrag bitte auch an die Hamburger Adresse. Berechnung und Zahlung erfolgen in Landeswährung zum Ladenpreis.

WICHTIG: Bei Ihren Bestellungen muß der linke Abschnitt der Zahlkarte Ihre vollständige Adresse enthalten, damit Sie die Hefte schnell und sicher erhalten. Überweisen Sie durch Ihre Bank, so muß die Überweisungskopie Ihre vollständige Anschrift gut leserlich enthalten.

SAMMELORDNER
Sie können die Sammelordner entweder direkt bei Ihrem Zeitschriftenhändler kaufen (falls nicht vorrätig, bestellt er sie gerne für Sie) oder aber Sie bestellen die Sammelordner für den gleichen Preis beim Verlag wie folgt:

Deutschland: Der Sammelordner kostet DM 12. Bitte füllen Sie eine Zahlkarte aus an: Marshall Cavendish International Ltd. (MCI), Sammelwerk-Service, Postgiroamt Hamburg 48064-202, Postfach 105703, 2000 Hamburg 1, Kennwort: Sammelordner Computer Kurs.

Österreich: Der Sammelordner kostet öS 98. Bitte füllen Sie eine Zahlkarte aus an: Computer Kurs Wollzeile 11, 1011 Wien, Postscheckkonto Wien 7857201 oder legen Sie Ihrer Bestellung einen Verrechnungsscheck bei. Kennwort: Sammelordner Computer Kurs

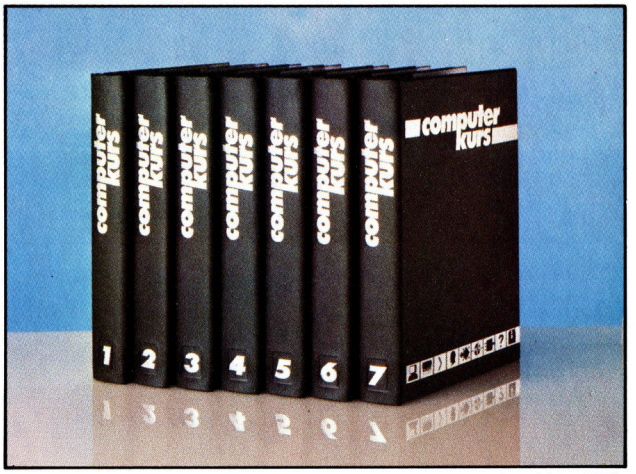
Schweiz: Der Sammelordner kostet sfr 15. Bitte überweisen Sie den Betrag durch die Post (grüner Einzahlungsschein) auf das Konto: Schmidt Agence AG, Kontonummer Basel 40-879, Kennwort: Sammelordner Computer Kurs, und notieren Sie Ihre Bestellung auf der Rückseite des Giroabschnittes (rechter Abschnitt).

INHALTSVERZEICHNIS

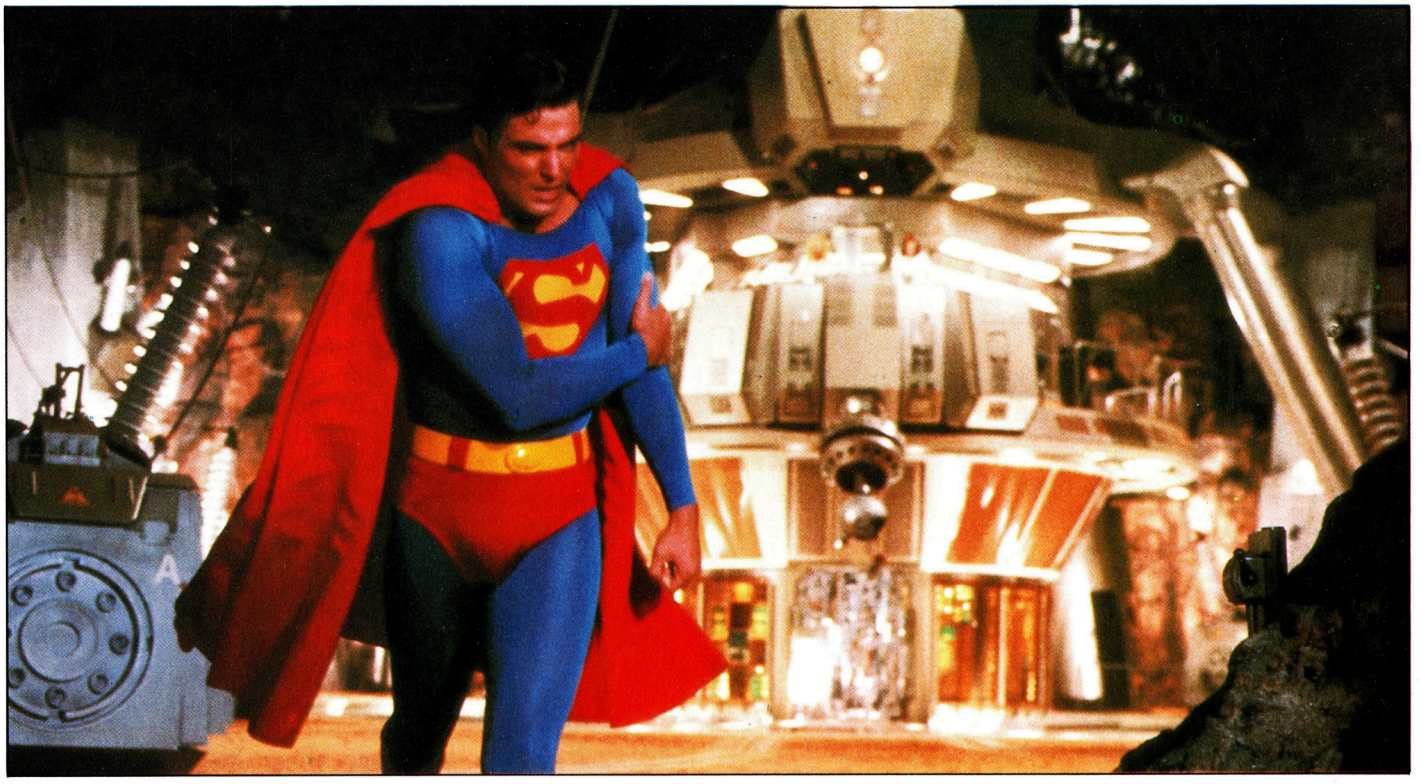
Alle 12 Hefte erscheint ein Teilindex. Die letzte Ausgabe von Computer Kurs enthält den Gesamtindex — darin einbezogen sind Kreuzverweise auf die Artikel, die mit dem gesuchten Stichwort in Verbindung stehen.

Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Joachim Seidel, Elke Leibinger, Susanne Brandt, Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1, Tel.: 040/23 40 80



Druck: E. Schwend GmbH, Schmollerstraße 31, 7170 Schwäbisch Hall



Computer-Kriminalität ist das Thema des 3. Supermann-Films. Richard Pryor verdient durch eine „Umbuchung" eines halben Cents bei jeder Bank-Transaktion ein Vermögen. Dieser Teil der Geschichte basiert übrigens auf mehreren realen Vorbildern. Der Film schließt mit der Zerstörung des weltgrößten Computers, der einzig für kriminelle Zwecke entworfen und gebaut worden ist.

Computer-Literatur

In der Science Fiction-Literatur hatte der Computer immer seinen festen Platz. Schriftsteller haben schon vor langer Zeit Technologien vorhergesagt, die erst heute Wahrheit werden.

Lange vor der Verwirklichung haben die Autoren von Zukunftsromanen und Science Fiction-Filmen technische Entwicklungen prophezeit, die heute eintreffen. Arthur C. Clarke, der den Film „2001 – Odyssee im Weltraum" schuf, veröffentlichte bereits in den frühen 50er Jahren – also fast zwanzig Jahre vor ihrer Entwicklung – einen Artikel über geostationäre Satelliten. Lange vor der Konstruktion von Robotern beschrieb Robert Heinlein in der Kurzgeschichte „Waldo" schon den ersten fernbedienten „Maschinenmenschen". Vorhersagen dieser Art haben Erfinder und Ingenieure zu revolutionären Neuentwicklungen angespornt.

Überlegene Intelligenz

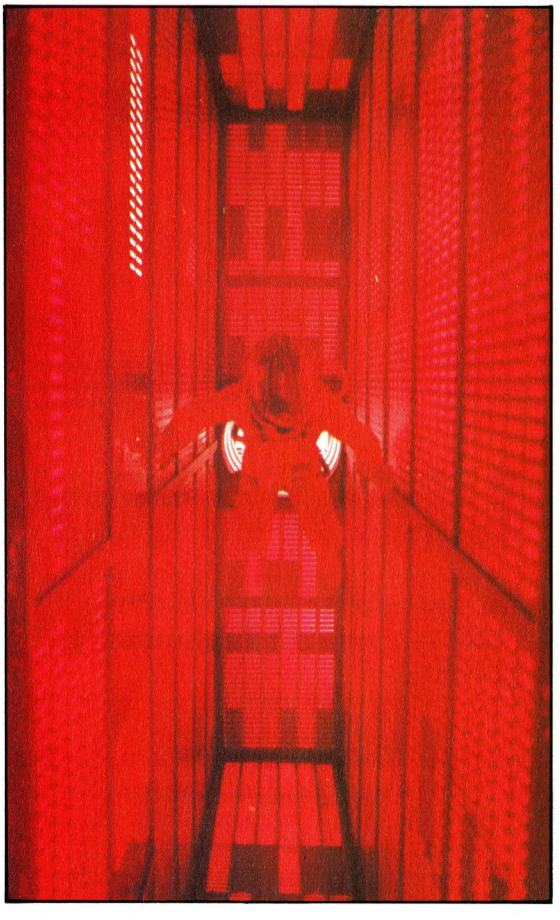
Computer aus den Fantasievorstellungen von Autoren sind zwangsläufig realitätsfern: So zum Beispiel in dem vor einigen Jahren gedrehten Film „Rollerball", wo ein kastenförmiger Roboter die perfekte Sprachein- und -ausgabe beherrscht. Die Fähigkeiten der mechanischen Statisten sind weniger interessant; sie dienen lediglich zur Dekoration. Der Anblick dieser Maschinen im Film hat aber in den 60er und 70er Jahren viel dazu beigetragen, den Geräten beim Publikum ein wenig von ihrem fast mystischen Ruf zu nehmen.

Die Forschungen von Charles Babbage auf dem Gebiet der mechanischen Rechenmaschinen Mitte des 19. Jahrhunderts bilden den Anfangspunkt kreativer Computer-Entwicklungen. Im Jahre 1879 beschrieb E. P. Mitchell die Verpflanzung einer Rechenmaschine in das Gehirn eines Verrückten, der dadurch zum Genie wurde. Damit nahm er schon eine zeitgenössische Entwicklung vorweg; denn seine „Denkhilfe" war miniaturisiert – klein genug, um im Schädel Platz zu haben, trotzdem aber so leistungsfähig, daß sie eine überlegene Intelligenz schuf. Auch die Idee der Verbindung von Mensch und Maschine wurde von Mitchell damit prophezeit. Heute, mehr als ein Jahrhundert später, gehen die ersten mit dem Zentralnervensystem verbundenen Geräte, etwa elektromechanische Prothesen, ihrer Perfektionierung entgegen.

Obwohl es unter den Schriftstellern auch berühmte Ingenieure gibt, verfügen die meisten Autoren über keine besonderen Kenntnisse

von der Arbeitsweise eines Computers. Aber die überzeugende Darstellung intergalaktischer Reisen erfordert auch nicht unbedingt das technische Wissen hochqualifizierter Astrophysiker oder Raketenexperten.

Der Computer in den Zukunftsromanen hat – verglichen mit dem heutigen technischen Standard – Fähigkeiten, von denen die Techniker noch kaum zu träumen wagen: So verfügt er zum Beispiel über eine enorme Speicherfähigkeit, kennt jede Information und alle einmal gedachten Ideen, zu denen er durch Methoden, die dem menschlichen Denken ähnlich sind, auch den sofortigen Zugang besitzt. Der Computer im Roman oder Film ist selbstverständlich auch allgegenwärtig, obwohl er jedem Benutzer vorspielt, dieser sei der einzige, der Zugang zur Maschine hätte. Natürlich kann der Rechner auch sprechen und Sprache verstehen. Dabei wirkt die Stimme nicht etwa synthetisch, sondern völlig „echt", und der menschliche Gesprächspartner wird durch sein individuelles Sprachmuster sofort erkannt. Für diese Computer der Superlative ist die Fähigkeit, Gegenstände zu sehen oder gar Lebensmittel aus elementaren Substanzen herzustellen, fast „normal". Oft haben die Rechner (über-)menschliche Charaktereigenschaften, die sie beinahe zu einem höheren Wesen machen. Das führt aber auch dazu, daß sie böswillig oder verwirrt sein können: Die computergesteuerte Bombe im Film „Dark Star" reagiert zunächst nach dem vorgegebenen Programm, entwickelt dann aber eine eigenständige Logik . . .



HAL – der Heuristisch programmierte ALgorithmische Computer in Clarkes „2001 – Odyssee im Weltraum" ist ein gutes Beispiel für den allmächtigen Rechner in Film und Literatur. Im Film kennt HAL – anders als die Besatzung – das Ziel der Raum-Mission und hält die Menschen deshalb für entbehrlich.

Diese Aussichten gehören zum Glück ins Reich der Fantasie. Einige der anderen Merkmale finden sich aber schon heute in modernen Computern wieder: Große Speicherkapazitäten mit kurzer Zugriffszeit sind schon Realität. Bereits in den frühen 80er Jahren wurden Gigabyte-(tausend Millionen Bytes)Speicher entwickelt. Hochtechnische Maschinen erreichten Millionen Operationen pro Sekunde. Bei der Sprachausgabe nähert man sich heute der Perfektion, die wir im Film schon bewundern können. Die Qualität hängt von der Speichergröße, der Rechengeschwindigkeit und der Programmqualität ab.

Größere Schwierigkeiten gibt es noch bei der Spracherkennung. Die unterschiedlichen Sprachmuster zweier Personen stellen den Computer vor das Problem des Erkennens von Ähnlichkeiten, ein Feld, auf dem der Mensch gegenwärtig noch haushoch überlegen ist.

Enorme Möglichkeiten

Das „Sehen", also das Erkennen von Gegenständen, steckt ebenfalls noch in den Kinderschuhen. Mit der Entwicklung leistungsfähiger Industrieroboter sind Computer allerdings heute schon in der Lage, mit Hilfe von angeschlossenen Kameras ein bestimmtes Teil aus einem ungeordneten Sortiment herauszusuchen. Das Wiedererkennen eines Gegenstandes hängt vom optischen „Vokabular" der Maschine, der Speichergröße und der Fähigkeit des Prozessors ab.

Zur Produktion von Nahrungsmitteln bleibt zu sagen, daß die Herstellung einer Mahlzeit, die etwa wie Fleisch und Kartoffeln riecht und schmeckt, wohl möglich wäre. Offen ist allerdings, wie dieses Gericht aussehen würde . . .

Nicht alle Schriftsteller führen Computer mit exotischen Eigenschaften vor: In einer 1969 erschienenen Erzählung von John Brunner, die im Jahre 2010 spielt, verfügt der Computer „Shalmaneser" zwar über erstaunliche Speichermöglichkeiten und eine hohe Geschwindigkeit – er steht mit jedem Fernsehgerät auf der Erde in direktem Kontakt –, seine Ein- und Ausgabeprozeduren erscheinen uns aber recht gewohnt:

PROGRAMM ABGELEHNT
F Grund der Ablehnung
FEHLER IN DEN BASISDATEN
F Spezifizieren
DATEN DER FOLGENDEN KATEGORIEN
NICHT BEKANNT: GESCHICHTE HANDEL
SOZIALVERHALTEN KULTUR
F Daten abarbeiten wie eingegeben
FRAGE BEDEUTUNGSLOS UND UNLÖSBAR

Brunner verwendet die Sprache hier so, daß auch ein nichtgeschulter Leser den Dialog Mensch–Maschine verstehen kann. Die Geschichte, in der die Menschheit durch Überbevölkerung und Hungersnöte bedroht ist, gewann mehrere Preise für Science Fiction-Literatur. In einigen Filmen und Büchern der letzten Jahre gewinnt der Computer als Element der Handlung zusätzlichen Raum – beispielhaft etwa in Walt Disneys „Tron". (Der Name stammt von einem Eingabekürzel der Computersprache ab – TRace ON.) Schauplatz des Films ist die Realität und eine zweite Welt innerhalb des Computers. In der „Außenwelt" treten Softwarefachleute und Programmierer auf. In der Maschine jedoch bildet ihre Konstruktion selbst den Hintergrund einer Auseinandersetzung, in der Programme und Betriebssysteme die Rollen der Akteure übernehmen.

Andere Werke der Literatur zeigen den Computer zwar nicht selbst, lassen aber keinen Zweifel, daß ohne Beteiligung leistungsfähiger Rechner das Szenario schlichtweg undenkbar wäre. Dazu zählen etwa Orwells „1984" und Huxleys „Schöne neue Welt". In diesen Zukunftsromanen beherrscht eine kleine Führungsschicht die Bevölkerung. Beide Bücher haben sicherlich zur Sensibilisierung gegenüber den Mißbrauchsmöglichkeiten des Computers beigetragen.

Unsere Darstellung der Rechner in der Zukunftsliteratur kann natürlich nicht vollständig sein. Die Kreativität der Autoren auf diesem Gebiet ist allerdings enorm: John Barth etwa schrieb das Buch „Giles Goat-Boy", welches als das Werk eines Supercomputers namens WESCAC dargestellt wird, der darin – auf mehr als 800 Seiten – seine Erlebnisse erzählt.

Man sollte zum Abschluß jedoch auch einmal daran erinnern, daß der Computer nicht nur Thema der Fantasy-Literatur ist. Aus Tausenden von – teilweise hochspezialisierten – Büchern über unsere Rechner und den Umgang damit sticht eins durch lebendige Schilderung besonders hervor: Tracy Kidders „Die Seele einer neuen Maschine" (bereits als Taschenbuch erschienen) ist die Entwicklungsgeschichte des 32-Bit-Minirechners „Eagle" von Data General. Obwohl die am Projekt beteiligten Manager und Ingenieure eine große Rolle darin spielen, ist erstmals die Maschine Hauptperson des Romans.

Traummaschine
Teils Fahrzeug, teils künstliche Intelligenz – so präsentiert sich Dr. Who's Tardis als Ausgeburt der Fantasie des Autors. Obwohl die Kommunikation mit dem Gerät über eine Tastatur und einen Bildschirm vor sich geht, scheint das Speichervermögen der Maschine unendlich groß zu sein.

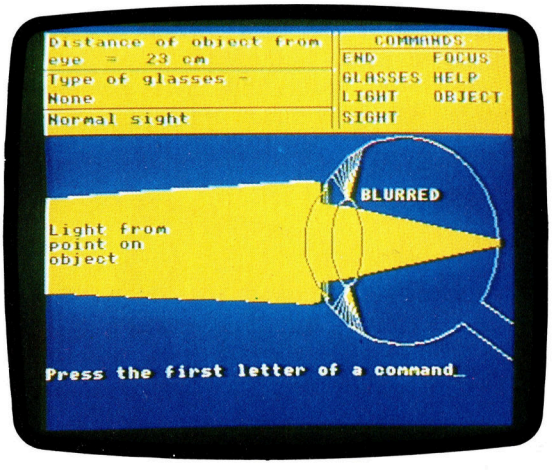
In dem Film „War Games" gerät ein Schüler, der mit einem Freund über das öffentliche Telefonnetz Daten austauschen möchte, versehentlich in den Verteidigungscomputer der NATO. Beim spielerischen „Hacken" setzt er dabei versehentlich den dritten Weltkrieg in Gang . . .

Spannende Spiele

Mit Hilfe von Simulations-Software können interessante Bildschirm-Experimente zu Hause und in der Schule durchgeführt werden.

So wie die bekannten Arcade-Spiele, bei denen Sie das Steuer eines Rennwagens oder eines Flugzeuges bedienen, versuchen Simulationsprogramme, die Realität so genau wie möglich abzubilden. Viele stellen dabei den erzieherischen Effekt in den Vordergrund und sind deshalb im Schulunterricht besonders geeignet, wo praktische Experimente zu gefährlich, zu zeitintensiv oder zu teuer wären.

Simulationsprogramme können aber auch anders eingesetzt werden. In einem Spiel mit dem Namen „Autoreise" müssen Kinder rechnerisches Geschick einsetzen, um ein Auto sicher zu lenken. Simulationsprogramme sind die vielleicht spannendste Software am Markt, doch leider nicht für alle Rechner verfügbar. Für den Commodore, den Sinclair, Apple oder Atari z. B., die auch in den Schulen bevorzugt werden, wurde gerade in der letzten Zeit hochwertige Simulationssoftware entwickelt.



