Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 15

Tips für die Praxis

Schnelle Sortierung

Der Laser-Speicher

Der berechenbare Zufall

BASIC: Verzweigungen

Auf Fehlersuche

Schneider CPC 464

Ein wöchentliches Sammelwerk

computer kurs

Heft 15

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 wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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.

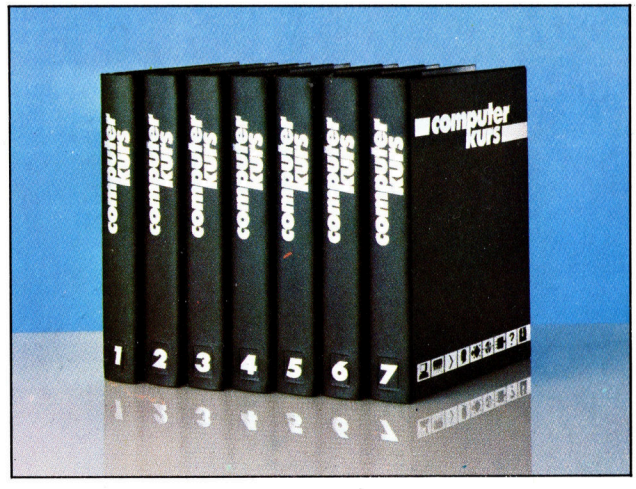
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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, Uta Brandl (Layout), Sammelwerk Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1, Tel. 040/23 40 85



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

Kalkuliertes Risiko

Wer sich genau auskennt, findet beim Pferderennen ein neues Einsatzgebiet für den Heimcomputer. Einige Züchter verwenden Computer, um die Abstammung der Pferde zu speichern – und um eine mögliche Gewinnchance zu ermitteln.

Rechner machen den Zufall berechenbarer: Schon heute gibt es zahlreiche Glücksspiel- und Wettprogramme für Heimcomputer.

Beim Glücksspiel steht die Wahrscheinlichkeit im Mittelpunkt, obwohl natürlich jeder Spieler sagt, daß es nur ums Gewinnen geht. Das stimmt nicht ganz, denn die Mehrzahl aller Spieler verliert sehr oft, zum Teil beträchtliche Summen. Das liegt daran, daß die wenigsten Glücksspiele „gerecht" sind und meistens nur der Buchmacher bzw. das Casino gewinnt. Um zu entscheiden, ob ein Computer diese Ungleichheiten aufheben kann, bedarf es einer Analyse.

Bei den meisten Spielen handelt es sich um Wetten auf ein künftiges, noch nicht bekanntes Ergebnis. Wenn aber beispielsweise bei einem Kartenspiel die Randbedingungen, also die Anzahl der Karten, bekannt sind, kann eine Aussage über den wahrscheinlichen Spielausgang getroffen werden. So hat das Roulette 37 Felder, die mit den Zahlen von 0 bis 36 numeriert sind. Außer der Null gibt es 18 gerade und 18 ungerade Zahlen, auf welche die Kugel fallen kann. Die Wahrscheinlichkeit, eine ungerade Zahl zu treffen, liegt daher bei 18 zu 37, etwas mehr als 48,6 %. Die Abweichung von einer reinen 50:50-Chance stellt den Profit des Casinos sicher.

Diese Gewinnmarge beschränkt bei jedem Glücksspiel den Erfolg des Spielers – zumindest auf lange Sicht. Trotz anderslautender Behauptungen, die Verkäufer von Spielsystemen gern in die Welt setzen, kann der Computer keine reale Gewinnchance errechnen. Professor Hans Sagan, eine Autorität auf diesem Gebiet, hat berechnet, das langfristig nicht ein einziges Casino-Spiel zu einem Gewinn führt – nur Blackjack bildet vielleicht eine Ausnahme.

Spielbegeisterte Programmierer haben sich von dieser Theorie nicht abschrecken lassen, so daß dem Besitzer eines Heimcomputers heute eine Vielzahl „narrensicherer" Spielsysteme angeboten werden, von denen einige sogar zu funktionieren scheinen. Bei Casino-Spielen beruhen diese meist auf der Methode des verdoppelten Einsatzes nach verlorenem Spiel – wozu für das sichere Gelingen ein unendlich großer Vorrat an Spielgeld vorhanden sein müßte. Zudem besteht ein weiteres Problem: Kein Casino gestattet den Einsatz von Computern, obwohl dies nicht unbedingt eine Verletzung der Regeln wäre. Man mißtraut dem Rechner, nicht so sehr jedoch als Hilfe für Strategie und Geschicklichkeit. Größer sind seine Fähigkeiten, dem Spieler die nötige „Spieldisziplin" zu verleihen und als eine Art Gedächtnisstütze zu wirken. Allerdings ist der Wert einer solchen Hilfe um so kleiner, je geübter der Spieler ist.

Geschicklichkeit und Erfahrung sind bei Glücksspielen nur selten von Bedeutung. Eine Ausnahme ist das Fußball-Toto. Von Professor Frank George stammt die F4-Fußball-Vorhersage, die man inzwischen für fast jeden bekannteren Heimcomputer bekommen kann. Sie stützt sich auf eine zehnjährige statistische Analyse. Die aus der Liga-Plazierung ermittelte langfristige Leistung einer Mannschaft wird mit den letzten Ergebnissen verglichen und bewertet. Daraus berechnet das Programm das wahrscheinliche Ergebnis des kommenden Spiels.

Beim Aufeinandertreffen einer Provinz- und einer Profi-Mannschaft ist die Berechnung meist überflüssig – interessanter wird es bei der Bewertung beinahe gleichstarker Kontrahenten. Die Genauigkeit der Vorhersage ist natürlich trotzdem beschränkt, immerhin verdreifacht sie jedoch, statistisch gesehen, die Trefferwahrscheinlichkeit. Professor George sagt dazu: „Natürlich sind die Chancen zu verlieren nach wie vor groß; aber sollte man nicht so intelligent spielen wie möglich?" All dieser Hilfsmittel zum Trotz halten sich die Erfolge in Grenzen. Ein englischer Toto-Veranstalter teilte mit, daß bisher kein größerer Preis von einem Computer-Spieler gewonnen worden sei. „Wenn es wirklich ein funktionierendes System gäbe, wir wüßten es als erste. Aber – es gibt keins!" Soweit Tony Hodge, Sprecher des Spiel-Konzerns. Obwohl für die Registrierung der Spielscheine Spezialmaschinen eingesetzt werden, geht der Computer-Einsatz des Unternehmens über die Aufzeichnung von Rekorden nicht hinaus.

Pferderennen können den Programmierer da schon mehr reizen. Ein Schüler aus England, dem klassischen Land des Pferderennens, entwickelte ein Programm für den Sinclair ZX 81 (inzwischen verbessert für den Spectrum), das einige richtige Vorhersagen lieferte. Zwar übertragen lokale BBC-Stationen die Renntips des Programms, sein Erfinder David Steward hatte bisher aber noch kein Glück damit.

Bis heute bleiben die Profis der Rennsport-Wetten bei der Bewertung der Pferde der traditionellen „Kopfarbeit" verhaftet, wenn auch die Ergebnisse schon im Rechner gespeichert werden. Da machen die „Bibeln der Pferde-Wetter", ihre Magazine, keine Ausnahme: „Den Computer nutzen wir nur zur Berechnung der normalen Rennzeit für jedes Pferd, wobei der Windwiderstand berücksichtigt wird", erklärt der Fachmann – „eine genaue Bewertung mit dem Rechner gibt es nicht!".

Die „Zero" auf dem Roulett sichert dem Casino seinen Gewinn. Die Chancen des Spielers sind auch durch den Computer nicht zu verbessern; ein Programm kann also höchstens ein Wettsystem sein.

Zur Unterstützung der Totospieler gibt es eine Vielzahl von Programmen, die angeblich die Chancen vergrößern. Gute Systeme sind auf die Daten der früheren Spiele angewiesen und können bei Vorausscheidungen nützlich sein. Computergestütztes Spiel kann jedoch nur eine kleine Hilfe auf dem Weg zum Hauptgewinn sein. Die Hersteller der Programme lehnen jedenfalls ein Garantieversprechen mit Nachdruck ab.

Bargeldloses Wetten

Dagegen setzen die größeren Buchmacher-Ketten immer mehr Computer ein – nicht für die Chancen-Berechnung, sondern zur Erleichterung des bargeldlosen Wettgeschäfts. Dadurch kann ein Spieler seinen Tip telefonisch plazieren, die Daten und der Einsatz werden registriert und das eingesetzte Geld vom Konto des Spielers abgebucht. Gewinnt er, erfolgt auch die Gutschrift vollautomatisch.

Buchmacher stehen dem Computer eher skeptisch gegenüber: „Kein System garantiert dauerhafte Gewinne, sonst wären wir nicht mehr da" – so lautet eine häufig geäußerte Meinung. Sie wird auch vom Sprecher der Buchmacher-Firma William Hill geteilt, obwohl dieses Unternehmen eine außergewöhnliche und umstrittene Renn-Simulation per Computer veranstaltete: Ein Programm wurde mit den Eigenschaften der weltbesten Rennpferde aller Zeiten gefüttert. Die Leser einer Zeitung sollten die sechs besten Pferde vorhersagen. Später gab es dann Streit: Niemals hatte das berühmte italienische Pferd „Ribop" ein Rennen verloren. Der Computer ließ es aber an vierter Stelle durchs Ziel gehen!

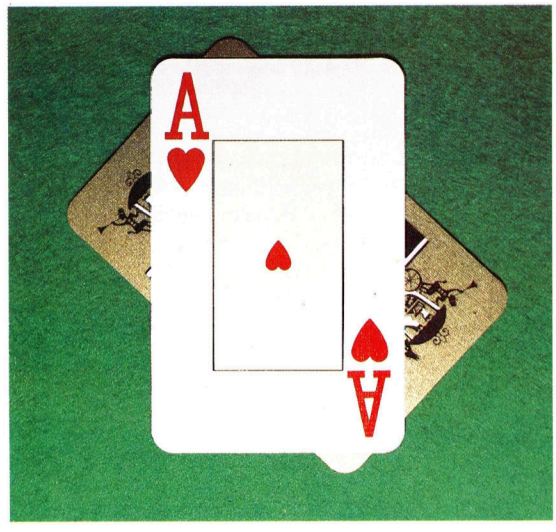
Weitaus populärer ist in England der Com-

puter ERNIE (Electronic Random Number Indicator Equipment), der die Zahlen eines lottoähnlichen Gewinnspiels ermittelt. ERNIE ist nicht programmierbar und eigentlich kein „echter" Computer, aber er läßt ein festes Programm ablaufen. Die Maschine erzeugt 200 000 Zufallszahlen, die den Seriennummern aller je verkauften Lose entsprechen. Danach werden die Zahlen mit Nummern verglichen, die schon früher gewonnen haben und nicht „mitspielen". Nach dieser Sortierarbeit druckt das Gerät auch noch die Preiszertifikate und benachrichtigt die Gewinner.

Seit der Inbetriebnahme haben die beiden ERNIEs 22,2 Millionen Preise im Gesamtwert von 1.181.843.400 Pfund verliehen, also fast 5 Milliarden DM. Das hört sich gut an, trotzdem liegt die Gewinnchance innerhalb eines Monats nur bei 1:15 000.

Ein anderes Glücksspiel wird von den englischen Zeitungen angeboten: Preise in Millionenhöhe sind ausgesetzt, doch es ist unwahrscheinlich, daß sie je ausbezahlt werden müssen: Die Mitspieler bekommen Karten mit einer zwölf Ziffern langen Zahlenreihe. Damit gibt es eine Million mal eine Million Kombinationen für den Hauptgewinn. Zwar publizieren die Zeitungen zwei Zahlen täglich, aber es scheint, die Verleger gehen auf Nummer Sicher – den Hauptgewinn werden sie wohl nicht so bald „berappen" müssen.

Bildlich kann man sich die Chancen als einen Sack, gefüllt mit 2,5 Millionen weißen Kugeln (für die Teilnehmer) und einer Billion schwarze Kugeln (für die möglichen Zahlenkombinationen), vorstellen. Man muß nicht besonders betonen, daß es sehr unwahrscheinlich ist, beim zufälligen Herausnehmen einer Kugel auf eine weiße zu stoßen. Auch die vielen Zahlenkugeln, die im Laufe eines Jahres gezogen werden, verbessern die Chancen nur unwesentlich. Statistiker haben mit dem Computer ausgerechnet, daß der Hauptgewinn in Höhe von einer Million englischen Pfund nur einmal in 667 Jahren fällig wird.



Blackjack-Programme gibt es für die meisten Heimcomputer. Diese eignen sich noch am besten für das computergestützte Spiel, weil die schon gespielten Karten gespeichert werden können. Da Casinos den Gebrauch von Computern nicht gestatten, versuchen hartnäckige Spieler ihr Glück mit versteckten Geräten oder auch durch Funkverbindungen zu entfernt untergebrachten Geräten.

Würfel programmieren

Die Zufallszahlenerzeugung, Grundlage jeden Glücksspiels, können Sie auf Ihrem Rechner ganz leicht programmieren. Die meisten BASIC-Versionen verfügen über einen Zufallszahlengenerator. Manchmal sind die erzeugten Zahlen jedoch nicht tatsächlich zufällig:

```
10 LET A = RND
20 LET B = RND
30 LET C = RND
40 PRINT A,B,C
```

In den ersten drei Programmzeilen werden den Variablen A,B und C Zufallszahlen zugeordnet. Zeile 40 gibt die Werte aus. Das sieht dann etwa so aus (Ihr Rechner zeigt natürlich andere Zahlen):

```
.014007 .964370 .457397
```

Wenn Sie jetzt das Programm noch einmal starten, wiederholt sich bei fast allen Computern dieselbe Zahlenfolge. Das liegt daran, daß der Rechner auf das Kommando RND mit der ersten Zahl aus einer festgelegten Reihe beginnt. Die Zahl ist meist kleiner als die 1 und hat sechs Stellen, stammt also aus dem Bereich von 0,000000 bis 0,999999. Das ergibt eine Million verschiedener Möglichkeiten, die bei RND jeweils einmal auftauchen – natürlich nicht in der „richtigen" Reihenfolge.

Manche BASIC-Versionen arbeiten nacheinander an deren Syntax, in der ein Klammerausdruck nötig ist. Er wird als das „Argument" bezeichnet. Dann heißt es LET A = RND (X). Der Effekt ist derselbe: RND und RND(X) werden genau wie andere Variablen eingesetzt.

Um nun einen Würfel zu simulieren, brauchen Sie ganze Zahlen zwischen 1 und 6, Bruchzahlen müssen mit Hilfe der INT (Integer)-Funktion gerundet werden. Man erhält zum Beispiel nach PRINT INT (6,99) ebenso eine 6 im Display wie nach PRINT INT (6,01). Werte hinter dem Dezimalpunkt werden einfach weggelassen.

RND erzeugt als größte Zahl 0,999999, die mit INT ausgedrückt nur den Wert 0 ergibt. Daher muß vor INT noch manipuliert werden. Die Anweisung dafür lautet:

```
LET A = INT(6 * RND)+1
```

Da der Würfel sechs Seiten hat, wird mit sechs multipliziert. Das „plus Eins" veranlaßt, daß die Ergebnisse zwischen 1 und 6, nicht aber zwischen 0 und 5 liegen.

Mark I-Triumph

An der Hochschule von Manchester wurde der erste programmierbare Rechner der Welt entwickelt.

Kurz nach Ende des Zweiten Weltkrieges stellte die Universität Manchester zwei neue Professoren ein: Max Newmann, der vorher am ersten elektromechanischen Computer „Colossus" in Bletchley Park mit Decodieraufgaben betraut war, erhielt den Titel „Professor der Mathematik". Der Radartechniker F. C. Williams wurde neuer Leiter des Bereichs Elektrotechnik. Sein Assistent Tom Kilburn, der ihn nach Manchester begleitete, hatte im Krieg bereits an elektronischen Speichereinheiten für Radaranlagen gearbeitet. Kilburn wurde später erster Professor der neuen Abteilung „Computer-Technik".

Während seiner Studienreise in die Vereinigten Staaten wurde Williams 1946 ein Prototyp des Röhren-Computers „ENIAC" vorgeführt. Nach seiner Rückkehr brachte er die englische „Royal Society" dazu, 35 000 Pfund in das neue Labor für Rechenmaschinen zu investieren. Im Rennen um neue Computer, die ein Programm speichern konnten, stand seine Hochschule nicht allein: In Pennsylvania wurde bereits am EDVAC, in Cambridge am EDSAC gearbeitet. Auch das Nationale Physikalische Labor beteiligte sich mit seinem ACE-Programm. Das Team der Universität von Manchester verwendete jedoch einen von Williams konstruierten Speicher mit Kathodenstrahlröhre, während die Konkurrenten mit Quecksilberröhren arbeiteten. Bereits im Herbst 1947 erreichte man, daß 2048 Bits über mehrere Stunden gespeichert werden konnten.

Erstes Programm 1948

Auf dem „Manchester Mark I", dem ersten programmierbaren Computer der Welt, lief dann 1948 das erste Programm. Für einen Rechenschritt brauchte der „Mark I" nur 1,2 Millisekunden. Durch die Verwendung der Kathodenstrahlröhre (CRT) konnte der Inhalt des Hauptspeichers und der verschiedenen Register dargestellt werden. Zudem war der Zugriff auf einzelne Speicherzellen unabhängig vom Zustand der anderen Zellen.

Nachdem die Brauchbarkeit erwiesen war, wurde der Mark I für optische Berechnungen und zur Primzahl-Erzeugung weiterentwickelt. Die Vorführung dieses Gerätes beeindruckte den wissenschaftlichen Inspektor der Regierung Sir Ben Lockspeiser so sehr, daß er für die kommerzielle Produktion durch eine Firma in Manchester sorgte. Fünf Monate vor der Fertigstellung des UNIVAC-Rechners konnte man dann im Februar 1951 mit dem „Ferranti Mark I" den ersten kommerziellen Computer der Welt kaufen.

Befehlsänderungen möglich

Neu am Mark I war die Möglichkeit, Befehle während des Rechenvorganges zu ändern und in den Programmspeicher zu übertragen – die Verarbeitungsgeschwindigkeit nahm zu. Für ihre ersten Rechner übernahm später IBM einige der Manchester-Patente. Als Williams das IBM-Gebäude in New York besuchte, stieß er überall auf das Firmenmotto „THINK" (DENKE). Später wurde ihm die Frage gestellt, wie sein Team die Aufgaben lösen könnte, bei denen selbst IBM passen mußte. Ein schnelles „Wir haben uns eben nicht die Zeit genommen, viel zu denken" war seine Antwort.

1948 kam Alan Turing, der schon 1959 das erste Handbuch zum Programmieren herausgeben sollte, nach Manchester. Zwei Jahre später plante die Gruppe dann einen kleinen, leistungsfähigen Rechner, ein Vorhaben, das durch die Einführung des Transistors vorangetrieben wurde. Im November 1953 arbeitete in Manchester der erste Transistor-Computer der westlichen Welt.

In den späten 50er Jahren versuchte die englische Industrie noch einmal, den Amerikanern die Spitzenposition in der Computertechnologie streitig zu machen. Tom Kilburns Team entwickelte mit Unterstützung der Regierung den „Atlas". Der Rechner mit 48-Bit-Worten, 16 KByte-Speicher und einem 8 KByte-Trommelspeicher war seiner Zeit weit voraus und blieb lange ein Spitzenreiter auf der „Computer-Hitliste". Sogar die British Petroleum (BP) und das Atomenergie-Forschungszentrum in Harwell arbeiteten lange damit.

Das erste Programm lief im Juni 1948. Damit war der Manchester Mark I der erste programmierbare Computer. Die damals noch kleine Firma Ferranti erhielt den Auftrag, aus dem Mark I eine kommerzielle Version zu entwickeln. Diese kam im Frühjahr 1951 auf den Markt.

Schneider CPC 464

Dieser Rechner verfügt über mehr Fähigkeiten als die meisten Microcomputer, kostet aber vergleichsweise wenig. In diesem „Computerpaket" sind standardmäßig ein integrierter Cassettenrecorder und ein Datenmonitor enthalten.

Der in England entwickelte Amstrad CPC 464 wird in Deutschland unter dem Namen Schneider CPC 464 angeboten und in zwei verschiedenen Konfigurationen ausgeliefert. Beide bestehen aus dem gleichen Grundgerät, sind aber mit unterschiedlichen Monitoren ausgerüstet: einem einfarbigen Bildschirm oder einem RGB (Farb-)Monitor. Das Netzteil des Computers befindet sich im Gehäuse des Monitors und versorgt das Hauptgerät über ein Verbindungskabel mit Strom. Da der Cassettenrecorder ebenfalls in den Computer eingebaut ist, führt die gesamte Stromversorgung über ein einziges Kabel.

Der Monochrom-Monitor erzeugt ein äußerst klares Bild, das sich gut für den kommerziellen Gebrauch und die Textverarbeitung eignet. Der Farbmonitor verfügt nur über Grafik mit mittlerer Auflösung. Er kann zwar volle Farbgrafik anzeigen, nicht aber alle 80 Zeichen in lesbarer Form.

