Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Codierfehler schnell korrigiert

Wissenswertes über Spielprogramme

Der tragbare Osborne-1

Computer Aided Design

Tintenstrahldrucker

Heft 14

Ein wöchentliches Sammelwerk

Programmierkurse BASIC und LOGO

computer kurs

Heft 14

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 oS 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 Zeitschriftenhandler 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 oS 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.

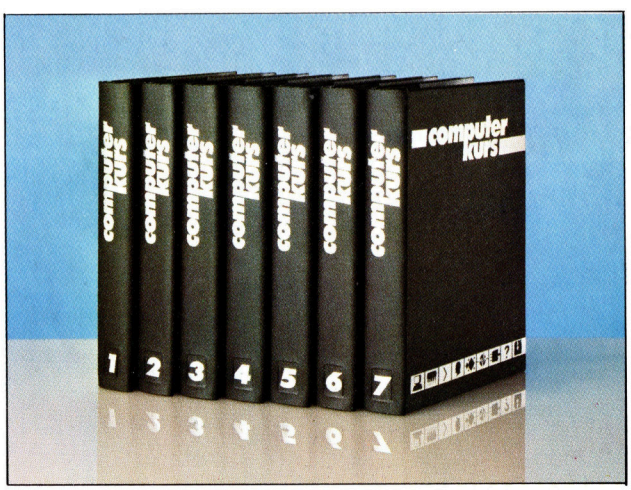
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk, dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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

CAD-Zeichnungen

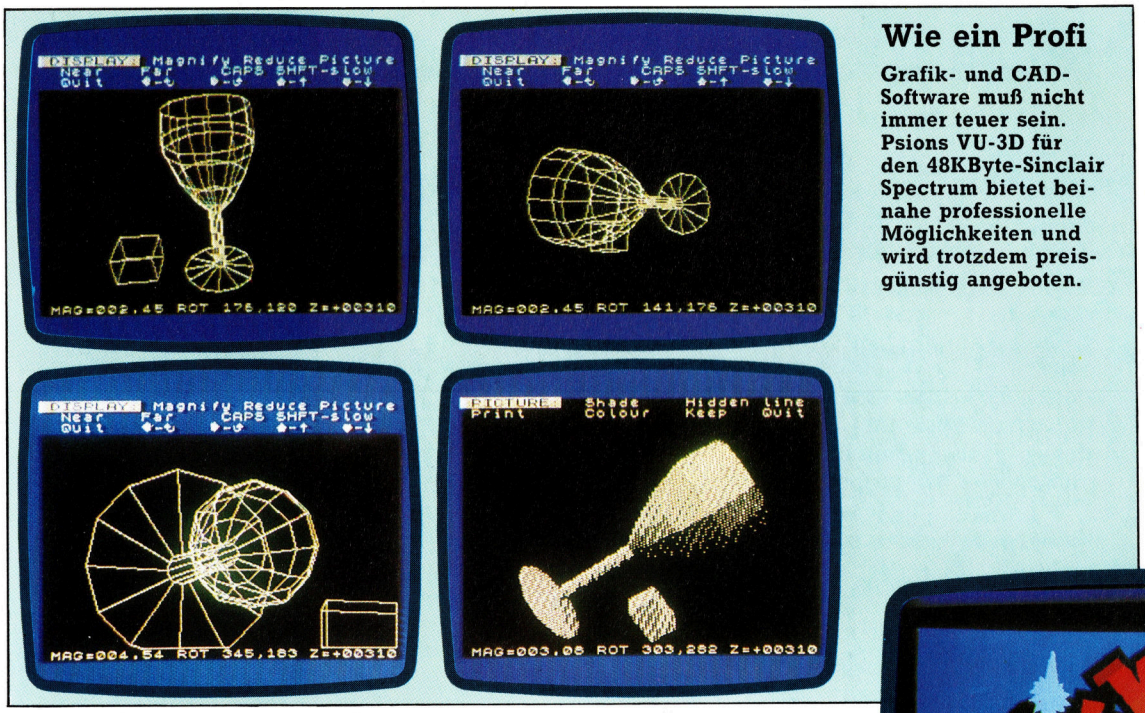
Computer Aided Design, kurz CAD genannt, beruht auf komplexen mathematischen Berechnungen und erzeugt so grafische Darstellungen mit hoher Qualität. Einige Heimcomputer lassen sich für CAD-ähnliche Zwecke aufrüsten.

Vom „Massachusetts Institute of Technology" (MIT) stammen die ersten Versuche aus den fruhen 60er Jahren, Computer mit den Aufgaben des industriellen Designs zu betrauen Aber erst ein Jahrzehnt später wurde es durch die Fortschritte der Technologie möglich, die auf dem Bildschirm dargestellten Entwürfe mit einem Lichtgriffel wie auf dem Reißbrett zu bearbeiten. Heute gehören Lichtgriffel, Digitalisierer und Plotter zum Handwerkszeug des Planens und Entwerfens. Durch Zeichnen auf dem Grafik-Tablett läßt sich ein Entwurf verandern, vorher angefertigte Zusatze und Einzelheiten konnen dazugemischt und das Ganze kann mit dem Plotter ausgedruckt werden. Aus dem Rechner wird ein Zeichen-System – ähnlich wie bei der Textverarbeitung laßt sich auch eine Zeichnung umstellen, ergänzen und erweitern.

Die Qualitat der Darstellung auf dem Monitor des Computers hängt dabei von zwei Dingen ab. von der Auflösung des Bildschirms, also der Größe eines einzelnen Bildelements oder Bildpunktes (Pixels), und von der Leistung und dem Speicherumfang des angeschlossenen Rechners Diese Anforderungen sind die gleichen wie bei der Bilderzeugung mit dem Rechner. Der Monitor sollte etwa 1000 × 1200 Punkte darstellen konnen, und der Computer muß diese 1,2 Millionen Bildelemente in weniger als den 24sten Teil einer Sekunde verarbeiten.

Eine der Anforderungen, die das Programm erfüllen muß, besteht darin, das Bild so zu speichern, daß es ohne Schwierigkeiten manipuliert und verändert werden kann. Beim Erzeugen eines Modells gibt es zwei Grundmethoden: ein additives und ein subtraktives Vorgehen. Additives Vorgehen ist etwa wie das Modellieren mit Ton, bei dem das Objekt Stück fur Stuck bis zur endgültigen Form zusammengesetzt wird. Die subtraktive Methode ist die des Bildhauers, der mit dem Abtragen von Material zum selben Ergebnis kommt. Der Computer bildet in Analogie zu einem massiven Block eines Rohmaterials ein dreidimensionales Feld. Nun wird die Arbeitsweise und Leistung des Rechners wichtig: Ist das Feld groß genug, um die Daten für ein Pixel in einem Byte unterzubringen, steht fur jedes Element

Auch die ausgefeilteste Software eines rechnergestützten Zeichensystems ersetzt nicht die Phantasie des Benutzers. Die Zeichnung oben wurde mit Versawriter erstellt. Trotz der Verwendung eines Digitalisierers brauchte der erfahrene Anwender einen nicht unerheblichen Zeitaufwand für die Eingabe.



Wie ein Profi

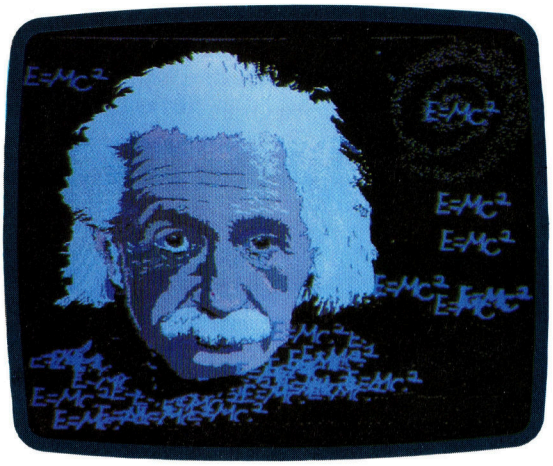
Grafik- und CAD-Software muß nicht immer teuer sein. Psions VU-3D für den 48KByte-Sinclair Spectrum bietet beinahe professionelle Möglichkeiten und wird trotzdem preisgünstig angeboten.

viel Platz für die entsprechenden Informationen zur Verfügung (256 Einzelinformationen bei einem 8 Bit-Prozessor, bei 16 oder 32 Bit-Prozessoren noch erheblich mehr). Dazu wäre insgesamt jedoch eine große Speicherkapazität erforderlich.