Das Auge

Dieses Programm zeigt die Funktionsweise des Auges. Die Aufgabe besteht darin, alle Elemente genau zu justieren, um ein klares Sehen zu gewährleisten. Es simuliert den Weg der Lichtstrahlen vom Objekt zur Retina (Netzhaut, auf der die Bilder erzeugt werden). Sie können dabei die Gehirnfunktion nachahmen, indem Sie die Objektentfernung, die Größe der Iris und die Brennweite der Linse steuern, um das Bild auf der Retina scharf zu stellen.

Das auf dem Bildschirm dargestellte Schnittdiagramm des Auges enthält alle wichtigen Elemente mit ihren Bezeichnungen. Mit dem Befehl „LICHT" kann der Weg eines Lichtstrahls vom Objekt zum Auge konstruiert werden. Sowie dann die anderen Variablen richtig eingegeben sind, erscheint die Meldung „IN FOCUS" (Einstellung scharf). Beherrscht der Anwender die normalen Augenfunktionen, können Besonderheiten wie z. B. Kurzsichtigkeit simuliert werden. Zusätzlich läßt sich eine weitere Linse vor dem Auge simulieren, wenn es unmöglich ist, ein entferntes Objekt scharf einzustellen. Falls Sie die richtige Linse gewählt haben, wird das normale Sehvermögen wieder hergestellt, und Sie haben damit Ihre erste eigene Brille konstruiert. Dieses Programm wurde für den Physik- und Biologieunterricht entwickelt, hilft aber auch, Jugendliche mit dem Computer vertraut zu machen.

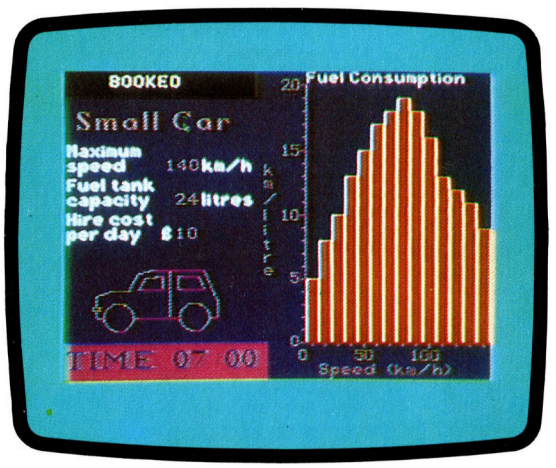
Ballonfahrt

Ballonfahrt ist ein Lehrprogramm fur Kinder im Alter zwischen acht und zwölf Jahren. Der Anwender übernimmt die Rolle des Piloten, der einen Heißluftballon zu steuern hat. Auf dem Bildschirm ist das Gelände, auf dem der Ballon anfangs steht. Am unteren Rand sind vier Instrumente dargestellt: Variometer, Thermometer, Höhenmesser und Kraftstoffanzeige. Es gibt nur zwei Steuerungsmöglichkeiten: den Gasbrenner, der die Luft anheizt und den Ballon steigen läßt, sowie ein Ventil, mit dem man Luft ablassen kann, um zu sinken. Eine einfache Einführungslektion lehrt den Schüler, den Ballon zu starten, zu fahren und zum Schluß sicher zu landen.

Nachdem der Pilot die wichtigsten Regeln beherrscht, kann er eigene Ausfluge unternehmen. Die Aufgabe besteht zum Beispiel darin, an einem vorbestimmten Punkt zu landen, wo der Ballonflieger neue Anweisungen erhält,

beispielsweise die Schafe eines Farmers zu retten, die auf einem markierten Feld zu finden sind. Geht unterwegs das Gas aus, so ist eine Zwischenlandung erforderlich.

Nach einigen Fehlstarts, Landungen in Bäumen und ähnlichen Mißgeschicken lernt man den Ballon richtig zu fliegen. Ferner können Sie bald anhand der Instrumente vorhersagen, wann Sie das Ventil betätigen oder an Höhe gewinnen müssen. Das Wichtigste ist dabei, den Umgang mit einem System zu üben, das eine gewisse Zeitverzögerung in Anspruch nimmt. Dieses Programm wird von Jungen und Mädchen gleichermaßen gern benutzt, da es viel Spaß macht.

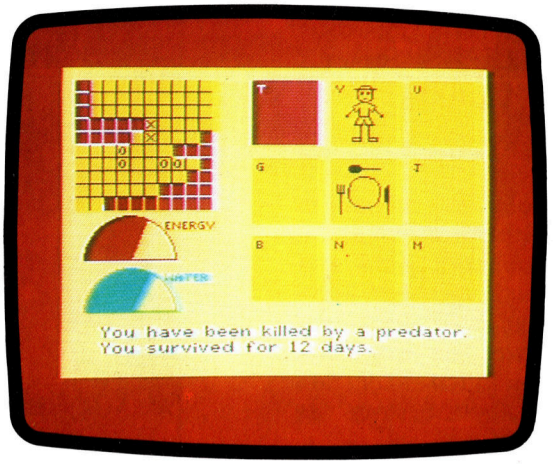


Autoreise

Bei diesem Programm übernimmt der Spieler die Rolle eines selbständigen Lieferanten. Hier müssen verschiedene Entscheidungen getroffen werden, z. B. welcher Liefervertrag anzunehmen ist, welches Fahrzeug verwendet werden kann oder wie schnell man fahren muß. Der Anwender hat dabei in seiner Kalkulation auch Geld, Entfernungen, die Zeit und den Benzinverbrauch zu berücksichtigen. Auf dem Bildschirm wird eine Landkarte von England gezeigt, die 15 Städte und die wichtigsten Straßenverbindungen beinhaltet. Ferner werden ein Tachometer, ein Meilenzähler, eine Tankanzeige und eine Uhr eingeblendet.

Zuerst muß entschieden werden, in welcher Stadt die Fahrt beginnt. Danach kann der Spieler aus einer Liste einen Kontrakt aussuchen, von dem er meint, daß er ihn erfüllen kann. Beispielsweise gilt es, eine Sendung Diamanten in Bristol um 12 Uhr abzuholen und in Dover vor 18 Uhr am gleichen Tag abzuliefern. Hierfür muß zuerst ein Fahrzeug gemietet werden. Ist der Spieler erfolgreich, bekommt er 1200 Mark sowie einen Zuschlag von 30 Mark, falls er noch früher ankommt. Anschließend wird ein neuer Auftrag übernommen. Hier müssen Zwischenstopps während der Nacht eingelegt werden, Reparaturen am Fahrzeug ausgeführt oder Strafe wegen zu schnellen Fahrens bezahlt werden. Und – wird der Auftrag nicht rechtzeitig erledigt – bekommt der Spieler eine 300-Mark-Vertragsstrafe. Soll eine Schwergutladung übernommen werden, muß das Fahrzeug gegen ein größeres ausgewechselt werden.

Dadurch steigen zwangsläufig Miete und Kraftstoffverbrauch. Die Autoreise vermittelt dem Spieler neben Grundkenntnissen über Fahrzeuge und Straßen zusätzliche Geschicklichkeit in der Entscheidungsfindung.



Überlebenstraining

Falls Sie schon immer wissen wollten, wie es ist, im Fell eines Löwen oder gar einer Maus zu stecken, dann ist das Überlebenstrainings-Programm genau das Richtige für Sie. Es ermöglicht dem Spieler, in die Rolle eines Tieres zu schlüpfen (man hat die Auswahl zwischen Habicht, Rotkehlchen, Löwe, Maus, Fliege und einem Schmetterling) und interessante Erfahrungen zu sammeln.

Die Umwelt wird auf dem Bildschirm als ein Gitter von Feldern dargestellt, und der Spieler bewegt sich über diese Felder mit Tasteneingaben. Die Hauptaufgabe ist es, Nahrung zu finden (Felder mit einem 0) sowie Feinden zu entkommen (mit einem X gekennzeichnet). Nähert sich der Spieler einem markierten Feld, wird auf der rechten Bildschirmseite vergrößert angezeigt, welcher Nahrung bzw. welchem Gegner er sich gegenübersieht. Außerdem werden zwei Anzeigen eingeblendet, die den bisherigen Energie- und Wasserverbrauch angeben. Wenn der Energievorrat zur Neige geht, muß schnellstens Nahrung gefunden werden; bei Wassermangel führt der Weg zu einem blau markierten Feld (Fluß). Fällt das Tier aber hinein, ertrinkt es.

Einige Lebewesen haben ein schweres Los. So etwa der Schmetterling, der nur die schwer zu findenden Blumen als Nahrungsquelle gebrauchen kann. Dem Habicht hingegen dienen Schnecken, Fliegen und Mäuse als Nahrung. Der Raubvogel hat in dem Jäger einen ernstzunehmenden Gegner. Im Spiel lernen Sie, wie sich verschiedene Tierarten ernähren, und wie Sie in der Wildnis überleben können.

Feuer-Knüppel

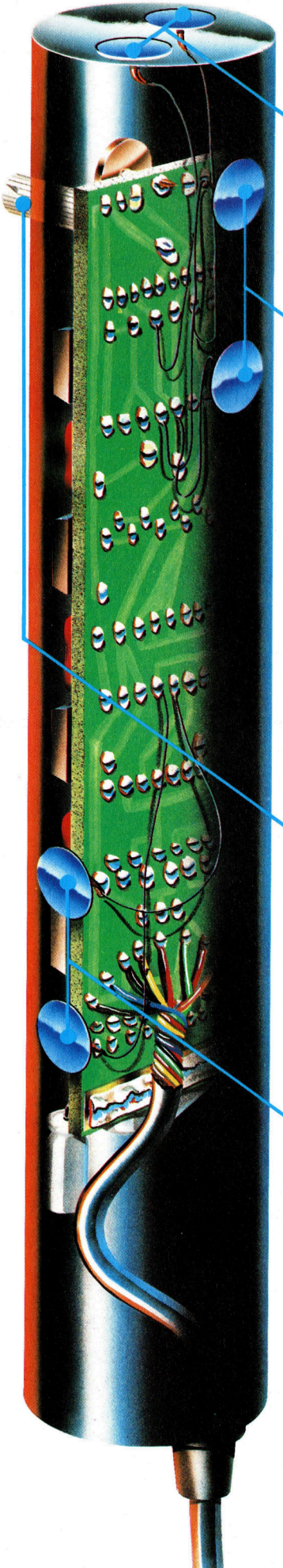
Joysticks gibt es auch ohne die übliche Schaltermechanik. Die eine Variante arbeitet mit Quecksilberschaltern, die andere nutzt elektromagnetische Signale des Körpers.

In der Computer-Branche ist man an eine rapide technische Entwicklung gewöhnt, und zwar nicht nur bei den Rechnern selbst, sondern auch bei der Peripherie und beim Zubehör. So gibt es neben dem bereits erklärten „klassischen" Joystick inzwischen zwei völlig anders arbeitende Steuerknüppel, die mit dem konventionellen mechanischen System fast nichts mehr gemeinsam haben.

Eine Methode zur Umsetzung von Ansteuerung in computergerechte Signale bietet der „Trickstick" der Firma East London Robotics. Dieses Gerät nutzt den menschlichen Körper als Erdung – der Körper führt nämlich infolge der harmlosen elektromagnetischen Abstrahlung aller Installationsleitungen in Häusern eine meßbare Wechselspannung. Der Trickstick besteht aus einem beidseitig geschlossenen Rohr mit Kunststoffüberzug, das senkrecht

Die Funktion des Trickstick beruht auf elektrischer Spannung, die infolge elektromagnetischer Abstrahlung von Installationsleitungen auftritt. Die Sensorflächen registrieren je nach Stärke des Fingerdrucks unterschiedlich hohe Spannungen und lösen so die Schaltung aus.

Trickstick



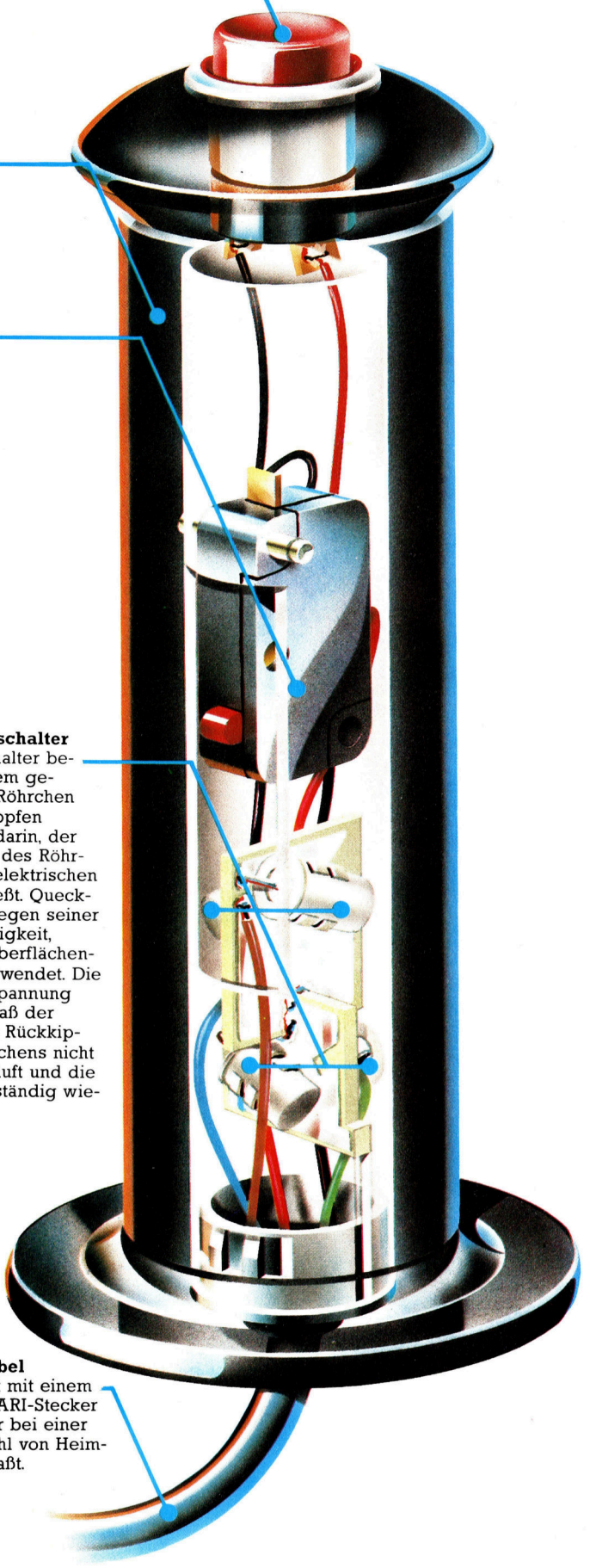
Horizontale Bewegungssteuerung
Indem Sie den Daumen zwischen diesen beiden Kontaktflächen hin- und herbewegen, können Sie horizontale Bewegungen auf dem Bildschirm steuern.

Vertikale Bewegungssteuerung
Beim Berühren der oberen Sensorfläche wird eine Aufwärtsbewegung, beim Berühren der unteren Fläche eine Abwärtsbewegung am Bildschirm ausgelöst.

Empfindlichkeitseinstellung
Damit muß der Trickstick individuell auf den bei jedem Benutzer unterschiedlichen Antenneneffekt eingestellt werden.

Feuerknöpfe
Die beiden Sensorflächen erzeugen voneinander unabhängige Signale, so daß man z. B. gleichzeitig die eine zum Bombenabwurf und die andere zum Abfeuern der Laserkanone benutzen kann.

Le Stik



in beiden Händen gehalten wird. In der Rohrwand sind paarweise Sensorflächen angeordnet: Das eine Paar steuert die Hin- und Herbewegung (horizontal), das zweite dient zum Auf- und Abwärtsfahren, und das dritte Paar sind die Feuerknöpfe.

Die vom Körper aufgefangene Spannung wird über die Sensorfelder auf eine empfindliche Verstärkerschaltung gegeben, die die Informationen in computergerechte Daten umwandelt. Je stärker Sie auf die Sensorfläche drücken, desto höher wird (bei entsprechender Verstärkerauslegung) die eingespeiste Spannung. Daher vereint der Trickstick in sich die Proportionaleigenschaften eines Analog-Joysticks mit der unmittelbaren digitalen Reaktion eines schaltergesteuerten Geräts. Da die Elektronik des Trickstick auf jeden Benutzer anders anspricht, muß das Gerät individuell eingestellt werden; dazu dient ein Empfindlichkeitsregler am Gehäuse. Leider liegen derzeit nur wenige Informationen über die langfristige Anwendung des Gerätes in der Praxis vor.

Bei „Le Stik", so lautet der Name eines anderen Steuerknüppels, wurde eine neue Technik angewendet. Das Gerät besteht aus einem Handgriff mit einem Feuerknopf für den Daumen und einer Pausentaste im Fingerbereich. Im Gegensatz zum üblichen Joystick, der auf einem festen Unterteil aufgesetzt ist, wird Le Stik frei in der Luft gehalten und aus der vertikalen Grundposition in die gewünschte Richtung bewegt.

Das „Gleichgewichtsorgan" des Knüppels besteht aus vier Quecksilberröhrchen, die mit Schaltern verbunden sind. Beim Kippen des Joystick fließt das Quecksilber in die Neigungsrichtung und überbrückt die Kontakte wie beim Schließen eines gewöhnlichen Schalters. Beim Wiederaufrichten fließt das Quecksilber zurück, und die Kontakte sind wieder offen. Das System arbeitet präziser als die üblichen Steuerknüppel.

In Reih und Glied

Und weiter geht es mit dem computerverwalteten Adreßbuch. Wie muß die Datei in Datensätze und Felder unterteilt werden?

Der letzte Teil endete damit, daß wir Ihnen die Aufgabe stellten, die Elemente der Programmierung mit Hilfe einer „Pseudo-Sprache" bis zu dem Punkt zu verbessern, an dem die Beispiele in BASIC umgesetzt werden können. So sah die erste Zusammenfassung der Ziele aus:

INPUT (EINGABE)
Ein Name (in jeder beliebigen Form)
OUTPUT (AUSGABE)
1. Ein Vorname
2. Ein Familienname

In der ersten Verbesserung haben Sie herausgefunden, daß sich dies in sechs Schritte unterteilen läßt. Die Namen dieser Unterroutinen wurden in Klammern geschrieben. Leider können in den meisten BASIC-Versionen Unterroutinen nicht über einen Namen aufgerufen werden. Daher ist es notwendig, daß Sie beim Schreiben des Programms statt dessen die entsprechenden Zeilennummern verwenden.

Optimierungen

Trotzdem ist es während der Entwicklungsphase erheblich einfacher, mit Namen zu arbeiten. Diese Namen können dann später in REM-Anweisungen eingetragen werden. Wir verwenden hier zur Kennzeichnung der Unterroutinen-Namen je einen Stern vor und hinter dem Namen. In einigen Programmiersprachen (zum Beispiel Pascal oder LOGO) werden die Unterroutinen, die durch einen Namen aufgerufen werden, als Prozeduren bezeichnet.

Ein wichtiger Punkt ist, daß Ihre BASIC-Version eventuell keine langen Variablennamen wie ZAEHLER oder STRASSE$ verarbeiten kann. Auch hier arbeiten wir in der Pseudo-Sprache mit langen Namen, da es die Arbeit erleichtert und das Ganze übersichtlicher bleibt. Versuchen Sie, Namen zu verwenden, die den Sinn einer Variable beschreiben. Es ist viel verständlicher, eine temporäre Variable mit dem Namen TEMSTRING$ zu versehen als z. B. XV$. Glücklicherweise sind jedoch inzwischen eine große Anzahl BASIC-Versionen imstande, lange Variablennamen zu verarbeiten.