Der Schneider CPC 464 ist mit einer Schreibmaschinentastatur und separatem Zehnerblock ausgerüstet. Alle Zeichen der Tastatur lassen sich umdefinieren; ferner können die Tasten des Zehnerblocks als frei programmierbare Funktionstasten eingesetzt werden. Jeder Taste kann ein Befehl bzw. eine Befehlsfolge von bis zu 32 Zeichen Länge zugeordnet werden. So läßt sich über einen einzigen Tastendruck z. B. ein Programm laden und listen.

Der integrierte Cassettenrecorder entspricht dem normalen Standard. Soll ein Programm geladen oder gespeichert werden, schaltet der Computer nur den Motor des Recorders ein oder aus. Der Schneider bietet zwei mögliche Ladegeschwindigkeiten, 1000 und 2000 Baud, die über Software ausgewählt werden können. Beim Laden eines Programms testet der Computer automatisch, mit welcher Geschwindigkeit das Programm gespeichert wurde und stellt sich entsprechend ein.

Der CPC 464 ermöglicht auch vielfältige Anschlußmöglichkeiten für Peripheriegeräte. Über eine einfach konstruierte Steckleiste läßt sich ein Centronics-Drucker mit dem Computer verbinden. Auch für Diskettenlaufwerke ist eine Schnittstelle vorgesehen, mit der dem Computer noch zusätzliche externe Speicherkapazitäten zur Verfügung gestellt werden sollen. Für die nähere Zukunft sind die Programmiersprache LOGO und das CP/M-Betriebssy-

Die preisgünstige Version des Schneider CPC 464 wird mit einem Monochrom-Monitor ausgeliefert; aber auch mit Farbmonitor kostet das Gerät noch weniger als 1400 DM. Die beiden hier gezeigten Programme sind die Varianten des „Hangman" und „Admiral Graf Spee".

stem angekündigt. An der Rückseite des Gerätes befindet sich ein Anschluß für Atari-kompatible Joysticks. Für Spiele, bei denen zwei Joysticks benötigt werden, liefert Schneider einen Zweierset.

Der eingebaute Lautsprecher hat einen eigenen Lautstärkeregler. Die Signale der drei Tongeneratoren können auch über einen externen Verstärker in Stereo an HiFi-Lautsprecher ausgegeben werden. Töne lassen sich nicht nur in der gewünschten Tonhöhe und Lautstärke ein- und ausschalten, auch ihre Hüllkurve kann festgelegt werden. Der Lautstärkeverlauf eines Tones läßt sich auf den Sound z. B. eines Klaviers oder einer Glocke einstellen, während unabhängig davon Klangeffekte wie z. B. eine Sirene oder Pfeifen über die Tonhöhe gesteuert werden können.

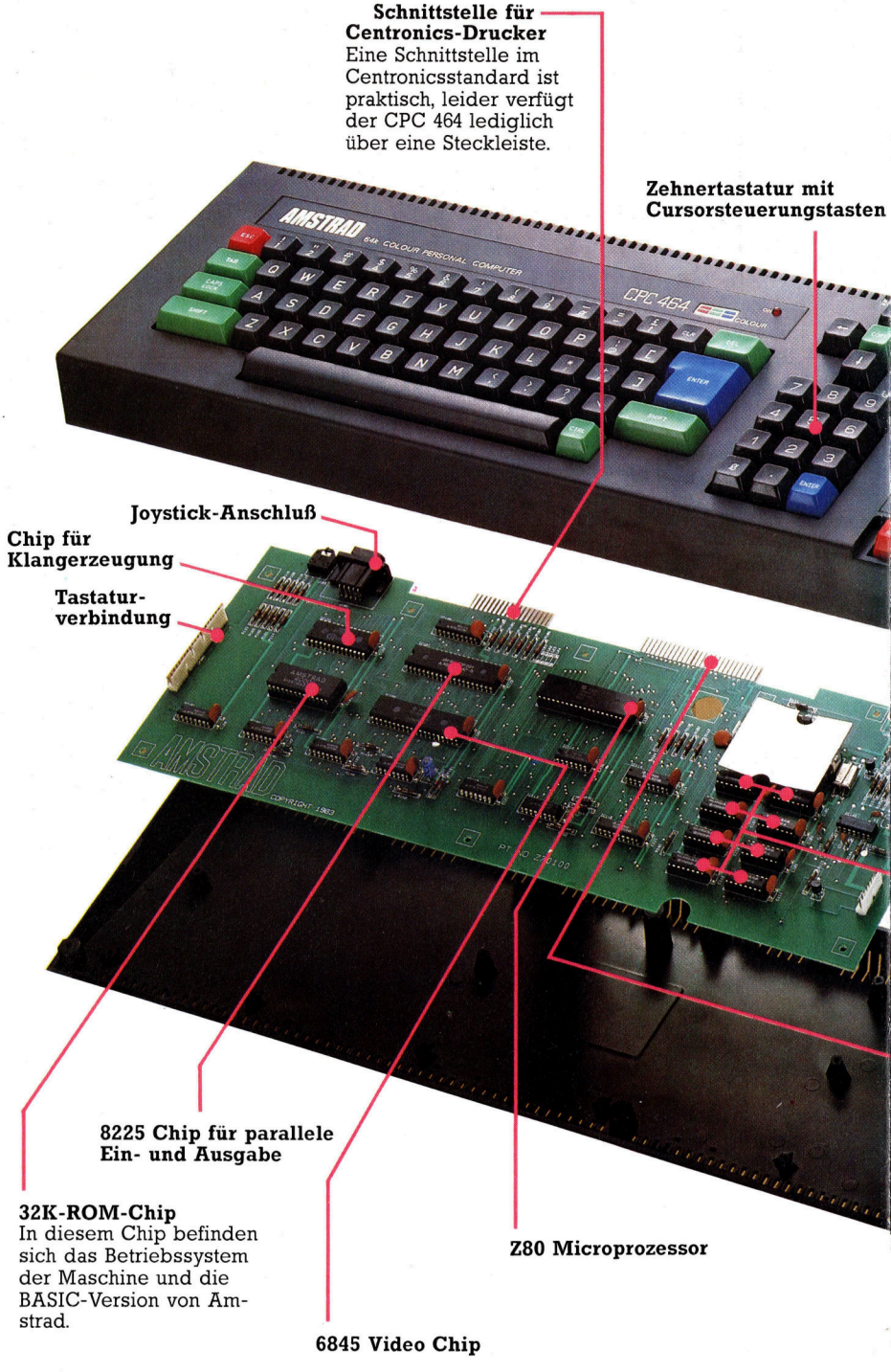
Die Grafikmöglichkeiten des CPC 464 sind beeindruckend. Er verfügt über drei Darstellungsarten, von denen jede eine unterschiedliche Anzahl Zeichen und Farben auf den Bildschirm bringt. In der höchsten Auflösung können zwei Farben gleichzeitig auf dem Bildschirm (Vorder- und Hintergrund) dargestellt werden. Pro Bildschirmzeile stehen dann 80 Zeichen zur Verfügung. Die höchste grafische Auflösung beträgt 640 × 200 Bildelemente. Am anderen Ende der Skala können 16 Farben gleichzeitig erscheinen, allerdings nur mit 20 Zeichen pro Zeile. Die dritte Darstellungsart verfügt über vier Farben und 40 Zeichen pro Zeile auf dem Schirm.

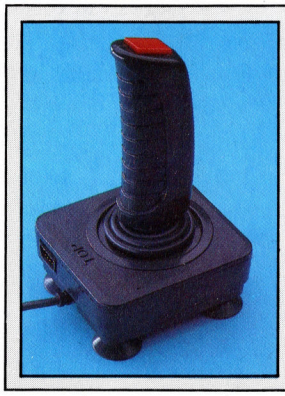
27 Farbtöne

Wenn auch die Anzahl der Farben, die sich gleichzeitig auf den Schirm bringen lassen, begrenzt ist, steht doch eine Palette von 27 Farbtönen zur Verfügung. Jede der Farben kann in zwei Abstufungen mit verschiedenen Geschwindigkeiten blinken. Der Schneider eignet sich sowohl für feststehende Abbildungen mit vielen Einzelheiten als auch für die Darstellung von Bewegung.

Obwohl der Bildschirmspeicher mit seinen 16 KByte einen großen Teil des Arbeitsspeichers einnimmt, wird dadurch der verfügbare RAM-Bereich nicht kleiner. Der Bildschirmspeicher und das ROM des BASIC liegen in der Belegungstabelle des Speichers auf dem gleichen Platz. Ein Spezialchip schaltet je nach Bedarf zwischen diesen beiden Funktionen um, so daß für Programme und Daten 42 KByte Speicherkapazität zur Verfügung stehen.

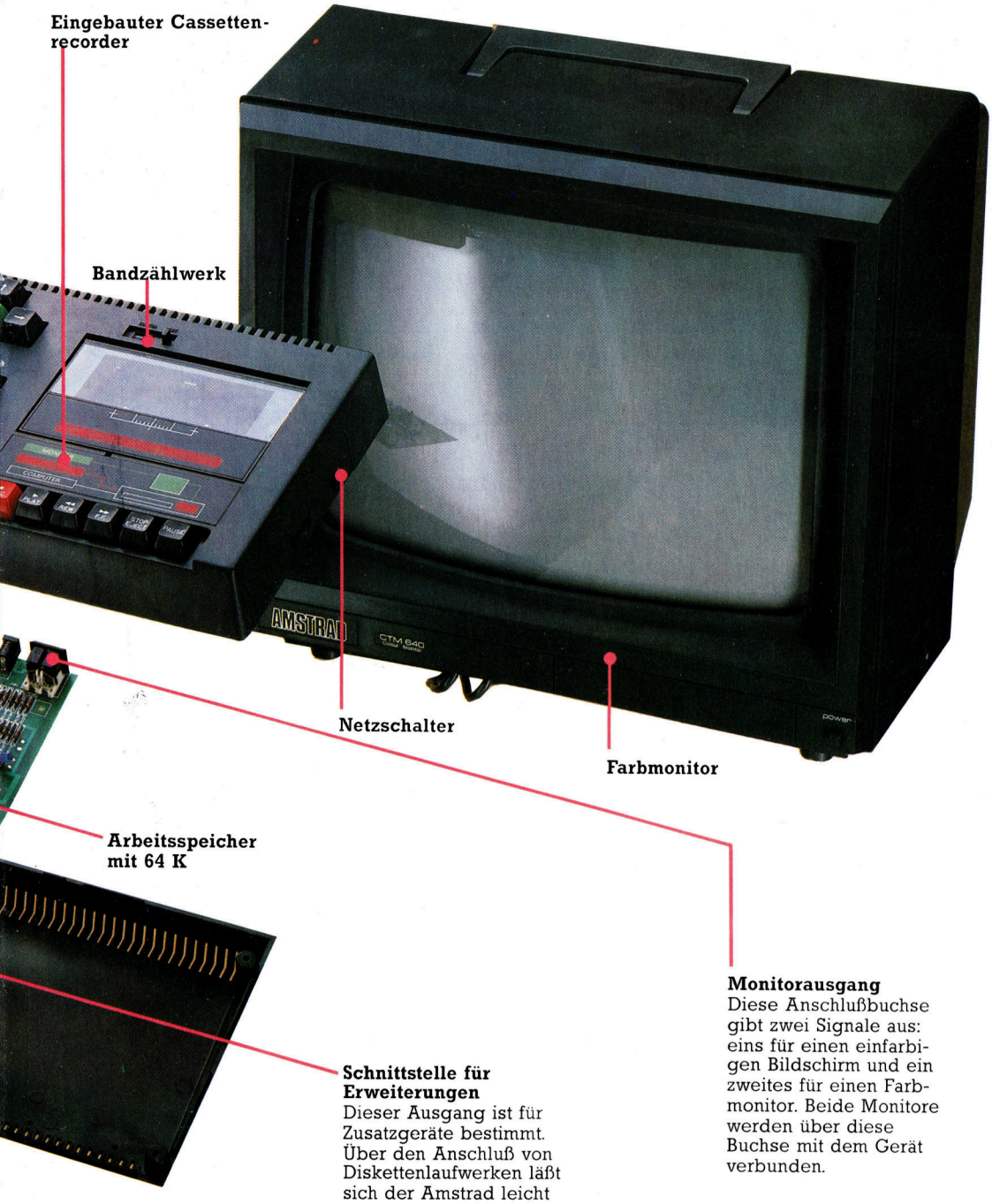
Das Amstrad-BASIC ist eine der höchstentwickelten Versionen dieser Sprache. Es unterstützt besonders die Grafikhardware, während Spezialfunktionen die Bildschirmdarstellung erleichtern. Der Nullpunkt der Grafik liegt in der linken unteren Ecke des Bildschirms, kann aber auf jeden beliebigen Punkt des Schirms gesetzt werden. Außerdem besteht die Möglichkeit, ein Fenster zu definieren, das grafi-





Vielseitige Joysticks

An den Schneider kann ein Standard-Atari-Joystick angeschlossen werden. Die Firma bietet jedoch ein eigenes Set an, mit dem sich zwei Joysticks gleichzeitig betreiben lassen. Die Signale des zweiten Joysticks werden über den ersten geleitet.

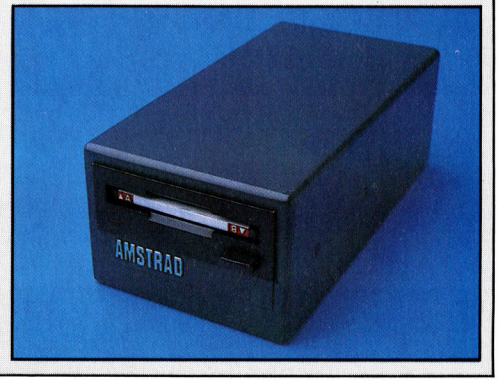


Monitorausgang
Diese Anschlußbuchse gibt zwei Signale aus: eins für einen einfarbigen Bildschirm und ein zweites für einen Farbmonitor. Beide Monitore werden über diese Buchse mit dem Gerät verbunden.

Schnittstelle für Erweiterungen
Dieser Ausgang ist für Zusatzgeräte bestimmt. Über den Anschluß von Diskettenlaufwerken läßt sich der Amstrad leicht in eine CP/M-Maschine verwandeln.

Künftige Erweiterung

Die englische Firma Amstrad verspricht sich von der Diskettenstation einigen Erfolg. Diese Erweiterung kostet etwa 800 Mark und schließt das CP/M-Betriebssystem mit ein, auf dessen Grundlage die meisten kommerziellen Programme arbeiten.



Schneider CPC 464

PREIS
ca. 900 Mark mit Monochrom-Monitor, ca. 1400 Mark mit RGB-Monitor

ABMESSUNGEN
Tastatur/Cassette:
565 × 170 × 70 mm
Farbmonitor:
380 × 350 × 350 mm

CPU
Z80

SPEICHERKAPAZITÄT
64 KByte RAM, von denen 42 K für BASIC-Programme zur Verfügung stehen; 32 KByte ROM

BILDSCHIRMDARSTELLUNG
Drei Darstellungsarten mit der Möglichkeit, Text und Grafik zu mischen:
640 × 200 Punkte (2 Farben),
320 × 200 Punkte (4 Farben),
160 × 200 Punkte (16 Farben).
27 Farben stehen zur Verfügung.

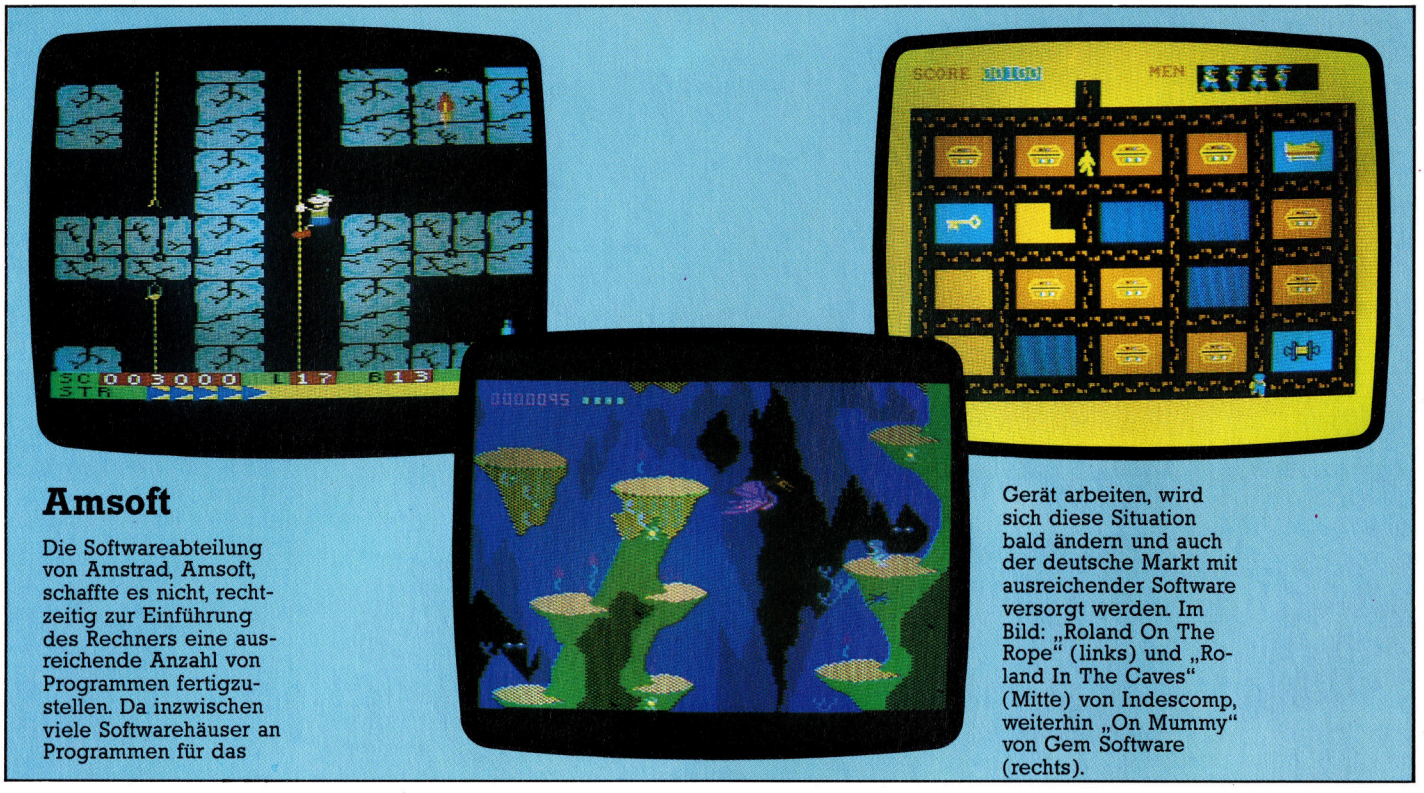
SCHNITTSTELLEN
Joystick, 2 Centronics-Druckeranschlüsse, Erweiterungssteckleiste für Diskettenlaufwerke, Stereoausgang, Monitoranschluß

PROGRAMMIERSPRACHEN
BASIC (eingebaut) und PASCAL (auf Cassette)

TASTATUR
Schreibmaschinentastatur mit 74 Tasten

DOKUMENTATION
Das Bedienerhandbuch für Anfänger ist einfach zu verstehen. Außerdem stehen Nachschlagewerke für BASIC und den technischen Aufbau zur Verfügung.

STÄRKEN
Der Schneider verfügt über einen großen Speicherbereich, ausgezeichnete, flexible Grafik und hervorragende Möglichkeiten der Tonerzeugung in Stereo.



Amsoft

Die Softwareabteilung von Amstrad, Amsoft, schaffte es nicht, rechtzeitig zur Einführung des Rechners eine ausreichende Anzahl von Programmen fertigzustellen. Da inzwischen viele Softwarehäuser an Programmen für das Gerät arbeiten, wird sich diese Situation bald ändern und auch der deutsche Markt mit ausreichender Software versorgt werden. Im Bild: „Roland On The Rope" (links) und „Roland In The Caves" (Mitte) von Indescomp, weiterhin „On Mummy" von Gem Software (rechts).

sche Darstellungen auf einen bestimmten Bereich des Bildschirms begrenzt. Bei Textdarstellung lassen sich bis zu acht Fenster gleichzeitig auf den Schirm zaubern.

Leider fehlt ein Grafikbefehl, mit dem sich ein begrenzter Bereich des Bildschirms mit Farbe füllen läßt. Da nur Striche und Punkte gezeichnet werden können, lassen sich keine Farbflächen darstellen und bestehende Umrahmungen auch nicht mit Farbe ausfüllen. Nur über viele dicht beieinanderliegende Linien kann ein Block farbig dargestellt werden.

Einzigartige Stellung

Auch andere ROMs, die an die Stelle des BASIC-ROMs gesetzt werden, können in die Speicherbelegungstabelle ein- und ausgeblendet werden, um Platz für den Bildschirmspeicher zu schaffen. Fähigkeiten, die dem Schneider jetzt noch fehlen, werden mit Sicherheit in zukünftigen ROMs vorhanden sein. Dies gilt auch für Sprachen wie PASCAL, FORTH und LOGO. Da diese „Zusatz"-ROMs nicht mehr Speicherplatz belegen würden als das bestehende BASIC, ständen auch dabei 42 KByte RAM für Programme und Daten zur Verfügung. Es ist auch möglich, zusätzlichen Speicherplatz in die Belegungstabelle zu integrieren, um den Standard von 64 KByte RAM zu erweitern.