Es gibt jedoch einen Kompromiß: Statt mit einem ganzen Byte wird jedes Element nur mit einem Bit beschrieben. Damit läßt sich jedoch nur festlegen, ob dort ein Bildpunkt vorhanden ist oder nicht.

Die Software für computergestütztes Entwerfen (CAD) beinhaltet zahlreiche Möglichkeiten wie etwa: Kurvenzeichnung, Beseitigen verdeckter Bildelemente, Grauwertabstufung und farbige Darstellung. Für die Kurvenzeichnung ist nur die wiederholte Berechnung einer einfachen Gleichung nötig. Wenn der Anfangs- und Endpunkt einer Linie sowie der maximale Abstand der gewünschten Kurve von dieser Linie bekannt ist, besitzen wir die Lösung der Gleichung für einen einzelnen Punkt. Mit dieser Lösung kann die Gleichung selbst ermittelt werden, aus der sich dann auch alle anderen Punkte der Kurve berechnen lassen.

Die große Stärke heutiger CAD-Systeme besteht darin, daß ein Bild aus Standard-Komponenten zusammengesetzt werden kann. Die einzelnen Teile eines Entwurfs müssen so nicht immer wieder neu konstruiert werden. Einmal definiert, lassen sie sich ja nach Bedarf aufrufen und in neue Entwürfe integrieren. Ein gutes Beispiel dafür ist der Computer-Einsatz



Mit einem schnellen Prozessor und zusätzlichem Speicherplatz macht das System „Pluto" von Io Research hochauflösende Grafik auch für kleine Microcomputer möglich. Der Grundbaustein läßt die Verwendung von acht Farben bei einer Auflösung von 670 × 576 Bildpunkten zu.

bei der Planung zukünftiger Rechner-Generationen:

Der Entwurf von Platinen erfordert eine äußerst präzise Arbeitsweise. Bei der Komplexität dieser Aufgabe ist der Planer oft auf das Prinzip von Versuch und Irrtum zurückgeworfen – CAD-Systeme bieten gerade in diesem Bereich wertvolle Hilfestellungen. Die erforderlichen Einzelteile werden als Bildmuster definiert und gespeichert, um sie während der Arbeit ständig parat zu haben. So kann auf dem Bildschirm ein Entwurf gefertigt werden, der vor dem Ausdruck genau auf seine Eignung zu überprüfen ist, sich leicht abwandeln läßt und sogar im Vergleich mit anderen Lösungen betrachtet werden kann. – All das in der gleichen Zeit, die vorher für eine einzige Zeichnung nötig war. Der Ausdruck des Plans erfolgt erst nach Fertigstellung des gesamten Schaltungsentwurfs.

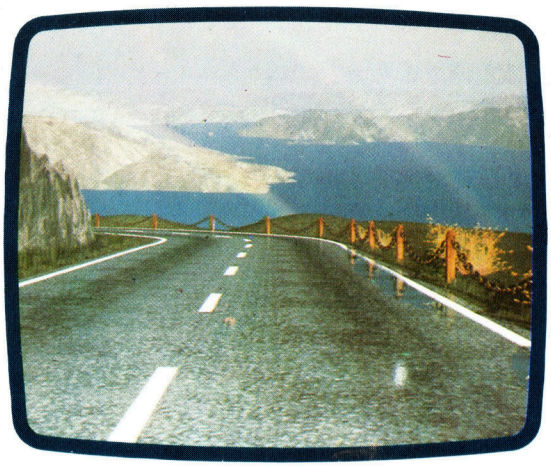
Ähnlich werden auch integrierte Schaltkreise entwickelt, für die zusätzlich eine andere Software-Hilfe bereitsteht: Teile der Schaltung können vergrößert, im vergrößerten Maßstab bearbeitet und danach wieder in den übrigen Entwurf eingepaßt werden. Diese Fähigkeit der Software hat CAD noch um einiges effizienter werden lassen, weil sich der gesamte Entwurf zwar auf einer einzigen Zeichnung befindet, der Maßstab jedoch nach den Wünschen des Benutzers verändert werden kann.

Farbige Sub-Systeme

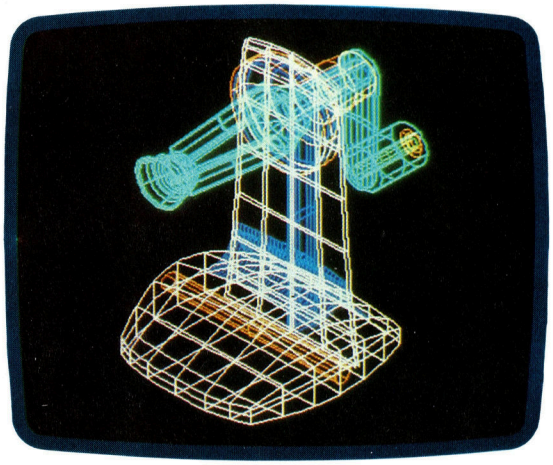
Bei komplexeren Entwürfen, wie sie zum Beispiel für ein Auto nötig sind, lassen sich zusätzlich die verschiedenen Sub-Systeme, aus denen sich die komplette Grafik zusammensetzt, in verschiedenen Farben darstellen. Dadurch kann ein Ingenieur, der etwa mit der Konstruktion der Auspuffanlage oder des Hydraulik-Systems beschäftigt ist, sein Aufgabengebiet aus dem Entwurf „herausziehen", um besser daran arbeiten zu können. Bei Bedarf lassen sich die Farben natürlich auch unterdrücken.

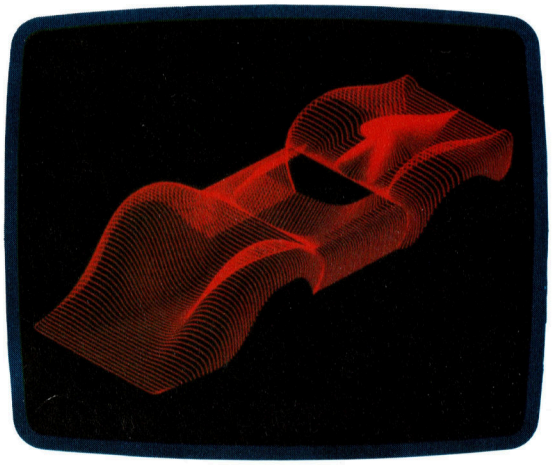
Ein wirklicher Durchbruch stellte sich bei den rechnergestützten Entwurfssystemen ein, als auch andere Eigenschaften eines Gegenstandes darstellbar wurden: Nicht nur die Form wird gespeichert, sondern ebenso Informationen über den Werkstoff, das Gewicht und sogar die Herstellungskosten. Exakte Angaben über die Form sind damit nur noch eine unter vielen Funktionen des Systems. Diese nunmehr „visuell unterstützte Datenbank" leistet die verschiedensten Dienste: Material bestellen, Produktionstermine abstimmen, Kosten analysieren und sogar den Produktionsprozeß überwachen. Schon heute läßt sich der nächste Fortschritt absehen – die voll rechnergesteuerte Produktion. Die schnelle Ausbreitung der Industrieroboter rückt diese Entwicklung bereits jetzt in greifbare Nähe.