Im letzten Teil des Kurses haben wir bereits den zweiten Punkt unserer Liste entwickelt (Umwandlung aller Buchstaben in Großbuchstaben). Dabei wurde eine zweite und dritte Verbesserung durchgeführt und anschließend ein kurzes BASIC-Programm geschrieben, das diese Aufgabe löste. Jetzt sollten Sie dies mit den anderen Punkten der Liste versuchen:

ZWEITE VERBESSERUNG
3. (Finde letzte Leerstelle)
BEGIN (STARTE)
LOOP (SCHLEIFE) solange nicht untersuchte
  Zeichen im NAME$ sind
    IF (WENN) Zeichen = " "
    THEN (DANN) speichere Position in einer
      Variablen
    ELSE (SONST) mache nichts
    ENDIF (WENN FERTIG — WEITER)
ENDLOOP (ENDE DER SCHLEIFE)
END (ENDE)

DRITTE VERBESSERUNG
3. (Finde letzte Leerstelle)
BEGIN (STARTE)
READ KOMPLETTNAME$ (LESE
  KOMPLETTNAME$)
LOOP (SCHLEIFE) (solange nicht untersuchte
  Zeichen übrig sind)
    FOR L=1 bis zur Länge von
      KOMPLETTNAME$
    READ (LESE) Zeichen aus
      KOMPLETTNAME$
    IF (WENN) ZEICHEN = " "
      THEN LET COUNT (DANN ZÄHLE) =
        Position des Zeichens
      ELSE (SONST) mache nichts
    ENDIF (WENN FERTIG — WEITER)
ENDLOOP (ENDE DER SCHLEIFE)
END (ENDE)

Jetzt sind Sie an dem Punkt angekommen, an dem das Programm von der Pseudo-Sprache in die richtige Programmiersprache umgesetzt werden kann. Achten Sie dabei auf die unterschiedliche String-Behandlung:

```
10 INPUT "BITTE VOLLSTAENDIGEN NAMEN
   EINGEBEN";KOMPLETTNAME$
20 FOR L=1 TO LEN(KOMPLETTNAME$)
30 LET ZEICHEN$=MID$
   (KOMPLETTNAME$,L,1)
40 IF ZEICHEN$=" " THEN LET ZAEHL=L
50 NEXT L
60 PRINT "LETZTE LEERSTELLE IST AN
   POSITION ";ZAEHL
70 END
```

Beachten Sie, daß Zeile 10 nur eine Eingabe zum Testen der Routine darstellt. Zeile 60 dient zur testweisen Ausgabe. Im fertigen Programm muß in Zeile 70 dann ein RETURN eingesetzt werden, da die Routine ja als Unterroutine verwendet werden soll.

ZWEITE VERBESSERUNG
4. (Lese Familiennamen)
BEGIN (STARTE)
Ordne Zeichen rechts nach der letzten
  Leerstelle FAMNAME$ zu
END (ENDE)

DRITTE VERBESSERUNG
4. (Lese Familiennamen)
BEGIN (STARTE)
READ (LESE) KOMPLETTNAME$
Lokalisiere letzte Leerstelle (rufe
  * LEERSTELLE *-Unterroutine auf)
LOOP (SCHLEIFE) solange in dem String
  noch Zeichen nach der Leerstelle sind
    READ (LESE) Zeichen und integriere sie in
      FAMNAME$
ENDLOOP (ENDE DER SCHLEIFE)
END (ENDE)

Bevor Sie mit der Programmierung in BASIC beginnen, sollten Sie einige Fehlermöglichkeiten beachten. Bei der Lokalisierung der letzten Leerstelle in der eben behandelten letzten Verbesserung rufen Sie in unserer Pseudo-Sprache die Unterroutine *LEERSTELLE* auf. Dies ist jedoch in dem BASIC-Programm so lange nicht möglich, bis wir die *LEERSTELLE*-Unterroutine geschrieben haben. Als generelle Regel gilt, daß die Programmierung der einzelnen Unterroutinen in der endgültigen Programmsprache so lange keinen Sinn hat, bis alle Routinen des Programms in der Pseudo-Sprache entwickelt worden sind. Trotzdem können Sie zum Testen einer Routine Ein- und Ausgabeanweisungen einbauen. In unserem Beispiel ist ZAEHL die Variable, die den Wert der Position der letzten Leerstelle im KOMPLETTNAME$ beinhaltet. Der Testlauf:

```
10 LET KOMPLETTNAME$="INA
     SCHROETER"
20 LET ZAEHL=4
30 FOR L= ZAEHL+1 TO LEN
   (KOMPLETTNAME$)
40 LET FAMNAME$=FAMNAME$+MID$
50 NEXT L
60 PRINT "DER FAMILIENNAME LAUTET ",
   FAMNAME$
70 END
```

Und jetzt folgt der Ablauf zum Finden des Vornamens. Die Eingabe für den Vornamen wurde so definiert, daß er alle alphabetischen Zeichen beinhalten kann.

ZWEITE VERBESSERUNG
5. (Lese Vornamen)
BEGIN (STARTE)
LOOP (SCHLEIFE) solange Zeichen im
  KOMPLETTNAME$ sind; bis zur letzten
    Leerstelle
  Suche Zeichen
  IF (WENN) Zeichen kein Buchstabe ist
    THEN (DANN) mache nichts
    ELSE (SONST) füge Zeichen in
      VORNAME$ ein
  ENDIF (WENN FERTIG — WEITER)
ENDLOOP (ENDE DER SCHLEIFE)
END (ENDE)

DRITTE VERBESSERUNG
5. (Lese Vornamen)
BEGIN (STARTE)
LOOP (SCHLEIFE) solange Zeichen bis zum
  Wert von ZAEHL vorhanden sind
  LET TEMPZEICHEN$ = das L'te Zeichen im
    String
  IF (WENN) TEMPZEICHEN$ kein Buchstabe
    ist
  THEN (DANN) mache nichts
  ELSE (SONST) LET VORNAME$ =
    VORNAME$ + TEMPZEICHEN$
  ENDIF (WENN FERTIG — WEITER)
ENDLOOP (ENDE DER SCHLEIFE)

Jetzt sind wir fast soweit, daß das Programm in BASIC programmiert werden kann. Da es sich aber immer noch um eine Zwischenstufe handelt, verwenden wir keine Zeilennummern.

PROGRAMMIERUNG
```
5. (Lese Vornamen)
REM BEGIN (STARTE)
REM LOOP (SCHLEIFE)
  FOR L=1 TO COUNT - 1
    LET TEMPZEICHEN$ = MID$
      (KOMPLETTNAME$, L, 1)
    LET ZEICHEN = ASC (TEMPZEICHEN$)
    IF ZEICHEN > 64 THEN VORNAME$ =
      VORNAME$ + CHR$ (ZEICHEN)
    REM ENDIF (WENN FERTIG — WEITER)
  NEXT L: REM ENDLOOP (ENDE DER
    SCHLEIFE)
REM END (ENDE)
```

In normalem BASIC sieht das dann so aus:

```
10 FOR L=1 TO ZAEHL-1
20 LET TEMPZEICHEN$ = MID$
   (KOMPLETTNAME$,L,1)
30 LET ZEICHEN=ASC(TEMPZEICHEN$)
40 IF ZEICHEN > 64 THEN VORNAME$ =
   VORNAME$ + CHR$(ZEICHEN)
50 NEXT L
60 END
```

So wie das Programm jetzt geschrieben ist, wird es nicht funktionieren. Insgesamt gibt es drei Probleme: ZAEHL muß ein Wert zugewiesen werden; es fehlt eine Eingaberoutine, um KOMPLETTNAME$ eine Zeichenkette zuzuweisen; und letztlich fehlt eine Ausgaberoutine, die eine Programmüberprüfung ermöglicht.

Wenn diese Routine Bestandteil einer Unterroutine wäre, würden die Parameter, die in ihr verarbeitet werden (die Eingabe), und die Parameter, die sie als Ergebnis produziert (die Ausgabe), in anderen Routinen des Programms verarbeitet. Dies ist eine sehr wichtige Feststellung: Der Fluß der Daten innerhalb eines Programms sollte sehr gut durchdacht werden, bevor man mit der Programmierung beginnt. Dies ist besonders wichtig, wenn Sie Variablen (ZAEHL als Beispiel) verwenden und derselbe Variablenname in verschiedenen Teilen des Programms verwendet wird. Es hat also keinen Sinn, eine Unterroutine aufzurufen, die die Variable ZAEHL verwendet, wenn dieser Variablen weder zuvor noch in der Unterroutine ein Wert zugewiesen wurde. Wenn der Variablen ZAEHL innerhalb einer Unterroutine ein Wert zugewiesen wurde, so bleibt dieser Wert so lange bestehen, bis ein neuer Wert zugewiesen wird – eventuell sogar in einer anderen Unterroutine. Dies ist ein Grund, weshalb es als eine schlechte Programmiertechnik angesehen wird, aus einer Schleife zu springen, ohne zu wissen, welchen Wert die Schleifenvariable hat.

Um Chaos bei der Verwendung von Variablen zu vermeiden, ist es nützlich, wenn man sich eine Liste aller verwendeten Variablen anfertigt. Am besten beginnen Sie damit bereits während der Entwicklung in der Pseudo-Sprache, und notieren Sie sich auch, wozu Sie jede Variable verwenden. Einige Programmiersprachen gestatten es, Variablen als „lokal" oder „global" zu deklarieren. Das bedeutet, daß ihre Werte entweder nur in einem Teil des Programms zur Verfügung stehen (lokal) oder aber im gesamten Programm (global). Viele Variablen, wie zum Beispiel die in Schleifen (z. B. das L in LET L=1 TO 10), sind häufig ohnehin nur lokal. Daher ist es ratsam, den Wert einer Variablen zu initialisieren, bevor sie verwendet wird (z. B. LET L=0).

Bis jetzt haben wir also eine unseren Bedürfnissen entsprechende Definition für einen Namen entwickelt sowie einige Routinen zur Verarbeitung von Namen. Nun zu der Struktur von „Verzeichnissen" in der Adreßbuch-„Datei".

Die Ausdrücke „Verzeichnis" (record), „Datei" (file) und „Feld" (field) haben sehr spezielle Bedeutungen in der Welt der Computer. Eine Datei ist eine ganze Anzahl an speziellen Daten. In einem Computer-System wäre dies eine bestimmbare Anzahl an Daten, die auf einer Diskette oder Cassette unter einem eigenen Namen, normalerweise als Dateiname bezeichnet, gespeichert sind. Das gesamte Adreßbuch könnte als Datei bezeichnet werden, dem wir hier den Namen ADBUCH geben.

Eine Datei ist in Verzeichnisse unterteilt. Diese enthalten ebenfalls eine bestimmbare Anzahl an Daten. Angenommen, daß unser Adreßbuch aus einem Karteikasten mit Karteikarten bestünde, dann wäre die Datei der ganze Kasten mit allen Karten, und die Verzeichnisse wären die einzelnen Karten – jede mit ihrem eigenen Namen, der Adresse und der Telefonnummer.

Eingabe der Daten

Die Felder des Verzeichnisses können Sie sich als eine oder mehrere Reihen mit spezifischen Daten vorstellen. Jedes der Verzeichnisse in unserer ADBUCH-Datei wird die folgenden Felder umfassen: NAME, ADRESSE und TELEFONNUMMER. Ein typisches Verzeichnis sieht so aus:

Peter Emmerich
Tubastraße 23
6200 Wiesbaden
Deutschland
06121-1234567

Dieses Verzeichnis enthält drei Felder: das Namens-Feld, das alphabetische Buchstaben beinhaltet (möglicherweise auch Sonderzeichen), das Adressen-Feld, das einige Zahlen und viele Buchstaben beinhaltet, und das Telefonnummern-Feld, das nur Zahlen beinhaltet. Bevor wir mit dem Schreiben eines Programms zur Verarbeitung solch komplexer Informationen beginnen können, müssen wir entscheiden, wie die Daten im Computer dargestellt werden sollen. Eine Möglichkeit ist, daß alle Daten in einem Verzeichnis eine lange Zeichenkette bilden. Nehmen wir einmal an, die folgenden Eingaben wären eine einzige lange Kette:

Oswin Heider
Schlugaweg 13
8000 München
Deutschland
(089) 987 6543

Was wäre, wenn wir auch die Landesvorwahl eingegeben hätten, wie z. B.: 001 (089) 987 6543? Die Nummer würde dann insgesamt 18 Zeichen umfassen. Um solche Probleme zu vermeiden, wird die Telefonnummer als separates Feld eingerichtet. So braucht uns das Programm dann nur alle in diesem Feld abgelegten Zeichen auszugeben.

Die Schwierigkeit bei dieser Methode ist, daß die verschiedenen Felder unterschieden werden müssen. Sonst kann es passieren, daß bei Zugriff auf ein bestimmtes Feld auch die Inhalte der anderen Felder ausgegeben werden. Eine Möglichkeit zur Lösung dieses Problems wäre, ein weiteres Feld in Verbindung mit jedem Verzeichnis einzurichten, das als Index dient. Wenn ein Verzeichnis beispiels-

weise das 15te in einer Datei ist, würde das Index-Feld die Zahl 15 beinhalten. Dieses Feld könnte dann als Zeiger zu den verschiedenen Feldern verwendet werden. Um dies zu verdeutlichen, betrachten Sie das folgende Verzeichnis:

Joachim Schmidt	NAME-Feld
Gasallee 54	STRASSE-Feld
6000 Frankfurt	STADT-Feld
Deutschland	STAAT-Feld
0401 234567	TELEFON-Feld
015	INDEX-Feld

Wenn wir den Namen dieser Person kennen und ihre Telefonnummer aufrufen wollen, müßten wir alle Felder mit den Namen durchsuchen, bis ein Vergleich positiv ausfällt. Wir erhielten dann die Angabe, in welchem Verzeichnis der Name enthalten ist – in diesem Beispiel Verzeichnis Nummer 15. Jetzt müssen Sie nur noch das 15te Feld der TELEFON-Felder finden, um die gesuchte Nummer zu erhalten.

Vielleicht möchten Sie alle Adressen Ihrer Bekannten im Raum Wiesbaden ausgeben lassen. Das Programm müßte dann alle STADT-Felder durchsuchen und dabei jeweils diejenigen Verzeichnisse ausgeben, bei denen der Vergleich in bezug auf Wiesbaden zutrifft. Wenn man diese Methode verwendet, braucht man das INDEX-Feld nicht zu untersuchen. Außerdem hat diese Technik den Vorteil, daß es eine relativ einfache Aufgabe ist.

Im nächsten Teil unseres Kurses werden wir uns mit den Problemen befassen, die das Durchsuchen von Dateien auf spezifische Informationen mit sich bringt.

Übung

■ Nehmen Sie an, daß Verzeichnisse mit den folgenden Feldern fur unser Adreßbuch ausreichen:

NAME-Feld
STRASSE-Feld
STADT-Feld
STAAT-Feld
TELEFON-Feld

Stellen Sie sich vor, daß eine der Optionen, die in einem Menu unseres Adreßbuches zur Verfügung stehen werden, die folgende wäre:

5. ERSTELLEN EINES NEUEN EINTRAGES

Sie geben 5 ein, und das Programm verzweigt zu dem Teil, mit dem Sie neue Eintragungen vornehmen können. Da das Programm vollständig Menu-gesteuert ist, werden Sie immer auf erforderliche Eingaben hingewiesen – mit Meldungen wie z. B. GEBEN SIE DEN NAMEN EIN, GEBEN SIE DIE STRASSE EIN und so weiter. Hier ist eine Liste mit den erwarteten Ergebnissen:

1. Ein Element im Feld für den Namen
2. Ein Element im Feld für die Straße
3. Ein Element im Feld für die Stadt
4. Ein Element im Feld für den Staat
5. Ein Element im Feld für die Telefonnummer

Ihre Aufgabe ist, dies durch einen Prozeß einer „Top-Down"-Programmierung unter Verwendung einer Pseudo-Sprache bis zu einem Punkt hin zu entwickeln, ab dem eine Umwandlung in BASIC möglich wird. Die Pseudo-Sprache können Sie nach Ihren eigenen Vostellungen entwickeln. Wir empfehlen Ihnen jedoch Großbuchstaben für Schlüsselwörter wie IF (WENN), LOOP (SCHLEIFE) und so weiter, sowie Kleinbuchstaben für Beschreibungen von Funktionsabläufen. Beachten Sie dabei die zuvor gezeigten Beispiele.

BASIC-DIALEKTE

SPECTRUM

Schritt 3

```
10 INPUT "GEBEN SIE DEN
   KOMPLETTEN NAMEN EIN";
   F$
15 LET ZAEHL=0
20 FOR L=1 TO LEN F$
30 LET C$=F$(L)
40 IF C$=" " THEN LET
   ZAEHL=L
50 NEXT L
60 PRINT "LETZTE LEERSTELLE
   BEFINDET SICH AN
   POSITION ";ZAEHL
70 STOP
9990 DEF FN M$(X$,P,N)=
     X$(P TO P+N-1)
9991 DEF FN L$=X$( TO N)
9992 DEF FN R$=X$ (LEN
     X$-N+1 TO )
```

In diesem Programmierprojekt werden sehr oft die String-Funktionen MID$, LEFT$ und RIGHT$ verwendet. Ihre Äquivalente in Sinclair-BASIC sind:

```
LEFT$ (F$, N)
  ersetzen durch F$( TO N)
RIGHT$(F$,N)
  ersetzen durch
    F$ (LEN(F$)-N+1 TO )
MID$(F$,P,N)
  ersetzen durch
    F$(P TO P+N-1)
MID$(F$,P,1)
  ersetzen durch F$(P)
```

Beachten Sie, daß Stringvariablen-Namen auf einigen Geräten nie länger als ein Buchstabe sein dürfen (plus dem $-Zeichen).

Schritt 4

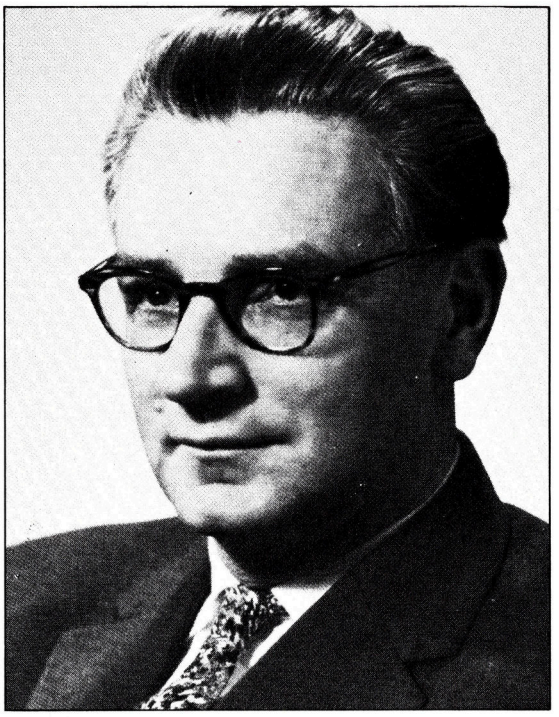
```
5 LET S$=" "
10 LET F$="KLAUS GARMS"
20 LET ZAEHL=4
30 FOR L=ZAEHL+1 TO LEN F$
40 LET S$=S$+F$(L)
50 NEXT L
60 PRINT "FAMILIENNAME IST
   ";S$
70 STOP
```

Schritt 5

```
5 LET C$=" "
10 FOR L=1 TO ZAEHL-1
20 LET T$=F$(L)
30 LET CHAR=CODE T$
40 IF CHAR > 64 THEN LET
   C$=C$+CHR$ CHAR
50 NEXT L
60 STOP
```

VARIABLES

In diesem Programmfragment wurde das Problem, Stringvariablen-Namen nur mit einem Buchstaben darstellen zu dürfen, noch größer. F$ ist beim Spectrum das Äquivalent zur Variablen KOMPLETTNAME$ (FULLNAME$), weshalb C$ für die Variable VORNAME$ (FORNAME$) verwendet werden muß.
Einige Computer bieten die Möglichkeit zur Verwendung langer Variablennamen wie KOMPLETTNAME$. Der Spectrum gestattet die Verwendung langer numerischer Variablennamen, jedoch nur einzelner Buchstaben zur Kennzeichnung von String-Variablen. Andere Geräte lassen lange Variablennamen zu, von denen jedoch nur die ersten zwei Buchstaben relevant sind. Man kann also den Namen KOMPLETTNAME$ verwenden, doch wird z. B. durch die Variable KOMMATA$ dieselbe Speicherstelle angesprochen, da beide Namen mit denselben zwei Buchstaben beginnen.
Auf dem Oric zum Beispiel können Variablennamen nicht länger als zwei Zeichen sein (zuerst ein Buchstabe und dann entweder eine Zahl oder ein zweiter Buchstabe).



Konrad Zuse

Während John von Neumann seine Pionierarbeiten in den USA vollbrachte, arbeitete Konrad Zuse in Deutschland an denselben Problemen.

Als in den USA gerade der erste relaisgesteuerte Computer (ENIAC) entstand, arbeitete der deutsche Ingenieur Konrad Zuse bereits an einem programmierbaren Rechner – höchstwahrscheinlich dem ersten Computer der Welt.

Zuse, 1910 in Berlin geboren, arbeitete nach dem Studium an der Technischen Universität der Stadt als Luftfahrttechniker bei den Henschel-Flugzeugwerken, wo er neue Tragflächenkonstruktionen entwarf. Die grundlegenden mathematischen Methoden für die Berechnung der Form und Stabilität von Tragflächen waren schon seit den 20er Jahren bekannt. Der Rechenaufwand für derartige Neuentwicklungen war jedoch enorm. Mit der Planung war ein ganzes Team beschäftigt, das sich mit mechanischen Rechenmaschinen und Rechenschiebern abplagen mußte. Zuse suchte Alternativen zu dieser zeitraubenden Arbeit. Zusammen mit Freunden begann er in der elterlichen Wohnung mit dem Bau eines ersten „Computers".