Der originellste Aspekt des Amstrad-BASIC ist seine Behandlung von „Interrupts" (Unterbrechungsroutinen). Die meisten Computer sind so konzipiert, daß der Programmierer die Möglichkeit hat, seine Maschinenprogramme über die Interrupt-Routinen des Betriebssystems laufen zu lassen. Amstrad hat diese Idee direkt auf die BASIC-Version übertragen. So verzweigt der BASIC-Befehl AFTER nach einer festgelegten Zeit auf ein bestimmtes Unterprogramm, während EVERY diesen Vorgang ständig wiederholt. Mit diesen Möglichkeiten läßt sich jede Art von zeitabhängigen Programmen von der Datenabfrage für Laborgeräte bis zu Arcadespielen problemlos programmieren.

Eine einzigartige Stellung unter den Heimcomputern nimmt der CPC 464 schon allein durch den mitgelieferten Monitor ein. Leider gibt es für die Geräteversion mit Monochrom-Monitor nicht die Möglichkeit, einen Farbfernseher anzuschließen, obwohl ein entsprechender Adapter verfügbar ist. Es läßt sich zwar ein zusätzlicher Farbmonitor ohne Adapter anschließen, aber da der Schneider seine Elektrizität über das im Monitor eingebaute Netzteil bezieht, müssen in diesem Fall beide Monitore gleichzeitig laufen.

Mit seinen hervorragenden Grafikfähigkeiten, seiner soliden Bauweise, dem hochentwikkelten BASIC und den Erweiterungsmöglichkeiten ist der Schneider CPC 464 einer der ausgefeiltesten Heimcomputer, die der Markt im Augenblick bietet.

Ausschlaggebend für den derzeitigen Erfolg des Schneider CPC 464 ist der sehr günstige Preis. Dieser Erfolg wird sich sicher noch weiter fortsetzen, wenn erst genügend Software am Markt ist; neue Programme für den CPC 464 sind in Vorbereitung. Der Hersteller selbst bemüht sich ebenso: Gerade ist ein passendes Floppy-Laufwerk auf den Markt gekommen, mit einem Preis unter achthundert Mark.

Shell-Sortierung

Die Shell-Sortierung arbeitet nach dem Prinzip der Kettenbildung. Bei großen Datenmengen eignet sich dieses Verfahren besser als die Bubble- oder Vergleichssortierung.

Das nach seinem Erfinder D. Shell benannte Verfahren hat den Vorzug, daß die ungeordneten Informationen in einem Datenfeld frühzeitig geordnet werden und die einzelnen Daten nicht weit von ihren wirklichen Positionen entfernt sind. Außerdem bietet dieses Verfahren die Möglichkeit, Daten über relativ lange Strecken auszutauschen. Die Anwendung ist bei großen Datenmengen sinnvoll, die nicht mit einem Blick überschaut werden können. Um zu sehen, wie die Methode arbeitet, genügt jedoch das folgende einfache Beispiel „Sortieren von Spielkarten":

1) Legen Sie alle Karten einer Farbe in gemischter Reihenfolge aus. Diese Karten sollen jetzt in fallender Ordnung (König ganz links, As ganz rechts) ausgelegt werden. Gehen Sie folgendermaßen vor: Zählen Sie die Karten (hier 13), und teilen Sie die Summe durch zwei. Das Ergebnis, in diesem Fall sechs, ist die „Kettenzahl".

2) Legen Sie über die äußerste linke Karte (Position Nr. 1 innerhalb der Kartenreihe) und über die Karte der Position Nr. 6 (dies entspricht der Kettenzahl) ein Markierungszeichen, zum Beispiel eine Münze. Jede Karte der Positionen eins bis sechs stellt nun eine „linke Eckkarte" der jetzt zu bildenden Ketten dar.

Karten positionieren

Bilden Sie nun die erste Kette: Beginnen Sie mit der Karte der Position eins und addieren Sie zur eins die Kettenzahl sechs, um die Position der nächsten Karte zu erhalten (Nr. 7). Addieren Sie nun zur sieben wieder sechs. Sie erhalten die Karte Nr. 13.

3) Sortieren Sie die Karten der ersten Kette nach ihrer Wertigkeit, und legen Sie diese auf die freien Plätze zurück.

4) Bilden Sie jetzt die zweite Kette auf die gleiche Weise: Position Nr. 2+6=8. Dann wieder sortieren und zurücklegen. Wiederholen Sie diesen Vorgang, bis sechs Ketten gebildet sind. In der ersten Kette finden sich die Karten Nr. 1, 7 und 13, in der zweiten Kette die Karten Nr. 2 und 8 und so weiter.

5) Teilen Sie die Kettenzahl nun durch 2 (6/2=3), und kennzeichnen Sie die Positionen eins, drei, sechs, neun und zwölf. Bilden Sie Ketten wie zuvor, diesmal jedoch mit der Kettenzahl 3.

6) Teilen Sie die Kettenzahl 3 nochmals durch zwei. Sie erhalten unter Vernachlässigung des Restes die Kettenzahl 1. Die jetzt vorliegende Kartenreihe ist folglich die letzte Kette. Sortieren Sie diese Kette nach dem Vergleichsverfahren. Die Kartenreihe muß jetzt geordnet sein, denn ein weiteres Teilen der Kettenzahl durch zwei würde eine Zahl kleiner als eins als Zeichen dafür ergeben, daß ein weiteres Ordnen nicht mehr nötig ist.

```
7999 REM**********************
8000 REM*        SHELL       *
8001 REM**********************
8025 PRINT "SHELL SORT - GO !!!!!"
8050 LET LK=LT
8100 FOR Z=0 TO I STEP 0
8150 LET LK=INT(LK/II)
8200 FOR LB=I TO LK
8250 LET LL=LB+LK
8300 FOR P=LL TO LT STEP LK
8350 LET D=R(P)
8400 FOR Q=P TO LL STEP-LK
8450 LET R(Q)=R(Q-LK)
8500 IF D<=R(Q) THEN LET R(Q)=D:LET Q=LL
8550 NEXT Q
8600 IF D>R(LB) THEN LET R(LB)=D
8650 NEXT P
8700 NEXT LB
8750 IF LK=I THEN LET Z=I
8800 NEXT Z
8850 PRINT "SHELL SORT - STOP !!!!!"
8900 RETURN
```

Um diese Routine in das Sortierprogramm auf Seite 159 einzufügen, sind die Zeilen 350 und 900 folgendermaßen zu ändern:

```
350 LET I=1:LET 0=0:LET II=+:LET TH=3

900 ON SR GOSUB 6000,7000,8000
```

Shell-Sortiertafel

Position Nr.									Kettenzahl	Vorgang
1	2	3	4	5	6	7	8	9		
2	8	9	3	10	5	K	6	7	(9/2)=>4	Anfang Durchgang
*	+	@	$	*	+	@	$	*		Ketten bilden
10				7				2		Kette 1 sortieren
	8				5					Kette 2 sortieren
		K				9				Kette 3 sortieren
			6				3			Kette 4 sortieren
10	8	K	6	7	5	9	3	2		Anfang Durchgang
10	8	K	6	7	5	9	3	2	(4/2)=>2	Ende Durchgang
*	+	*	+	*	+	*	+	*		Ketten bilden
K		10		9		7		2		Kette 1 sortieren
	8		6		5		3			Kette 2 sortieren
K	8	10	6	9	5	7	3	2		Ende Durchgang
K	8	10	6	9	5	7	3	2	(2/2)=>1	Anfang Durchgang
*	*	*	*	*	*	*	*	*		Kette 1 bilden
K	10	9	8	7	6	5	3	2		Ende Durchgang

Das links gezeigte Listing für eine Shell-Sortierung ist in Verbindung mit dem Programm auf Seite 159 zu verwenden. Es stellt eine wesentliche Verbesserung gegenüber anderen Sortierverfahren dar, wenn mehr als 40 Daten zu ordnen sind.
Der Kasten zeigt eine Shell-Sortierung am Beispiel eines reduzierten Kartensatzes. Die Kartenreihe wird in Ketten aufgegliedert, deren Anzahl durch die jeweilige Kettenzahl bestimmt wird. Jede Kette wird in sich sortiert, in diesem Beispiel nach dem Vergleichsverfahren. Die Kettenbildung wird so lange fortgesetzt, bis die Kettenzahl kleiner als 1 wird.

Zeichenerklärung

*	Teil der Kette 1
+	Teil der Kette 2
@	Teil der Kette 3
$	Teil der Kette 4

Verzweigungen

Wenn ein umfangreiches Programm entwickelt wird, ist seine Struktur mit einem Baum vergleichbar, an dem sich immer mehr Äste verzweigen, je weiter das Entwicklungsstadium fortschreitet.

Im letzten Teil des BASIC-Programmierkurses haben wir einen Blick auf die Probleme geworfen, die beim Suchen nach einem speziellen Verzeichnis in einer Datei auftreten. Jetzt soll die „Top Down"-Programmierung für die nächsten Teile des Hauptprogramms weiterentwickelt werden. Diese enthalten vier Aufrufe für Unterroutinen bzw. Prozeduren:

```
HAUPTPROGRAMM
BEGIN (STARTE)
   INITIALISIEREN (Prozedur)
   BEGRUESSUNG (Prozedur)
   AUSWAHL (Prozedur)
   AUSFUEHRUNG (Prozedur)
END (ENDE)
```

Die erste Unterroutine, *INITIALISIEREN*, bringt zahlreiche komplexe Aktivitäten mit sich – Festlegen von Bereichen, Einlesen von Daten in diese Bereiche, Testen von Bedingungen und so weiter. Die nächsten zwei Teile des Hauptprogramms umfassen die BEGRUESSUNG- und AUSWAHL-Unterroutinen.

Das Problem der ständigen Verbesserungen während der Programmentwicklung liegt darin, daß die Anzahl an notwendigen Schritten in diesem Stadium nicht genau bestimmbar ist. Für einfache Routinen können zwei oder drei Schritte völlig ausreichen. Doch etwas schwierigere Routinen bedürfen eventuell sehr vieler Schritte, bis das Problem ausreichend analysiert ist.

Ein sehr effektiver Weg, die Dokumentation eines Programms zu organisieren, ist, die einzelnen Stufen seiner Entwicklung systematisch zu numerieren: zum Beispiel römische Zahlen zur Kennzeichnung der Stufe der Verbesserung und arabische Zahlen zur Kennzeichnung der jeweiligen Unterstufe des Programms. Hier ist das Kennzeichnungssystem für unser Adreßbuch-Programm:

```
I HAUPTPROGRAMM
BEGIN (STARTE)
   1. INITIALISIEREN
   2. BEGRUESSUNG
   3. AUSWAHL
   4. AUSFUEHRUNG
END (ENDE)
```

Nun folgt die Verbesserung der BEGRUESSUNG-Routine.

```
II 2 (BEGRUESSUNG)
BEGIN (STARTE)
   1. Stelle Begrüßungsmeldung dar
   2. LOOP (SCHLEIFE) (bis die Leertaste
      gedrückt wird)
      ENDLOOP (ENDE DER SCHLEIFE)
   3. Verzweige zu *AUSWAHL*
END (ENDE)
```

```
III 2 (BEGRUESSUNG) 1
      (stelle Meldung dar)
BEGIN (STARTE)
   1. Lösche Bildschirm
   2. PRINT (DRUCKE) Begrüßungsmeldung
END (ENDE)
```

```
III 2 (BEGRUESSUNG) 2 (LOOP warte
      bis Leertaste gedrückt wird)
BEGIN (STARTE)
   1. LOOP (SCHLEIFE) (bis die Leertaste
      gedrückt wird)
         IF (WENN) Leertaste gedrückt
         THEN (DANN)
      ENDLOOP (ENDE DER SCHLEIFE)
END (ENDE)
```

```
III 2 (BEGRUESSUNG) 3
      (verzweige zu *AUSWAHL*)
BEGIN (STARTE)
   1. GOSUB *AUSWAHL*
END (ENDE)
```

An diesem Punkt sollte klar sein, daß III-2-1 und III-2-3 bereits direkt in BASIC geschrieben werden können. Abschnitt III-2-2 bedarf jedoch noch einer weiteren Verbesserung:

```
IV 2 (BEGRUESSUNG) 2 (LOOP)
BEGIN (STARTE)
   1. LOOP (SCHLEIFE) (bis die Leertaste
      gedrückt wird)
         IF (WENN) INKEY$ ist ungleich
         Leerzeichen THEN (DANN) mache
         weiter
      ENDLOOP (ENDE DER SCHLEIFE)
END (ENDE)
```

Wir sind jetzt an einem Punkt angekommen, an dem die Programmierung der Routine BEGRUESSUNG in BASIC nur noch geringfügig verbessert werden kann:

IV 2 (BEGRUESSUNG) 1 (stelle Meldung dar) BASIC CODE

```
REM *BEGRUESSUNG* UNTERROUTINE
PRINT
PRINT
PRINT
PRINT
PRINT TAB(13);"*WILLKOMMEN ZUM*"
PRINT TAB(11);"*ADRESSBUCH-
  PROGRAMM*"
PRINT TAB(8);"*DES COMPUTERKURSES*"
PRINT
PRINT TAB(1);"(DRUECKE DIE LEERTASTE,
  UM FORTZUFAHREN)"
```

V 2 (BEGRUESSUNG) 2 (LOOP warte bis Leertaste gedrückt wird) BASIC CODE

```
LET L=0
FOR L=1 TO 1
IF INKEY$ < >" " THEN LET L=0
NEXT L
```

IV 2 (BEGRUESSUNG) 3 (verzweige zu *AUSWAHL*) BASIC CODE

```
GOSUB *AUSWAHL*
RETURN
```

Beachten Sie, daß die Variablen mit Anweisungen wie LET I=0 initialisiert werden. Genau genommen ist dies nicht in allen Fällen unbedingt notwendig. Doch es ist eine gute Angewohnheit, die der Übersichtlichkeit dient. Dafür gibt es drei Gründe: Erstens ist es als Erinnerungshilfe sehr nützlich, am Anfang einer Routine eine Liste aller aktuell verwendeten Variablen zur Verfügung zu haben. Zweitens kann man nie ganz sicher sein, welchen Wert eine Variable bei der letzten Verwendung angenommen hat. Drittens kann die Verwendung von Anweisungen wie LET I=0 an entsprechender Stelle die Verarbeitungsgeschwindigkeit eines Programms beschleunigen.

Wir haben die Form, in der FOR. . .NEXT-Schleifen zur Simulation von DO. . .WHILE- oder REPEAT. . .UNTIL-Strukturen eingesetzt werden, bereits in früheren Abschnitten des Kurses geändert. Anstatt FOR I=0 TO 1 oder FOR I=0 TO 1 STEP 0 heißt es jetzt FOR I=1 TO 1. Die Anweisung FOR I=1 TO 1. . .NEXT I führt die Schleife nur einmal aus. Sollte jedoch innerhalb der Schleife der Wert von I auf 0 gesetzt werden, wird die Schleife immer wieder neu ausgeführt.

Der BASIC-Code, der gerade entwickelt wurde, ist alles, was für das komplette BEGRUESSUNG-Modul im Hauptprogramm notwendig ist. Wir haben noch keine Zeilennummern eingefügt, da dies erst nach Fertigstellung aller Programmroutinen möglich ist. Außerdem läßt sich momentan auch noch nicht festlegen, wie die gültigen Zeilennummern für die GOSUB-Anweisungen lauten werden. Wenn Sie das Programm trotzdem testen wollen, ist es notwendig, einige Test-Eingaben und -Unterroutinen zu integrieren. Zwei Punkte, die bei diesem Programmteil zu beachten sind: zum einen die Verwendung der TAB-Funktion und zum anderen die „clear screen – lösche Bildschirm"-Anweisung. TAB verschiebt den Cursor in einer Zeile entsprechend der in Klammern spezifizierten Zahl. Die von uns angegebenen Zahlen geben die Meldung auf einem 40 Zeichen-Bildschirm jeweils zentriert aus. Wenn Sie weniger Zeichen pro Zeile zur Verfügung haben, müssen Sie die Werte der TAB-Anweisungen entsprechend anpassen. Der Befehl zum Löschen des Bildschirms ist in vielen BASIC-Versionen CLS, doch die Version des Microsoft-BASIC, die wir zur Entwicklung unseres Programms verwenden, stellt diese Anweisung nicht zur Verfügung. Statt dessen haben wir PRINT CHR$(12) verwendet, da der Computer den ASCII-Wert 12 als „clear screen"-Zeichen verwendet. Andere Rechner verwenden gewöhnlich ASCII 24 für diese Funktion.

```
10 REM PROVISORISCHES
   HAUPTPROGRAMM
20 PRINT CHR$(12)
30 GOSUB 100
40 END
100 REM UNTERROUTINE *BEGRUESSUNG*
110 PRINT
120 PRINT
130 PRINT
140 PRINT
150 PRINT TAB(12);"*WILLKOMMEN ZUM*"
160 PRINT TAB(9);"*ADRESSBUCH-
    PROGRAMM*"
170 PRINT TAB(6);"*DES COMPUTER-
    KURSES*"
180 PRINT
190 PRINT TAB(5);"(DRUECKE DIE
    LEERTASTE, UM FORTZUFAHREN)"
195 LET L=0
200 FOR L=1 TO 1
210 IF INKEY$< > " " THEN LET L=0
220 NEXT L
230 PRINT CHR$(12)
240 GOSUB 1000
250 RETURN
1000 REM PROVISORISCHE UNTERROUTINE
1010 PRINT "PROVISORISCHE UNTER-
     ROUTINE"
1020 RETURN
```

Und jetzt wird genau dieselbe Methode angewendet, um die AUSWAHL-Unterroutine zu verbessern.

II 3 (AUSWAHL)

```
BEGIN (STARTE)
  1. PRINT (DRUCKE) Menü
  2. INPUT WAHL
  3. Verzweige zur WAHL-Unterroutine
END (ENDE)
```

III 3 (AUSWAHL) 1 (PRINT Menü)
BEGIN (STARTE)
1. Lösche Bildschirm
2. PRINT (DRUCKE) Menü und Meldung
END (ENDE)

III 3 (AUSWAHL) 2 (INPUT WAHL)
BEGIN (STARTE)
1. INPUT WAHL
2. Überprüfe, ob Wahl im zugelassenen Bereich liegt
END (ENDE)

III 3 (AUSWAHL) 3 (verzweige zu WAHL)
BEGIN (STARTE)
1. AUSFÜHRUNG DER WAHL
ENDCASE (ENDE DER AUSFÜHRUNG)
END (ENDE)

III-3-1 (PRINT Menü) kann jetzt in BASIC geschrieben werden:

IV 3 (AUSWAHL) 1 (PRINT Menü) BASIC CODE

```
REM LOESCHE BILDSCHIRM
PRINT CHR$(12):REM ODER 'CLS'
PRINT
PRINT
PRINT
PRINT
PRINT "1. VERZEICHNIS DURCH NAMEN
  SUCHEN"
PRINT "2. NAMEN DURCH TEIL EINES
  NAMENS SUCHEN"
PRINT "3. VERZEICHNISSE NACH
  STADTANGABEN SUCHEN"
PRINT "4. VERZEICHNISLISTE DURCH
  INITIALEN"
PRINT "5. LISTE ALLER VERZEICHNISSE"
PRINT "6. HINZUFUEGEN EINER ADRESSE"
PRINT "7. AENDERN EINER ADRESSE"
PRINT "8. LOESCHEN EINER ADRESSE"
PRINT "9. PROGRAMM BEENDEN UND
  DATEN SPEICHERN"
```

Lassen Sie uns zuerst die nächste Entwicklungsstufe von III-3-2 betrachten.

Der Variablen WAHL einen Wert zuzuordnen, ist ausgesprochen einfach: Eine „INPUT WAHL"-Anweisung nach der entsprechenden Aufforderung erfüllt diese Aufgabe. Es gibt jedoch nur neun gültige Möglichkeiten. Was würde passieren, wenn man versehentlich eine 0 oder sogar 99 eingibt? Da die WAHL, die Sie treffen, bestimmt, welcher Teil des Programms als nächstes aufgerufen wird, ist sicherzustellen, daß keine unerwünschten Fehler auftreten können. Zu diesem Zweck müssen wir eine „Eingabe-Überprüfungs"-Unterroutine entwickeln. Dabei handelt es sich um eine kleine Routine, die vor der Fortsetzung des Programmlaufs überprüft, ob die eingegebene Zahl im gültigen Bereich liegt.

```
1 REM ROUTINE
10 LET L=0
20 FOR L=1 TO 1
30 INPUT "WAEHLE 1 BIS 9";WAHL
40 IF WAHL < 1 THEN LET L=0
50 IF WAHL > 9 THEN LET L=0
60 NEXT L
70 PRINT "IHRE WAHL WAR ";WAHL
80 END
```

INPUT bewirkt, daß das Programm stoppt und auf eine Eingabe über die Tastatur wartet. Eine „anwenderfreundlichere" Methode wäre, wenn das Programm nach Eingabe einer gültigen Zahl automatisch fortfahren würde. Dies ist durch die INKEY$-Funktion möglich. Bei dieser Funktion liest BASIC ein Zeichen von der Tastatur, sobald INKEY$ verwendet wird. Üblicherweise wird INKEY$ in Schleifen eingesetzt. Die Schleife zur Überprüfung, ob eine Taste gedrückt wurde, kann so aussehen: IF INKEY$ = " " THEN . . . Eine für unsere Zwecke geeignete Routine sieht so aus:

```
LET I=0
FOR I=1 TO 1
LET A$=INKEY$
IF A$=" " THEN LET I=0
NEXT I
```

Der einzige Nachteil bei der Verwendung von INKEY$ ist, daß sowohl Buchstaben als auch Zahlen von der Tastatur „gelesen" werden. Wenn in einem Programm eine Auswahl zwischen mehreren Möglichkeiten besteht, ist es in BASIC einfacher, mit Zahlen zu arbeiten.