Die meisten hier vorgestellten Anwendungen erfordern Großrechner oder zumindest sehr leistungsfähige Kleincomputer. Damit soll aber nicht gesagt sein, daß ein kleinerer Computer keine Aufgaben beim Entwurf übernehmen kann. Für Rechner mit CP/M-Betriebssystem gibt es diverse CAD-Programme, und selbst für den relativ einfachen Sinclair ZX 81 kann man ein entsprechendes Software-Paket erwerben. Der Heimcomputer stößt aufgrund seiner relativ kleinen Speicherkapazität bei CAD zwar an seine Grenzen, zeigt für bescheidenere Ansprüche aber trotzdem interessante Nutzungsmöglichkeiten.



Der Hintergrund dieser Filmaufnahme besteht fast ausschließlich aus Fragmenten, die unter Verwendung einer neuen CAD-Technik zusammengefügt wurden. Je genauer diese Fragmente betrachtet werden, um so komplexer erscheinen sie. Zu Anfang waren die Hügel und Berge nur als einfacher Linienzug im Computer gespeichert. Die Grundform wird dann mehrfach – in immer kleinerer Struktur – wiederholt und auf die Flächen der Geländeform projiziert. Ein Zufallsgenerator sorgt dabei für realistische Abwechslung.





Drahtmodelle sind der erste Schritt zum dreidimensionalen Entwurf. Das Bild wird durch eine Reihe von Punkten definiert, die durch gerade Linien miteinander verbunden sind. Durch Kurven-Algorithmen und das Weglassen verdeckter Linien mittels Einfärbung der Flächen wird das Modell realer, und es entsteht der Eindruck von Tiefe.

Osborne-1

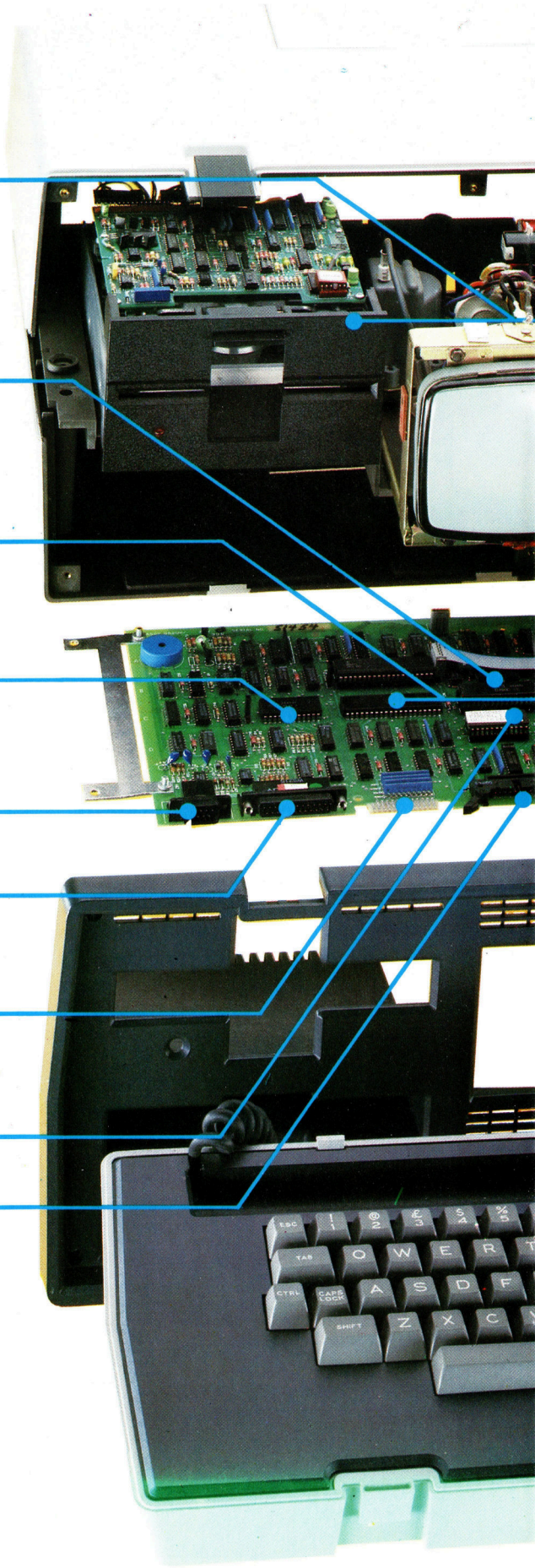
Er war nicht nur der erste tragbare Microcomputer, sondern auch das erste Gerät, in dessen Kaufpreis Software enthalten war.

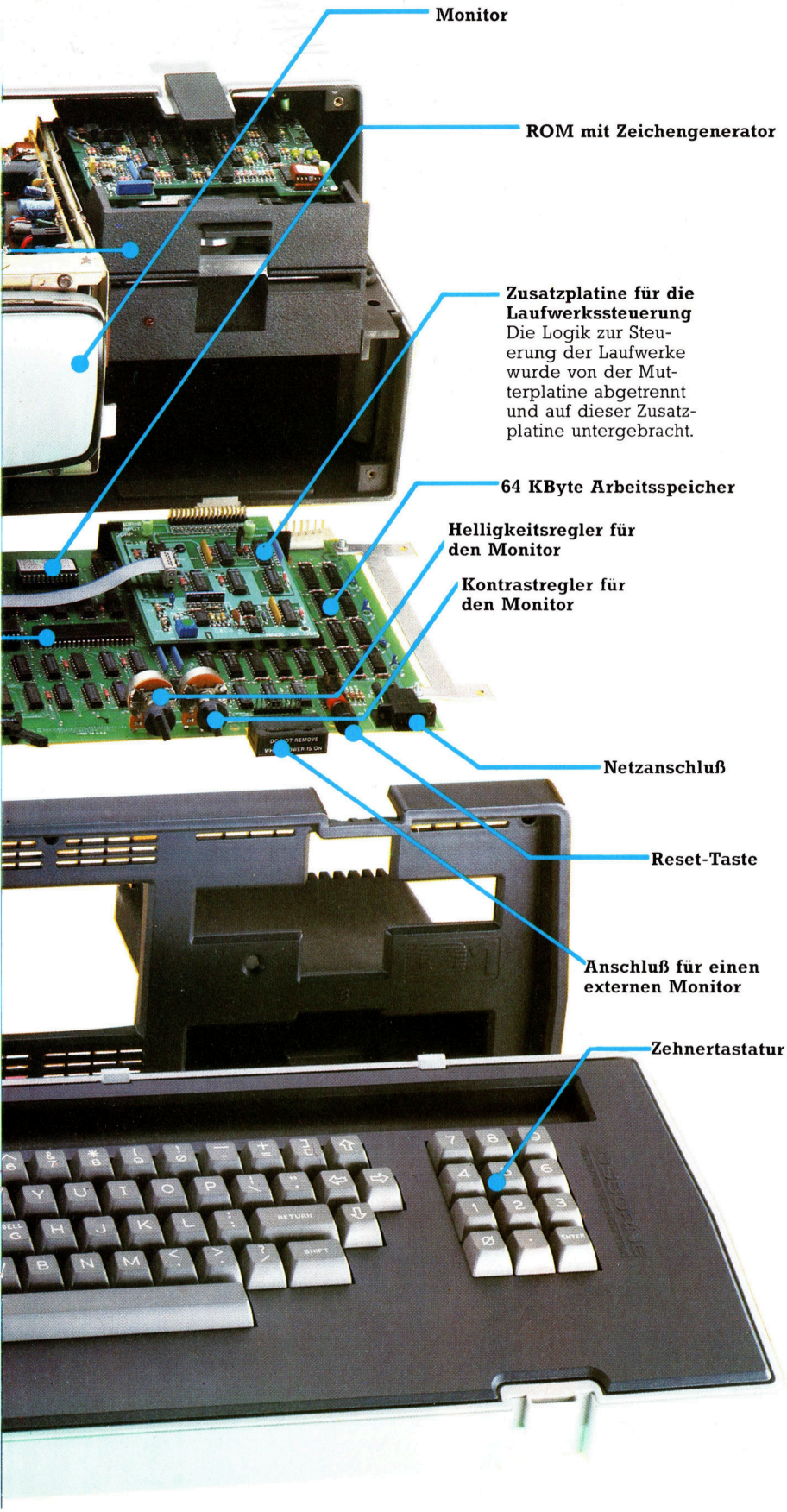
Obwohl der Osborne-1 nicht unter die Kategorie „Heimcomputer" fällt, ist er schon allein wegen der Tatsache interessant, daß er der erste völlig eigenständige tragbare Microcomputer war. Mit zwei integrierten Diskettenlaufwerken und einem kleinen Bildschirm ausgestattet, bietet er alle Vorteile einer transportablen Datenverarbeitungsanlage. Nur eine Batterie hatte der Hersteller nicht eingebaut. Das Gewicht des Gerätes ware dadurch zu hoch geworden (der Osborne-1 wiegt bereits 10,5 kg). In der Frontplatte befindet sich deshalb ein Netzanschluß, der das Gerät mit den nötigen 12 V (für die Diskettenlaufwerke) und 5 V (fur die interne Verarbeitung) versorgt.