Der Einstieg war ein rein mechanisches Gerät, der „Z1". Damit ließen sich neben den vier Grundrechenarten schon Quadratwurzeln ziehen und dezimale in Binärziffern verwandeln. Die wichtigste Erkenntnis bestand jedoch in der simplen Feststellung, daß ein Hebel jeweils zwei (Schalt-)Positionen einnehmen kann, Ein bzw. Aus. Dadurch eignet er sich sowohl zum „Speichern" von Daten als auch zum Steuern des Rechenvorganges selbst.

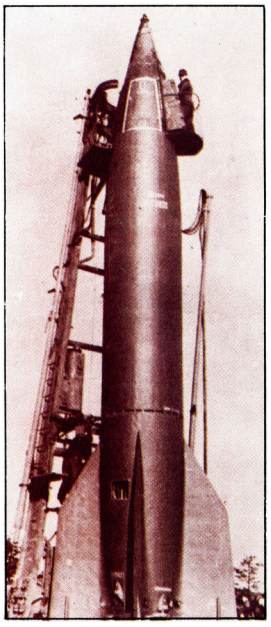
Steiniger Weg

Zuse beschritt daraufhin konsequent den Weg, sowohl Daten als auch Rechenanweisungen in rein binärer Form darzustellen, wodurch er 1941 zur Konstruktion der „Z2" fand, seines ersten elektromechanischen Rechners. Nachdem die militärischen Einsatzmöglichkeiten des Gerätes erkannt wurden, stellte die vorher eher uninteressierte Regierung dann auch entsprechende Mittel zur Weiterentwicklung bereit, die dem Bau des „Z3" zugute kamen. Hier waren mechanische Verbindungen erstmals vollständig durch elektrische Verdrahtung ersetzt, wodurch die Maschine viel kleiner und technisch eleganter wurde.

Der Krieg legte Zuse bei seiner Arbeit einen Stein nach dem anderen in den Weg: Zuerst wurde er zweimal an die Ostfront einberufen, dann mußte er wegen alliierter Bombenangriffe mehrfach die Werkstatt wechseln. Zudem gab es kaum geeignetes Material. Ausgeschlachtete Telefone dienten ebenso als Hilfsmittel wie gebrauchte Filmstreifen, die Codestreifen aus Papier ersetzen mußten. Trotzdem speicherte der „Z3" schon 64 Worte à 22 Bit. Daten konnten über eine Tastatur eingegeben und Ergebnisse auf einem Anzeigebrett mit Lämpchen abgelesen werden. Der „Z3" wurde leider zusammen mit seinen Vorgängern bei den Bombenangriffen auf Berlin im Jahre 1945 zerstört. Vorher wurde das Gerät aber auch praktisch eingesetzt: bei der Konstruktion eines unbemannten, fernlenkbaren Bombenflugzeuges.

Der letzte Computer aus Kriegszeiten war der „Z4", der Worte bis zu 32 Bit verarbeiten konnte. Er wurde beim Vorrücken der Alliierten nach Göttingen ausgelagert und später bis zum Jahre 1954 in Basel eingesetzt – lange einer der wichtigsten Computer, die es in Europa zu dieser Zeit gab

Mit dem Zusammenbruch endete für Zuse vorläufig die Möglichkeit, in Deutschland einen Computer zu bauen. Er wandte sich daher der Rechner-Theorie zu und entwickelte die Programmiersprache „Plankalkül", mit der sich sowohl mathematische als auch allgemeinere Probleme bearbeiten ließen. Eine später von Zuse gegründete Firma blieb bis zur Übernahme durch Siemens Deutschlands bedeutendste Computer-Herstellerin. Heute ist Professor Zuse nur noch wissenschaftlich tätig.



Zuse entwickelte seine Rechner, um die großen Teams zu entlasten, die bei der Flugkörper-Konstruktion langwierige Kalkulationen mit dem Rechenschieber anstellen mußten. Auch bei der Entwicklung der V 1- und der V 2-Raketen wurden seine Geräte eingesetzt.

Sharp MZ-711

Ein Heimcomputer der unteren Preisklasse, dessen Peripheriegeräte im Gehäuse integriert sind.

Der Sharp MZ-711 ist ein interessantes Gerät mit ungewöhnlichen Eigenschaften. Die Maschine entspricht dem hohen Standard japanischer Konstruktion und funktioniert gut. Text läßt sich ausgezeichnet lesen, die Darstellung der Farben ist gut, wenn auch das Bild bei einigen Farbkombinationen etwas verwischt erscheint.

Der Rechner weist jedoch einige Eigenheiten auf. So erzeugt der normale Schreibmodus Großbuchstaben, während Kleinbuchstaben nur durch das Drücken der SHIFT-Taste zu erzeugen sind. Obwohl das ohne Zweifel gut für BASIC-Programmierungen ist, werden andere Funktionen dadurch erschwert. Drückt man die CTRL-Taste und das E gleichzeitig, kehrt sich der normale Schreibmodus in Kleinbuchstaben um, während CTRL und F ihn wieder auf Großbuchstaben umschalten. Das Cassettenlaufwerk wie auch die zusätzlichen Geräte des Sharp MZ-711 sind als Extras erhältlich. Das Cassettensystem ist langsam, aber zuverlässig, obwohl der dafür zuständige Teil des Betriebssystems einige Schwachstellen aufweist. Der Motor des Cassettenrecorders kann nicht über den Computer gesteuert werden. Durch diese Beschränkungen läßt sich, will man den Bandzähler nicht benutzen, nur eine Datei pro Cassette speichern, was besonders erstaunt, da Sharp speziell auf dem HiFi-Sektor über einige Fachkenntnisse verfügt und Solenoid-gesteuerte Cassettendecks in diesem Bereich nichts Besonderes mehr sind.

Mehrere Sprachen

Der eingebaute Drucker/Plotter (wiederum ein zusätzliches Extra, das in das Gehäuse eingesetzt wird) hat einen Mechanismus zur Steuerung des Druckkopfes und der Zeichenstifte, der außerordentlich empfindlich ist. Wird der Drucker über längere Zeit nicht benötigt, müssen die Stifte ausgebaut und mit Kappen geschützt werden. Da diese sehr klein sind, könnten sie außerhalb der Maschine leicht verloren gehen; ferner trocknen die

Die Tastatur des Sharp MZ-711 gibt der Maschine ein professionelles Aussehen. Die Tasten haben ausreichenden Abstand voneinander, lassen sich voll durchdrücken und sind für Textverarbeitung geeignet. Mit fünf doppelt belegten programmierbaren Funktionstasten kann die Bedienung vereinfacht werden. Die grafischen Symbole sind in logischer Reihenfolge angeordnet. Die Tastatur ist nach dem QWERTY-Format ausgelegt – auf der rechten Seite befinden sich die vier Cursorsteuerungstasten.

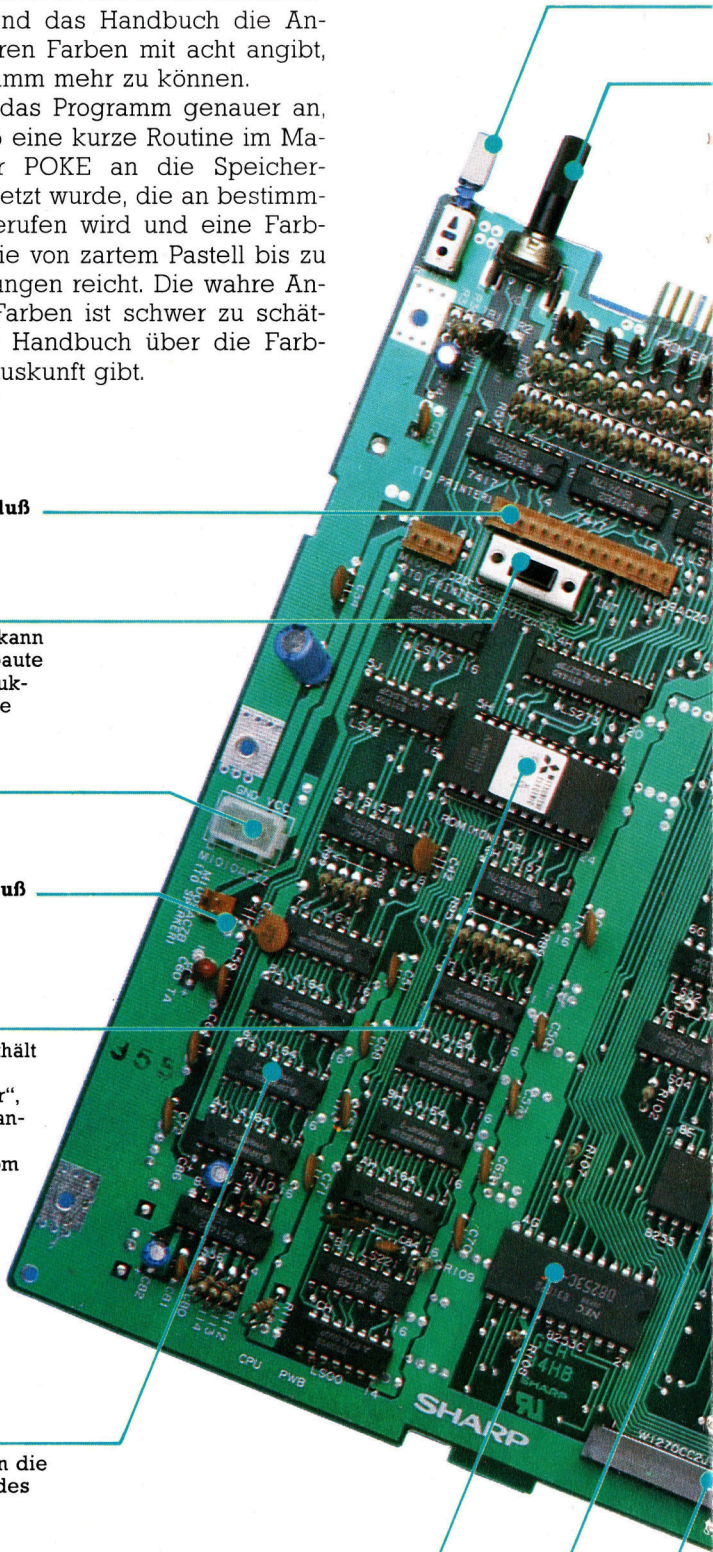
Stifte leicht aus, so daß die Schrift nach längerer Benutzung kaum lesbar ist. Der BASIC-Interpreter ist nicht im ROM integriert, sondern wird vom Band geladen. Damit läßt sich die Maschine für mehrere Sprachen nutzen.

Die technische Dokumention ist auf einem hohen Standard. Darin eingeschlossen sind Schaltpläne, eine Darstellung der Speicherorganisation (Memory MAP), eine Auflistung der eingebauten Software in der Assemblersprache und andere technische Einzelheiten. Aus unerklärlichen Gründen sind die Hersteller bei der Darstellung einiger ungewöhnlicher Eigenschaften des Sharp MZ-711 sehr bescheiden. Ein Beispiel dafür ist das Demonstrationsprogramm. Während das Handbuch die Anzahl der verfügbaren Farben mit acht angibt, scheint das Programm mehr zu können.

Sieht man sich das Programm genauer an, entdeckt man, daß eine kurze Routine im Maschinencode über POKE an die Speicheradresse 40960 gesetzt wurde, die an bestimmten Punkten aufgerufen wird und eine Farbpalette darstellt, die von zartem Pastell bis zu dunklen Schattierungen reicht. Die wahre Anzahl verfügbarer Farben ist schwer zu schätzen, da auch das Handbuch über die Farbsteuerung kaum Auskunft gibt.

Cassettendeck
Das Cassettendeck des Sharp MZ-711 wird in das Hauptgehäuse eingebaut. Der Motor des Cassettenrecorders läßt sich jedoch nicht vom Bildschirm aus steuern.

Drucker/Plotter
Wie das Cassettenlaufwerk ist auch der Drucker/Plotter ein zusätzliches Extra, das in das Gehäuse integriert werden kann. Mit vier Miniatur-Kugelschreiberminen werden Text und Grafik auf das 11,5 cm breite Papier gebracht.



Druckeranschluß

Schalter für die Druckerauswahl
Mit diesem Schalter kann entweder der eingebaute oder ein externer Drukker zur Druckausgabe gewählt werden.

Netzanschluß

Lautsprecheranschluß

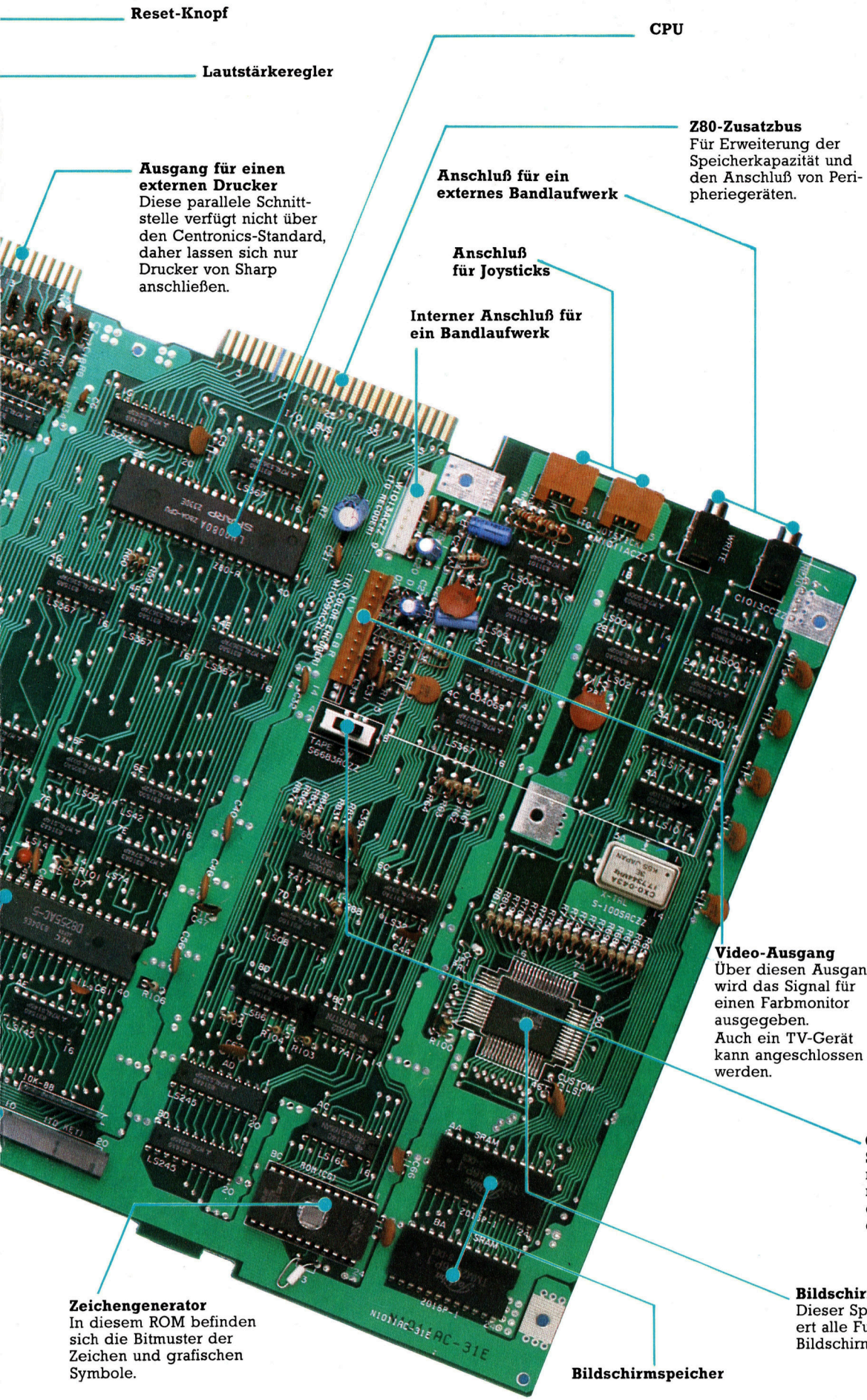
ROM
Das System ROM enthält nur ein kleines Programm, den „Monitor", das Speicherstellen ansprechen oder den BASIC-Interpreter vom Band laden kann.

RAM
Acht Chips enthalten die Standard-64K-RAM des Arbeitsspeichers.

Chip mit Taktgeber und Zeitmodul

Schnittstellenadapter
Dieser PIA(Peripheral Interface Adaptor)-Chip steuert die Schnittstelle der Tastatur und des Cassettendecks.

Tastaturanschluß



Reset-Knopf

Lautstärkeregler

Ausgang für einen externen Drucker
Diese parallele Schnittstelle verfügt nicht über den Centronics-Standard, daher lassen sich nur Drucker von Sharp anschließen.

CPU

Z80-Zusatzbus
Für Erweiterung der Speicherkapazität und den Anschluß von Peripheriegeräten.

Anschluß für ein externes Bandlaufwerk

Anschluß für Joysticks

Interner Anschluß für ein Bandlaufwerk

Video-Ausgang
Über diesen Ausgang wird das Signal für einen Farbmonitor ausgegeben.
Auch ein TV-Gerät kann angeschlossen werden.

Cassettenschalter
Sollte das Cassettendeck nicht funktionieren, können mit diesem Schalter die Signalphasen geändert werden.

Bildschirmsteuerung
Dieser Spezialchip steuert alle Funktionen der Bildschirmausgabe.

Zeichengenerator
In diesem ROM befinden sich die Bitmuster der Zeichen und grafischen Symbole.

Bildschirmspeicher

SHARP MZ-711

PREIS
Computer und Cassettendeck rd. 800 Mark, Drucker/Plotter rd. 520 Mark

ABMESSUNGEN
440 x 305 x 86 mm

ZENTRALEINHEIT
Z80A

TAKTFREQUENZ
4 MHz

SPEICHERKAPAZITÄT
ROM 4 K, RAM 64 K

BILDSCHIRM-DARSTELLUNG
25 Zeilen mit je 40 Zeichen, oder Grafik mit niedriger Auflösung 50 x 80 Pixel, 8 angegebene Farben mit einer großen Anzahl von Schattierungen, Vorder- und Hintergrund unabhängig einstellbar; 127 fest definierte Zeichen, 127 Zeichen vom Anwender programmierbar

SCHNITTSTELLEN
RS232 seriell, Cassettenlaufwerk, Systembus, paralleler Drucker

PROGRAMMIERSPRACHE
BASIC auf Cassette

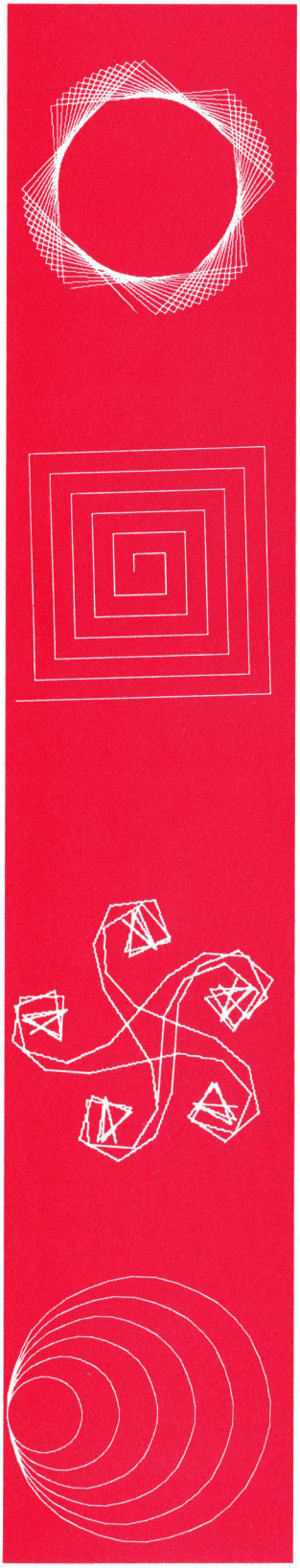
WEITERE PROGRAMMIER-SPRACHEN
PASCAL, ASSEMBLER, FORTRAN, FORTH, DAFMOD, TOOLKIT

ZUBEHÖR
Handbücher für Installation und BASIC, Fernseherkabel, Cassettenkabel, BASIC und Demonstrationscassetten

TASTATUR
Vollwertige Tastatur mit fünf programmierbaren Funktionstasten, die über SHIFT zehn Funktionen ausführen können

DOKUMENTATION
Sehr ausführlich

Ausführung wiederholen!