Jetzt kommen die BASIC-Funktionen NUM und VAL ins Spiel. Sie wandeln Zahlen in Zeichen-Strings in „reale" Zahlen um. Das sind numerische Werte. Die realen Zahlen werden wie folgt verwendet:

```
LET N=VAL(A$) oder LET N=NUM(A$)
```

Durch Einsetzen von NUM- oder VAL-Funktionen können wir das Programm mit INKEY$ dazu veranlassen, Eingaben in numerische Variablen umzuwandeln.

```
FOR L=1 TO 1
  PRINT "GEBEN SIE IHRE WAHL EIN (1—9)"
    FOR I=1 TO 1
      LET A$=INKEY$
      IF A$=" " THEN LET I=0
    NEXT I
  LET WAHL=VAL(A$)
    IF WAHL < 1 THEN LET L=0
    IF WAHL > 9 THEN LET L=0
NEXT L
```

Jetzt wollen wir für die *WAHL*-Routine ein komplettes Programm in BASIC erstellen, einschließlich Eingabe- und Unterroutinen zu Testzwecken. Die Zeilennummern wurden nur

zu Testzwecken eingefügt und müssen bei der Zusammenstellung des endgültigen Programms ersetzt werden.

```
10 PRINT CHR$(12)
20 PRINT "WAEHLEN SIE AUS"
30 PRINT
40 PRINT
50 PRINT
60 PRINT "1. VERZEICHNIS DURCH NAMEN
   SUCHEN"
70 PRINT "2. NAMEN DURCH TEIL EINES
   NAMENS SUCHEN"
80 PRINT "3. VERZEICHNISSE NACH
   STADTANGABE SUCHEN"
90 PRINT "4. VERZEICHNISLISTE DURCH
   INITIALEN"
100 PRINT "5. LISTE ALLER VERZEICHNISSE"
110 PRINT "6. HINZUFUEGEN EINER
   ADRESSE"
120 PRINT "7. AENDERN EINER ADRESSE"
130 PRINT "8. LOESCHEN EINER ADRESSE"
140 PRINT "9. PROGRAMM BEENDEN UND
   DATEN SPEICHERN"
150 PRINT
160 PRINT
170 LET L=0
180 LET I=0
190 FOR L=1 TO 1
200 PRINT "WAEHLE 1 BIS 9"
210 FOR I=1 TO 1
220 LET A$=INKEY$
230 IF A$=" " THEN LET I=0
240 NEXT I
250 LET WAHL=VAL(A$)
260 IF WAHL < 1 THEN LET L=0
270 IF WAHL > 9 THEN LET L=0
280 NEXT L
290 ON WAHL GOSUB 310, 330, 350, 370,
   390, 410, 430, 450, 470
300 END
310 PRINT "TEST-UNTERROUTINE 1"
320 RETURN
330 PRINT "TEST-UNTERROUTINE 2"
340 RETURN
350 PRINT "TEST-UNTERROUTINE 3"
360 RETURN
370 PRINT "TEST-UNTERROUTINE 4"
380 RETURN
390 PRINT "TEST-UNTERROUTINE 5"
400 RETURN
410 PRINT "TEST-UNTERROUTINE 6"
420 RETURN
430 PRINT "TEST-UNTERROUTINE 7"
440 RETURN
450 PRINT "TEST-UNTERROUTINE 8"
460 RETURN
470 PRINT "TEST-UNTERROUTINE 9"
480 RETURN
```

Im nächsten Teil unseres Kurses werden wir uns mit Datei-Strukturen befassen und mit der Ausarbeitung der *INITIALISIEREN*-Unterroutine beginnen.

BASIC-Dialekte

SPECTRUM

In dem provisorischen Hauptprogramm sowie in allen anderen Routinen muß die Anweisung PRINT CHR$(12) durch CLS und END durch den Befehl STOP ersetzt werden.

PRÜFROUTINE

```
1 REM ROUTINE
10 LET L=0
20 FOR L=1 TO 1
30 INPUT "WAEHLE 1 BIS 9";WAHL
40 IF WAHL < 1 THEN LET L=0
50 IF WAHL > 9 THEN LET L=0
60 NEXT L
70 PRINT "IHRE WAHL
   WAR ";WAHL
80 STOP
```

FERTIGES LISTING

```
10 CLS
```

Danach geben Sie das Programm bis Zeile 230 wie im Haupttext angegeben ein.

```
240 NEXT I
250 LET WAHL=CODEA$-48
260 IF WAHL < 1 THEN LET L=0
270 IF WAHL > 9 THEN LET L=0
280 NEXT L
290 GOSUB (WAHL*20+290)
300 STOP
```

TAB

Einige Dialekte verfügen nicht über die TAB-Anweisung. Fügen Sie am Anfang folgendes ein:

```
5 LET S$=" "
```

Zwischen den Anführungszeichen sollten sich soviele Leerzeichen befinden, wie in eine Bildschirmzeile passen. Wenn TAB(12) erscheint, ersetzen Sie es durch LEFT$(S$,12). Verwenden Sie dabei den Wert innerhalb der TAB-Anweisung in der LEFT$()-Funktion.

CHR$(12)

Beim Oric, dem Dragon 32, dem Lynx und dem Acorn B müssen Sie PRINT CHR$(12) durch CLS ersetzen. Beim Commodore 64 und dem VC 20 muß CHR$(12) durch PRINT „shift-Taste + CLR/HOME-Taste" ersetzt werden. Als Ergebnis sollte ein „inverses Herz" erscheinen.

ON.. GOSUB

Diese Funktion ist auf einigen Computern nicht verfügbar, kann jedoch durch Zeile 290 des fertigen Spectrum-Listings ersetzt werden.

VARIABLES

Siehe vorherige BASIC-Dialekte.

INKEY$

Besitzer eines Commodore müssen LET A$=INKEY$ durch GET A$ ersetzen, und aus IF INKEY$=" " THEN wird:

```
GET A$:IF A$=" " THEN
```

Laser-Show

Optische Datenträger (Laserplatten) eröffnen interessante Möglichkeiten für Heimcomputer: als „interaktive" Videosysteme und als Massenspeicher.

Wann immer es um Heimcomputer geht, steht die Speicherkapazität im Mittelpunkt. Die Größe des Arbeitsspeichers ist natürlich wichtig, aber auf längere Sicht ist eher die Kapazität des angeschlossenen Massenspeichers bei der Beurteilung eines Gerätes entscheidend. Nach ein paar Monaten verfügen die meisten Anwender über einen ansehnlichen Stapel von Cassetten und Disketten – mit Programmen, die größtenteils nie wieder verändert werden. Sie wären in einem ROM-Modul besser untergebracht als auf den empfindlichen Magnetträgern. Aber dazu müßte ein Festspeicher mit der vielfachen Kapazität eines ROM-Moduls vorhanden sein.

So etwas gibt es in Form der Laserplatte. Für den Hausgebrauch gibt es allerdings lediglich Bildplatten, die Video-Cassetten ersetzen, und Compact Discs anstelle konventioneller Schallplatten.

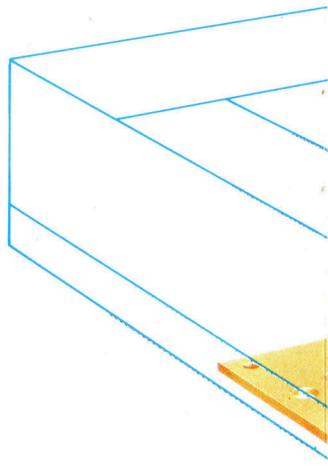
Bild- und CD-Platten unterscheiden sich im Durchmesser (35 cm bzw. 12 cm). In beiden Fällen werden die Informationen laseroptisch gespeichert und ausgelesen, bei der CD-Platte digital. Damit wird eine extrem hohe Pakkungsdichte und Störsicherheit erreicht. Beim Abspeichern muß das ursprüngliche Signal, etwa eine Mikrofonspannung, zunächst durch einen Analog/Digital-Wandler in Digitalwerte umgesetzt werden, d. h. in eine Folge von Nullen und Einsen. Bei der Wiedergabe wird daraus mit einem Digital/Analog-Wandler wieder das Tonfrequenzsignal rekonstruiert. Videoaufnahmen auf Magnetträgern dagegen sind aufgrund der häufig auftretenden elektrischen Ladung der Umgebung sehr störanfällig. Die Speicherkapazität einer einzigen Bildplatte ist mit einigen Tausend Megabyte viel größer als die einer Winchesterplatte.

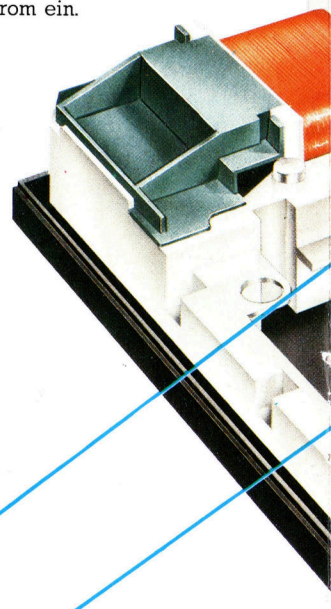
Bei den Bildplattenspielern dominiert bisher das Philips-System. 99 % der Platte sind im Grunde nur ein Schutz für den Informationsträger: eine dünne Metallfolie mit Milliarden mikroskopischer Vertiefungen („Pits"). Die Informationen sind wie bei einer Diskette katalogisiert, so daß ein sofortiger Zugriff auf beliebige Abschnitte möglich ist. Die Daten werden nach Positionieren des Lesekopfes mit einem Laser abgetastet. Das Laserlicht dringt durch die Schutzschicht und wird von der glatten Metalloberfläche voll auf einen Lichtdetektor reflektiert, während es von den „pits" gestreut wird. Die Aufzeichnung erfolgt in Spiralform, je Umdrehung ein Einzelbild. Auf eine Plattenseite passen 54 000 Bilder, einer Spieldauer von 36 Minuten entsprechend.

Im Verbund mit Computern sind die Anwendungsmöglichkeiten für Laserplatten wesentlich in zwei Bereichen zu sehen – erstens bei den „interaktiven" Video-Systemen, die bereits auf dem Markt sind. Hier ist die Abfolge von Bild- und Textinformationen per Rechner willkürlich steuerbar. Die Bildplatte wird dabei als „Bibliothek" verwendet. Textzeilen können dem Bild auf einem normalen Fernsehschirm überlagert werden. Im Dialog mit dem Rechner kann der Benutzer beliebige Einzelszenen abrufen. Auch für Lehrprogramme ist die Bildplatte einsetzbar: Auf dem TV-Schirm werden Standbilder oder Abläufe mit unterlegten Fragen dargestellt; dazu gibt der Lernende seine Antworten zur Überprüfung in den Rechner ein. Philips bietet eine Profi-Version seines Systems an, die bereits ohne Zusatzrechner interaktives Video ermöglicht, aber auch über eine IEEE488- oder RS232-Schnittstelle an einen Computer angeschlossen werden kann.

Das zweite große Einsatzfeld für die Laserplatte ist die Bereitstellung von Computer-Software. Stellen Sie sich vor, daß Sie die gesamte System-Software für Ihren Rechner – Textverarbeitung, Datenbank, Tabellenkalkulation, dazu noch einige Dutzend Spiele – auf einer einzigen robusten Compact Disc bereithalten könnten!

Bislang konnte man die Verbindung Laserdisc – Heimcomputer und die dadurch erzeugte Darstellungsqualität nur auf Messen bewundern. Angesichts der sinkenden Preise für technische Geräte bleibt abzuwarten, wann die Laserdisc die herkömmlichen Speichermedien auch im Heimbereich ablöst.

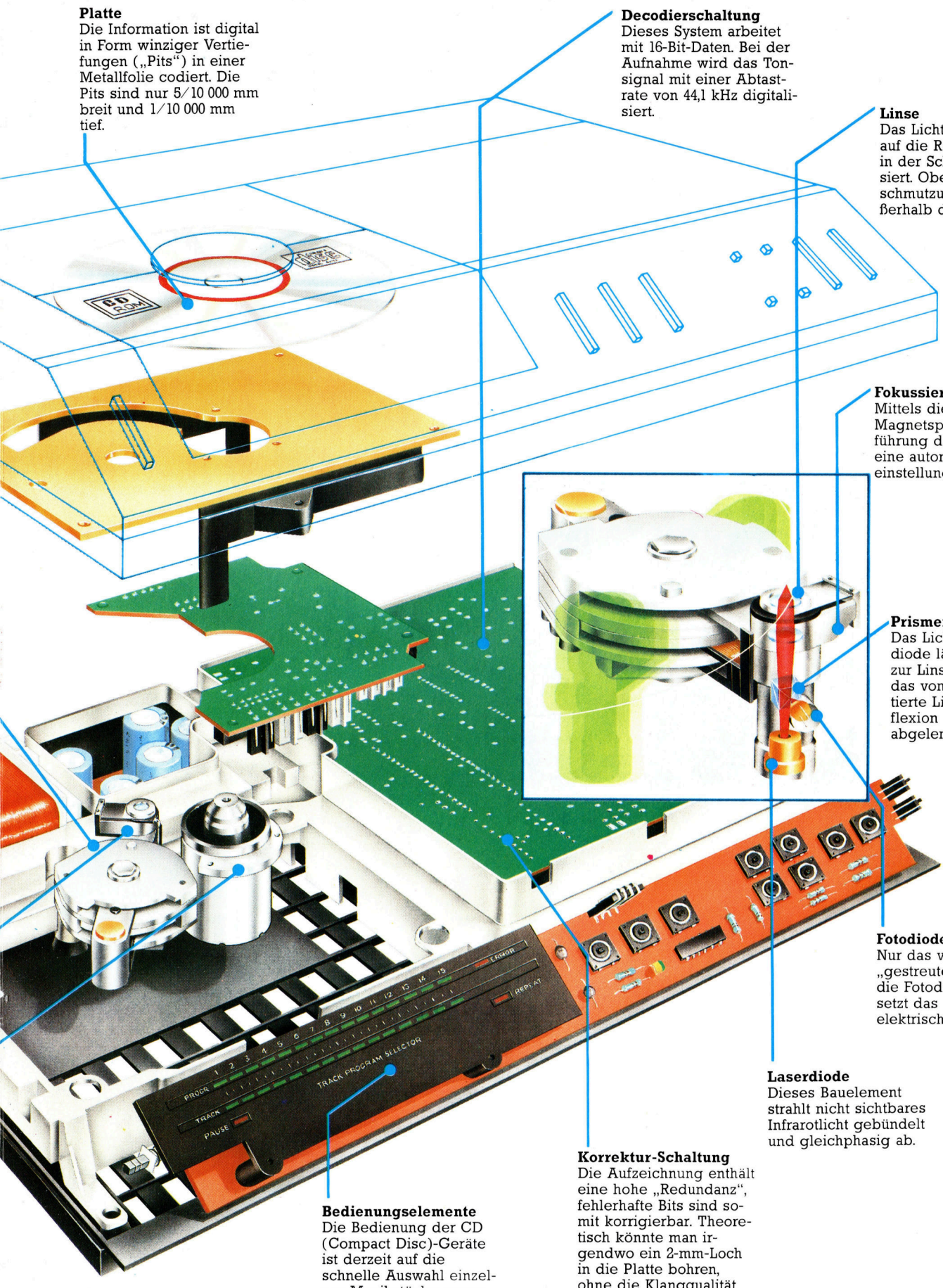




Linearmotor
Das Schwenken des Abtastarms besorgt eine daran befestigte kleine Spule, die sich in einem Magnetfeld gegen eine Rückholfeder bewegt. Der Schwingarm stellt sich entsprechend dem Spulenstrom ein.

Abtastarm
Der Arm ist in seiner Mitte leichtgängig gelagert und mit einem Gegengewicht versehen. Der Lesekopf bewegt sich beim Schwenken des Arms unterhalb der Platte auf einem Kreisbogen.

Motor
Für die Steuerung der Plattendrehzahl sorgt ein quarzüberwachter Regelkreis. Wenn das Abtastsystem von innen nach außen läuft, wird die Drehzahl von 500 auf 200 Upm gesenkt (konstante Aufzeichnungsdichte).



Platte
Die Information ist digital in Form winziger Vertiefungen („Pits") in einer Metallfolie codiert. Die Pits sind nur 5/10 000 mm breit und 1/10 000 mm tief.

Decodierschaltung
Dieses System arbeitet mit 16-Bit-Daten. Bei der Aufnahme wird das Tonsignal mit einer Abtastrate von 44,1 kHz digitalisiert.

Linse
Das Lichtbündel ist exakt auf die Reflexionsschicht in der Schutzhülle fokussiert. Oberflächliche Verschmutzungen liegen außerhalb der Schärfezone.

Fokussierspule
Mittels dieser kleinen Magnetspule zur Nachführung der Optik erfolgt eine automatische Scharfeinstellung.

Prismensystem
Das Licht von der Laserdiode läuft ungebrochen zur Linse hin, während das von den Pits reflektierte Licht durch Totalreflexion auf die Fotodiode abgelenkt wird.

Fotodiode
Nur das von den Pits „gestreute" Licht trifft auf die Fotodiode. Diese setzt das Licht in ein elektrisches Signal um.

Laserdiode
Dieses Bauelement strahlt nicht sichtbares Infrarotlicht gebündelt und gleichphasig ab.

Korrektur-Schaltung
Die Aufzeichnung enthält eine hohe „Redundanz", fehlerhafte Bits sind somit korrigierbar. Theoretisch könnte man irgendwo ein 2-mm-Loch in die Platte bohren, ohne die Klangqualität merklich herabzusetzen.

Bedienungselemente
Die Bedienung der CD (Compact Disc)-Geräte ist derzeit auf die schnelle Auswahl einzelner Musikstücke ausgelegt.

Schalldicht

Klangsynthese mit dem Dragon 32.

Obwohl der Dragon 32 nur über einen Tongenerator für Rechteckschwingungen verfügt, kann schon ein einziges Kommando mit den außerordentlich einfachen Tonsteuerungsbefehlen des Microsoft Extended Colour BASIC eine interessante Melodie spielen. Erstaunlicherweise fehlt die Möglichkeit, Rauschen zu erzeugen. Es gibt kaum Arcadespiele, die ohne diesen Effekt auskommen.

Mit dem Befehl SOUND können nur Töne erzeugt werden. Er hat folgendes Format:

```
SOUND P,D
```

wobei P die Höhe des Tones ist (1 – 255) und D die Dauer (1 – 255). Die Tonhöhe ist sehr ungenau und hat nur wenig Ähnlichkeit mit einer normalen Tonleiter. Der Wert 89 entspricht ungefähr dem mittleren C, und der Kammerton A mit 440 Hz liegt etwa bei 159. Ähnlich ungenau ist auch die Tondauer, wobei jedoch 16 etwa einer Sekunde entspricht, 32 zwei Sekunden und so weiter.

Soundeffekte

Das folgende Programm zeigt, wie SOUND Spezialeffekte erzeugen kann. In diesem Fall hebt gerade – mit ein wenig Phantasie – ein UFO ab:

```
10 FOR P=10 TO 170 STEP 10
20 FOR D=16 TO 1 STEP —1
30 SOUND P,D
40 NEXT D
50 NEXT P
```

Der Befehl PLAY legt die exakte Höhe, Dauer und Lautstärke eines Tones fest. Auch eine Anzahl von Tönen mit entsprechenden, dazwischenliegenden Pausen kann damit in den unterschiedlichsten Geschwindigkeiten gespielt werden. Die Zusammensetzung von Tönen mit verschiedenen Längen und Pausen läßt sich sehr einfach bewerkstelligen:

```
PLAY "T;O;V;L;N;P"
```

wobei T = Geschwindigkeit (T1–T255), O = Oktave (01–05), V = Lautstärke (V0–V15), L = Tondauer (L1–L255), N = Notenwert (1–12 oder Buchstabe der Note) und P = Pause vor dem nächsten Ton (P1–P255).