Gegen eine Einstufung als Heimcomputer spricht außerdem sein Preis – ca. 4000 DM. Die mitgelieferte Software enthält einige der besten kommerziellen Programme: CBASIC von Microsoft, ein BASIC, das sich besonders zur Compilierung eignet und Programme um ein Vielfaches schneller macht als herkömmliches BASIC, SuperCalc, eines der anerkannt besten Kalkulationsprogramme; WordStar und Mailmerge, die Bestseller unter den transportablen (d. h. auf einer breiten Palette von Maschinen einsetzbaren) Textprogrammen und schließlich das Betriebssystem CP/M (Control Program for Microcomputers) von Digital Research, das den Osborne-1 fur eine Vielzahl von Programmen zugänglich macht.

Laden von Diskette

Wie der Apple II lädt auch der Osborne-1 sein Betriebssystem von einer Diskette. Außer der internen Steuerung des Computers erledigt CP/M noch eine Reihe von Dienstfunktionen, wie beispielsweise das Kopieren von Dateien und Disketten, das Initialisieren von Disketten und das Anzeigen der Inhaltsverzeichnisse. CP/M hat aber noch andere Stärken. Die geschriebenen Programme sind maschinenunabhängig. Aufgrund hoherer Verkaufszahlen konnten mehr Geld und Zeit in die Erstellung von Software investiert werden, womit sich wiederum die Qualität der Programme erhohte. Es kann dem erfahrenen CP/M-Anwender fast völlig gleich sein, welchen Maschinentyp er vor sich hat, bei der Hardwareerweiterung unter CP/M mussen Daten nicht nochmals eingegeben werden.





Osborne-1

PREIS
ca. 4000 DM

ABMESSUNGEN
510 × 325 × 255 mm

GEWICHT
10,5 kg

CPU
Z80A

TAKTFREQUENZ
4 MHz

SPEICHERKAPAZITÄT
64 KByte RAM,
4 KByte ROM

BILDSCHIRM-DARSTELLUNG
24 Zeilen mit je 52 Zeichen; 32 Zeilen mit je 128 Zeichen sind im Bildschirmspeicher „virtuell" vorhanden.

SCHNITTSTELLEN
RS232 seriell, IEEE, Modem

PROGRAMMIER-SPRACHEN
BASIC, Z80-Assembler

WEITERE PROGRAMMIER-SPRACHEN
Alle Programmiersprachen, die unter CP/M laufen.

ZUBEHÖR
CP/M, WordStar, CBASIC, MBASIC, Mailmerge, SuperCalc, Handbücher

TASTATUR
Schreibmaschinentastatur mit 69 Tasten und Zehnertastatur.

Control Program for Microcomputers

Seit Mitte der sechziger Jahre die zweite Computergeneration vorgestellt wurde, profitierten die Geräte der Groß-EDV und die Minicomputer von maschinenunabhängigen Betriebssystemen. Es sollten jedoch noch weitere zwölf Jahre vergehen, bis diese Art der Steuerung auch für Microcomputer zur Verfügung stand. Mit CP/M brachte Digital Research das erste dieser Betriebssysteme auf den Markt. Entworfen für den 8080 von Intel und die Z80-Serie von Zilog, enthält es eine Anzahl von Dienstprogrammen und die Fähigkeit, laufende Programme zu unterbrechen (interrupt) und fortzuführen.

Ein weiterer großer Vorteil ist die Festlegung der Dateistrukturen und Speicherformate durch das Control Program for Microcomputers.

Pech für Osborne war, daß sich die Aufmerksamkeit der amerikanischen Geschäftswelt auf die Einführung des Personal Computers von IBM richtete, der – als 16 Bit-Maschine konzipiert – auf dem 8088-Chip von Intel aufbaute. Obwohl von IBM eigentlich nur als Zwischenlösung gedacht (der 8088 hat zwar eine 16 Bit-Adressierung, aber nur einen 8 Bit-Datenbus), entwickelte sich dieser Chip aufgrund der Macht des Namens IBM zu einem Industriestandard.

Der IBM Personalcomputer verfügt über ein speziell für ihn entwickeltes Betriebssystem, das PC-DOS. Um konkurrenzfähig zu bleiben, stellte Digital Research zwei neue Versionen seines CP/M-Betriebssystems vor: Concurrent CP/M, ein echtes Mehrplatzsystem, unter dem mehrere Programme gleichzeitig ablaufen können, und CP/M86, das für den 8086-Chip von Intel konzipiert ist, der eine 16 Bit-Adressierung mit einem „echten" 16 Bit-Datenbus enthält.

Osborne-Konkurs

All diese Entwicklungen waren vermutlich die Ursache dafür, daß der Osborne-1 vom Markt verdrängt wurde: 1983 meldete die Osborne Computer Corporation – die Muttergesellschaft in den Vereinigten Staaten – Konkurs an. Dennoch ist der Osborne-1 mit seinen 64 KByte Arbeitsspeicher (von denen dem Anwender 60 KByte zur Verfügung stehen) und den beiden Diskettenlaufwerken mit je 102 KByte Kapazität ein durchaus interessantes Gerät. Berücksichtigt man weiterhin die eingebauten Schnittstellen RS232 und IEEE 488 sowie die Möglichkeit des Batteriebetriebes, dann ist leicht zu verstehen, warum dieser Computer ein Bestseller wurde und auch nach dem Konkurs seines Herstellers noch populär blieb.

Eine bemerkenswerte Eigenschaft des Osborne-1 (zum Teil auch auf dem HX-20 von Epson vorhanden) ist der „virtuelle Bildschirm", der die dreifache Kapazität des eingebauten Bildschirms von 24 Zeilen mit je 52 Zeichen besitzt. Mit Hilfe der Control-Taste (ein Standard-Bedienungselement unter CP/M) und den Cursor-Steuertasten kann ein „Fenster" über den gesamten Bildschirmspeicher bewegt werden. Damit wurden die meisten Nachteile des kleinen Bildschirms (8,75 cm × 6,6 cm) ausgeglichen.