In LOGO wird die mathematische Technik der Recursion häufig angewendet. Verbunden mit Eingabevariablen erzeugt die Recursion interessante Bildschirmresultate.

Eines der ersten Programme dieses LOGO-Kurses war das Quadrat-Beispiel. Die Turtle wurde angewiesen, einige Schritte geradeaus zu laufen, sich um 90 Grad nach rechts zu drehen und diesen Bewegungsablauf dreimal zu wiederholen. Hier ist eine andere Lösung:

```
TO QUADRAT
  FD 50
  RT 90
  QUADRAT
END
```

Wie Sie sehen, zeichnet die Turtle ein Quadrat und läuft anschließend so lange auf den bereits gezogenen Linien entlang, bis Control-G oder BREAK gedrückt wird. Das bedeutet, daß sich die Prozedur immer wieder selbst aufruft – also eine Recursion ausführt.

Um zu verstehen, was die Turtle veranlaßt, unermüdlich umherzulaufen, sollten Sie sich genauer mit der Programmstruktur beschäftigen. Zuerst liest LOGO die beiden Befehle FORWARD 50 und RIGHT 90. Die darauf folgende Anweisung heißt QUADRAT, also kehrt LOGO zum Prozedurnamen QUADRAT zurück und beginnt erneut mit der Ausführung der Anweisungen. Dieses Verfahren wiederholt sich, bis der Programmablauf unterbrochen wird.

Die Recursion läßt sich ebenfalls in Prozeduren einsetzen, deren Variablenwerte durch Eingaben des Anwenders definiert werden:

```
TO POLY :SEITE :WINKEL
  FD :SEITE
  RT :WINKEL
  POLY :SEITE :WINKEL
END
```

Mit dieser Prozedur können auch all die Vielecke (Polygone) modifiziert werden, die Sie an früherer Stelle des Kurses bereits kennengelernt haben. Das folgende Beispiel berechnet SEITE bei jeder Ausführung erneut:

```
TO POLYSPI :SEITE :WINKEL
  FD :SEITE
  RT :WINKEL
  POLYSPI ( :SEITE + 5) :WINKEL
END
```

Der einzige Unterschied zwischen dieser Prozedur und POLY besteht darin, daß zu der Variablen SEITE bei jedem erneuten recursiven Aufruf die Zahl fünf addiert wird. Wenn Sie zum Beispiel die Zeile POLYSPI 10 90 eingeben, wird die erste Linie mit einer Länge von 10 gezeichnet, die zweite mit dem Wert 15, danach 20 und so weiter. Das Ergebnis dieser Prozedur ist eine Spirale. Experimentieren Sie nun mit verschiedenen Eingaben wie: 10 90, 10 95, 10 120, 10 117, 10 144 oder 10 142. Sie haben auch die Möglichkeit, das Programm zu modifizieren, zum Beispiel indem Sie die Addition in eine Subtraktion oder Multiplikation umwandeln.

```
TO INSPI :SEITE :WINKEL :ERH
  FD :SEITE
  RT :WINKEL
  INSPI : SEITE ( :WINKEL + :ERH) :ERH
END
```

Neben der recursiven Ausführung eines bestimmten Programmteils wird in LOGO auch der bereits vorgestellte Befehl REPEAT verwendet. In anderen Programmiersprachen werden statt dessen Anweisungen wie FOR . . . NEXT, REPEAT . . . UNTIL oder WHILE . . . END eingesetzt.

Alle bislang vorgestellten recursiven Prozeduren laufen nach dem Aufruf endlos weiter, da kein Befehl eingegeben wurde, der die Programmausführung anhalten kann. Soll die Prozedur an einem bestimmten Punkt abbrechen, muß also eine zusätzliche Anweisung in das Programm integriert werden. Um beispielsweise die Turtle in der QUADRAT-Prozedur anzuhalten, sobald die erste Figur vollendet ist und sich die Turtle wieder in der Ausgangsposition (HEADING 0) befindet, geben Sie folgende Befehle ein:

```
TO QUADRAT :SEITE
  FD :SEITE
  RT 90
  IF HEADING = 0 THEN STOP
  QUADRAT :SEITE
END
```

Diese Prozedur enthält zwei neue Befehle: STOP und IF – THEN. Der erste bewirkt, daß die Turtle stehenbleibt, sobald diese in Richtung Null zeigt. IF – THEN dagegen überprüft eine Bedingung. WENN die der IF-Anweisung folgende Bedingung erfüllt oder „wahr" ist, DANN werden die Befehle nach THEN ausgeführt.

Zur Verdeutlichung nun noch eine geänderte Version der POLYSPI-Prozedur mit Bedingungsabfrage:

```
TO POLYSPI :LAENGE
  IF LAENGE > 15 THEN STOP
  FD :LAENGE
  RT 90
  POLYSPI ( :LAENGE + 5)
END
```

Haben Sie überlegt, was passieren wird, wenn man POLYSPI 10 eingibt? Also, zuerst ruft LOGO die Prozedur POLYSPI auf und weist der lokalen Variable den Wert 10 zu. Da diese Zahl nicht größer ist als 15, werden die Befehle FD 10 und RT 90 ausgeführt. Anschließend ruft LOGO erneut POLYSPI auf, diesmal enthält die Variable allerdings den Wert 15. An dieser Stelle wird intern eine 'Kopie' der Prozedur angefertigt, die den zuletzt eingegebenen Wert enthält. Da LAENGE noch immer nicht den Wert 15 übersteigt, lautet die Anweisung: FD 15 RT 90 – verbunden mit einem weiteren Aufruf der Prozedur. Jetzt enthält die lokale Variable die Zahl 20, so daß das Programm unterbrochen wird und LOGO zur letzten ausgeführten Prozedur (POLYSPI 15) zurückkehrt. Da diese bereits beendet ist, wird die Kontrolle wieder an das Ausgangsprogramm übergeben.

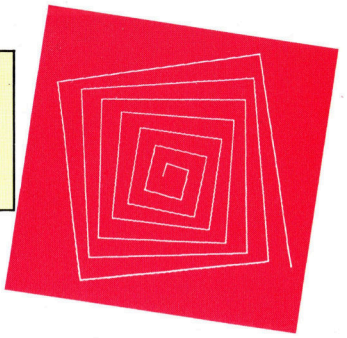
```
TO POLYSPI :LAENGE
  IF :LAENGE > 15 THEN STOP
  POLYSPI ( :LAENGE + 5)
  FD :LAENGE
  RT 90
END
```

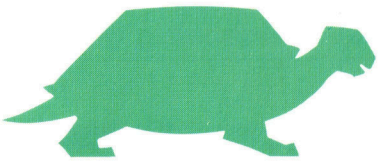
Dieses Programm zeichnet die Spirale 'rückwärts'. Im Gegensatz zu den vorherigen Prozeduren verlaufen die Linien von außen nach innen. Die Arbeitsweise ist deutlicher zu erkennen, wenn Sie die Bedingungsüberprüfung wie folgt ändern:

```
IF :LAENGE > 50 THEN STOP
```

Anders als in der zuvor gezeigten Prozedur, bei der eine Linie gezeichnet und danach erst der Wert für die nächste berechnet wurde, verläuft die Ausführung hier in entgegengesetzter Richtung. Die Berechnung und Definition der lokalen Variablen sowie das Anfertigen von Prozedur-Kopien erfolgt vor Erstellung der Spirale. Dieses Verfahren belegt jedoch viel Speicherplatz.

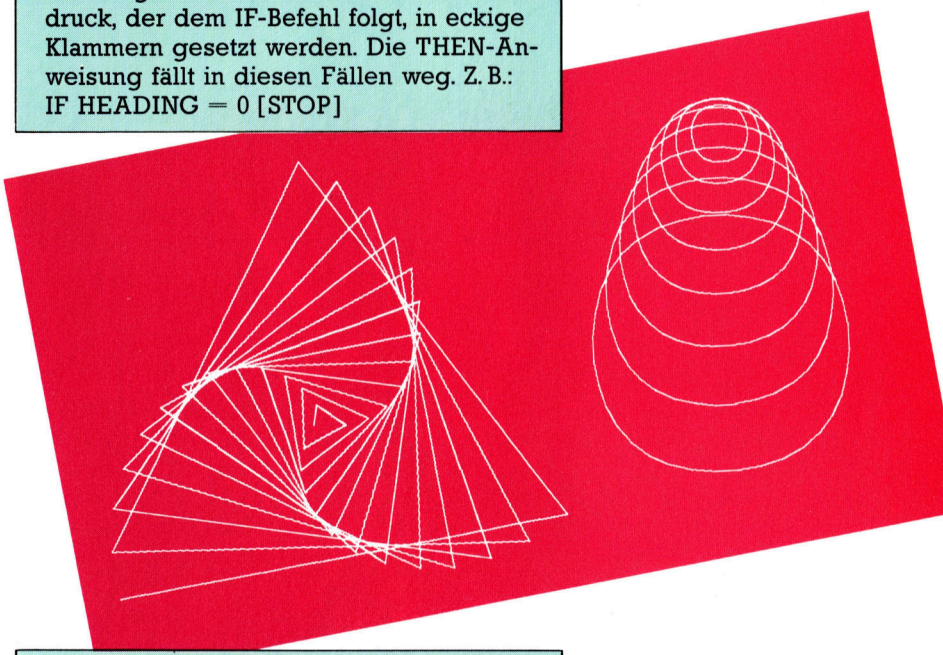
Übung
Schreiben Sie eine Prozedur, die einen Turm aus übereinanderliegenden Quadraten baut. Die Seitenlänge soll bei jedem Quadrat reduziert werden.





LOGO-Dialekte

Bei einigen LOGO-Versionen muß der Ausdruck, der dem IF-Befehl folgt, in eckige Klammern gesetzt werden. Die THEN-Anweisung fällt in diesen Fällen weg. Z. B.:
IF HEADING = 0 [STOP]



Lösungen

Hier ein Beispiel, das einen Kreis zeichnet, wobei der Radius die Eingabevariable darstellt:

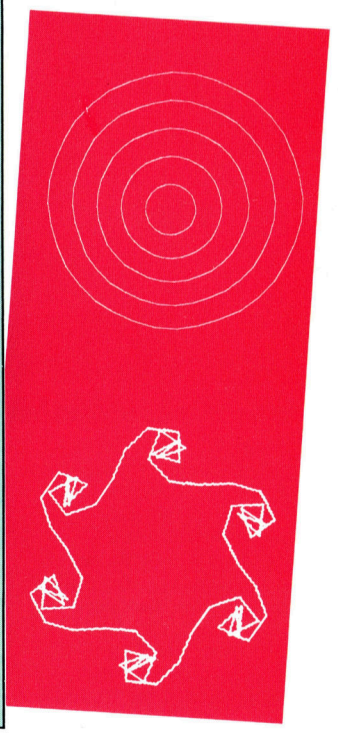
```
TO CIRCLE :RADIUS
  REPEAT 36 [FD(2* :PI* :RADIUS/36)
  RT 10]
END
MAKE "PI(3.14159)
```

Die verbesserte Programmversion sieht so aus:

```
TO C.CIRCLE :RADIUS
  PU LT 90 FD :RADIUS RT 90 PD
  CIRCLE :RADIUS
  PU LT 90 BK :RADIUS RT 90 PD
END
```

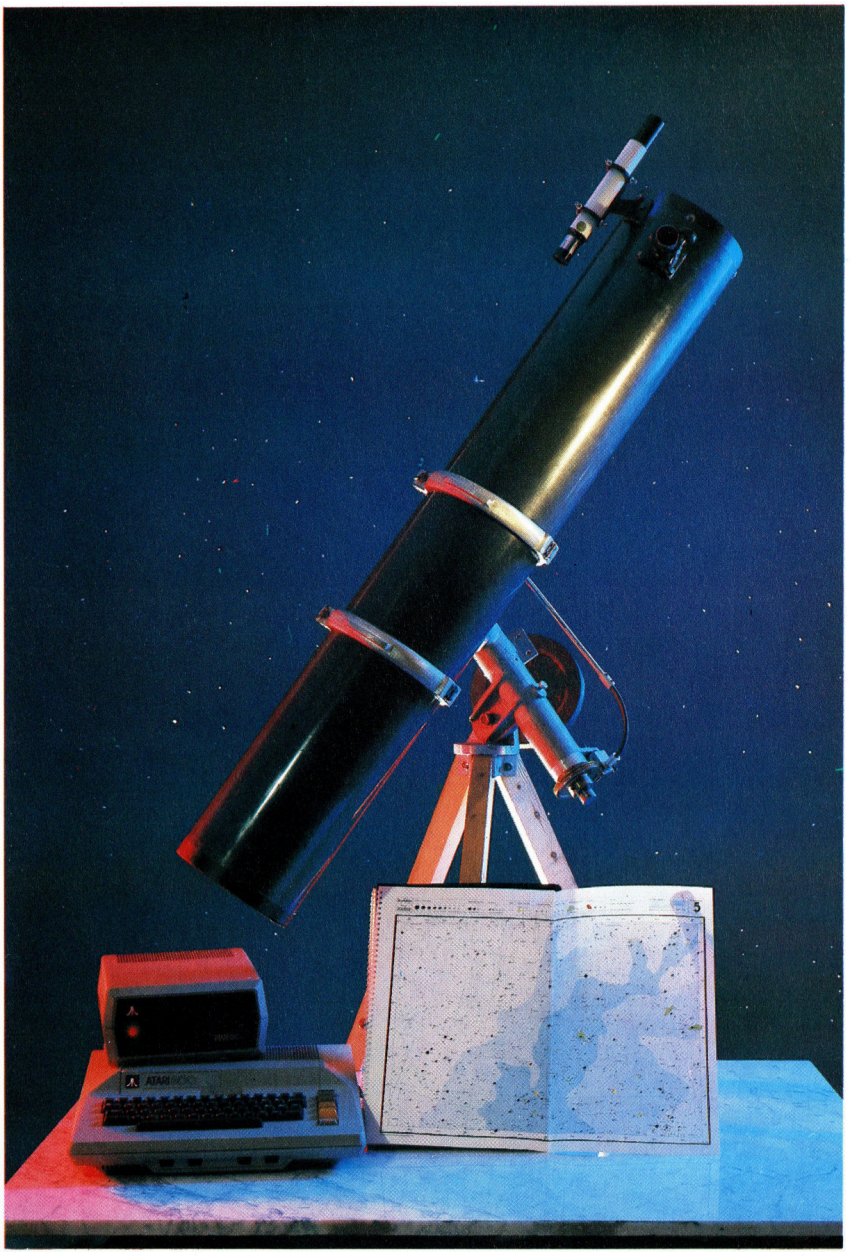
Die Prozedur TARGET zeichnet fünf konzentrische Kreise:

```
TO TARGET
  C.IRCLE 10 C.CIRCLE 20 C.CIRCLE 30
  C.CIRCLE 40 C.CIRCLE 50
END
```



Sternschnuppen

In der Astronomie gibt es zwei Hauptaufgaben für den Computer: Er dient zum Speichern der Daten beobachteter Objekte und richtet das Teleskop gleichzeitig genau aus.



Optische Astronomie wird viel einfacher, wenn eine genaue Vorhersage über den Standort eines Sterns oder Planeten für einen bestimmten Tag vorhanden ist. Der Heimcomputer als Datenbank ermöglicht die Berechnung der Koordinaten in Sekunden.

Wenn Sie den nächtlichen Himmel beobachten und dabei den uns nächsten Stern sehen, erkennen Sie diesen tatsächlich nur so, wie er vor vier Jahren war: So lange ist das Licht von Proxima Centauri zur Erde unterwegs. In eben diesen vier Jahren hat sich der Microcomputer vom seltenen und teuren Hobby zum beinahe selbstverständlichen „Hausgenossen" entwickelt. Gerade in der Astronomie lassen sich die Fähigkeiten des Heimcomputers beim Verarbeiten von Daten, Rechnen und Steuern voll nutzen.

Jeder Hobby-Astronom kennt die drei Probleme, die bei der Beobachtung des Himmels auftreten: die erste Beobachtung eines Himmelskörpers, den Umgang mit den dabei ermittelten Daten sowie deren sinnvolle Analyse. In jedem dieser drei Bereiche hilft der Rechner. Die Aufgaben des Computers beginnen manchmal aber schon bei der Konstruktion des Handwerkszeuges – des Teleskops. Durch mathematische Berechnungen der Linsen und Spiegel werden hier die besten Ergebnisse erzielt. Viele „Himmelsfreunde" hätten schon früher gern ein Teleskop selbst gebaut. Vor der Einführung des Heimcomputers griffen viele lieber auf einen Bausatz zurück, als die schwierigen Berechnungen des Strahlenganges selbst durchzuführen. Diese – oft wochenlange – Arbeit kann man mit dem Computer in wenigen Minuten erledigen.

Positionen festlegen

Bei der Suche nach einem bestimmten Stern am Himmel kann uns der Rechner ebenfalls unterstützen. Himmelskörper sind keine feststehenden Objekte. Durch die Erdrotation beschreiben sie Bahnen am Himmel, so daß ihre Position je nach Tages- und Jahreszeit verschieden ist. Zur Lokalisierung dienen dabei Systeme von Längen- und Breitengraden, wie sie auch in der herkömmlichen Geographie üblich sind. Denkt man sich ein auf die Himmelsfläche projiziertes Koordinatensystem, läßt sich jede Sternenposition durch Angabe von Deklination und Rektaszension unverwechselbar festlegen.

Mit diesen Angaben kann man den Stern dann in einer Karte einzeichnen und aus vielen dieser Karten einen Himmelsatlas zusammenstellen. Für Planeten-, Satelliten- oder auch Kometenbeobachtung sind diese Fixpunkte unverzichtbar. Zusammen mit Werten der Leuchtkraft und spektralen Zusammensetzung der Strahlenemission sowie der Art und dem Alter eines Sterns können diese Daten im Computer gespeichert und verwaltet werden. Dadurch kann der Rechner eine Sternkarte für einen beliebigen Standort und jeden Zeitabschnitt auf den Bildschirm projizieren.

Durch die Erdrotation bewegt sich jeder

Stern innerhalb weniger Minuten aus dem Sichtfeld eines Teleskops, sogar, wenn es einen relativ großen Blickwinkel hat. Wer längere Beobachtungen in einem kleinen Gebiet vornehmen möchte, muß die Position des Teleskops daher ständig justieren. Dazu werden seit Jahren mechanische Antriebe verwendet. Heute stehen auch dem Amateur computergesteuerte Systeme zur Verfügung. Das Teleskop wird auf einer nach Norden gerichteten Achse montiert und von einem kleinen Servomotor genau mit der richtigen Geschwindigkeit bewegt, die nötig ist, um die Erdrotation auszugleichen. Auf der Schwenkachse und dem Teleskop selbst sind Meßfühler montiert, die über ein digitales Signal dem Computer die jeweilige Neigung des Teleskops angeben.

Die Überwachung des Motorantriebs durch den Rechner erlaubt beispielsweise das Verfolgen einer Sternenbahn während der ganzen Nacht, so daß auch Fotografien mit sehr großer Belichtungszeit möglich sind. In den USA ist ein Adapter namens „Celestial Navigator MK II" im Handel, mit dem sich die Verbindung zwischen Teleskop und Rechner über die normalen Schnittstellen bewerkstelligen läßt.

Amerikanische Hobby-Astronomen sind beim Computer-Einsatz aber noch weiter gegangen: Sie nutzen die Möglichkeit der Sprachein- und -ausgabe. Durch den Zuruf „Auf" bzw. „Kuppel drehen" läßt sich die Kuppel des Observatoriums öffnen und bewegen. Synthetische Sprache ist in der Dunkelheit eines Observatoriums ebenfalls eine große Hilfe bei der Datenausgabe durch den Computer: So kann der Rechner etwa Zeitintervalle laut ansagen, um dem Beobachter beim Fotografieren exakt die notwendige Belichtungszeit anzugeben.

Profi-Astronomen verwenden auch Teleskope, die anstelle von Licht Radio- oder Röntgenstrahlen registrieren. Die Größe und das Gewicht dieser Geräte führen zu schwierigen Problemen bei der Handhabung. Das 1964 bei Manchester installierte elliptische 38 x 25 Meter-Teleskop war als erstes mit einem Computer verbunden, der die angewählten Himmelskoordinaten in Steuerungsbefehle für die verschiedenen Antriebsmotoren umwandelte. Dabei wurden Ist- und Sollposition des Teleskops viermal in jeder Sekunde verglichen.

Mit Computern lassen sich auch atmosphärische Einflüsse ausgleichen, die durch veränderte Lichtbrechung bei Feuchtigkeits- und Temperaturdifferenzen entstehen. Auch Verzerrungen des Objektbildes durch Eigenschaften des Teleskops selbst können durch den Computer kompensiert werden.