Bei der Festlegung einer Tondauer über L und P lassen sich die Werte als ganze Töne oder als Teile davon betrachten, wobei L1 oder P1 einen ganzen Ton darstellen, L2 oder P2 einen halben etc. Die eigentliche Spieldauer wird über den Geschwindigkeitsparameter T gewählt, wobei T1 langsam ist (ein langer Ton), und T255 sehr schnell (kurzer Tonimpuls). Weiterhin kann die Tondauer auch noch durch Hinzufügen von Punkten verlängert werden: L1. . . oder L5. . . Dabei dehnt jeder Punkt die Länge des Tones um die Hälfte des angegebenen Wertes L1. . . ist also 1 + 1/2 + 1/2 + 1/2 = 2 1/2 Tonlängen und L5 ist 1/5 + 1/10 = 3/10 Tonlängen.

Der Parameter O legt die Oktave fest, in der der nächste Ton erklingen soll. O1 fängt bei einem tiefen C (131 Hz) an und hört bei dem B mit 2093 Hz auf. Innerhalb einer Oktave lassen sich Töne auf zwei verschiedene Weisen spielen. Im ersten Fall wird eine Zahl angegeben, die dem Ton nach folgendem Schema entspricht:

1	2	3	4	5	6
C	C#	D	D#	E	F
7	8	9	10	11	12
F#	G	G#	A	A#	B

Nach dieser Methode läßt sich ein Ton als Variable in die angewählte Oktave einsetzen. Im zweiten Fall kann die Buchstabenbezeichnung des Tones direkt angegeben werden.

Die geschilderten Möglichkeiten lassen sich am besten anhand eines Beispiels darstellen.

```
PLAY "T3;L2;6;P4;O3;V20;L1;A#"
          F                  A#
PAUSE
```

Eine der brauchbarsten Fähigkeiten des Dragon ist die Möglichkeit, Teile einer Zeichenkette (Substrings) einzusetzen. Die Substrings werden zunächst festgelegt und lassen sich dann in beliebiger Reihenfolge spielen oder auch wiederholen.

```
10 A$="F;A#;G"
20 B$="C;D#;F;P4;XA$;"
30 PLAY B$
```

Diese Programmzeilen definieren zunächst A$ und schließen die Variable dann als den Substring XA$ in B$ ein. Das Ergebnis ist die Melodie C-D#-F-P4-F-A#-G. Diese Technik erweist sich als besonders nützlich, wenn innerhalb eines Musikstückes die gleiche Tonfolge mehrfach wiederholt wird. Dabei muß auf den Substring ein Semikolon folgen.

Leichte Unterhaltung

Die zweite Folge über die Grafikfähigkeiten des Acorn B.

Das BASIC des Acorn B verfügt nicht wie das anderer Heimcomputer über eine ganze Reihe von Befehlen für hochauflösende Grafik. So besitzt er beispielsweise kein CIRCLE- oder PAINT-Kommando. Die meisten dieser Möglichkeiten lassen sich mit einigen BASIC-Anweisungen selber programmieren.

Der Grafikschirm wird immer über die gleichen Koordinaten gesteuert, welcher Auflösungsgrad auch gewählt wurde. Nullpunkt ist die linke untere Ecke. Mit folgenden Befehlen läßt sich die grafische Darstellung steuern:

```
MOVEx,y
```

Unabhängig von dem Textcursor bewegt dieser Befehl den Grafikcursor auf einen Punkt mit den (x,y)-Koordinaten, zeichnet aber keine Linie.

```
DRAWx,y
```

zeichnet von der augenblicklichen Position des Grafikcursors eine Linie zu dem Punkt mit den (x,y)-Koordinaten.

```
PLOTk,x,y
```

ist ein Vielzweckbefehl, dessen Funktion vom Wert der Variablen k gesteuert wird:

k	Funktion
0	Bewegung relativ zum letzten Punkt
1	Zeichnet vom Nullpunkt eine Linie in der Vordergrundfarbe
2	Zeichnet vom Nullpunkt eine Linie in invertierter Farbe (Farbumkehr)
3	Zeichnet vom Nullpunkt eine Linie in der Hintergrundfarbe
4	entspricht dem Befehl MOVE
5	entspricht dem Befehl DRAW
6	entspricht dem Befehl DRAW, aber in invertierter Farbe (Farbumkehr)
7	entspricht dem Befehl DRAW, aber in der Hintergrundfarbe

Höhere Zahlen wiederholen diese acht Funktionen, rufen aber Spezialeffekte hervor wie beispielsweise gepunktete statt durchgehender Linien. Die Werte von k zwischen 80 und 87 erfüllen eine besonders brauchbare Funktion. PLOT80,x,y verbindet den Punkt (x,y) mit den beiden zuvor gezeichneten Punkten zu einem Dreieck und füllt dessen Fläche mit der Vordergrundfarbe aus. Dies ist die einzige Funktion, mit der sich Umrisse ausfüllen lassen.

VDU x entspricht dem BASIC-Befehl PRINT CHR$(x). In der letzten Folge über die Grafikmöglichkeiten des Acorn B haben wir gesehen, daß der Befehl VDU von einer ganzen Reihe von Zahlen gefolgt werden kann. VDU v,w,x,y,z entspricht:

```
PRINT CHR$(v);CHR$(w);CHR$(x);CHR$(y);
  CHR$(z)
```

Mit dem VDU-Befehl hat der Anwender Zugang zu dem Teil des Betriebssytems, das die Grafik- und Bildschirmdarstellung steuert.

Zeichen lassen sich auf dem Acorn B mit dem Befehl VDU 23 sehr leicht neu definieren. Im Artikel über definierbare Grafikelemente haben Sie gesehen, daß sich die Zeichen des normalen ASCII-Alphabets aus Blöcken von acht mal acht Pixeln (Bildpunkten) aufbauten. Eine 1 stellt die sichtbaren Pixel dar und eine 0 die unsichtbaren. Die acht Bits jeder Zeile lassen sich in die entsprechende Dezimalzahl umsetzen, wobei insgesamt acht Zahlen ein neues Zeichen ergeben. Der Befehl VDU 23 gibt dem Anwender die Möglichkeit, die Zeichen des ASCII-Codes zwischen 224 und 255 neu festzulegen. Ein Beispiel:

```
10 REM DEFINIERE EIN ZEICHEN
20 MODE 2
30 VDU 23,240,16,56,124,146,16,16,16,0
40 PRINT CHR$(240)
50 END
```

Dieses kurze Programm wandelt das Zeichen 240 des ASCII-Codes in einen Pfeil um. Die letzten acht Zahlen legen die neue Form fest, während Zeile 40 das Zeichen auf dem Bildschirm abbildet.

Die Befehle VDU 24 und VDU 28 steuern jeweils den Aufbau von Bildschirmfenstern für Grafik und Text. Mit diesen Funktionen läßt sich die Abbildung von Text und Grafik auf einen bestimmten Bereich begrenzen. Für Programme, in denen mehrere Funktionen gleichzeitig gesteuert werden, ist diese Möglichkeit besonders praktisch. Zur Konstruktion eines Bildschirmfensters brauchen nur die Koordinaten der linken unteren und der rechten oberen Ecke angegeben werden.

Dieses kurze Programm zeichnet eine farbenprächtige Blume spiralenförmig auf den Bildschirm. Die Auflösung ist dabei MODE 1. Beachten Sie, wie die farbigen Blütenblätter aus ausgefüllten Dreiecken aufgebaut werden.

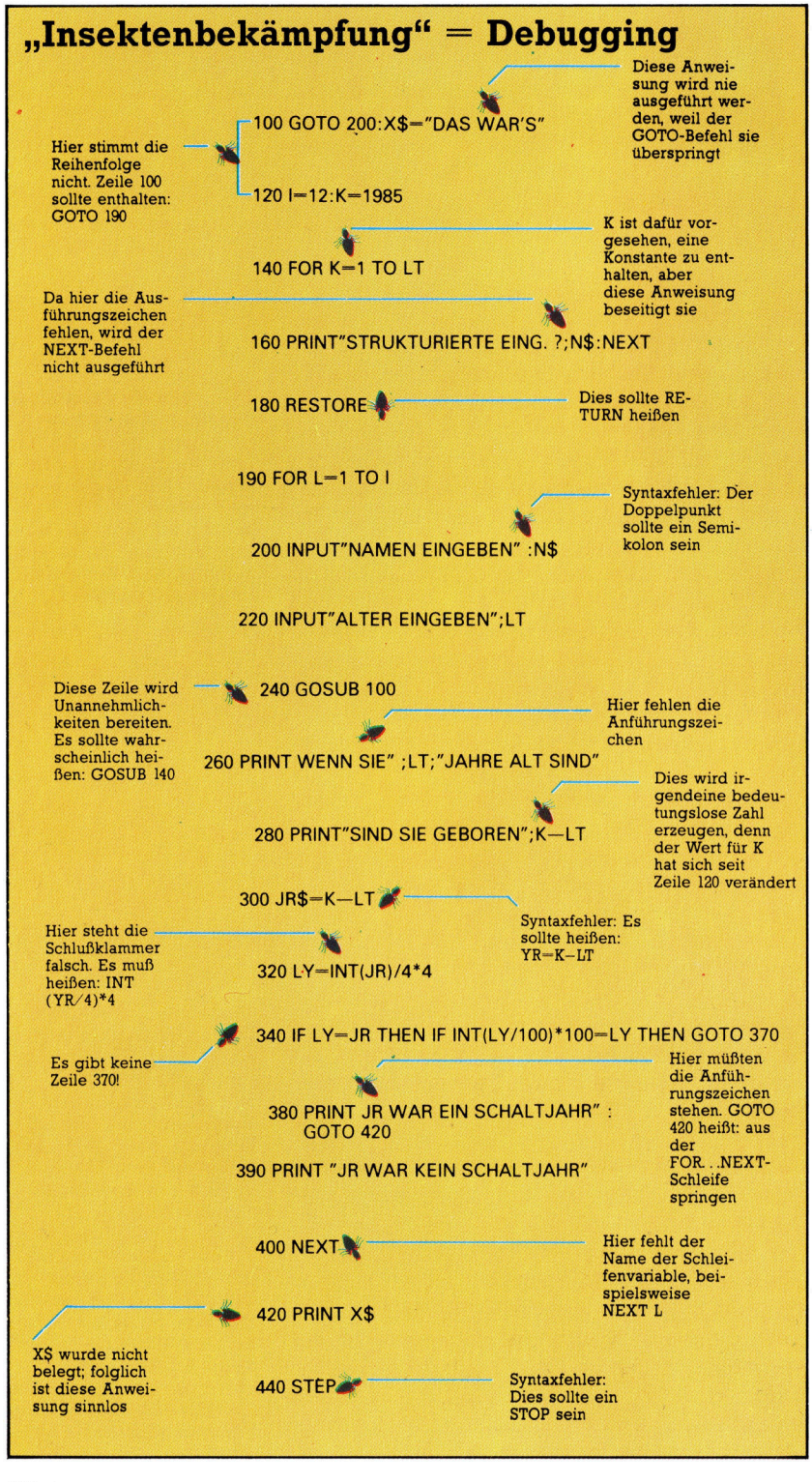
```
10 REM BLUETE
20 CLS
30 MODE 1
40 FOR D=1 TO 3
50 A=600:B=500
60 MOVEA,B
70 FOR C=1 TO 550
   STEP3
80 GCOL0,RND(3)
90 S=(C/RND(5)+10))
100 X=S*5*SIN(C/16)+A
110 Y=S*5*COS
    (C/16)+B
120 PLOT85,X,Y
130 NEXT C
140 NEXT D
150 END
```

Das Spiralmuster wird durch eine Verbindung von Sinus und Cosinus in Zeile 100 und 110 hervorgerufen. Normalerweise erzeugt dieses Verhältnis zwischen x- und y-Koordinaten einen Kreis. Die FOR. . .NEXT-Schleife erhöht jedoch den Radius C bei jedem Durchlauf und baut damit eine Spirale auf. Soll die Blüte an eine andere Stelle gesetzt werden, müssen nur die Koordinaten des Spiralenzentrums verändert werden.

Auf Fehlersuche

Durch sorgfältiges Planen und schrittweises Vorgehen beim Programmieren kann der Zeitaufwand für die Fehlerbeseitigung erheblich verkürzt werden.

„Insektenbekämpfung" = Debugging



Mit zunehmender Programmiererfahrung wird auch das „debugging" (das Beseitigen von Programmierfehlern) geläufiger. Fangen Sie damit an, sich selbst die Aufgabenstellung so klar wie möglich darzulegen. Teilen Sie die Gesamtaufgabe dann in einzelne, logisch in sich geschlossene Teilbereiche, wie Eingabe, Ausgabe, Algorithmen, Datenstruktur, Verarbeitungsprozesse, und betrachten Sie jeden Teilbereich separat.

Beginnen Sie erst mit der Programmeingabe, wenn Sie genau wissen, wie Sie Stück für Stück des Gesamtproblems lösen können! Diese durchdachte Vorgehensweise wird Ihnen viel Zeit beim späteren Ausmerzen von Programmierfehlern ersparen.

Nach dieser Vorarbeit sind Sie in der Lage, strukturierte Programme zu schreiben, die aus einer Anzahl von Subroutinen sowie einem als Gerüst dienenden Hauptprogramm bestehen. In einem solchen Programm sind Fehler leichter zu entdecken. Und es läßt sich eine „Bibliothek" fehlerfreier Subroutinen aufbauen, auf die bei späteren Programmierarbeiten zurückgegriffen werden kann.

Geben Sie den Variablen möglichst aussagekräftige Namen oder Abkürzungen. NET=BRUTT-TAX beispielsweise spricht für sich selbst. Während N=B-T dagegen kaum mehr erkennen läßt, welchen Wert die Variable beinhaltet. Es empfiehlt sich, eine Variablenliste zu führen, die alle im Programm verwendeten Variablen nach Namen und Verwendungszweck enthält. Damit ist die Voraussetzung für eine gewisse Standardisierung geschaffen (beispielsweise können Sie immer einen bestimmten Buchstaben als Variable für Schleifenzähler verwenden). Es sollte ferner vermieden werden, die gleiche Variable für verschiedene Zwecke zu verwenden. Genauso empfiehlt sich, konstante Werte am Anfang des Programms in Variablen zu speichern und später auf diese Variablen zu verweisen. Das Programm wird dadurch schneller und übersichtlicher, und Sie können diese Werte ändern, ohne das Programm danach absuchen zu müssen, wo dieser Wert auftritt.

Trotz des hier beschriebenen Vorgehens sind Programmierfehler nicht ganz zu vermeiden. Es ist darum wichtig, ein Verfahren zur Hand zu haben, mit dessen Hilfe Fehler methodisch aufgefunden werden können. Die bekanntesten „Schnitzer" sind Syntaxfehler – sie können gewöhnlich sofort, wenn der Computer diese anzeigt, beseitigt werden, jedoch kann dies schwierig sein.

Am folgenden Beispiel wollen wir diesen Vorgang einmal besonders deutlich machen. Es handelt sich um einen recht häufigen Fehler:

```
10 PRINT "SCHEINBAR BEDEUTUNGSLOSE
   FEHLER"
20 PRINT "HABEN OFTMALS GROSSE AUS-
   WIRKUNGEN"
```

Solche Zeilen lösen in manchen Fällen eine Fehlermeldung aus, wenn sie nicht als getrennte Zeilen eingegeben wurden. Zeile 10 enthält mehr als 40 Zeichen. Vergessen Sie beim Eingeben, am Ende der Zeile 10 RETURN zu drücken, ehe Sie mit Zeile 20 beginnen, liest der Computer die beiden so perfekt aussehenden Zeilen als eine Zeile.

Falsche Fährte

Fehlermeldungen können aber auch auf die falsche Fährte führen. Die beiden Programmzeilen

```
25 DATA 10.2,34,56.9,0.O08, 15.6
30 FOR K=1 TO 5:READ N(K):NEXT K
```

beispielsweise werden wegen eines Fehlers nicht ausgeführt, der scheinbar in Zeile 30 auftritt. Tatsächlich liegt der Fehler aber in Zeile 25: Eine der Nullen wurde fälschlicherweise als Buchstabe O eingegeben.

Codierfehler, die nicht zu Syntaxfehlern führen, sind häufig schwierig zu finden. Hier ist methodisches Vorgehen besonders wichtig. Versuchen Sie zunächst grob herauszufinden, in welchem Programmbereich der Fehler versteckt sein könnte. Bei einem gut strukturierten modularen Programm ist dies gar nicht so schwer. Die Suche wird weiter vereinfacht durch die TRACE-Funktion, die diejenige Programmzeile auf den Bildschirm bringt, die gerade abgearbeitet wird. Fehlt diese Funktion bei Ihrem Computer, können Sie sich durch periodische TRACE-Anweisungen helfen, die Sie in das Programm einbauen (zum Beispiel PRINT "LINE 150" zu Beginn der Zeile 150). Auch der STOP-Befehl ist recht nützlich – mit ihm kann das Programm an bestimmten Stellen angehalten werden, um die Werte kritischer Variablen zu überprüfen. Dies kann direkt durch PRINT oder durch eine Subroutine am Programmende veranlaßt werden:

```
11000 REM DRUCKE DIE VARIABLEN AUS
11100 PRINT "SCORE,SIZE,FLAGS"
11200 PRINT SC;SZ;F1;F2
11300 PRINT "BOARD ARRAY"
11400 FOR K=1 TO 10:PRINT BD$(K):NEXT K
```

Die Folge ist, daß Sie immer dann, wenn das Programm einen STOP-Befehl erreicht, GOTO 11000 eingeben können und der gegenwärtige Variablenstand angezeigt wird. Sie können sogar Änderungen durchführen (zum Beispiel durch Eingeben von SZ=17, RETURN und CONT).

Haben Sie den Fehler bis auf einige bestimmte Zeilen oder eine Variable eingegrenzt, sind Sie nahe am Ziel, aber Vorsicht ist trotzdem geboten. Gehen Sie immer nur um einen Schritt vor und stellen Sie die Auswirkung dieses Schrittes genau fest, ehe Sie weitermachen. Ein besonders fruchtbarer Boden für „Bugs" ist die Anhäufung von Schleifen und Verzweigungen. Hier ist besondere Sorgfalt am Platze und zwar sowohl beim Schreiben als auch bei der Fehlersuche. Betrachten Sie folgendes Beispiel:

```
460 IF SM<0 AND SC< >—1 THEN IF
    SC>0 OR SM=SC—F9 THEN LT=500
470 FOR C1=1 TO LT:FOR C2=LT TO C1
    STEP—1
480 SC=SM+SC*C2
490 NEXT C2:SM=0:NEXT C1
```

Was bedeutet dies alles? Selbst wenn Sie wissen, was geschehen sollte, wissen Sie noch lange nicht, ob das Programm auch läuft. Ein „sicherer" Weg, Fehler einzuschleusen, ist das Einsetzen von Anweisungen innerhalb mehrfach verschachtelter Schleifen. Genauso fehlerträchtig ist es, wenn versäumt wird, beim Schreiben von IF. . .THEN-Anweisungen alle möglichen Konditionen zu beachten. Dies ist besonders dann der Fall, wenn nach IF. . .THEN mehrere Anweisungen folgen. Hier ein Beispiel:

```
655 IF A$=" " THEN GOTO 980:A$=B$
660 PRINT A$
```

Die Anweisung A$=B$ wird nie ausgeführt werden. Entweder ignoriert der Computer A$=" " und das Programm springt auf Zeile 980, oder aber A$< >" ", dann wird der Rest der Zeile 655 nicht beachtet.

Erfahrung ist der beste Lehrer bei der Fehlersuche. Schrittweises Vorgehen und strenge Methodik sind jedoch wertvolle Hilfen.

Ungeziefer

Für Programmierneulinge scheinen Fehler häufig lebendige Wesen zu sein.
Der Fachjargon „debugging" für Fehlersuche wird tatsächlich auf „Ungeziefer" zurückgeführt. Der Ausdruck entstand, als Captain Grace Hopper bei der Suche nach einem Programmfehler einen großen Nachtfalter entdeckte, der sich im elektromechanischen Teil des Computers verfangen hatte und Ursache des „Fehlers" war. Der Ausdruck „debugging" für Fehlersuche war geboren.

Do the LOGOmotion

In diesem Teil des LOGO-Kurses stellen wir Ihnen ein kleines Weltraum-Abenteuer vor, das durch verschiedene Ein- und Ausgaben gesteuert wird.

Stellen Sie sich vor, daß sich die Turtle im weiten Weltraum verloren hat – viele Lichtjahre entfernt von der Heimbasis, zu der Sie sie nun zurückbringen müssen. Diese „Rettungsaktion" setzt viel Programmierarbeit und vor allem zahlreiche Dateneingaben voraus.