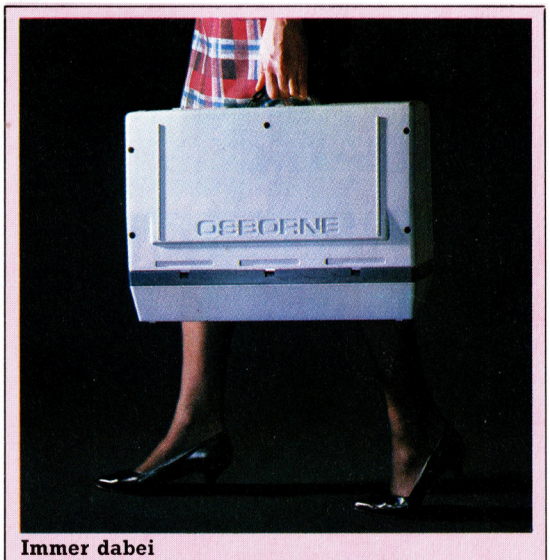
Nur wenige Anwender hatten Schwierigkeiten mit dem Mini-Bildschirm. Zudem besitzt der Osborne auch einen Anschluß, über den der Inhalt des kleinen Screens auf einem externen Monitor dargestellt werden kann. Von den Anwendern trafen sogar Nachfragen ein, wie die gesamten 4 KByte des virtuellen Bildschirms (32 Zeilen mit je 128 Zeichen) zur Verfügung gestellt werden könnten.

Osborne baute daher eine Veränderung ein, mit der zwischen drei verschiedenen Darstellungsformaten gewählt werden kann: 52, 96, oder 128 Zeichen pro Zeile.

Die Tastatur des Osborne-1 verfügt über 69 Tasten und läßt sich als Deckel über die Vorderseite des Geräts klappen. Sie besteht aus normalen Schreibmaschinen- und zusätzlichen Control- und Escape-Tasten. Auf der rechten Seite befinden sich die Ziffern mit einer zusätzlichen Eingabetaste. Mit Hilfe des CP/M-Programms SETUP können die numerischen Tasten mit bis zu 96 Zeichen neu belegt werden. Diese Möglichkeit ist besonders hilfreich, wenn ein Wort, ein Satz oder eine Befehlsfolge oft benötigt wird. Die mit SETUP programmierten Funktionen werden auf die Disketten geschrieben und sind sofort wiederzeit abrufbar. Neben 96 Standardzeichen gibt es noch weitere 32 Grafikzeichen.

Da der Osborne-1 Chips der 6800er-Serie einsetzt, und nicht (wie bei einer CP/M-Maschine üblich) die der 8080-Familie, ist die Tastaturabfrage (Polling) etwas ungewohnlich gestaltet. Dabei wird ein bestimmter Speicherbereich für die Interpretation der Tastatureingaben abgestellt. Das ROM des Systems fragt diesen Bereich ständig ab, um festzustellen, ob eine Taste gedrückt wurde. Da die Übersetzungslogik nicht fest in die Tastatur eingebaut wurde und Tastenfunktionen im Arbeitsspeicher untergebracht werden, bereitet die neue Belegung der Funktionstasten keine Schwierigkeiten.

Nachdem die Osborne Computer Corporation Konkurs angemeldet hatte, etablierte sich ihre englische Niederlassung als eigenständige Firma. Doch wie die Zukunft der Maschine auch aussehen mag, gegen die hohe Qualität des Osborne-1 läßt sich nichts sagen.



Immer dabei
Außer seinem ausgezeichneten Preis-Leistungsverhältnis hat der Osborne-1 noch den zusätzlichen Vorteil der Tragbarkeit. Er ist mit 10,5 kg nicht gerade ein Fliegengewicht, aber da er gut austariert ist, bereitet das Tragen dieses Computers kaum Probleme.

Intelligente Spiele

Selbst für einen erfahrenen Programmierer ist es besonders schwer, ein Schachprogramm zu schreiben – einfache Spiele mit eigener „Intelligenz" können jedoch schon von Anfängern programmiert werden.

Schachcomputer bestehen aus den gleichen Bauteilen wie Heimcomputer: CPU, ROM und RAM. Nur die Ein- und Ausgabemethode ist anders. Das „Phantom" (Foto) besitzt einen Servomechanismus und Magneten, mit denen er die Schachfiguren selbständig bewegt. Springt zum Beispiel ein Pferd über eine andere Figur, dann rückt das Gerät diese Figur einem komplizierten Algorithmus folgend aus dem Weg und stellt danach den korrekten Spielstand wieder her.

Viele Computer-Neulinge träumen davon, ein Programm zu schreiben, das Schach spielen kann. Eine der Ursachen dafür ist vermutlich die Tatsache, daß dieses Spiel als „hochintelligent" angesehen wird: Ein Computer, der Schach spielen kann, ist ein weiterer Schritt in Richtung auf die Konstruktion künstlicher Intelligenz. Leider läßt sich im Rahmen dieses Artikels der Aufbau eines vollständigen Schachprogramms nicht erklären. Wir werden jedoch einige der Grundlagen darlegen, auf denen intelligente Spiele aufbauen, und Ihnen zeigen, wie Sie in BASIC „lernfähige" Programme schreiben können.

Die meisten Kinder (und Erwachsene) kennen das Spiel „Schere – Stein – Papier". Die Regeln sind einfach: Beide Spieler denken an eins der drei Objekte und halten dann gleichzeitig eine Hand hoch, die das gedachte Objekt darstellt. Der Gewinner läßt sich leicht bestimmen: Schere schlägt Papier (durch Zerschneiden), Papier schlägt Stein (durch Einwickeln) und Stein schlägt Schere (durch Abstumpfen).

Jeder, der unseren BASIC-Kurs verfolgt hat, wird ohne Probleme ein Programm schreiben können, das die Ergebnisse der Spieler verknüpft und anzeigt. Dabei wird durch die RND-Funktion aus einer Matrix mit den drei STRING-Elementen „Stein", „Schere", „Papier" ein Element ausgewählt und auf dem Bildschirm angezeigt, sobald die Leertaste gedrückt wird. Der Spieler gibt jetzt seine eigene Wahl ein. Das Programm berechnet, wer gewonnen hat, und zeigt das aktuelle Ergebnis sowie die Gesamttreffer beider Spieler an. Wenn die RND-Funktion echte Zufallszahlen erzeugt, nähern sich die Trefferquoten unabhängig von der Spielstrategie nach zahlreichen Spielen einander an. Untersuchen Sie jetzt einmal, welche Strategie der Computer verfolgen muß, um mehr Treffer erzielen zu können.

In dem Artikel über Zufallszahlen wurde festgestellt, daß es für Menschen wie für Computer unmöglich ist, eine Reihe von echten zufälligen Zahlen zu erzeugen. Sie können ein Unterprogramm schreiben, das mit Hilfe einer Matrix feststellt, wie oft der Spieler ein be-

stimmtes Objekt gewählt hat. Nennen wir die Elemente der Matrix Wahl (1), Wahl (2) und Wahl (3). Bei jeder Auswahl wird auf das Matrixelement des entsprechenden Objektes eine Eins addiert. Auf diese Weise kann der Computer feststellen, welches Objekt von seinem Gegner bevorzugt gewählt wird und dann eine darauf abgestimmte Spielstrategie verfolgen, die (wiederum über eine größere Anzahl Spiele) zum Sieg führt.

Bei dieser Methode entstehen einige Probleme. Der Spieler könnte nämlich schnell erkennen, wenn der Computer, entsprechend seiner Strategie, das gleiche Objekt mehrfach wiederholt. Der Mensch könnte dies zu seinem Vorteil ausnutzen. Der Computer muß daher seine drei Objekte hauptsächlich über die RND-Funktion auswählen, während ein gesondertes Unterprogramm dafür sorgt, daß ein Objekt, das den Spieler schlagen kann, häufiger gewählt wird.