Im Gegenzug zur Hilfe des Computers in der Astronomie haben Himmelskundler aber auch die Computerentwicklung gefördert: Die Programmiersprache FORTH ist eine Entwicklung des Astronomen C. H. Moore vom Kitt Peak Observatorium aus Arizona, der damit ursprünglich das Radioteleskop und dessen Prozeßdaten steuern wollte.

Musterschüler
Die Leistungen von Schülern und Lehrern der Kettering-Schule in Northamptonshire haben der Schule auf dem Gebiet der Astronomie einen besonderen Ruf eingetragen. Die Kettering-Schüler haben mehrfach als erste das Auftauchen eines neuen Satelliten im Orbit beobachtet.

Dieses Riesen-Teleskop wurde nach der Entdekkung von Radiostrahlung aus entfernten Galaxien in Jodrell Bank errichtet. Mit Radioteleskopen können Objekte ausgemacht werden, die auch dem leistungsfähigsten optischen Teleskop verborgen bleiben.

. . . gegen die Zeit

Beachtet man einige Besonderheiten in der Programmstruktur und bei der Verwendung von Variablen, läßt sich die Geschwindigkeit fast jedes BASIC-Programms erhöhen.

BASIC ist eine äußerst flexible Programmiersprache, die sich als Lernhilfe eignet. Verfügt man über genug Speicher und ist die Ausführzeit nicht von Bedeutung, dann läßt sich fast jedes Programm in BASIC schreiben. Da die Sprache jedoch meist in interpretierter Form und nur selten mit einem Compiler verwandt wird, sind die Ausführzeiten oft sehr lang.

Ein Sortierprogramm besteht aus einer Hauptschleife mit verschachtelten kleineren Schleifen. Sollen 100 Daten sortiert werden, muß das Programm die gleiche Schleife zwischen 2500- und 5000mal ausführen. Obwohl ein Sortierprogramm in BASIC immer langsam ist, läßt sich die Bearbeitungsgeschwindigkeit mit Hilfe sorgfältig durchdachter Befehlsanordnung optimieren.

Will man die Unterschiede zwischen einem „guten" und einem „schlechten" Programmaufbau feststellen, braucht man ein Rahmenprogramm und eine Methode zur Zeitmessung. Die Ausführzeit läßt sich exakt feststellen, wenn Sie am Programmanfang ein BEEP-Signal einbauen.

Unser Rahmenprogramm sieht so aus:

```
1000 L=500
2000 PRINT "*** ANFANG *** ":REM HIER
     "BEEP" BEFEHL
2100 TI$="000000"
2200 FOR K=1 TO L
.....
2900 NEXT K:T9=TI
2950 REM HIER "BEEP" BEFEHL
3000 PRINT "*** ENDE ***"
3100 PRINT "AUSFUEHRZEIT "; (T9/60);
     "SEKUNDEN"
```

Die Zeilen 2100 und 3100 sind nur auf dem Commodore möglich, bei anderen Maschinen müssen sie gelöscht oder entsprechend geändert werden. Zwischen den Zeilen 2200 und 2900 können die Befehle untergebracht werden, deren Ausführungszeit gemessen werden soll. Beachten Sie, daß die Variable L bestimmt, wie oft die Schleife wiederholt wird.

Es gibt einige allgemeine Regeln für das Schreiben von „schnellem" BASIC (hier nach Wichtigkeit geordnet):

1. Keine Rechenoperationen in Schleifen planen.

Potenzierung (z. B. x^3, d. h. x hoch 3) und mathematische Funktionen (wie z. B. cos(y), d. h. der Cosinus des Winkels) sind besonders langsam. Wenn auch Multiplikation und Division langsamer sind als Addition und Subtraktion, so benötigt selbst die schnellste Rechenoperation (die Addition) immer noch relativ viel Zeit.

Fügen Sie jetzt die folgenden Zeilen in das Rahmenprogramm ein:

```
900 Z=1.1
2300 X=Z↑3
```

und starten Sie es. Auf unserem Testgerät brauchte das Programm für 500 Wiederholungen 27,95 Sekunden. Verändern Sie die Zeile 2300 jetzt in

```
2300 X=Z*Z*Z
```

und starten Sie das Programm nochmals. Jetzt beträgt die Laufzeit nur 3,55 Sekunden.

Untersucht man dieses Phänomen weiter, wird man irgendwann an einen Punkt kommen, an dem bei der Potenzierung die einfache Multiplikation besser durch die Exponentialfunktion ersetzt würde. Auf unserem Computer fanden wir diese Grenze in der 18. Potenzierungsebene (X = Z hoch 18). Will man jedoch $Z^{2,3}$ berechnen, ist die wiederholte Multiplikation nicht zu gebrauchen, während die Exponentialfunktion alle realen Zahlen – auch die negativen – liefert.

Vergleichen Sie einmal die anderen mathematischen Operationen mit dem Rahmenprogramm, und überprüfen Sie, ob es schneller geht, eine Zahl z. B. durch 2 zu dividieren oder mit 0,5 zu multiplizieren.

2. Gebrauchen Sie Variablen anstelle von numerischen Konstanten.

In einem BASIC-Programm muß jede numerische Konstante (z. B. 7280) erst in eine verwendbare Form gebracht werden.

3. Springen Sie bei einem GOTO-Befehl möglichst auf höhere Zeilennummern statt auf niedrigere. Müssen Sie im Programm rückwärts springen, ist es besser, an den Anfang des Programms zu gehen statt nur ein paar Zeilen zurück.

Dies gilt auch für GOSUB. Findet der BASIC-Interpreter einen GOTO- oder GOSUB-Befehl, vergleicht er die Zeilennummer, auf die er springen soll, mit der augenblicklichen Zeilennummer. Ist die Zeilennummer des Befehls größer als die augenblickliche, sucht er vorwärts, bis er die entsprechende Programmzeile gefunden hat; ist sie jedoch kleiner, sucht der Interpreter die entsprechende Zeilennummer vom Anfang des Programms an. Es ist daher besser, Subroutinen und oft benutzte Programmzeilen entweder an den Anfang oder das Ende eines Programms zu stellen. Setzen Sie jetzt 56 REM-Zeilen zu Beginn des Rahmenprogramms, um die Länge eines typischen Programms zu simulieren, und fügen Sie folgende Zeilen ein:

```
2300 GOTO 2400
2400 GOTO 2500
2500 GOTO 2900
```

Für 500 Wiederholungen brauchten wir 2,33 Sekunden, während

```
2300 GOTO 2500
2400 GOTO 2900
2500 GOTO 2400
```

4,85 Sekunden benötigte.

4. Initialisieren Sie alle Variablen in der Reihenfolge der Häufigkeit ihres Vorkommens.

Alle Variablennamen werden vom BASIC-Interpreter in einer Symboltabelle in der Reihenfolge gespeichert, in der sie zum ersten Mal im Programm auftreten. Je später eine Variable in dieser Symboltabelle auftaucht, desto länger braucht der Interpreter, um sie zu finden. Sie sollten deshalb möglichst keine neuen Variablen einsetzen, wenn Sie eine zuvor benutzte, aber gegenwärtig nicht mehr gebrauchte, wiederverwenden können.

Wird eine Variable in verschachtelten Schleifen verwandt – wie etwa in Sortierprogrammen –, dann wird sie oft angesprochen und sollte gleich am Anfang des Programms vor jeder anderen Variablen – wenn nötig mit einem Dummywert – initialisiert werden.

```
1000 L=500:C=7280:X=0:Z=1.1
2300 A=0
```

brauchte 2,2 Sekunden für 500 Wiederholungen, während

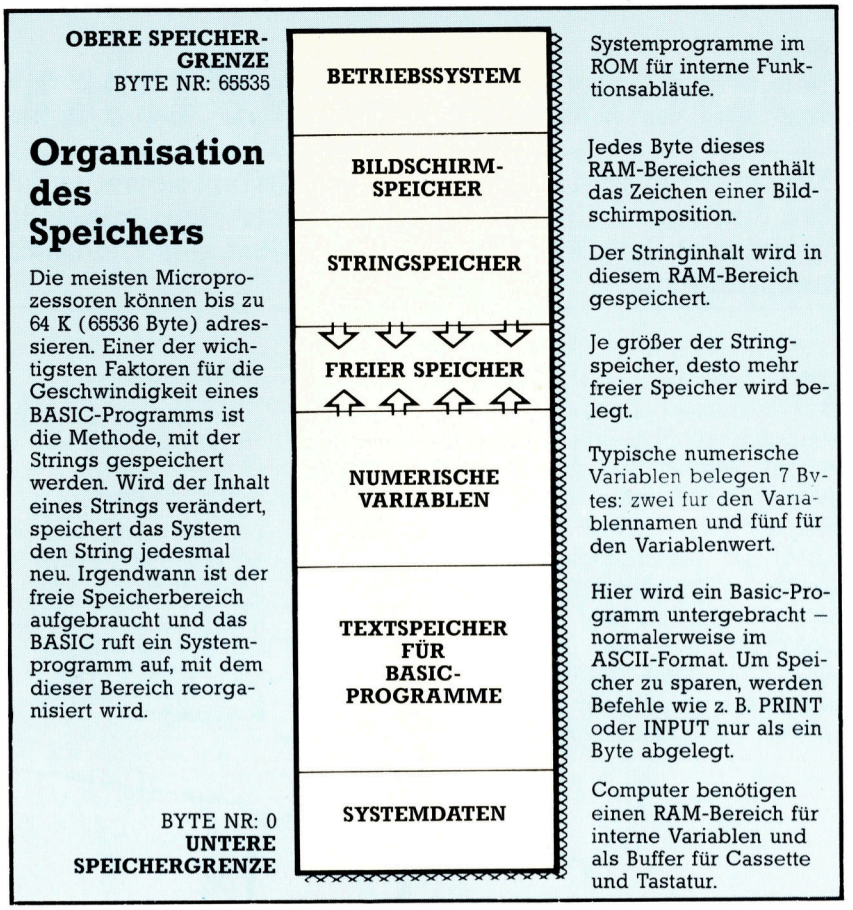
```
1000 A=0:L=500:C=7280:X=0:Z=1.1
2300 A=0
```

nur 2,06 Sekunden benötigte.

5. Vermeiden Sie Zeichenketten (Strings).

Organisation des Speichers

Die meisten Microprozessoren können bis zu 64 K (65536 Byte) adressieren. Einer der wichtigsten Faktoren für die Geschwindigkeit eines BASIC-Programms ist die Methode, mit der Strings gespeichert werden. Wird der Inhalt eines Strings verändert, speichert das System den String jedesmal neu. Irgendwann ist der freie Speicherbereich aufgebraucht und das BASIC ruft ein Systemprogramm auf, mit dem dieser Bereich reorganisiert wird.



Es ist schwierig, für für die String-Verarbeitung ein allgemeingültiges Testprogramm zu schreiben, da jedes Computermodell seinen Speicher anders verwaltet. Auf unserem Gerät gaben wir ein:

```
40 POKE 52,32:POKE 56,32:CLR
```

um den für BASIC-Programme verfügbaren Speicher radikal zu verkleinern, und dann

```
1000 L=500:DIMT$(L)
1100 FOR K=1 TO L
1200 T$(K)="A"+"B"
1300 PRINT K
1400 NEXT K
```

womit wir viele Speicherzellen im Stringbereich belegten. Bei jedem Durchlauf zeigt der PRINT-Befehl die Schleifenzahl an. In unserem Testprogramm wurde diese Anzeige unterbrochen, wenn das Systemprogramm den Stringspeicher neu organisieren mußte. Die Pause dauerte manchmal mehr als drei Sekunden. Das Programm geht weiter mit:

```
2300 A$=LEFT$(T$(L),1:BS=A$+RIGHT$
   (T$(L),1)
```

und brauchte für 500 Wiederholungen 30,03 Sekunden. Bei einem zweiten Durchlauf stellten wir mehr Speicher zur Verfügung und brauchten 8,66 Sekunden.

Speicherstruktur

Höhere Programmiersprachen wie beispielsweise BASIC verwalten den Speicher automatisch. Detaillierte Kenntnisse über den Speicheraufbau helfen, die Fähigkeiten des Computers optimal auszunutzen.

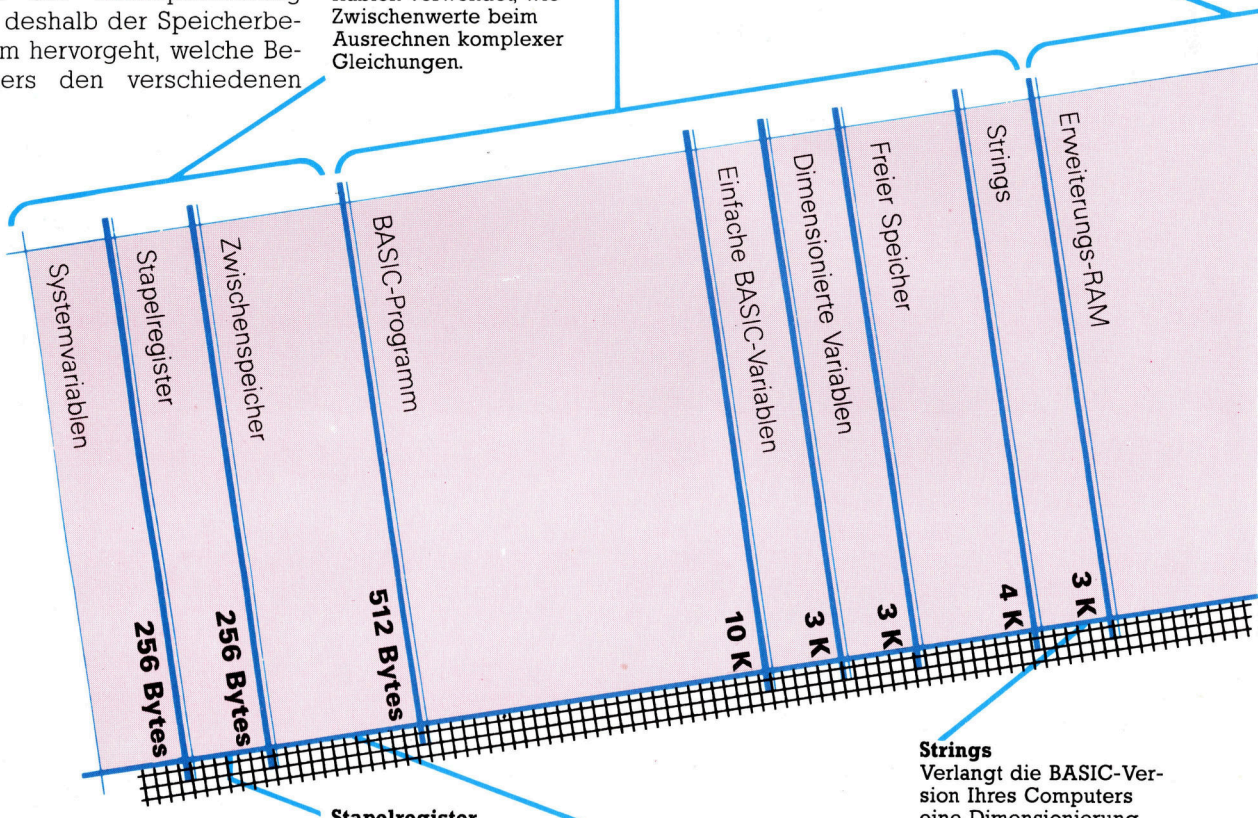
Die Zentraleinheit CPU (Central Processing Unit) eines Computers ist mit einem festen Adressenbereich ausgestattet, der bestimmt, wieviel Speicherplatz zur Verfügung steht. Bei den meisten Heimcomputern sind dies 64 KByte. Diese Kapazität umfaßt alle RAM- und ROM-Speicher sowie die Speicherplätze für Erweiterungsmodule, Schnittstellen und Anschlußstellen. Einer der wichtigsten Gesichtspunkte bei der Konzeptionierung eines Computers ist deshalb der Speicherbelegungsplan, aus dem hervorgeht, welche Bereiche des Speichers den verschiedenen Rechnerfunktionen zugeordnet sind. Wer sich mit BASIC-Programmierung begnügt, braucht sich um die Speicherorganisation nicht zu kümmern, wer aber in die Maschinensprache einsteigen oder Hardware bauen möchte, muß die Speicherorganisation genau kennen.

Hier wird gezeigt, wie ein typischer Speicherbelegungsplan aussieht. Dabei stützen wir uns mehr auf einen Computer mit einem 6502-Prozessor als CPU als auf einen Z80-Prozessor. Leider liefern nicht alle Computerhersteller genaue Angaben über die Speicherbelegung. Gewöhnlich läßt sich jedoch die Speicherorganisation durch Experimentieren herausfinden.

Systemspeicher
Bei einem Computer mit einer RAM-Kapazität von vier KByte stehen dem Anwender möglicherweise nur drei KByte frei zur Verfügung. Den Rest beansprucht der Systemspeicher für das Betriebssystem. Ein Teil dieses Speicherbereichs wird für Systemvariablen verwendet, wie Zwischenwerte beim Ausrechnen komplexer Gleichungen.

Anwender-RAM
Der Umfang dieses Speichers bestimmt, ob komplexere, leistungsfähige Programme auf diesem Computer laufen können. Beim Computerkauf ist dies einer der wichtigsten Gesichtspunkte.

Nicht belegt
Im Speicher muß auch Platz für Erweiterungs-RAMs reserviert sein. Es gibt Computer, die eine Erweiterung von mehr als 64 KByte erlauben, wobei eine besondere Schaltung die entsprechende RAM-Sektion je nach Bedarf in diesen Speicherbereich schaltet.



Stapelregister
Dieser Bereich ist ausschließlich für die CPU reserviert und als „LIFO" (Last In/First Out) strukturiert. Wird beispielsweise eine GOSUB-Routine in BASIC ausgeführt, schiebt die CPU denjenigen Speicherplatz im Register nach oben, auf den sie zurückkehren (RETURN) muß. Das Stapelregister wird besonders stark bei arithmetischen Rechnungen und in FOR ... NEXT-Schleifen in Anspruch genommen.

Zwischenspeicher
Für die Tastatur muß ein Zwischenspeicher reserviert sein, der Zeichen aufnimmt, die schneller eingegeben werden, als sie das Programm verarbeiten kann. Auch Peripheriegeräte, wie z. B. Cassettenrecorder, benötigen einen Zwischenspeicher, weil die meisten Betriebssysteme die Daten blockweise übertragen.

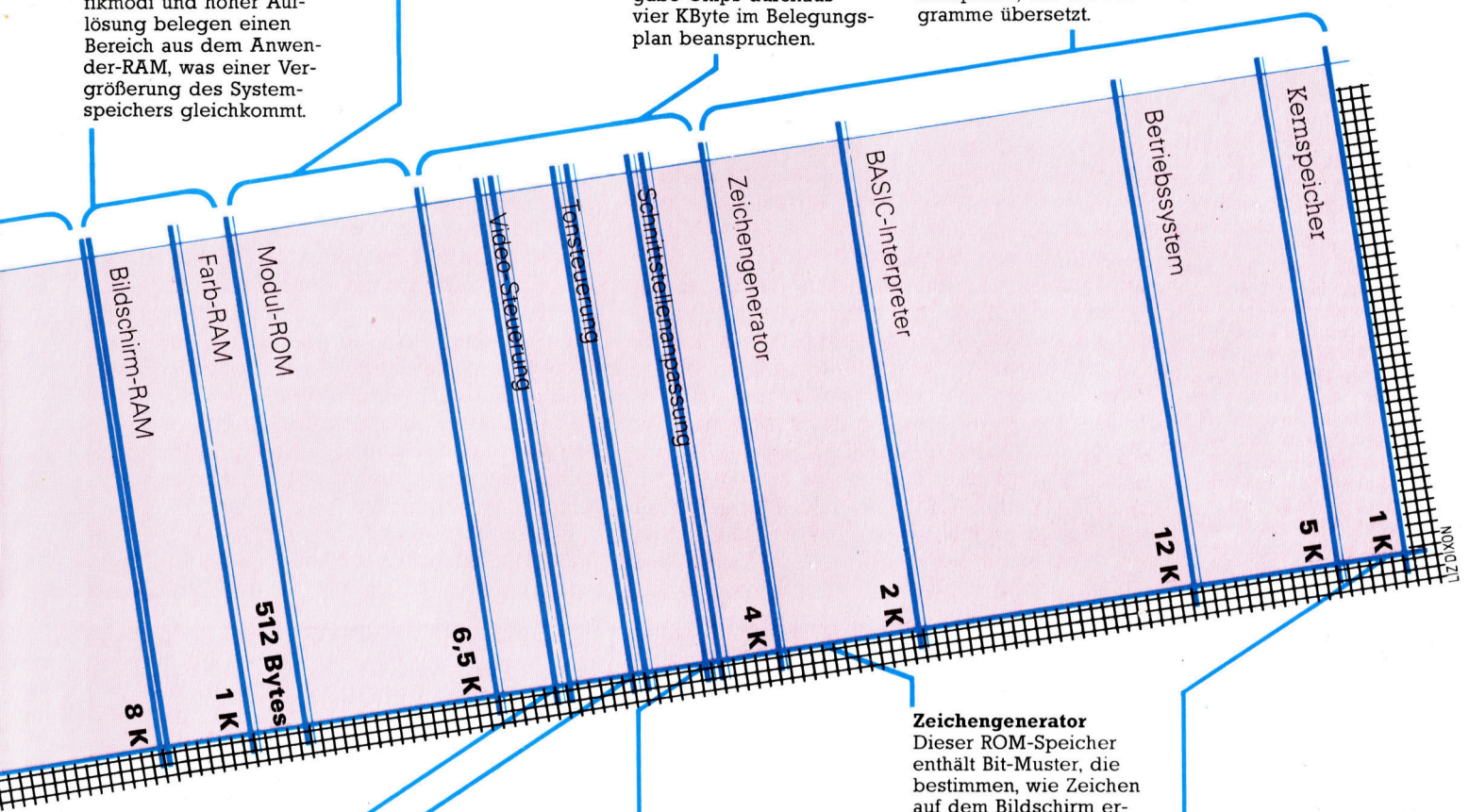
Strings
Verlangt die BASIC-Version Ihres Computers eine Dimensionierung der benötigten Strings, werden diese nach dem gleichen Schema abgelegt wie dimensionierte Variablen. Sind jedoch „dynamische" Strings möglich, deren Länge veränderbar ist, werden die tatsächlichen Daten getrennt in einem Speicherteil abgelegt, dessen Größe variabel ist. In gewissen Zeitabständen sorgt dann das Betriebssystem für eine „Müllaktion", d. h. nicht mehr benötigte Daten werden gelöscht.