Wie Sie bereits wissen, gibt die Anweisung PRINT "HALLO das Wort HALLO auf dem Bildschirm aus. PRINT" dagegen wird benutzt, wenn ein „leeres Wort" (ein Wort, das keine Zeichen beinhaltet) dargestellt werden soll. Das Ergebnis dieses Befehls ist also eine leere Zeile. Sind jedoch mehrere Worte auszugeben, ist der dem PRINT folgende Text (Liste) in ekkige Klammern zu schreiben.

```
PRINT [DIE ZEIT IST ABGELAUFEN]
```

Zudem kann mit PRINT der Inhalt einer Variablen angezeigt werden, etwa PRINT :PUNKTE, wodurch der in "PUNKTE abgelegte Wert auf dem Schirm erscheint. Texte und Variablenwerte lassen sich aber auch in einer PRINT-Anweisung kombinieren, indem der gesamte Ausdruck in runde Klammern gesetzt wird.

```
(PRINT [SIE ERREICHTEN] :PUNKTE)
```

PRINT1 hat die gleiche Funktion wie PRINT. Der Cursor bleibt allerdings nach Beendigung der Textausgabe auf der letzten Position stehen und springt nicht in die nächste Zeile.

```
PRINT1 [GEBEN SIE IHREN NAMEN EIN?]
```

Nun zu den Ausgabeanweisungen. Der Befehl PRINT XCOR zum Beispiel gibt die X-Koordinatenposition aus, auf der sich die Turtle gerade befindet. Beträgt der gegenwärtige Wert von XCOR 20, dann erscheint nach Eingabe von PRINT XCOR als Antwort 20. Geben Sie jedoch nur XCOR ein, wird die Meldung RESULT: 20 (bei manchen LOGO-Versionen auch YOU DON'T SAY WHAT TO DO WITH 20) dargestellt. Tatsächlich handelt es sich hierbei um eine Fehlermeldung, da das PRINT vergessen wurde.

Die Prozeduren, die bislang vorgestellt wurden, enthielten ausschließlich Befehle. Das LOGO-Vokabular beinhaltet selbstverständlich auch mathematische Operationen, die in Verbindung mit der Anweisung OUTPUT ausgegeben werden. Das folgende Beispiel berechnet zuerst die Entfernung der Turtle von ihrer Ausgangsposition und ermittelt die Quadratwurzel dieses Wertes:

```
TO ENTFERNUNG
  OUTPUT SQRT [XCOR * XCOR + YCOR *
    YCOR]
END
```

Mit PRINT ENTFERNUNG erhalten Sie das Ergebnis.

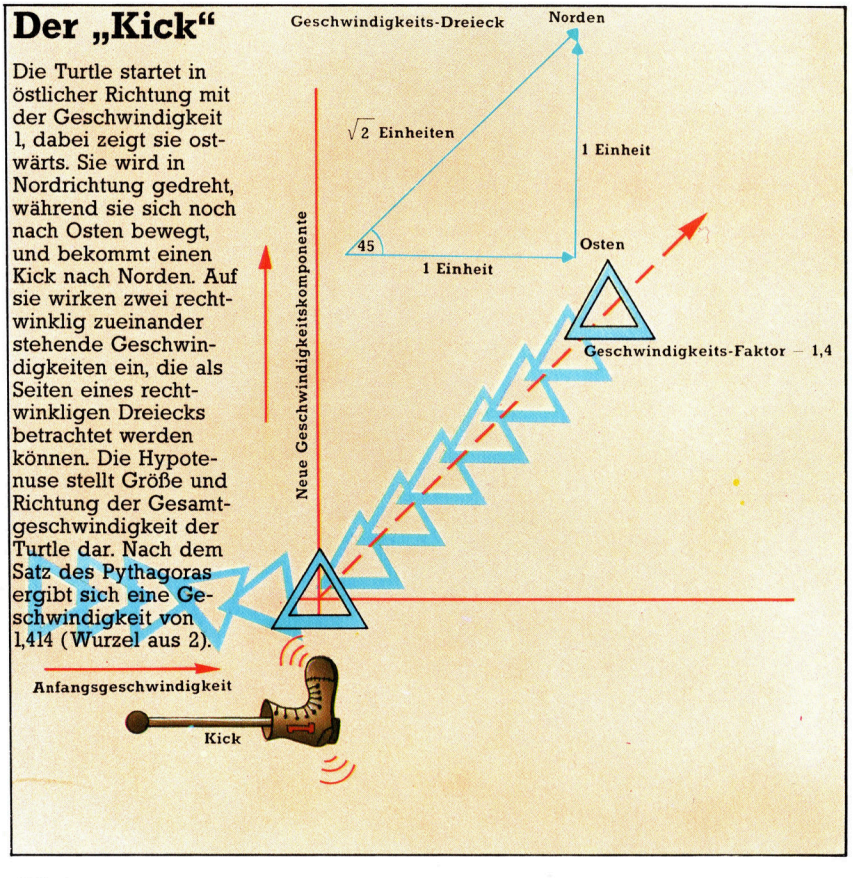
Das nächste Programm, bei dem wiederum der OUTPUT-Befehl eingesetzt wird, vergleicht zwei Zahlen und gibt den größeren Wert aus:

```
TO MAX :X :Y
  IF :X > :Y THEN OUTPUT :X
  OUTPUT :Y
END
```

Die Eingabe PRINT MAX 6 2 stellt die Zahl 6 dar. Versuchen Sie nun, eine Prozedur zu entwickeln, die den absoluten Wert einer Zahl

Der „Kick"

Die Turtle startet in östlicher Richtung mit der Geschwindigkeit 1, dabei zeigt sie ostwärts. Sie wird in Nordrichtung gedreht, während sie sich noch nach Osten bewegt, und bekommt einen Kick nach Norden. Auf sie wirken zwei rechtwinklig zueinander stehende Geschwindigkeiten ein, die als Seiten eines rechtwinkligen Dreiecks betrachtet werden können. Die Hypotenuse stellt Größe und Richtung der Gesamtgeschwindigkeit der Turtle dar. Nach dem Satz des Pythagoras ergibt sich eine Geschwindigkeit von 1,414 (Wurzel aus 2).



(ohne Vorzeichen) ausgibt. Eingaben wie PRINT ABS 4 und PRINT ABS (−4) müssen als Resultat nur die 4 anzeigen.

Bei der folgenden Prozedur wird nach dem Namen gefragt. Die Eingabe ist mit RETURN zu beenden.

```
TO EING.NAME
  SPLITSCREEN
  PRINT1 [GEBEN SIE IHREN NAMEN EIN?]
  MAKE "NAME FIRST REQUEST
  (PRINT "HALLO :NAME)
END
```

Nachdem die Eingabe abgeschlossen und die Prozedur ausgerufen wurde, veranlaßt FIRST die Darstellung des ersten Elementes der Liste (NAME).

Nun zu unserem Spiel: Mit Hilfe der Tasten R, L und K können Sie die Bewegung der Turtle steuern, wobei R bewirkt, daß sie sich im Uhrzeigersinn um 30 Grad nach rechts bewegt und bei L um den gleichen Winkel nach links. Mit K dagegen geben Sie der Turtle einen „Kick". Das heißt, die Geschwindigkeit, mit der sie sich in der vorgegebenen Richtung bewegt, wird erhöht. Und so sieht das Listing aus:

```
TO COMMAND
  MAKE "COM READKEY
  IF :COM = "R THEN RIGHT 30
  IF :COM = "L THEN LEFT 30
  IF :COM = "K THEN KICK
END
```

Leider gibt es den Befehl READKEY, der in einigen anderen Programmiersprachen enthalten ist, bei LOGO nicht. Mit den folgenden Zeilen wird jedoch diese Funktion simuliert:

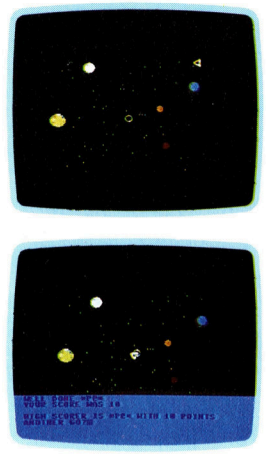
```
TO READKEY
  IF RC? THEN OUTPUT READCHARACTER
  OUTPUT "
END
```

Sobald Sie eine Taste drücken, wird das jeweilige Zeichen in einem Buffer abgelegt. READCHARACTER liest einfach das letzte Zeichen aus diesem Buffer und ordnet es dem entsprechenden Befehl zu. Die Bedingungsabfrage (IF RC? THEN . . .) ist „wahr", wenn sich im Buffer Zeichen befinden, und „falsch", wenn dieser geleert ist.

Dynaturtle

Die Hauptrolle in unserem Weltraum-Abenteuer wird die sogenannte „Dynaturtle" übernehmen, die exakt den Newtonschen physikalischen Gesetzen folgt. Das Bewegungsschema läßt sich am besten anhand der Illustration erklären. Angenommen, die Turtle läuft mit dem Geschwindigkeitswert 1 von links nach rechts über den Bildschirm. Drückt man jetzt die L-Taste, dreht sich die Dynaturtle zum oberen Bildschirmrand und führt die Bewegung weiter aus. Mit K können Sie nun die Geschwindigkeit auf den Wert 1,4 steigern — die Richtung bleibt jedoch konstant. Dieses Programm eignet sich hervorragend zum Experimentieren, da Sie die Eingaben beliebig verändern können.

Die Faktoren, die zur Berechnung der Geschwindigkeit auf den X- und Y-Koordinaten führen, werden anhand der SIN- und COS-Funktion ermittelt. Beginnen Sie das Spiel mit der START-Eingabe. Der jeweilige Punktestand wird mit Hilfe einer Unterroutine errechnet und angezeigt. Das Programm gibt Ihnen eine festgelegte Zeit vor, in der Sie das Spielziel erreichen müssen. Die Turtle-Steuerung erfolgt über die Tasten L,R und K.



Das Programm stellt die Turtle mit ihrer Heimbasis dar. Die abgebildeten Planeten entstehen durch zusätzliche Kreis-Unterroutinen.

LOGO-Dialekte

MIT LOGO	LCSI LOGO
DRAW	CS
PRINT1	TYPE
RC?	KEYP
READCHARACT.	RC
REQUEST	RL
SETHEADING	SETH
SETXY	SETPOS
FULLSCREEN	FS (nicht vorhanden beim Spectrum-LOGO)
SPLITSCREEN	SS (nicht vorhanden beim Spectrum-LOGO)

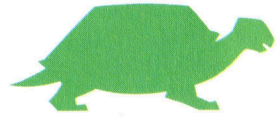
IF variiert bei einigen LOGO-Versionen, z. B.:
IF DISTANCE < 5 [DONE STOP]

Übungen

Entwickeln Sie nun ein Weltraum-Spiel, das folgende Merkmale aufweist:
Sie steuern ein Raumschiff, das über der Mondoberfläche entlangfliegt. Der Treibstoffvorrat ist limitiert, und durch die Anziehungskraft verlieren Sie permanent an Energie. Durch Drücken der F-Taste erhöht sich die Sinkgeschwindigkeit, jedoch reduziert sich dadurch gleichzeitig der Treibstoffvorrat. Ziel ist es, das Schiff (mit der F-Taste) sicher zu landen. Guten Flug!

Abkürzungen

OUTPUT	OP
PRINT	PR
READCHARACTER	RC
REQUEST	RQ
SETHEADING	SETH



Lösungen

1. Verschachtelte Dreiecke

```
TO TRI :SIZE :LEVEL
   IF :LEVEL = 0 THEN REPEAT 3 [FD :SIZE RT 120]
      STOP
   TRI ( :SIZE / 2 ) ( :LEVEL — 1 )
   FD ( :SIZE / 2)
   TRI ( :SIZE / 2 ) ( :LEVEL — 1 )
   RT 60
   TRI ( :SIZE / 2 ) ( :LEVEL — 1 )
   FD ( :SIZE / 2 )
   RT 60
   TRI ( :SIZE / 2 ) ( :LEVEL — 1 )
   LT 60
   BK ( :SIZE / 2 )
   LT 60
   BK ( :SIZE / 2 )
END
```

2. Schneeflocken-Kurve

```
TO SNOW 1:SIZE :LEVEL
   REPEAT 4 [SIDE1 :SIZE :LEVEL RT 90]
END
TO SIDE1 :SIZE :LEVEL
   IF :LEVEL = 0 THEN FD :SIZE STOP
   SIDE1 ( :SIZE / 3 ) ( :LEVEL — 1 )
   LT 90
   SIDE1 ( :SIZE / 3 ) ( :LEVEL — 1 )
   RT 90
   SIDE1 ( :SIZE / 3 ) ( :LEVEL — 1 )
   RT 90
   SIDE1 ( :SIZE / 3 ) ( :LEVEL — 1 )
   LT 90
   SIDE1 ( :SIZE / 3 ) ( :LEVEL — 1 )
END
```

3. Kurve ohne Gradienten

```
TO W :XSTEP :YSTEP :LEVEL
   WUP :XSTEP :YSTEP :LEVEL
   WDOWN :XSTEP :YSTEP :LEVEL
END

TO WUP :XSTEP :YSTEP :LEVEL
   IF :LEVEL = 0 THEN SETXY ( XCOR + :XSTEP )
      ( YCOR + :YSTEP ) STOP
   WUP ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1)
   WDOWN ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1 )
   WUP ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1 )
   WUP ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1 )
   WDOWN ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1 )
   WUP ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1 )
END

TO WDOWN :XSTEP :YSTEP :LEVEL
   IF :LEVEL = 0 THEN SETXY ( XCOR + :XSTEP )
      ( YCOR — :YSTEP ) STOP
   WDOWN ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( : LEVEL — 1)
   WUP ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1 )
   WDOWN ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1 )
   WDOWN ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1 )
   WUP ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( : LEVEL — 1 )
   WDOWN ( :XSTEP / 6 ) ( :YSTEP / 2 ) ( :LEVEL — 1 )
END
```

Weltraum-Spiel

```
TO START
   MAKE "MAX 0
   MAKE "BEST "
   DRAW
   HT
   TARGET
   PLAY
END

TO TARGET
   PU SETXY 0 5 PD
   RT 90
   REPEAT 36 [FD 31.4/36 RT 10]
   PU
END

TO PLAY
   GET.NAME
   INIT
   DRIVE
END

TO GET.NAME
   SPLITSCREEN
   PRINT1 [WHAT IS YOUR NAME?]
   MAKE "NAME FIRST REQUEST
END

TO INIT
   MAKE "SCORE 200
   SETXY 100 100
   SETH 270
   MAKE "XVEL 0
   MAKE "YVEL 0
   FULLSCREEN
   ST
END

TO DRIVE
   COMMAND
   DYNA.MOVE
   IF DISTANCE <5 THEN DONE STOP
   MAKE "SCORE :SCORE - 1
   IF :SCORE = 0 THEN OUT.OF.TIME
   STOP
   DRIVE
END

TO COMMAND
   MAKE "COM READKEY
   IF :COM = "R THEN RIGHT 30
   IF :COM = "L THEN LEFT 30
   IF :COM = "K THEN KICK
END

TO READKEY
   IF RC? THEN OUTPUT
   READCHARACTER
   OUTPUT "
END

TO KICK
   MAKE "XVEL + 3 * SIN HEADING
   MAKE "YVEL + 3 * COS HEADING
END

TO DYNA.MOVE
   SETXY XCOR + :XVEL
   YCOR + :YVEL
END

TO DISTANCE
   OUTPUT SORT
   (XCOR * XCOR + YCOR * YCOR)
END

TO DONE
   PRINT"
   SPLITSCREEN
   (PRINT [WELL DONE] :NAME)
   (PRINT [YOUR SCORE WAS]
   :SCORE)
   REPORT
   AGAIN
END

TO REPORT
   IF :SCORE > :MAX THEN MAKE
   "MAX :SCORE MAKE "BEST :NAME
   PRINT "
   (PRINT [HIGH SCORER IS] :BEST
   [WITH] : MAX [POINTS])
END

TO AGAIN
   PRINT1 [ANOTHER GO?]
   MAKE "ANS FIRST REQUEST
   IF :ANS = "YES THEN REPLAY
   STOP
   IF :ANS = "NO THEN STOP
   PRINT [MAKE YOUR MIND UP, YES
   OR NO?]
   AGAIN
END

TO OUT.OF.TIME
   PRINT "
   SPLITSCREEN
   PRINT [YOUR TIME HAS RUN OUT]
   AGAIN
END

TO REPLAY
   HT
   GET.NAME
   INIT
   DRIVE
END
```

Alan Turing

Der britische Mathematiker entwickelte einen allgemein anerkannten Test zur Feststellung künstlicher Intelligenz. Während des Zweiten Weltkrieges arbeitete er als Codeknacker für den Geheimdienst.

Schon in sehr jungen Jahren zeigte Alan Turing eine bemerkenswerte Begabung für Naturwissenschaften. Er hatte gerade erst Lesen und Schreiben gelernt, als er bereits magische Zahlen in Primfaktoren zerlegte und einen Entwurf für ein „schwimmfähiges Fahrrad" erstellte. Jahre später, während sein Vater im fernen Madras bei der indischen Verwaltung tätig war, erhielt Alan ein Stipendium für das King's College in Cambridge. Sein ganzes Interesse konzentrierte sich nunmehr auf die Probleme mathematischer Logik – zunächst noch als Student, später als wissenschaftlicher Mitarbeiter der dortigen Hochschule.

Im Jahre 1931 verblüffte der tschechische Mathematiker Kurt Gödel die Fachwelt mit der Entdeckung wahrer, aber nicht beweisbarer mathematischer Lehrsätze. Alan Turing begann jedoch mit der Untersuchung beweisbarer Lehrsätze. Er entwarf das Modell einer Maschine, das auf mechanische Weise Operationen durchführen konnte – Arbeiten, die sonst von Mathematikern verrichtet werden. So entstanden Geräte zum Addieren, zum Integrieren, für die Division und für andere Rechenarten. Die Denkmodelle wurden später unter der Bezeichnung „Turing-Maschinen" bekannt.

Turing untersuchte die Funktionsweise dieser imaginären Geräte und kam zu einem bemerkenswerten Schluß. Statt für jede mathematische Operation einen eigenen Rechner bereitzustellen, mußte es möglich sein, einen „Universalrechner" zu entwerfen, der durch entsprechende Programmierung jeden „Spezialrechner" ersetzen kann.

Als der Zweite Weltkrieg ausbrach, wurde Turing von der Universität zur „Government School of Codes and Ciphers" in Bletchley Park, Buckinghamshire, eingezogen. Ohne den Krieg wären seine Maschinen vielleicht nur trockene Theorie geblieben, aber Bletchley Park war mit der dringenden und streng geheimen Aufgabe beschäftigt, die deutschen Militär-Codes zu knacken.

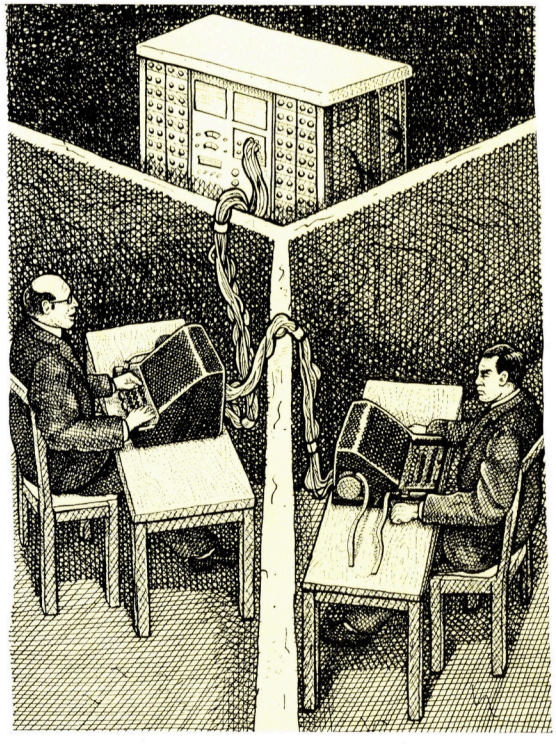
Weil diese jeden Tag geändert wurden, benötigte man Maschinen, um den jeweils neuesten Schlüssel zu finden. Bletchley Park wuchs zu einem riesigen Zentrum für Informationsverarbeitung. Alan Turing wurde während des Krieges auch nach Amerika geschickt, um für die transatlantischen Nachrichtenverbindungen der Alliierten Streitkräfte sichere Codes zu entwickeln.

Da seine Arbeit zu dieser Zeit geheim war, sind nur wenige Berichte über seine Reisen vorhanden. Es wird jedoch allgemein angenommen, daß er während seines Aufenthalts in Princeton, New Jersey, mit John von Neumann zusammentraf. Gegen Ende des Krieges wurde Turing gebeten, einen britischen Computer mit Namen ACE für das National Physical Laboratory zu entwickeln.

Die „Automatical Computing Engine" (Automatisch rechnende Maschine) wurde zu Ehren von Charles Babbage „Analytical Engine" benannt. Wie bei diesem Vorläufer der Computertechnik dauerte auch der Bau von ACE sehr lange, aber der Rechner war dem amerikanischen ENIAC in vielerlei Hinsicht weit voraus. Enttäuscht vom schleppenden Fortgang der Arbeiten ging Turing resigniert nach Manchester, wo er sich am Computerprojekt der Universität beteiligte. Zur gleichen Zeit wurde er Berater der Ferranti-Gesellschaft und beschäftigte sich in der Folgezeit mit den ersten Computern, die in Großbritannien gebaut wurden.

Ein Freund sagte einmal, daß Turing, was die Fehler bei anderen angehe, eher großzügig verfahren sei, sein eigenes wissenschaftliches Genie dagegen war unbestritten. 1952 wurde er wegen homosexueller Delikte verurteilt und beging zwei Jahre später Selbstmord.