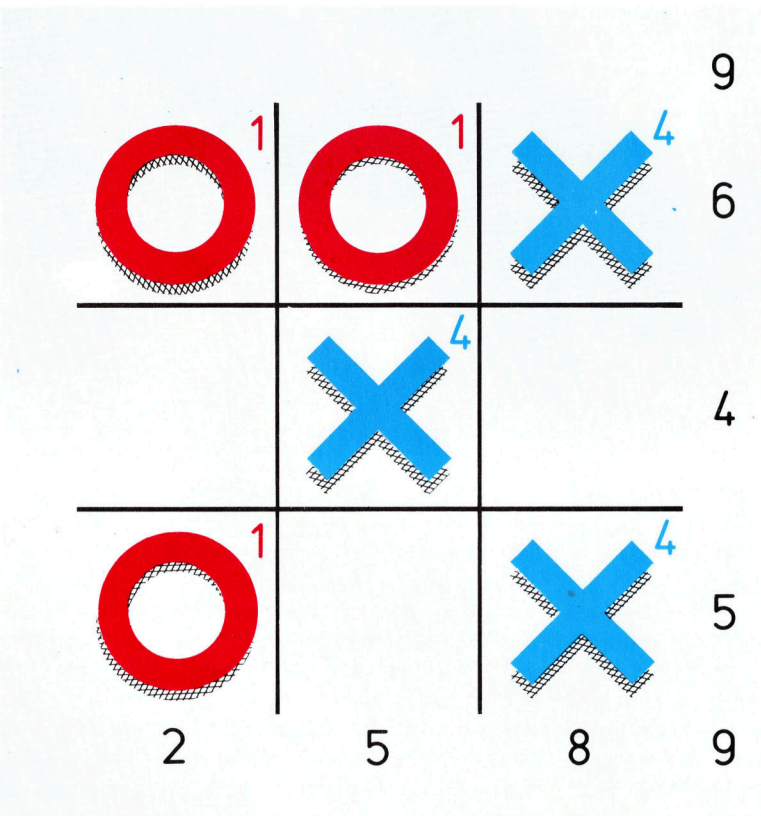
Ein zweites Problem entsteht aus der Tendenz des Spielers, sein bevorzugtes Objekt im Verlauf des Spiels zu wechseln. Statt die Auswahl des Gegners von Anfang des Spiels an aufzuzeichnen, müßte der Rechner daher beispielsweise nur die letzten 20 gewählten Objekte speichern. Dafür müßten eine „WAHL"-Matrix, die aus 20 × 3 Elementen besteht, und ein Unterprogramm eingesetzt werden, das die drei Spalten addiert und über die höchste Summe das geeignete Objekt für die nächste Runde voraussagt.

Der Nachteil dieser Methode wird jedoch schnell offensichtlich, wenn ein Spieler die Strategie des Computers errät. Es ist dann relativ einfach, so zu spielen, daß der Computer mehr als die Hälfte der Runden verliert. Der Spieler könnte beispielsweise eine Zeitlang immer das gleiche Objekt angeben, um dann (für den Computer) völlig unerwartet auf ein anderes überzuwechseln. Daher wird ein Algorithmus benötigt, mit dem sich diese Probleme vermeiden lassen.

Computer-Trefferquote

Da Menschen nicht imstande sind, völlig zufällige Entscheidungen zu treffen, ist die logische Folgerung, daß sich jede neue Wahl aus den vorhergehenden Entscheidungen ergibt. Der Computer kann diese jedoch in der Annäherung errechnen und wird in seiner Trefferquote mehr und mehr vorn liegen. Da jeder Spieler nach einer eigenen unterbewußten Formel spielt und sich diese während des Verlaufs eines langen Spiels ändern kann, muß das Programm die Berechnungsgrundlage ebenfalls während des Spiels entwickeln und ändern können.

Mit einem solchen Programm kann ein Computer Strategieveränderungen des Gegners erkennen und seine eigene Methode entsprechend abändern. Beispielsweise werden die letzten 50 Entscheidungen beider Spieler in einer Matrix gespeichert, und diese Informationen werden ständig nach einer Methode, die Korrelation genannt wird, untersucht.



Tic-Tac-Toe

Die Bewertung der Position ist die Grundlage für jedes Brettspielprogramm – selbst bei einfachen Spielen wie Tic-Tac-Toe. In diesem Fall wird das Brett als eine Matrix mit dreimal drei Elementen gespeichert. Die Kreise des Spielers werden durch eine Eins dargestellt und die Kreuze des Computers von einer Vier. Mit Hilfe dieser Zahlen kann jede Position durch das Addieren der einzelnen Reihenelemente ermittelt werden. Das Ergebnis von 12 in einer der Reihen bedeutet, daß der Computer gewonnen hat; das Ergebnis 3 zeigt an, daß der Spieler der Gewinner ist; die Summe von 8 heißt, daß der Computer in einer Reihe zwei Kreuze hat und bei dem nächsten Zug gewinnen kann usw.

Das heißt, der Computer stellt Hunderte von Vergleichen zwischen der aktuellen Wahl des Gegners, der vorigen Wahl, der vorletzten oder auch der Wahl, die fünf Runden zurückliegt, an. Seine eigenen Entscheidungen untersucht der Computer ebenso. Nehmen wir als Beispiel die Korrelation zwischen dem letzten und dem vorletzten Zug des Spielers. Bezeichnen Sie Schere als Element 1, Papier als Element 2 und Stein als Element 3. Zunächst müssen Sie eine Matrix mit 3 × 3 Elementen anlegen, die KORR1 heißt, da darin die Ergebnisse des ersten Korrelationstestes dargestellt werden. Jetzt untersuchen Sie die letzten 50 Spielzüge. Für jede Runde, in der der Spieler nach dem Objekt Schere (1) das Objekt Stein (3) wählte, addieren Sie eine Eins auf das Element KORR1 (1,3); wird Stein (3) von Papier (2) gefolgt, dann wird eine Eins auf das Element KORR1 (3,2) addiert usw.

Würde der Spieler seine Auswahl rein zufällig treffen, wären in jedem Element der Matrix KORR1 ungefähr die gleichen Werte enthalten – ein derartiges Ergebnis ist aber höchst unwahrscheinlich. Hat sich der Spieler in der letzten Runde für Papier entschieden, dann gibt Ihnen das Element in Reihe 2 (Papier) mit dem höchsten Wert Auskunft darüber, für welches Objekt er sich vermutlich als nächstes entscheiden wird. Je größer die Zahlenunterschiede der Elemente einer Reihe sind, desto zuverlässiger fällt die Vorhersage aus. Da es jedoch auch vorkommen kann, daß zwischen der letzten und vorletzten Entscheidung des Spielers keine Korrelation besteht, müssen Sie weiter zurückliegende Runden oder die Beziehung zwischen der letzten Wahl des Spielers und dem vom Computer gewählten Objekt ebenfalls untersuchen.

Berechnung der Spielzüge

Eine Schwierigkeit ergibt sich, wenn die verschiedenen Korrelationsroutinen unterschiedliche Voraussagen über den nächsten Spielzug des Gegners machen. Das Programm muß dann entscheiden können, welche Vermutung am zuverlässigsten ist. In diesem einfachen Spiel hat der Computer die Aufgabe festzustellen, welcher Test die größten Wertunterschiede geliefert hat. So könnte z. B. KORR1 folgende Wahrscheinlichkeiten vorhersagen: Schere 51 %, Papier 29 %, Stein 20 %; während KORR2 (in dem z. B. die Beziehungen zwischen der Wahl des Gegners mit der vorigen Wahl des Computers verglichen werden) die Wahrscheinlichkeiten Schere 24 %, Papier 60 %, Stein 16 % ergibt. KORR2 enthält eindeutig die klareren Werte und bestimmt damit die Auswahl des nächsten Objektes. Ein intelligentes Spielprogramm besteht in der Tat oft aus einer ganzen Reihe von Unterprogrammen, die unterschiedlich arbeiten und das Hauptprogramm über den besten Zug informieren.

```
  5 CLS
 10 DIM C1(3,3),C2(3,3),C3(3,3)
 20 CR=0
 30 FOR I=1 TO 3
 40 IF C1(PL,I) > CR THEN BG=I:CR=C1(PL,I)
 50 IF C2(PP,I) > CR THEN BG=I:CR=C2(PP,I)
 60 IF C3(P3,I) > CR THEN BG=I:CR=C3(P3,I)
 70 NEXT I
 80 CT=BG-1
 90 IF BG=1 THEN CT=3
100 GET PT: IF PT=0 THEN 100
110 REM ZEILE 100 WARTET AUF EINE
120 REM ZAHLENEINGABE
130 IF CT=PT-1 THEN CS=CS+1
140 IF CT=PT-2 THEN PS=PS+1
150 IF CT=PT+1 THEN PS=PS+1
160 IF CT=PT+2 THEN CS=CS+1
170 CLS
180 PRINT "IHRE WAHL:";PT
190 PRINT "MEINE WAHL:";CT
200 PRINT "IHRE TREFFER:";PS
210 PRINT "MEINE TREFFER:";CS
220 C1(PL,PT)=C1(PL,PT)+1
230 C2(PP,PT)=C2(PP,PT)+1
240 C3(P3,PT)=C3(P3,PT)+1
250 P3=PP
260 PP=PL
270 PL=PT
280 GOTO 20
```