System-RAM
Manche Systeme weisen diesen RAM-Speicher nicht als Teil des Anwender-RAM-Speichers aus. Dies trifft gewöhnlich auf das Bildschirm-RAM zu, in dem ein Byte je einem Zeichenplatz auf dem Bildschirm entspricht, und ebenso für das Farb-RAM, in dem ein Byte die Vorder- und Hintergrundfarben für die Zeichenpositionen bestimmt. Computer mit vielen Grafikmodi und hoher Auflösung belegen einen Bereich aus dem Anwender-RAM, was einer Vergrößerung des Systemspeichers gleichkommt.

Nicht belegt
Programme auf Steckmodulen (Cartridges) werden im Speicherbelegungsplan als Erweiterungs-ROM behandelt. Manche Computer haben eigene ROM-Steckplätze für zusätzliche Sprachen, für die der Belegungsplan ebenfalls einen reservierten Bereich ausweist.

Eingabe-/Ausgabe-Chips
Eine CPU kann nur mit Einrichtungen kommunizieren, die als Speicherplatz im Belegungsplan erscheinen. Darin müssen alle Schnittstellen und Chips im Belegungsplan enthalten sein. Dazu gehören die Schnittstellen für die Tastatur, das Cassettenlaufwerk, die Videosteuerung und den Drucker. Da die CPU Speicherplätze blockweise adressiert, können die Ein-/Ausgabe-Chips durchaus vier KByte im Belegungsplan beanspruchen.

System-ROM
Der ROM-Speicher enthält Informationen, die permanent benutzt, jedoch nicht verändert werden können. Der wichtigste Teil des ROM-Speichers ist das Betriebssystem, das die Funktionen des Computers steuert. Diese Maschinenprogramme führen Funktionen wie Abfragen der Tastatur und Speichern oder Lesen von Daten auf Cassette aus. Eine andere Komponente ist der BASIC-Interpreter, der die Programme übersetzt.



Video-Steuerung
Hochentwickelte Grafiken, wie Sprites und Mehrfachauflösungen, werden zunehmend über die Hardware und nicht über die Software gesteuert. Im Belegungsplan erscheint die Video-Steuerung als eine Anzahl von Einbyte-Registern, die jede sichtbare Komponente bestimmen, angefangen von der Hintergrundfarbe bis zur genauen Position eines jeden Sprites.

Tonsteuerung
Einfache Toneffekte können über die Software erzielt werden. Computer mit großer Tonpalette oder eigenem Tongenerator verfügen jedoch über eine besondere Tonsteuerung mit eigenem Verstärker.

Schnittstellenanpassung
Die Schnittstellenanpassung PIA (Peripheral Interface Adaptor) dient dazu, die üblichen Schnittstellen für Tastatur, Cassetten, Joysticks (Steuerknüppel) und Drucker zu handhaben. Die meisten hochentwikkelten Chips können zwischen parallelen und seriellen Daten unterscheiden und haben eingebaute Zeitschalter, die beim Programmieren oder dazu, Übertragungsraten zu steuern, verwendet werden können.

Zeichengenerator
Dieser ROM-Speicher enthält Bit-Muster, die bestimmen, wie Zeichen auf dem Bildschirm erscheinen. Bei manchen Computern können alle oder ein Teil der gesetzten Zeichen in den RAM-Speicher kopiert werden, was für den Anwender die Möglichkeit schafft, selbst Zeichen zu definieren.

Kernspeicher
Dieser Speicher, der bei nahezu jedem Computer einen anderen Namen trägt, ist das Herz des Betriebssystems. Beim Einschalten des Computers springt die CPU automatisch zu dieser Speicherstelle und führt das „Quellprogramm" aus. Die CPU sucht den RAM-Bereich nach der verfügbaren Speicherkapazität ab und prüft, ob ein Cartridge eingesteckt ist. Vom Kernspeicher werden auch die meisten elementaren Formen der Eingabe und Ausgabe gehandhabt.

Terminal-Ergonomie

Die Gestaltung des Arbeitsplatzes am Computer unter Beachtung arbeitswissenschaftlicher Erkenntnisse.

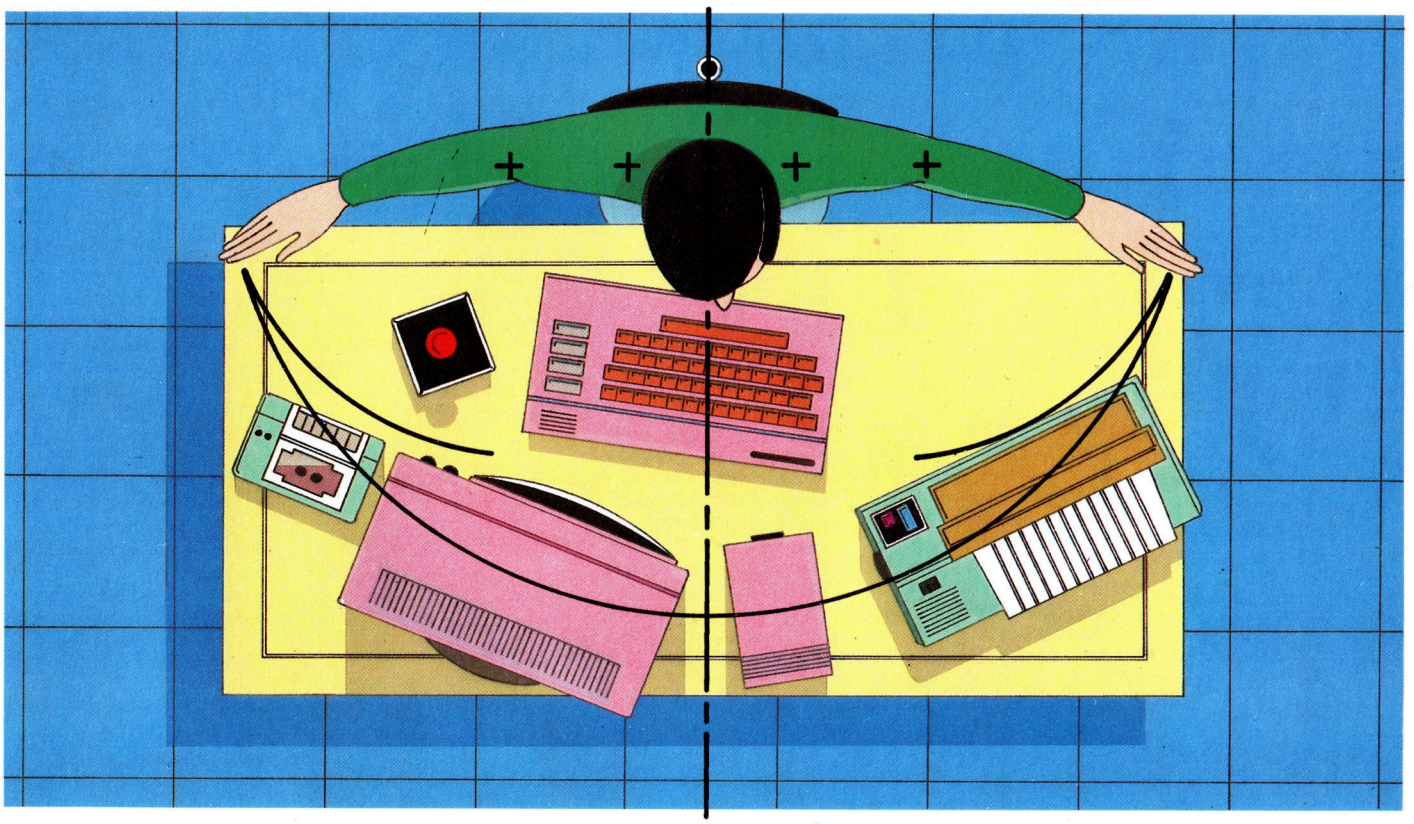
Obwohl der menschliche Körper in Größe und Form sehr unterschiedlich sein kann, bleiben seine Proportionen gleich. Die Arbeitswissenschaft nutzt diese Beständigkeit und stellt hieraus grundsätzliche Regeln für die Gestaltung der Arbeitsumgebung auf. Für den Besitzer eines Heimcomputers bedeutet dies, daß der Monitor eine Armlänge entfernt vor den Augen aufgestellt werden sollte, damit das Auge sich nicht zu sehr umstellen muß, wenn auch andere Gegenstände optisch zu erfassen sind. Entsprechend sollte nach diesen Regeln auch die Tastatur aufgestellt werden.

Beim Design eines Computers werden zwei Aspekte vordringlich berücksichtigt: Das ästhetische Empfinden und die Ergonomie, die die Beziehung zwischen dem Anwender und seiner Umgebung untersucht. Eine unschöne optische Gestaltung des Gerätes trägt nicht gerade zur Steigerung der Arbeitsfreude bei, unabhängig davon, wie gut eine Anlage funktioniert. Ebenso darf die Arbeitsumgebung nicht ablenkend oder unbequem sein.

Beim Kauf eines Micro-Computers wird die ergonomische Qualität wahrscheinlich weniger berücksichtigt als der Preis oder die technische Leistung. Es lohnt dennoch, sich einige Gedanken über das Umfeld zu machen, in dem der Computer eingesetzt werden soll. Arbeiten Sie an einem büroähnlichen Platz, der Ihnen ausreichende Tischfläche in der richtigen Höhe bietet? Oder schließen Sie Ihren Heimcomputer einfach an den Fernseher an und bedienen ihn auf den Knien, oder liegen Sie gar vor dem Fernseher auf dem Fußboden?

Das Programmieren von Computern ist schon kompliziert genug, als daß man sich die Tätigkeit durch eine unbequeme Arbeitshaltung noch erschweren sollte. Es gibt viele Möglichkeiten, sich einen bequemeren Arbeitsplatz zu schaffen. Die Überlegungen beginnen damit, die Bildschirmdarstellung zu optimieren. Ein gewöhnliches Fernsehgerat kann allerdings nicht die neuesten Entwicklungen berücksichtigen oder die Spiegelungen auf der Bildschirmoberfläche unterdrücken. Filter und ein speziell eingefärbter Phosphor werden eingesetzt, um die Reflexionen so gering wie möglich zu halten. Auch einfache Vorsatzfilter können die Darstellungsqualität eines Fersehers verbessern. Ebenso können Polarisationsfilter verwendet werden, die die Reflexion verringern und dazu beitragen, einen hohen Kontrast bei geringer Helligkeit zu erreichen, was einer unnötigen Augenbelastung rechtzeitig vorbeugt.

Wichtig ist auch die Raumbeleuchtung. Nachts ist es besser, mit einer niedrigen Tischlampe zu arbeiten, die die Tastatur und die Ar-



beitsvorgänge beleuchtet, den Bildschirm aber in einem dunkleren Bereich läßt. Ebenso ist der Augenabstand zum Monitor von Bedeutung. Er sollte etwa eine Armlänge betragen. Das Display selbst sollte kippbar sein, damit die Oberfläche in einen rechten Winkel zur Blickrichtung gebracht werden kann. Dies läßt sich auch einfach durch Unterlegen von Büchern erreichen. Zusätzlich tritt in manchen Fällen das Problem einer Spiegelung des Anwenders selbst auf. Ein matter Vorsatzfilter schafft hier schnelle Abhilfe.

Ist die Einstellung des Bildschirms zufriedenstellend, kann das Augenmerk auf das mechanische Layout und die Eigenschaften der Tastatur gerichtet werden. Die wichtigsten Faktoren sind hierbei die Höhe der Tasten in bezug auf die Tischoberfläche sowie der Winkel der Tastenreihen zueinander. Im Idealfall sollten die Handgelenke und die Unterarme vor der Tastatur auf dem Tisch abgestützt werden können, und der Neigungswinkel sollte insgesamt verstellbar sein. Unglücklicherweise haben aber nur wenige Heimcomputer das wünschenswerte niedrige Profil. Einige sind auch mit unkomfortablen Tastaturen aus Mehrschicht-Membranen oder aus Gummitasten statt der üblichen Federtasten ausgestattet. Membrantasten vermitteln keinen Druckpunkt, so daß die Eingabe langer Programme zu einer anstrengenden Tätigkeit wird. Beim Oric-1 und Spectrum beispielsweise wird versucht, den fehlenden Druckpunkt mit einem akustischen Signal zu kompensieren, das beim Drücken der Taste ertönt. Doch dies gelingt nur ungenügend. Einige Unternehmen bieten für diese Geräte Austausch-Tastaturen mit Federtasten an. Doch die besseren Modelle sind recht teuer.

Die optimale Gestaltung eines Keyboards erfordert eine konkave Tastenanordnung. Auch jede einzelne Taste sollte konkav geformt sein. Diese Oberflächenhöhlung vermindert das Abrutschen der Finger. Der Commodore 64, der Apple und der Acorn B beispielsweise erfüllen diese Anforderungen.

Das Layout der Tastatur selbst war lange Zeit ein großer Streitpunkt der Designer. Mit dem Aufkommen der ersten Schreibmaschinen im 19. Jahrhundert gab es ebensoviele verschiedene Gestaltungsarten wie Hersteller. Immerhin wurden die am häufigsten verwendeten Buchstaben in der Mitte der Tastatur angeordnet. Als 1870 der Typenkorb eingeführt wurde, entdeckten die Hersteller, daß sogar langsame Schreiber die Hebel verklemmen konnten. Das Problem tauchte besonders dann auf, wenn Worte geschrieben wurden, deren Buchstaben nahe beieinander im Typenkorb lagen. Also entschied man sich, solche Buchstaben, die oft in Wörtern aufeinanderfolgen, im Typenkorb auseinander zu legen. Hieraus entstand der heutige Standard. Die QWERTY-Tastatur wurde von Scholes und Gliden in den USA entwickelt. In Deutschland jedoch gilt das QWERTZ-Format als Standard. Es gibt keinen echten Grund mehr dafür, daß eine elektronische Tastatur sich an diese Auslegung hält, mit der Ausnahme, einen Standard zu erhalten (ein interessantes Beispiel dafür, wie ein defacto-Standard zwar unerwünscht wird, aber nicht mehr zu ändern ist).

Trotzdem wurden einige Anstrengungen unternommen, alternative Tastaturen zu entwikkeln. 1977 machte sich Frau G. Malt die Flexibilität der elektronischen Hardware zunutze und entwickelte ein Keyboard, das der Hand optimal angepaßt und weniger ermüdend war. Eine Schreibgeschwindigkeit von 300 Worten und mehr pro Minute war keine Seltenheit. Doch auch sie konnte die „Bürde" des QWERTY-Standards nicht durchbrechen.

Preiswerte Alternative

Einen wesentlichen Vorteil teilt diese Tastatur aber mit vielen Micro-Computern: Die Eingabeeinheit läßt sich vom Sichtgerät getrennt benutzen. Die meisten Heimcomputer haben keinen eingebauten Bildschirm und sind klein genug, um leicht bewegt zu werden. Bei Bürocomputern ist das aber nicht der Fall. Hier werden die Tastaturen zunehmend so flach wie möglich konstruiert und durch ein Spiralkabel mit dem Rechner verbunden. Noch fortschrittlicher ist der IBM-PC-Junior: Das Kabel wird hier wie bei der Fernbedienung eines Fernsehers durch Infrarotstrahlen ersetzt.

Da die Ergonomie aber keine absolut objektive Wissenschaft ist (die Anforderungen an die Beziehung zwischen Arbeiter und Umgebung sind unterschiedlich), ist es unmöglich, klare und eindeutige Regeln aufzustellen. Ziel ist ein bequemes Arbeiten über einen längeren Zeitraum hinweg, mit voller Konzentration und ohne Ermüdung. Der Computer-Anwender selbst sollte ausprobieren, wie seine Arbeitsbedingungen verbessert werden können. Als Vorbild sei die Menü-gesteuerte Software genannt, bei der eine Maus die Tastatur ergänzt. Eine preiswerte Alternative hierzu kann mit dem Joystick oder dem Trackball erstellt wer-

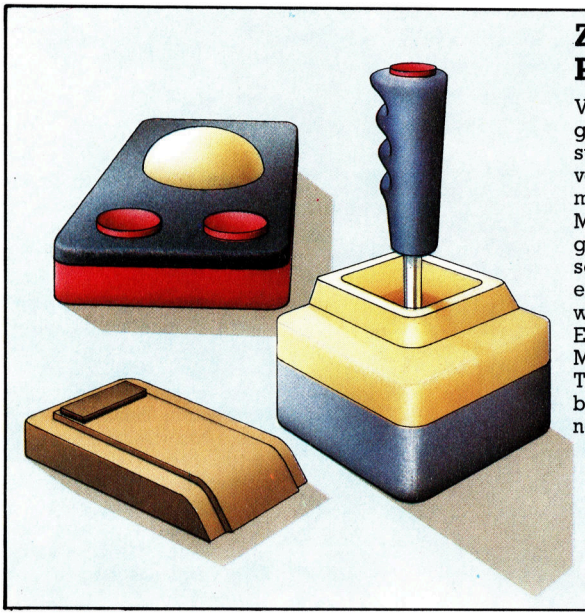
Wenn Gespräche mitgeschrieben werden sollen, der Stenograf aber der Geschwindigkeit des Sprechers nicht folgen kann, wird häufig eine Maschine eingesetzt. Sie ist als Stenorette bekannt. Diese Stenografiermaschine verwendet eine besondere Kurzschriftversion des phonetischen Alphabets.

Vor der Erfindung der elektronischen Tastatur durften die Buchstabenhebel von häufig nacheinander verwendeten Buchstaben nicht nebeneinander liegen, weil sie sonst verklemmen konnten. Obwohl diese Einschränkung heute nicht mehr besteht, wird immer noch an der gewohnten QWERTY-Tastatur festgehalten. Die MALTRON-Tastatur, deren Tasten entsprechend ihrer Anschlagshäufigkeit angeordnet wurden, konnte sich nicht durchsetzen.

den. Sie müssen sich allerdings hierfür kleine Programme schreiben. Doch unter Verwendung von PEEK und POKE im Bereich des Bildschirmspeichers ist dies keine schwierige Sache. Ebenso ist es möglich, einige Tasten mit neuen Werten zu belegen und entsprechend zu definieren. In diesem Fall ist die einfachste Lösung, mit dem PEEK-Befehl den Wert des Buchstabens in ein Feld mit acht Variablen zu bringen, ihn dort zu ändern und mit dem POKE-Befehl zurückzuschreiben. Sie können mit POKE den Wert allerdings auch direkt verändern, müßten aber zuvor den ursprünglichen Wert vorübergehend in einem Feld sichern und danach jedes Zeichen in seine neue Position einordnen.

Es bleibt aber wichtig, sich einen optimalen Arbeitsplatz zu schaffen. Im Fachhandel werden sogenannte „Computertische" angeboten, die dem Rechner und der Peripherie genügend Platz bieten. Nun nehmen diese Möbelstücke oft mehr Raum ein, als zur Verfügung steht; es bietet sich daher an, sich selbst einen Computertisch „nach Maß" anzufertigen.