Alan Turing (1912–1954) fand beim Langstreckenlauf Entspannung. Er war interessiert an den Auswirkungen körperlicher Anstrengung auf die Kreativität und die geistige Beweglichkeit.



Turing entwickelte das „Nachahmungsspiel", das später als Turing-Test bekannt wurde. Eine Person wird in einen Raum gebracht, in dem sich ein Fernschreiber befindet. Dieser ist mit einem weiteren Fernschreiber im Nebenraum verbunden, der ebenfalls von einer Person oder von einem Computer bedient wird. Die erste Person stellt eine beliebige Frage. Wenn nicht eindeutig zu bestimmen ist, wann mit einem Menschen und wann mit der Maschine kommuniziert wurde, kann die Maschine als intelligent bezeichnet werden.

Alles auf Band

Die Turing-Maschine ist ein rein theoretisches Hilfsmittel, um herauszufinden, ob ein Problem per Computer gelöst werden kann oder nicht.

Computerwissenschaften haben für die elektronische Datenverarbeitung die gleiche Bedeutung wie die reine Mathematik für den Ingenieurwissenschaftler – es handelt sich um theoretische Verfahren, die jedoch letztendlich die Grundlage für praktische Anwendungen darstellen.

Die Turing-Maschine ist beispielsweise ein theoretisches Denkmodell, entwickelt von Alan Turing für seine Studien mit Algorithmen und der Problem-Berechenbarkeit. Ein Problem läßt sich als „nicht berechenbar" bezeichnen, wenn man beweisen kann, daß es nicht mit der Turing-Maschine gelöst werden kann. Der Mathematiker stellte fest, daß ein solcher Minimalcomputer drei Teile benötigt: einen externen Speicher zur Aufzeichnung und Speicherung von Ein- und Ausgaben, die Mittel, um diesen Speicher zu beschreiben und zu lesen, und eine Kontrolleinheit, um die durchzuführenden Operationen zu bestimmen.

Eine Turing-Maschine wird daher üblicherweise definiert als ein Band (beispielsweise ein Magnetband) und als Kopfmechanismus, der die Symbole lesen oder schreiben kann und der das Bild bewegt. Dabei erhält er von einer Kontrolleinheit Anweisungen, welche Symbole geschrieben werden sollen.

Die Kontrolleinheit enthält ein Ausführungsprogramm, und in dieser Beziehung kann eine Turing-Maschine als speziell für eine bestimmte Anwendung „gebaut" bezeichnet werden, da in der Beschreibung keine Möglichkeit vorgesehen ist, ein Programm zu laden oder zu ändern. Dabei wird absichtlich die Schreibweise „gebaut" verwendet, weil die wenigen Turing-Maschinen, die wirklich konstruiert worden sind, ausschließlich zu Lehrzwecken dienten. Es ist jedoch ziemlich einfach, ein BASIC-Programm zu schreiben, das die Arbeit einer Turing-Maschine auf einem Heimcomputer simulieren kann.

Das Kontrollprogramm in einer Turing-Maschine besteht aus einer Reihe von „Fünf-Tupeln". Dies sind Ausdrücke, die fünf Elemente enthalten. Welcher Tupel in einem bestimmten Zustand ausgeführt wird, hängt von zwei Faktoren ab: Dem Symbol in dem Quadrat, das sich gerade unter dem Bandkopf befindet, und dem „Zustand" der Maschine. Dieser Zustand ist ein völlig willkürliches Kriterium: Sie können festlegen, daß die Maschine im Zustand S(A) startet und anhält, wenn sie den besonderen Zustand H erreicht und die Berechnung beendet ist. In der Zwischenzeit wird sich der Zustand gemäß den Anweisungen der Fünf-Tupel viele Male ändern.

Die fünf Elemente jedes Tupels sind:

1) Der augenblickliche Zustand der Maschine.
2) Das Symbol in dem Bandquadrat unter dem Kopf.
3) Das Symbol, das in dieses Quadrat geschrieben werden soll (das gleiche wie 2, wenn keine Änderung der Daten verlangt wird).
4) Der Zustand, in den die Maschine als nächstes übergehen sollte.
5) Die Richtung, in die der Bandkopf bewegt werden soll – links oder rechts.

Der Tupel (S(A), 5, 3, S(B), R) zum Beispiel wird immer dann ausgeführt werden, wenn die Maschine im Zustand S(A) ist und der Bandkopf eine 5 liest. Die 5 wird dann zu einer 3 verändert. Die Maschine geht über in den Zustand S(B) und der Kopf wird um ein Quadrat nach rechts bewegt.

Modell-Entwurf

Um eine Turing-Maschine zu entwerfen, die eine konkrete Aufgabe erfüllen soll, müssen Sie das Format angeben, in dem die Eingabedaten der Maschine auf dem Band geliefert werden, das Format der Ausgabedaten auf dem Band nach Beendigung der Rechnung (wenn die Maschine, in unserem Beispiel, im Zustand H ist) und die Fünf-Tupel, die erforderlich sind, um den Algorithmus auszuführen.

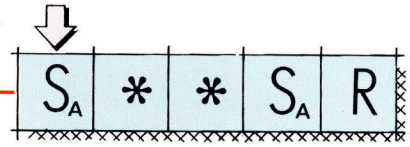
Im Kasten sehen Sie eine Turing-Maschine, die die AND-Funktion berechnen soll. Sie müssen die zwei Eingabebits (jedes eine 1 oder eine 0) in aufeinanderfolgende Quadrate schreiben, gefolgt durch ein Fragezeichen-Symbol, das durch die Antwort (wieder eine 1 oder eine 0, abhängig von der Eingabe) ersetzt wird. Zur Dekoration wurde ein Sternchen-Symbol an beiden Enden des Datenblocks angefügt. Sie starten die Maschine im Zustand S(A) auf dem linken Sternchen, diese stoppt auf dem rechten.

Insgesamt zehn Fünf-Tupel werden benötigt, um diese Maschine zu definieren, obwohl, wie Sie in dem abgearbeiteten Beispiel sehen können, für jeden Lauf nur fünf gebraucht werden. Wenn Sie mit derselben Maschine einen Versuch mit 0 AND 1 machen, werden Sie feststellen, daß ein anderer Teil von Tupeln aus den zehn ausgewählt wird.

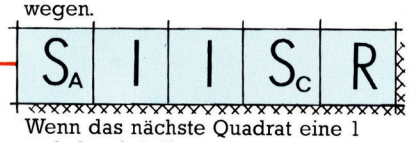
Turing-Maschine

Dieses Beispiel zeigt eine Turing-Maschine zur Berechnung der AND-Funktion. Die beiden Eingabebits werden in aufeinanderfolgende Quadrate geschrieben, gefolgt von einem Fragezeichen, das durch das Ergebnis ersetzt wird. Zwei Sternchen werden als Begrenzer an die Enden des Datenblocks plaziert. Die zehn Fünf-Tupel unten definieren die Operationen der Maschine, obwohl für jedes bearbeitete Beispiel (in diesem Fall 1 AND 1) nur fünf von zehn gebraucht werden.

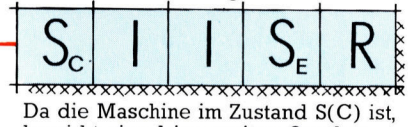
S_A	*	*	S_A	R
S_A	0	0	S_B	R
S_A	1	1	S_C	R
S_B	0	0	S_D	R
S_B	1	1	S_D	R
S_C	0	0	S_D	R
S_C	1	1	S_E	R
S_D	?	0	S_F	R
S_E	?	1	S_F	R
S_F	*	*	H	R



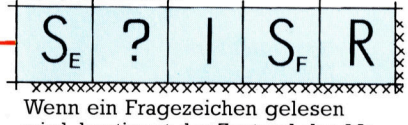
Die Maschine startet im Zustand S(A) mit dem Kopf über dem linken Stern. Der Effekt des Tupels ist, den Bandkopf einen Schritt nach rechts zu bewegen.



Wenn das nächste Quadrat eine 1 enthält, wird dieser Tupel angewählt: Die Maschine geht in den Zustand S(C) und wird angewiesen, das Band nach rechts zu bewegen.

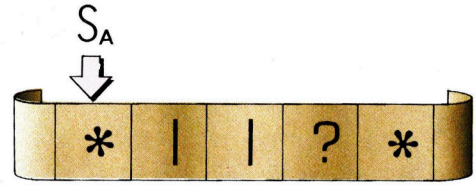


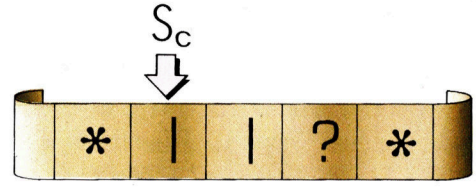
Da die Maschine im Zustand S(C) ist, bewirkt eine 1 im zweiten Quadrat den Zustand S(E). In allen anderen Fällen würde die Maschine Zustand S(D) annehmen.

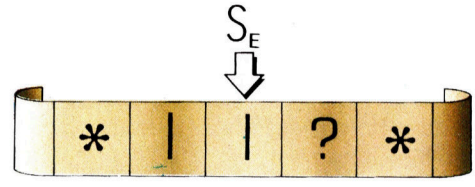


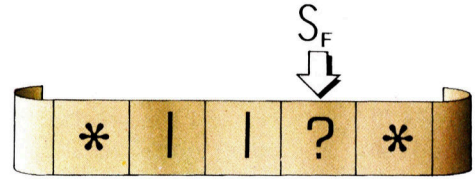
Wenn ein Fragezeichen gelesen wird, bestimmt der Zustand der Maschine, S(E) oder S(D), ob eine 0 oder eine 1 als Ergebnis an seine Stelle geschrieben wird.

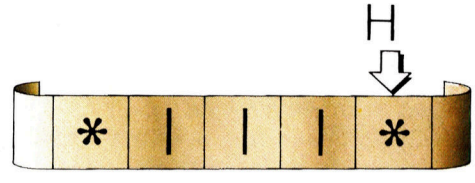
Die Maschine geht nur über dem zweiten Sternchen in den Haltezustand H. Sie können die Operation auf dem Papier für 1 AND 0, 0 AND 1 und 0 AND 0 ausprobieren.

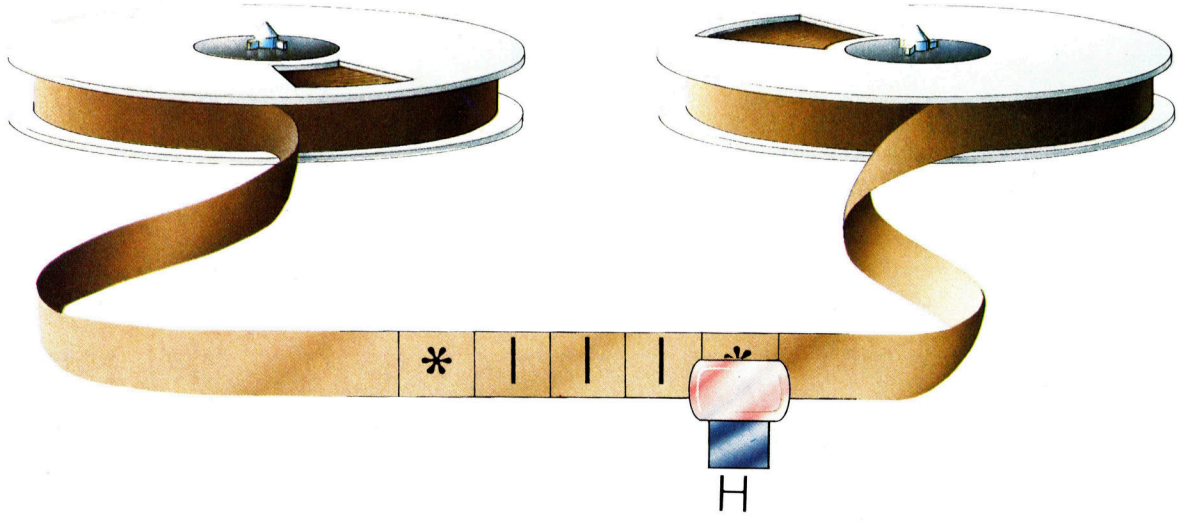












Einhand-Schreiber

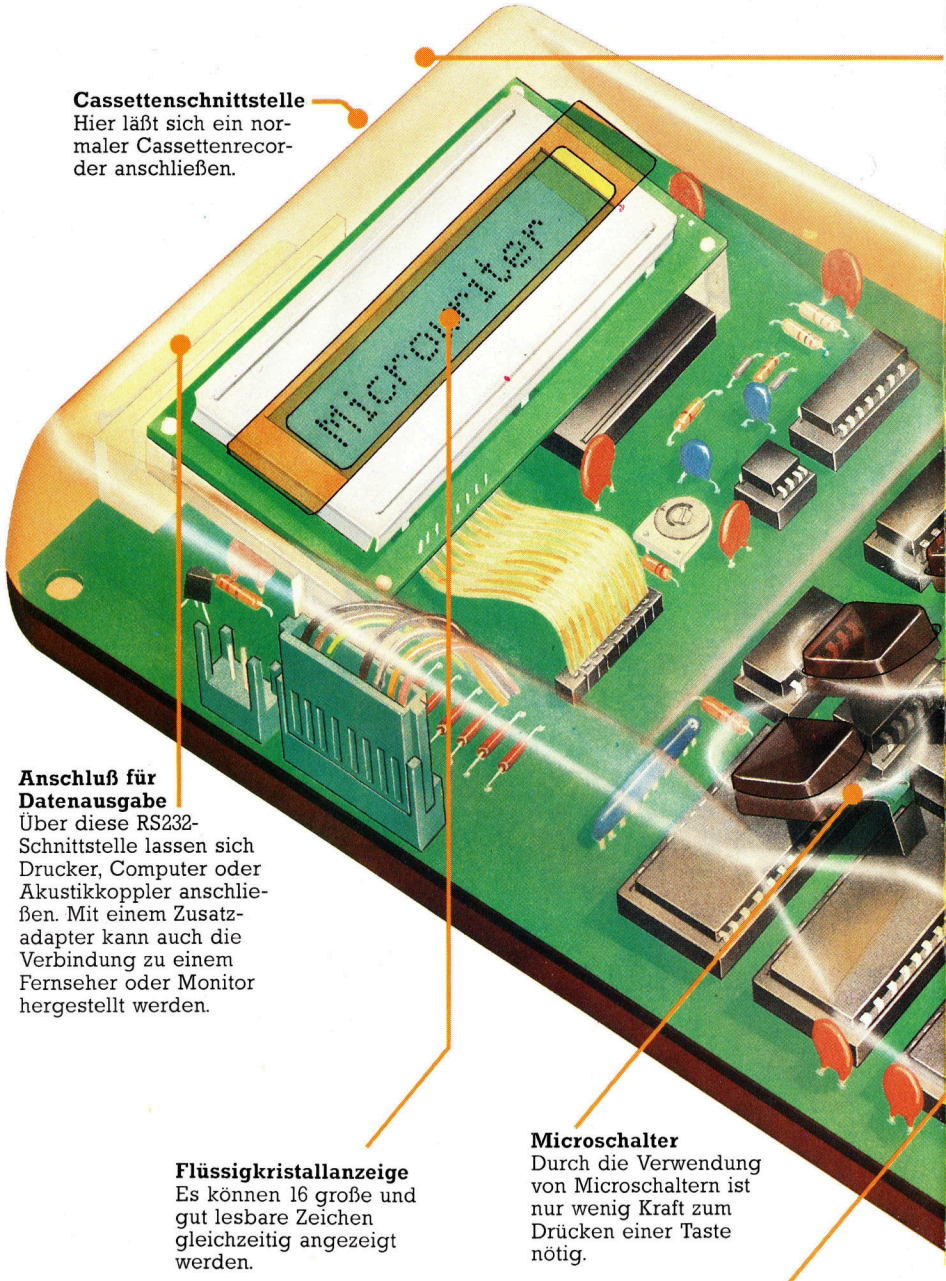
Der „Microwriter" ist ein tragbares Textsystem, das mit einer Hand bedient wird. Nur sechs Tasten sind für die Eingabe nötig.

Mit einem Textsystem – ob im Bürocomputer oder auf einem Heimgerät – läßt sich der Zeitaufwand für die täglich anfallenden Schreibarbeiten auf ein Minimum reduzieren. Auch die Dokumentation von Programmen, das Erfassen von Notizen und Führen von Adreßregistern kann damit sehr vereinfacht werden.

Der Markt für tragbare Computersysteme wächst ständig. Doch obwohl diese Geräte als Terminals von größeren Systemen oder als tragbare Textverarbeitung eingesetzt werden können, sind sie immer noch weitaus größer als ein Notizblock oder ein Diktiergerät. Wie wäre es deshalb mit einem Textsystem in Taschenformat, das mit Batterien arbeitet, mit nur einer Hand bedient wird und außerdem an einen Drucker oder sogar an einen Computer angeschlossen werden kann?

Ein Gerät dieser Art gibt es bereits seit mehreren Jahren. Es heißt „Microwriter" und wurde von dem Amerikaner Cy Endfield entwickelt. Der Microwriter verzichtet auf die übliche Tastatur und besitzt statt dessen ein einzigartiges System mit nur sechs Tasten, bei dem zur Erzeugung eines Buchstabens mehrere Tasten gleichzeitig gedrückt werden. Die Idee entwickelte sich aus der Konstruktion eines LCD-Minispiels, das durch Eingabe von Worten gesteuert werden sollte. Selbst eine Miniaturtastatur wäre dafür zu groß und zu teuer gewesen. Es wurde deshalb eine Spezialtastatur entwickelt, die zwar nur wenige Tasten besaß, deren Kombinationen aber alle alphanumerischen Zeichen ansprechen konnten. Die Erfindung eines symbolischen, nur für den Microwriter entwickelten Codes verhalf schließlich zum Durchbruch.

Auf den ersten Blick scheint es unmöglich, alle Buchstaben des Alphabets mit nur sechs Tasten zu erzeugen – ganz zu schweigen von Zahlen und Satzzeichen. Diese wenigen Tasten reichen jedoch völlig aus, und der Code läßt sich in einigen Stunden lernen. Der Hersteller behauptet sogar – mit einiger Berechtigung –, daß dieses System besonders von Anwendern, die mit der Schreibmaschinen-Tastatur nicht vertraut sind, leicht erlernt werden kann, da die Tastenkombinationen der Form der Zeichen entsprechen.



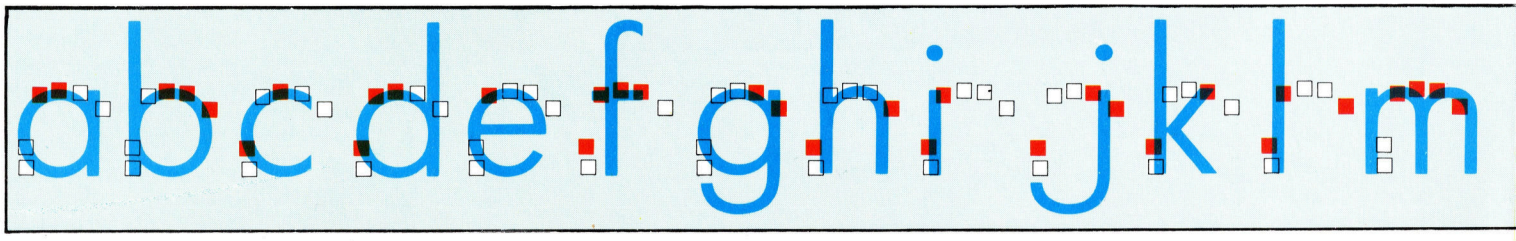
Cassettenschnittstelle
Hier läßt sich ein normaler Cassettenrecorder anschließen.

Anschluß für Datenausgabe
Über diese RS232-Schnittstelle lassen sich Drucker, Computer oder Akustikkoppler anschließen. Mit einem Zusatzadapter kann auch die Verbindung zu einem Fernseher oder Monitor hergestellt werden.

Flüssigkristallanzeige
Es können 16 große und gut lesbare Zeichen gleichzeitig angezeigt werden.

Microschalter
Durch die Verwendung von Microschaltern ist nur wenig Kraft zum Drücken einer Taste nötig.

Arbeitsspeicher
Das Gerät wird mit einem Standard von 8 K RAM geliefert, kann aber mit größeren Chips, die in die gleichen Steckleisten passen, aufgerüstet werden.