Dieses Programm hat seinen Ursprung in dem Spiel Schere – Stein – Papier. Es läßt sich damit leicht darstellen, wie ein Programm während eines Spieles „lernen" kann. Der Computer wählt eine Zahl zwischen 1 und 3, vergleicht sie mit der Wahl des Gegners und zeigt die Gesamtzahl der Treffer für beide Spieler an. Mit dem GET-Befehl lassen sich die Zahlen schnell nacheinander eingeben. Selbst wenn Sie versuchen, Ihre Wahl zufällig zu gestalten, werden Sie merken, daß die Trefferquote des Computers nach einigen Hundert Runden vorn liegt. Dieses Programm läßt sich natürlich „bluffen"; verfeinerte Unterprogramme könnten aber auch das verhindern.

Ergibt sich eine Korrelation zwischen den Zügen des Spielers und dem vorhergehenden Zug des Computers, kann in das Programm ein „Bluff" eingebaut werden, der den Spieler bewußt in die Irre führt. Am besten funktioniert diese Methode bei Spielen, in denen der Einsatz nach jeder Runde erhöht wird und bei denen es sich lohnt, die Anfangsrunden zu verlieren, um die späteren, höher bewerteten Runden zu gewinnen.

An der State University of New York in Buffalo ließ man beispielsweise eine Anzahl von Poker-Programmen (alle mit Lernfähigkeit ausgerüstet) einige Tausend Spiele gegeneinander spielen (Bericht im Scientific American, Juli 1978). Gesamtsieger war ein Programm mit dem Namen „Adaptive Evaluator of Opponents" (AEO), das das Blatt des Gegners anfänglich einmal bewertete, diese Bewertung im Verlauf des Spiels aber immer wieder korrigierte. Das Programm SBI („Sells and Buys Images") schnitt erstaunlich schlecht ab – seine Methode bestand darin, zu bluffen, um dem Gegner ein falsches Bild von seinem Blatt zu „verkaufen" oder durch „Kaufen" des Blatts die Spieltechnik des Gegners herauszufinden.

Das Programm „Adaptive Aspiration Level" (AAL) versucht, bestimmte menschliche Verhaltensmuster zu imitieren. Aufgrund des bisherigen Spielverlaufs wird versucht, das momentane Risiko exakt zu bestimmen.

Wie man sieht, ist es in der Tat möglich, „intelligente" Spiele zu programmieren.

Binäres Suchen

Wenn Sie eine neue Eintragung im „Adreßbuch" beabsichtigen, müssen Sie zunächst eine leere „Karteikarte" finden.

Im letzten Teil unseres Kurses wurde erklärt, wie man eine Datei aus Verzeichnissen erstellt. Jetzt folgen die Möglichkeiten des Suchens bestimmter Daten.

Das Erstellen von Verzeichnissen für das Adreßbuch ist nicht schwierig. Angenommen, für jedes Feld des Verzeichnisses existiert ein separater String mit einem Namen. Bei den acht aufgeführten Funktionen des Adreßbuch-Programms war Punkt sechs die Aufnahme neuer Adressen. Angenommen, daß wir schon einige Eintragungen haben, aber nicht mehr genau wissen wie viele. Selbstverständlich muß vermieden werden, daß neue Eintragungen über bereits bestehende geschrieben werden. Das bedeutet, daß eine der Aufgaben des Programms sein muß, die Elemente eines Feldes zu durchsuchen, um das erste zu finden, das noch keine Daten enthält.

Dieser Suchvorgang ist nicht schwer. String-Variablen können in BASIC genauso miteinander verglichen werden wie numerische. IF A$=„HAUS" THEN . . . ist genauso möglich wie IF A=61 THEN . . ., zumindest in den meisten BASIC-Versionen. Wenn irgendeines der Felder des Adreßbuches bereits eine Eintragung enthält, so besteht diese garantiert aus mindestens einem alphanumerischen Zeichen.

Da die Felder für Namen, Straße, Stadt und Telefonnummer zusammengehören, befinden sich die Namens-Daten des 15ten Verzeichnisses im 15ten Element des Namensbereiches, die Straßen-Daten im 15ten Element des Straßen-Bereiches usw. Daher brauchen Sie nur einen dieser Bereiche zu durchsuchen, um ein leeres Element zu finden.

Wenn die Variable POSITION die Nummer des ersten freien Elements in einem der Bereiche repräsentiert, könnte ein Programm zum Lokalisieren dieser POSITION aussehen:

```
PROZEDUR (finde freies Element)
BEGIN (STARTE)
  LOOP (SCHLEIFE)
    REPEAT UNTIL (WIEDERHOLE BIS)
      freies Element lokalisiert ist
    READ (LESE) Bereich (POSITION)
    POSITION = POSITION + 1
    IF (WENN) Bereich (POSITION) = " "
      THEN (DANN) merke POSITION
      ELSE (SONST) mache nichts
    ENDIF (WENN FERTIG — WEITER)
  ENDLOOP (ENDE DER SCHLEIFE)
END (ENDE)
```

In BASIC könnte das dann so aussehen:

```
1000 FOR L=0 TO 1 STEP 0
1010 LET POSITION = POSITION + 1
1020 IF KPTNAME$(POSITION) = " " THEN LET
     L=X
1030 NEXT L
1040 REM (weitere Teile des Programms)
```

Beachten Sie, daß der Wert von X in Zeile 1020 der Wert ist, der benötigt wird, um die FOR. . .NEXT-Schleife zu beenden. Ebenso ist daran zu denken, daß dies nur ein Programm-Fragment ist und daß vorausgesetzt wird, daß KPTNAME$() DIMensioniert und POSITION initialisiert wurde.

POSITION ist ein Wert, den Sie möglichst frühzeitig bestimmen sollten, da er bei jeder Benutzung des Adreßbuch-Programms verwendet wird. Auch während des Programmlaufs muß er oft aktualisiert werden. Die Initialisierungs-Routine für diese Variable kann bei jedem Programmstart ausgeführt werden.

Lassen Sie uns die Möglichkeiten, die das Programm bieten soll, noch einmal zusammenfassen. Versuchen wir, eine allgemeine Strategie zu entwickeln. Dieses Mal werden wir einfach annehmen, daß jede Möglickeit eine Unterroutine ist (der Name jeder Unterroutine steht zwischen zwei Sternen).