Sollten Ihre Interessen in den handwerklichen Bereich hineingehen, können Sie an den Bau eines speziellen Arbeitsplatzes denken, bei dem die Tastatur in die Arbeitsplatte eingelassen und der Bildschirm im richtigen Winkel aufgebaut ist. Professionelle Arbeitsplätze sehen üblicherweise zusätzlichen Platz für die Massenspeicher vor, so daß die Disketten oder Cassetten in Schubladen aufbewahrt werden können. Ergonomie ist im Grunde eine Frage des gesunden Menschenverstandes. Das Nachdenken wird mit bedeutend weniger Rückenschmerzen und einer geringeren Augenbelastung belohnt.



Zukunfts-Perspektiven

Viele Computer-Designer würden gern vollständig auf die Tastatur verzichten, wenn dies möglich wäre. Neuere Micro-Computer mit größerem Speicher und schnellerem Prozessor ermöglichen die Verwendung zusätzlicher Eingabegeräte wie Maus, Joystick oder Trackball. Allerdings bedarf es hierfür geeigneter Software.

Klanggebäude

Mehr über die Soundmöglichkeiten des VC 20.

Es wurde bereits untersucht, wie sich die drei Oszillatoren des Computers mit POKE-Befehlen steuern lassen und wie die Lautstärke und Dauer eines Tones festgelegt wird. Wir haben herausgefunden, daß man Tondauer und Pausen mit FOR . . . NEXT-Schleife oder eleganter mit der internen Uhr des VC 20 über Jiffys (dem sechzigsten Teil einer Sekunde) steuern kann.

Melodien

Bevor sich auf dem VC 20 eine Melodie spielen läßt, müssen erst die Töne zusammengestellt werden, die dafür gebraucht werden. Für die Erzeugung einfacher Klänge nehmen wir folgende Noten:

```
D# E F D# C A#  G# G G# D# D#
```

Mit den bereits beschriebenen Techniken kann man jetzt die Dauer der einzelnen Töne festlegen und die Pausen mit der TI-Funktion bestimmen. Das folgende Programm spielt die Melodie (Beachten Sie dabei, wie die POKE-Folgen mit Variablen vereinfacht werden):

```
10 V = 36878
20 FOR I = 1 TO 11
30 READ N  REM *TON*
40 POKE V,7:P=TI: REM *LAUTSTAERKE AN*
50 IF TI-P < 15 THEN 50: REM *PAUSE*
60 POKE V-3,N:D=TI: REM *TON AUSGE-
BEN*
70 IF TI-D < 20 THEN 70: REM *DAUER*
80 POKE V-3,0: REM *ENDE TON*
90 NEXT I
100 DATA 203, 207, 209, 203: REM
    *TONWERTE*
110 DATA 195, 187, 179, 175
120 DATA 179, 203, 203
130 POKE V,0: REM *LAUTSTAERKE AUS*
140 END
```

Nun spielt das Programm die Töne zwar in der richtigen Reihenfolge, durch gleiche Tonlängen und Pausen klingt die Melodie jedoch etwas hölzern. Mit ein wenig Experimentieren lassen sich Programme zusammenstellen, die für jeden Ton individuelle Zwischenräume und Längen festlegen.

Klangeffekte

Verwendet man zwei oder drei Oszillatoren nebeneinander, so ist es möglich, einfache Akkorde zu spielen. Das folgende Programm spielt den D-DUR Akkord (F#, A und D). Es fängt mit F# an und fügt nach jeweils einer Sekunde das A und D hinzu. Der Akkord setzt sich dann für weitere zwei Sekunden fort.

```
10 POKE 36878,7
20 POKE 36874,233:D=TI
30 IF TI-D<60 THEN 30
40 POKE 36875,219:D=TI
50 IF TI-D<60 THEN 50
60 POKE 36875,147:D=TI
70 IF TI-D< 120 THEN 70
80 POKE 36878,0: POKE 36874,0
90 POKE 36875,0: POKE 36876,0
100 END
```

Man kann den Klang dieses Akkords jedoch noch verfeinern, indem man z. B. über eine Variable die Lautstärke während der Dauer des Tones an- und abschwellen läßt:

```
100 V=36878
110 FOR I= 1 TO 12
120 POKE V,I
130 NEXT I
140 POKE V,0
```

Diese Programmzeilen erhöhen die Lautstärke in Schritten von 1 bis 12, wobei der gesamte verfügbare Bereich von 0 (aus) bis 15 (laut) reicht. Man kann den Ton durch abwechselnde Werte von Laut und Leise auch „pulsieren" lassen. Und auch die Tonhöhe (Frequenz) läßt sich auf diese Weise verändern – ein Ton wird „gebeugt", wenn man Zeile 120 folgendermaßen schreibt:

```
POKE V-3,203+I
```

Spaß macht es auch, die Möglichkeiten des Rauschens, der Oszillatorfrequenzen und der Lautstärken miteinander zu kombinieren. Oft läßt sich dadurch die Klangerscheinung angenehmer gestalten. Aber ganz gleich, ob man Musik oder Klangeffekte für Spiele erzeugen will, Ziel all dieser Kombinationen ist, monotone Klänge zu vermeiden.

Wir haben gezeigt, wie sich auch mit den beschränkten Möglichkeiten des VC 20 interessante Töne und Melodien spielen lassen. Hauptproblem ist dabei das Fehlen von speziellen Tonsteuerungsbefehlen, so daß man auch einfache Klänge nur über komplizierte Befehlsfolgen erzeugen kann. Dadurch entstehen lange, sehr zeitraubende Programme.

Leuchtendes Beispiel

Der erste Anlauf, die hervorragenden Grafikfähigkeiten des Acorn B einzusetzen.

In England zählt der Acorn B zu den beliebtesten Heimcomputern. Auf ihm lassen sich mit wenigen BASIC-Befehlen erstaunliche grafische Darstellungen erzeugen, und auch die Geschwindigkeit des Bildschirmaufbaus ist beeindruckend.

Das BBC-BASIC besitzt eine Reihe von Befehlen für hochauflösende Grafik, mit denen man unter anderem gerade Linien ziehen, einzelne Punkte ansprechen und Dreiecke zeichnen und ausfüllen kann. Will man jedoch beliebige Formen mit Farben ausfüllen, läßt sich das nur mit einer Serie von kleinen Dreiecken verwirklichen, da der Acorn über keinen PAINT-Befehl verfügt. Text und Grafik lassen sich auf dem Bildschirm gleichzeitig darstellen, mit Hilfe von zwei getrennt steuerbaren Cursors für Grafik und Text und einem speziellen VDU-(oder Bildschirm-)Befehlssatz für den direkten Zugang zum Bildschirmspeicher von einem BASIC-Programm aus. Zudem können auf dem Bildschirm „Fenster" definiert werden, mit denen sich der Bildschirminhalt in getrennte Bereiche von Text und Grafik aufteilen läßt. Für jedes dieser Fenster, die sich auch einzeln löschen lassen, können unterschiedliche Farben verwendet werden.

Darstellungsarten

Der Acorn B verfügt über acht Grafikmodi, von denen allein drei nur für die Textdarstellung bestimmt sind. Man kann zwischen 20, 40 oder 80 Zeichen pro Bildschirmzeile wählen. Zwei, vier oder sechzehn Farben stehen zur Auswahl, aber auch bei der Verwendung von nur zwei oder vier Farben ist man nicht festgelegt, sondern kann innerhalb eines Programms die verwendeten Farben gegen eine andere der verfügbaren austauschen.

MODE 7 unterscheidet sich von allen anderen Darstellungsarten, da dabei nicht der ASCII-Standard eingesetzt wird. MODE 7 verwendet den Teletex-Zeichensatz, wobei normale Grafikbefehle wie PLOT oder DRAW nicht funktionieren.

Die folgende Tabelle zeigt den Auflösungsgrad und die Anzahl der Farben sowie die Textdarstellung in Zeilen und Spalten, die in den einzelnen Modi angesprochen werden können:

Mode	Text	Grafik	Farben
0	80×32	640×256	2
1	40×32	320×256	4
2	20×32	160×256	16
3	80×25		2 (schwarz&weiß)
4	40×32	320×256	2
5	20×32	160×256	4
6	40×25		2 (schwarz&weiß)
7	40×25		Teletext

Der Nullpunkt der hochauflösenden Grafik befindet sich bei allen Darstellungsarten am linken unteren Bildschirmrand. Die vertikalen Werte liegen im Bereich von 0 bis 1023 und die horizontalen Werte reichen von 0 bis 1279.

Die Farben des Hintergrundes, des Textes und der Grafik lassen sich mit den Befehlen COLOUR und GCOL auf einfache Weise bestimmen. Die Farbauswahl kann über ein besonderes Konzept von „logischen" und „aktuellen" Farben definiert werden. Das läßt sich am besten an einem Beispiel erläutern, bei dem MODE 0 – also nur zwei Farben – gesetzt ist. Hierbei werden den beiden möglichen Vordergrundfarben die logischen Zahlen 1 und 0 zugeordnet (ohne besondere Anweisungen nimmt der Computer dafür automatisch 0 als Schwarz und 1 als Weiß an). Der COLOUR-Befehl wählt nun die Farbe für Textdarstellung aus; COLOUR 1 bedeutet: Die „logische Farbe Nummer 1" als Textfarbe. Diese automatische Auswahl kann über den Befehl VDU19 verändert werden. Dazu ein Beispiel:

```
VDU19, 1, 2, 0, 0, 0,
```

Nach dem GCOL-Befehl müssen zwei Zahlen eingegeben werden, von denen die zweite die logische Farbnummer für grafische Darstellung festlegt. Die erste Zahl bestimmt, wie die Farbe auf dem Schirm eingesetzt wird. In dem Befehl GCOL a,b können die Variablen a und b Werte von 0 bis 4 annehmen. Damit kann der Anwender bestimmen, ob ein bestimmter Punkt auf dem Bildschirm in der logischen Vordergrundfarbe dargestellt werden soll, ob er durch AND mit der bereits existierenden Farbe verbunden werden oder mit OR oder Exclusive-OR diese Farbe ausschließen soll, und ob die ursprüngliche Farbe zu invertieren ist.

Fachwörter von A bis Z

Hier werden einzelne Fachausdrücke eingehend behandelt. Da bei der Kommunikation mit dem Computer meist die englische Sprache verwendet wird, werden hier zunächst die englischen Begriffe genannt, dann die deutsche Übersetzung. In den Gesamtindex werden sowohl deutsche als auch englische Stichwörter aufgenommen, damit Sie es leichter haben, das von Ihnen Gesuchte zu finden.

Access Time = Zugriffszeit

Darunter versteht man allgemein die Zeit für das Aufsuchen einer bestimmten Information innerhalb einer großen Datensammlung. Meist ist speziell der Zugriff auf einen einzelnen Datenblock in einer Datei gemeint, vor allem bei den großen Datenbanksystemen.

Die Zugriffszeit ist etwas anderes als die „Datenübertragungsrate" – das ist die Geschwindigkeit, mit der nach der Lokalisierung die Information zwischen Speicher und Rechner übertragen wird. Beim Sinclair Microdrive beispielsweise beträgt die Zugriffszeit circa 3,5 Sek.: Sie kann auf Null reduziert werden, wenn die gesuchte Information bei Abruf gerade am Lesekopf anliegt, und maximal muß man sieben Sekunden warten, bis das Endlosband einmal ganz umgelaufen ist. Verglichen mit der üblichen Disketten-Zugriffszeit (unter 1/2 Sek.) ist das langsam; dagegen ist die Datenübertragungsrate des Microdrive mit 16 KByte pro Sekunde nicht kleiner als bei anderen Floppylaufwerken.

Accumulator = Akkumulator

Jeder Microprozessor enthält eine Reihe von „Registern" – das sind Ein-Byte-Speicher, über die alle arithmetischen und logischen Operationen abgewickelt werden. Das wichtigste und meistbenutzte Register ist der Akkumulator (kurz Akku), der direkt am Rechenwerk hängt. Die Hauptaufgabe des Akku ist die Bereithaltung von Werten und die Aufnahme von Ergebnissen. Man kann beispielsweise irgendein Byte zum Akku-Inhalt addieren oder davon abziehen. Der BASIC-Anwender hat keinen direkten Zugang zum Akku; dagegen bezieht sich bei Programmierungen im Maschinencode die Mehrzahl der Befehle unmittelbar darauf.

Acoustic Coupler = Akustikkoppler

Das öffentliche Fernsprechnetz ist so ausgelegt, daß eine Informationsübertragung nur bei Sinusfrequenzen von etwa 300–3400 Hertz möglich ist, weil das für eine verständliche Sprachwiedergabe ausreicht. Diese „Bandbreite" bestimmt auch den für digitalen Datenverkehr nutzbaren Frequenzbereich. Man benötigt daher eine Einrichtung zur Umsetzung der binären Information in geeignete Frequenzen (diese Umsetzung nennt man „Modulation"). Man kann beispielsweise der dualen Null einen Ton bestimmter Frequenz zuordnen und die duale Eins durch einen anderen Ton darstellen.

Ein Gerät zur Umwandlung der Digitalsignale in Tonfrequenzen und umgekehrt heißt „Modem" (MOdulator-DEModulator). Das Modem wird am besten direkt an die Telefonlei-

tung angeschlossen; das ist aber nur bei stationärem Betrieb möglich. Für den mobilen Einsatz benötigt man einen Akustikkoppler: Ein Modem mit zwei Gumminäpfen, in die Ohr- und Sprechmuschel des Telefonhörers eingesteckt werden. Nähme man während der Übertragung den Hörer ab, dann könnte man den Informationsfluß als Tonfolge hören.

Acronym = Akronym

BASIC ist ein Akronym, genau wie PET oder FIFO, EPROM, RAM und EMUF. Ein solches „Initialwort" wird aus den Anfangsbuchstaben der Wörter einer Bezeichnung gebildet (z. B. EMUF = Einplatinen-Mikrocomputer, Universell Festprogrammierbar). Akronyme sind in der Computerbranche sehr verbreitet, u. a. als Herstellerbezeichnungen, wobei oft der Verdacht naheliegt, daß erst das Akronym da war und nachträglich der Text dazu erfunden wurde – wie käme man sonst wohl auf Konstruktionen wie „Beginners All-purpose Symbolic Instruction Code".

ADA = ADA

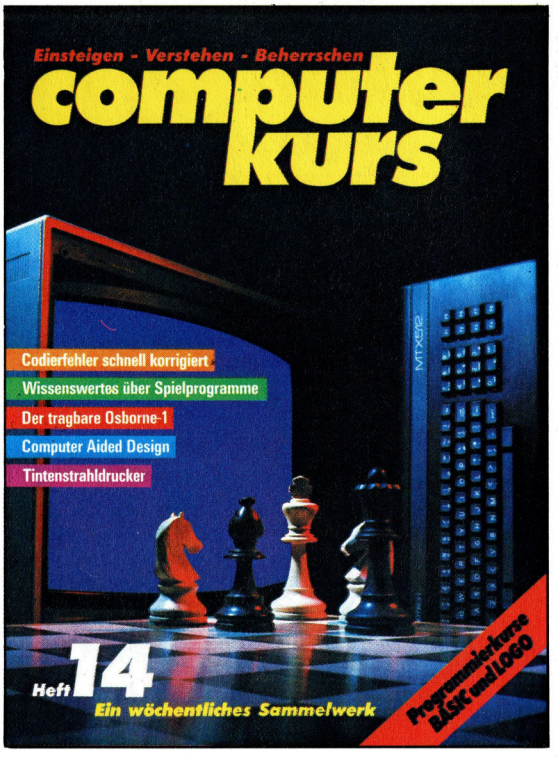
Ende der siebziger Jahre wurde von der Firma Honeywell-Bull eine neue Programmiersprache konzipiert – hauptsächlich für das Pentagon, wo sie die dortige Sprachvielfalt weitestgehend maschinenunabhängig ablösen sollte. Die Sprache ist sehr stark strukturiert. Die Namensgebung erfolgte zu Ehren der Gräfin Ada Lovelace, einer engen Mitarbeiterin des englischen Rechner-Konstrukteurs Charles Babbage.

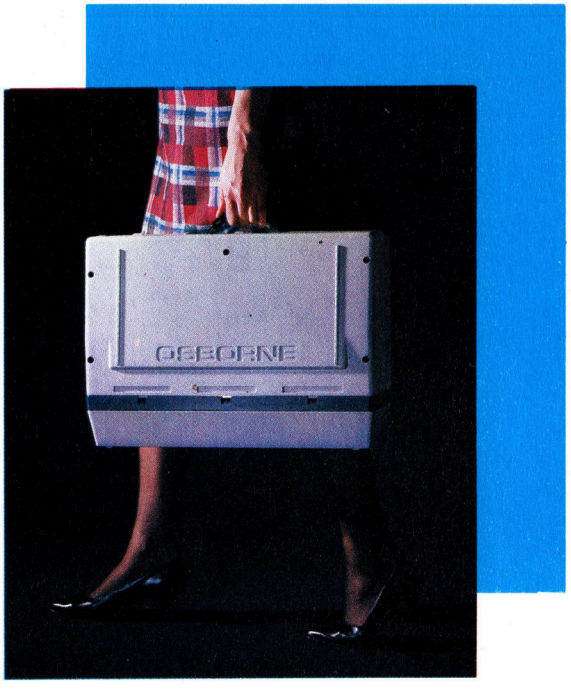
Bildnachweise

337: Columbia Warner
338: B. F. I. Stills
339: BBC Stills
339: United Artists
340, 341, 364: Ian McKinnell
348: Siemens-Museum, München
348: The Imperial War Museum
349: Chris Stevens
353: Brian Morris
354: Marcus Wilson-Smith
355: John Drysdale/Colorific
357, 358, 359: Liz Dixon
360: Roy Ingram
361: Martin Burke
362: Kevin Jones

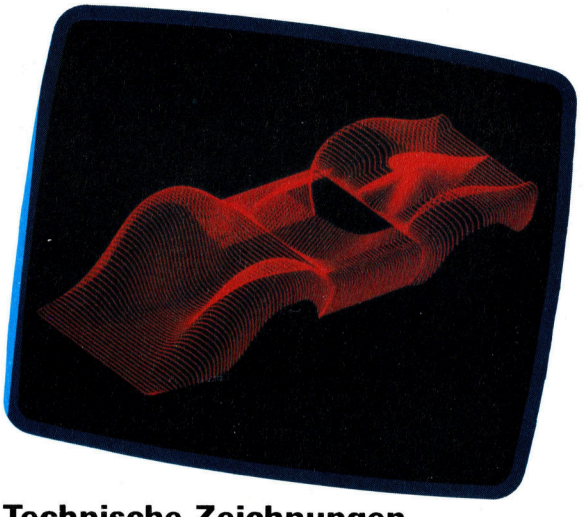
+++ Vorschau +++ Vorschau +++ Vorschau +++

computer kurs Heft 14

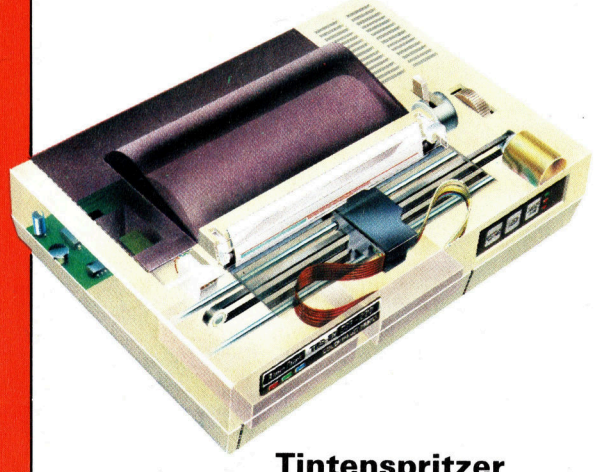




Tragbares System
Rechner, zwei Floppy-Laufwerke und ein Bildschirm verbergen sich in diesem Koffer: der Osborne-1.



Technische Zeichnungen
bereiten mit dem „Computer-Aided-Design" (CAD) keine Probleme mehr. Auch für Heimcomputer gibt es CAD-Programme.



Tintenspritzer
sind Drucker, bei denen die Schreibflüssigkeit durch kleine Düsen auf das Papier gebracht wird. Diese Geräte sind außerordentlich schnell und können mit verschiedenen Farben ausgerüstet werden.

+++ Abenteuerspiele +++ Fortsetzung der BASIC- und LOGO-Kurse +++ Programm-Optimierungen +++ Variabler Chip +++ Codier-Fehler finden +++ Wie Spielprogramme aussehen +++ Töne