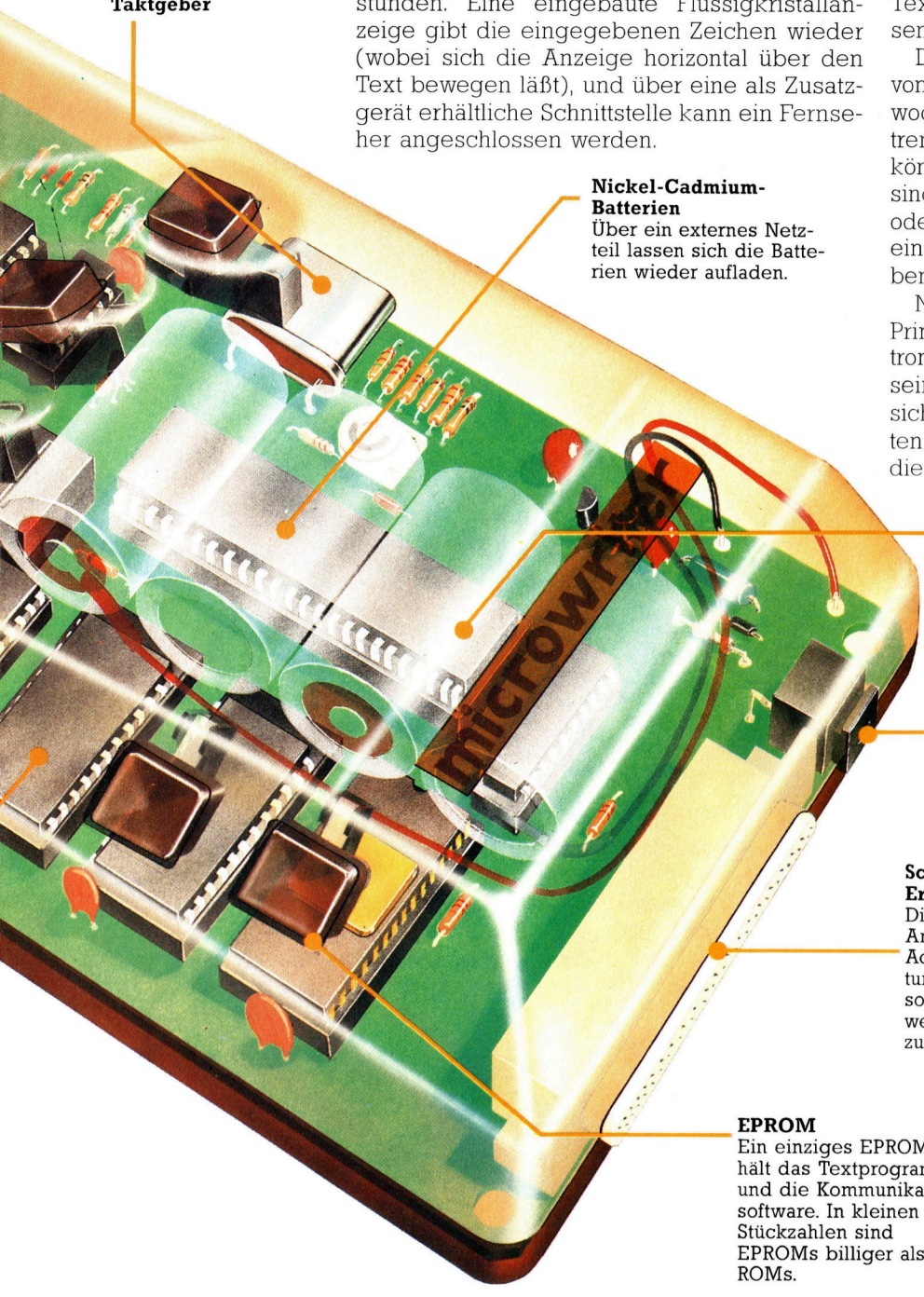


Der wichtigste Punkt bei der Konstruktion des Microwriters betrifft die „Tragbarkeit". Für den internen Microprozessor und den Speicher wurden CMOS-Elemente (Halbleiterbauelemente in MOS-Technologie mit komplementärem Aufbau) verwandt, die nur wenig Strom verbrauchen. Nickel-Cadmium-Batterien liefern genügend Elektrizität für 30 Betriebsstunden. Eine eingebaute Flüssigkristallanzeige gibt die eingegebenen Zeichen wieder (wobei sich die Anzeige horizontal über den Text bewegen läßt), und über eine als Zusatzgerät erhältliche Schnittstelle kann ein Fernseher angeschlossen werden.

Außer der seriellen Schnittstelle RS232 für den Anschluß eines Druckers besitzt der Microwriter eine Schnittstelle, über die im Gerät gespeicherte Texte auf Cassetten abgelegt oder wieder davon eingelesen werden können. Mit der seriellen Schnittstelle läßt sich der Microwriter als „Einhand-Terminal" an einen Computer oder ein Textsystem anschließen. Texte, die unterwegs eingegeben wurden, lassen sich so in andere Systeme einspeisen.

Der Speicher des Microwriters läßt sich in voneinander getrennte Bereiche unterteilen, wodurch mehrere unterschiedliche Texte getrennt eingegeben und bearbeitet werden können. Auch einfache Editiermöglichkeiten sind vorhanden: Texte lassen sich zufügen oder löschen, und größere Blöcke können über eine Zwischenspeicherung im Buffer verschoben werden.

Nach der Intention des Erfinders sollte das Prinzip des Microwriters auch in anderen elektronischen Geräten Verwendung finden. Trotz seiner vielen guten Eigenschaften verkaufte sich das Gerät nur mäßig. Es bleibt abzuwarten, ob die Hersteller von Heimcomputern diese Idee aufgreifen werden.



Hauptschalter
Selbst beim Abschalten des Gerätes sind die Daten nicht verloren; nach dem Anschalten kann sofort damit weitergearbeitet werden.

Interner Quarz-Taktgeber

Nickel-Cadmium-Batterien
Über ein externes Netzteil lassen sich die Batterien wieder aufladen.

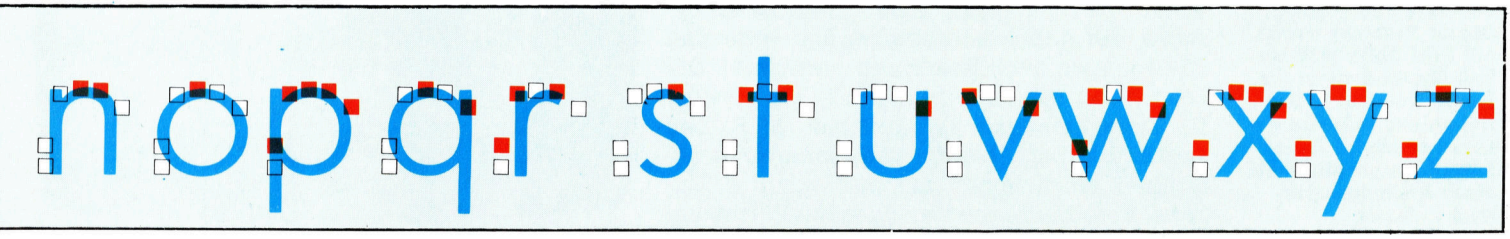
Hauptprozessor
CPU und RAM sind aus CMOS (Complimentary Metal Oxide Silicon)-Bauelementen aufgebaut, die den Stromverbrauch stark herabsetzen.

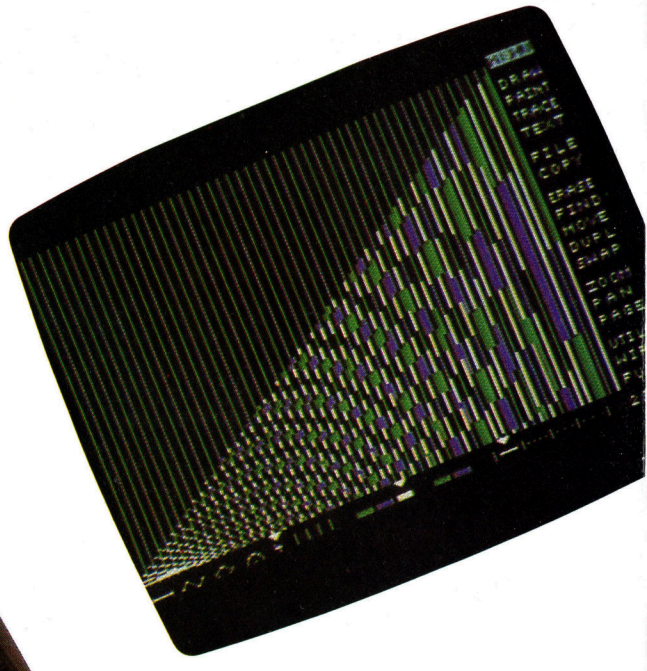
Stromanschluß
Hier läßt sich ein externes Netzgerät für den Netzbetrieb oder zum Aufladen der Batterien anschließen.

Schnittstelle für Erweiterungen
Dieser Ausgang enthält Anschlüsse für die Adressen- und Datenleitungen des Microprozessors, um zukünftige Erweiterungen des Gerätes zu ermöglichen.

EPROM
Ein einziges EPROM enthält das Textprogramm und die Kommunikationssoftware. In kleinen Stückzahlen sind EPROMs billiger als ROMs.

In dem Handbuch des Microwriters helfen Abbildungen und Merksätze, mit den unterschiedlichen Tastenkombinationen des Alphabets vertraut zu werden. Über die sechste Taste werden Satzzeichen und Editierbefehle eingegeben.





Grafik-System Robo 1000

Spezielle Software und die entsprechenden Peripheriegeräte ermöglichen präzise, detailgetreue Zeichnungen.

Um einen bestimmten Punkt auf dem Bildschirm anzusprechen, verwendet die Mehrzahl der Benutzer entweder Joysticks oder den Trackball. Einige dieser Peripheriegeräte sind aber nicht besonders exakt. Abhilfe schafft in diesem Falle qualitativ hochwertige Software, durch die sich Joystick und Trackball in präzise Eingabegeräte wandeln, mit denen sich saubere Grafiken erstellen lassen. Zeichnungen können zuerst in großem Maßstab angefertigt und danach auf das gewünschte Maß verkleinert werden – aus dem Peripheriegerät für Spiele wird so eine leistungsfähige „Zeichenmaschine".

Das Robo 1000 Bit Stik Grafik-System für den Apple ist ein gutes Beispiel, wie man präzise Grafiken erstellen kann: Es besteht aus der Software und speziell entwickelten Zubehörteilen einschließlich eines Präzisions-Joysticks, der neben horizontalen und vertikalen Bewegungen auch Rotationen ermöglicht. Die Leistung ist fast mit einem CAD/CAM-System für Spezialsysteme vergleichbar: 16 Farben sind einsetzbar, Ablegen und Einfügen vorgefertigter Zeichnungen auf und von der Diskette, Vergrößern von Details in der Zeichnung (Zooming) sowie Kurven- und Kreisprogramme sind möglich. Die Farbkarte und einen Farb-Monitor muß man dazu allerdings schon besitzen. Die im Standard-Apple von einem CAD/CAM-System weit entfernten Grafikmöglichkeiten werden auf diese Weise gesteigert. Zusätzliches Bonbon: Ein verständliches Handbuch, mit dem man sich schnell zurechtfindet.

Wenn man besonders exakte Zeichnungen per Computer anfertigen will, ist man meist auf relativ ungenaue Eingabegeräte angewiesen. Dieses Problem wurde mit dem Robo 1000 Bit Stik Grafik-System für Apple-Rechner gelöst. Neben der Software gehören spezielle Zubehörteile, einschließlich eines Präzisions-Joysticks, dazu.

Fachwörter von A bis Z

ALGOL
Der Heimcomputer-Benutzer hat heute eine verhältnismäßig große Auswahl an Programmiersprachen zur Verfügung, während man früher, als es praktisch nur Mainframes (Großrechner) gab, auf FORTRAN, ALGOL und COBOL angewiesen war. FORTRAN war eigentlich für Ingenieure gedacht, COBOL für Kaufleute; ALGOL (ALGOrithmic Language) dagegen war die Sprache der Naturwissenschaftler, Mathematiker und Informatiker. Daher ist ALGOL an Hochschulen auch heute noch eine der gängigsten Sprachen. Im Microcomputer-Bereich hat ALGOL jedoch kaum Verbreitung gefunden und ist nur für die wenigsten preisgünstigen Rechner verfügbar.

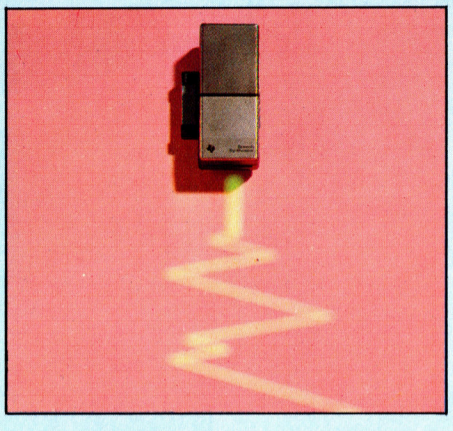
Die Stärke dieser höheren Programmiersprache liegt in der Unterprogramm-Technik: ALGOL ist ähnlich hochstrukturiert wie PASCAL und bietet dem Anwender ebenfalls eine große Anzahl fest definierter mathematisch-wissenschaftlicher Funktionen.

Algorithm = Algorithmus
Ein Algorithmus ist ganz allgemein ein mathematisches Verfahren zur Lösung eines bestimmten Problems. Algorithmen werden vorwiegend beim Programmieren eingesetzt. Manchmal besteht die ganze „Kunst" nur aus der Umsetzung eines bekannten Algorithmus (etwa für das Wurzelziehen) in die jeweilige Programmiersprache. Häufig aber liegt das Hauptproblem darin, erst einmal einen Algorithmus zur Lösung eines bestimmten mathematischen Problems zu entwickeln. Nach welchem Algorithmus finden Sie beispielsweise den Ausweg aus einem Labyrinth, oder wie muß der Algorithmus für die realistische Simulation eines Landeanflugs aussehen?

Für viele Aufgabenstellungen kann man leicht einen beliebigen Lösungsweg finden. Entscheidend ist aber, den besten ausfindig zu machen. Der eine Algorithmus führt mit einem Minimum an Speicherbedarf zum Ergebnis, ein anderer ist dafür wesentlich schneller.

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.

Allophones = Allophone
Allophone sind bei der Sprachsynthese besonders wichtig. Die kleinsten Elemente der Sprache, die nicht weiter zerlegbar sind, heißen Phoneme; sie ermöglichen die Identifikation von Wörtern. „Reh" und „Zeh" unterscheiden sich beispielsweise durch die Phoneme „r" und



„z". Ein Phonem kann als Sprechlaut in mehreren Varianten auftreten, die man als „Allophone" bezeichnet (vgl. das „e" in „her" und „Herr"). Eine Sprache, die aus Allophonen erzeugt wird, klingt etwas künstlich, weil die Übergänge fehlen, ist aber leicht programmierbar. Zur Vereinfachung kann man bei einigen Sprachsynthese-Geräten Worte und Sätze in der üblichen Schreibweise über die Tastatur eingeben, das dann in die entsprechenden Allophone umgesetzt wird. Probleme tauchen dann auf, wenn die richtige Aussprache erst aus dem Zusammenhang hervorgeht – Beispiel: „Statt noch zu rasten, rasten sie weiter." Die Sprachsynthese anhand von Phonemen und Allophonen heißt „Synthese nach Regeln".

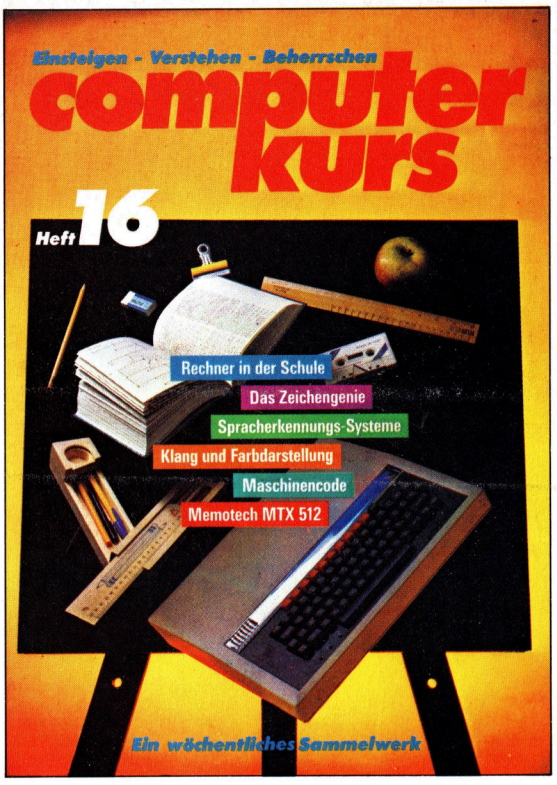
Das zweite Verfahren ist die „Synthese durch Analyse". Dabei spricht man zunächst die Worte ins Mikrofon, wobei der Rechner die Laute digitalisiert, die Frequenzen der Töne analysiert und in komprimierter Form abspeichert. Die Worte können dann durch umgekehrtes Vorgehen rekonstruiert werden. Dieses Verfahren ist so genau, daß man den Sprecher wiedererkennt. Hier wird das verfügbare Vokabular allerdings durch die Speicherkapazität begrenzt.

Alphanumeric = Alphanumerisch
„Alpha" steht für die Buchstaben des Alphabets, und „numerisch" für die Ziffern von 0 bis 9; alphanumerisch bedeutet also „in Buchstaben und Zahlen ausgedruckt". Gewöhnlich sind auch die Satzzeichen und einige Tastatur-Sonderzeichen mit eingeschlossen. Nicht inbegriffen sind Grafik-Symbole und verschiedene nicht ausdruckbare Zeichen wie „Wagenrücklauf", „Klingel" und „Bildschirmlöschung". Wenn das Wort „alphanumerisch" in der Funktionsbeschreibung eines Software-Pakets auftaucht, kann das vorteilhaft, aber auch stark einschränkend sein. Ein Lagerverwaltungsprogramm, das alphanumerische Zeichen (und nicht nur Zahlen) für jeden Artikel zuläßt, ist durchaus hilfreich, während Sie bei einem Drukker mit alphanumerischem Zeichensatz einen Programmausdruck erhalten, auf dem alle Grafik-Symbole fehlen.

Bildnachweise
393: Rex Features Ltd.
394: David W. Hamilton
396: University of Manchester
397, 398, 399, 400: Chris Stevens
407: Philips, Steve Cross
411: Tony Lodge
412, 417: Kevin Jones
413, U3: Ian McKinnell
415: Bob Venables
418: Steve Cross
419: Liz Dixon

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

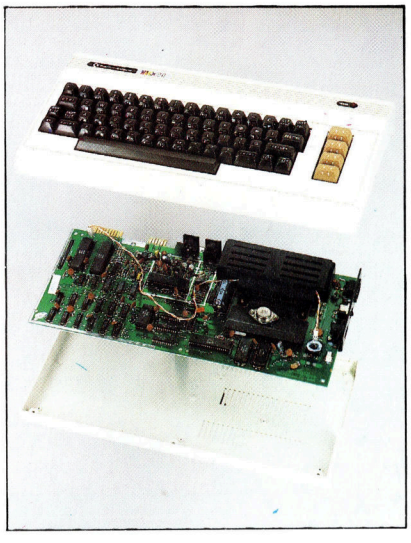
computer kurs Heft 16



LDA	**LoaD Accumulator** überträgt den Inhalt einer einzelnen Speicherstelle (Byte) in das interne Akkumulator-Register.
STA	**STore Accumulator** führt den gegenteiligen Vorgang zu LDA aus.
ADC	**ADd with Carry** addiert den Inhalt einer Speicherstelle zum Inhalt des Akkumulators und erzeugt, wenn notwendig, ein Übertrag-Bit.
SBC	**SuBtract with Carry** ist die Gegenfunktion von ADC.
JMP	**JuMP** überträgt eine Operation zu einer neuen Stelle (ähnlich wie GOTO in BASIC).

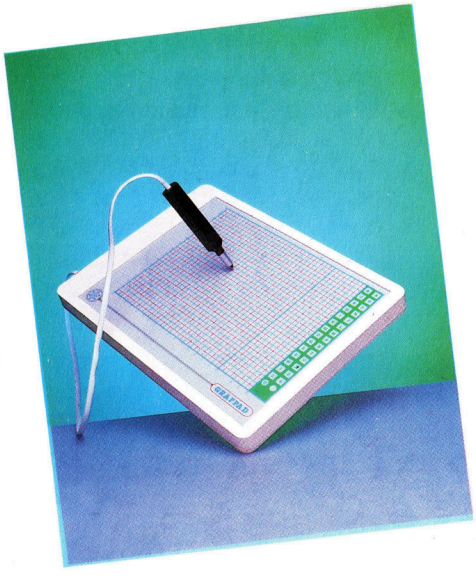
Maschinensprache

Wenn eine höhere Geschwindigkeit bei der Datenverarbeitung notwendig ist, muß das Programm durch Maschinensprache optimiert werden.



Memotech MTX 512

Dieser Heimcomputer zeichnet sich durch ausgereifte Konstruktion, interessante Standardprogramme und formschönes Design aus. Weniger gelungen ist dagegen die Gestaltung des Handbuchs. Ein empfehlenswertes MSX-Gerät zu einem vernünftigen Preis.



Zeichengenie

Grafiktabletts gehören zu den vielseitigsten und nützlichsten Peripheriegeräten. Hier das „Grafpad".

+++ Rechner im Klassenzimmer +++ BASIC: Mehr über Dateien +++ Anwendungsgeneratoren +++ Spracherkennung +++ Tips für die Praxis +++ LOGO: Animationseffekte +++ Aufregende Spiele +++ Pionier Norbert Wiener +++