1. Finde Verzeichnis (nach Namenseingabe)	*FINDVERZ*
2. Finde Namen (nach Eingabe eines unvollständigen Namens)	*FINDNAME*
3. Finde Verzeichnis (nach Stadteingabe)	*FINDSTADT*
4. Finde Verzeichnisse (Liste nach Eingabe von z. B. Initialen)	*FINDINIT*
5. Erstelle Liste (Komplettliste)	*ERSTLISTE*
6. Hinzufügen	*NEUEINGABE*
7. Ändern	*AENDERN*
8. Löschen	*LOESCHEN*
9. Verlasse Programm (speichern der Daten)	*ENDEPRG*

Alle weiteren Details können im Verlauf der verschiedenen Entwicklungsabschnitte ausgearbeitet werden. Wir wissen, daß einige Dinge initialisiert werden müssen, einschließlich des Wertes von POSITION. Da das Programm Menü-gesteuert sein soll, muß bei jedem Pro-

grammstart eine Liste mit Möglichkeiten auf dem Bildschirm erscheinen. Aufgrund dieser Kenntnis kann das Hauptprogramm bereits grob dargestellt werden:

HAUPTPROGRAMM
BEGIN (STARTE)
  INITIALISIERUNG (Prozedur)
  BEGRUESSUNG (Prozedur)
  AUSWAHL (Prozedur)
  AUSFUEHRUNG (Prozedur)
END (ENDE)

In BASIC würde das dann so aussehen (für die Namen der Unterroutinen müssen später Zeilennummern eingesetzt werden):

```
10 REM C. K. ADRESSBUCH-PROGRAMM
20 GOSUB *INITIALISIERUNG*
30 GOSUB *BEGRUESSUNG*
40 GOSUB *AUSWAHL*
50 GOSUB *AUSFUEHRUNG*
60 END
```

Die *BEGRUESSUNG*-Unterroutine bzw. -Prozedur würde für einige Sekunden eine Begrüßungs-Meldung auf dem Bildschirm darstellen. Danach folgt das Menü. Die Begrüßung kann beispielsweise so aussehen:

```
*WILLKOMMEN ZUM*
*ADRESSBUCH-PROGRAMM*
*DES COMPUTER-KURSES*
(DRUECKE DIE LEERTASTE,
  UM FORTZUFAHREN)
```

Sobald Sie die Leertaste drücken, verzweigt das Programm zur *AUSWAHL*-Unterroutine, und Sie sehen ein Menü, wie beispielsweise das folgende, auf dem Bildschirm:

```
*WOLLEN SIE*
1. VERZEICHNIS DURCH NAMEN SUCHEN
2. NAMEN DURCH TEIL EINES NAMENS
   SUCHEN
3. VERZEICHNISSE NACH STADTANGABE
   SUCHEN
4. VERZEICHNISLISTE DURCH INITIALEN
5. LISTE ALLER VERZEICHNISSE
6. HINZUFUEGEN EINER ADRESSE
7. AENDERN EINER ADRESSE
8. LOESCHEN EINER ADRESSE
9. PROGRAMM BEENDEN UND DATEN
   SPEICHERN
*WAEHLE 1 BIS 9*
*DRUECKE DANN RETURN*
```

An diesem Punkt verzweigt das Programm zur entsprechenden Unterroutine – je nachdem, welche Nummer eingegeben wurde. Jetzt nimmt die Struktur des Programms allmählich Form an. Alle Optionen, ausgenommen Nummer 9 (zum BEENDEN und SPEICHERN), müssen mit einer Anweisung beendet werden, die eine Rückkehr zur *AUSWAHL*-Unterroutine bewirkt. Doch es gibt noch viele Details der internen Organisation der Daten, die wir bisher noch nicht untersucht haben.

Nehmen Sie einmal an, daß das Programm gestartet wird, daß alle notwendigen Verzeichnisse bereits eingegeben wurden, und daß Sie ein Verzeichnis suchen wollen, indem Sie nur den Namen eingeben. Dazu müssen Sie Option 1 aufrufen – VERZEICHNIS DURCH NAMEN SUCHEN (*FINDVERZ*).

SUCHEN

Bücher über Programmiertechniken tendieren dazu, das Suchen und Sortieren zusammen zu behandeln. So kommen interessante Punkte über die Datenorganisation ans Tageslicht – sei es in einem Computer oder in einem anderen Informations-System.

Wenn bei einem „manuellen" Adreßbuch neue Eintragungen jeweils bei Bedarf ohne alphabetische Sortierung vorgenommen werden, spricht man von einer „Stapel"-Struktur. Ein Stapel ist eine Ansammlung von Daten, die in der Reihenfolge zusammengetragen wurden, in der sie anfielen. Es ist offensichtlich, daß ein Stapel die uneffizienteste Art der Datenorganisation ist. Jedes Mal, wenn Sie die Adresse und Telefonnummer einer Person finden wollen, müssen Sie das gesamte Adreßbuch durchblättern.

Eine Datenstruktur, die sowohl für den Menschen als auch für den Computer viel leichter zu handhaben ist, erreicht man, wenn die Daten anhand eines vorgegebenen einfachen Systems organisiert werden. Ein Telefonbuch ist ein gutes Beispiel für eine Ansammlung von Informationen, wobei das Namens-Feld nach den einfachen Regeln des Alphabetes geordnet ist. Die Zahlen und anderen Daten sind zufällig geordnet.

Mit unserem Adreßbuch-Programm werden die Daten als Stapel organisiert. Dabei wird eine Eintragung im Element X des Namen-Bereichs, Element X des Straßen-Bereichs usw. gespeichert. Die nächste Eintragung wird dann im Element X+1 des Namen-Bereichs, Element X+1 des Straßen-Bereichs usw. gespeichert. Wenn wir nun eine bestimmte Eintragung finden wollen – KARIN BOBEY zum Beispiel –, erfordert diese Datenorganisation, daß das erste Element im Namen-Bereich untersucht werden muß, ob es den Namen KARIN BOBEY enthält. Danach muß das zweite Element untersucht werden und so fort, bis entweder das entsprechende Element gefunden wird, oder aber die Tatsache klar wird, daß keine Eintragung unter KARIN BOBEY vorhanden ist.

Wenn sich die Daten jedoch schon in einer sinnvollen Ordnung befinden, wird der Vorgang viel einfacher. Stellen Sie sich vor, Sie

hätten eine Datei über Fußball-Vereine, und eines der Felder der Verzeichnisse enthielte die Wertung einer bestimmten Woche. Ein gutes Datenverwaltungsprogramm würde Ihnen ermöglichen, den Verein bzw. die Vereine herauszufinden, die in dieser Woche 11 Tore erzielt haben. Hier ist ein Bereich, der die Ergebnisse der Vereine einer Woche enthält:

1,6,2,2,1,9,0,0,2,1,4,11,4,2,12,5,2,1,0,1

Es ist offensichtlich, daß die Wertungen in der Reihenfolge der Vereine geordnet sind und nicht nach Punktzahl. Insgesamt sind 20 Vereine erfaßt. Nur einer war in der Lage, 11 Tore zu erzielen. Es war der zwölfte Verein.

Beim Analysieren der Daten stellt sich heraus, daß insgesamt 20 Wertungen in einem Bereich von 0 bis 12 Treffern erfaßt wurden. Dieses Beispiel ist natürlich sehr einfach, und es bedarf keines großen Zeitaufwandes, jeden Wert einzeln zu überprüfen. Doch was wäre, wenn wir es mit Tausenden von Elementen in einem großen Bereich zu tun hätten? Die Suche würde die Verarbeitungsgeschwindigkeit des Programms in einem unerwünschten Maße reduzieren.

Die Lösung ist, die Daten zuvor zu ordnen. So nimmt der Suchvorgang nur noch wenig Zeit in Anspruch. Hier ist wieder unser Bereich mit den Vereins-Wertungen, dieses Mal in numerischer Reihenfolge sortiert:

0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,9,11,12

Wenn wir wissen, daß die Anzahl der erfaßten Vereine 20 ist, wäre der einfachste Weg zum Finden der Position der gesuchten Wertung, den Bereich in zwei Hälften zu unterteilen und nur den Teil, der vermutlich die gesuchte Wertung enthält, zu durchsuchen. Der Algorithmus zum Lokalisieren der gesuchten Wertung sähe nun so aus:

Finde den Bereich, der die Wertungen enthält.
Lese die Zahl, nach der wir suchen.
Finde die Länge des Bereiches.
Finde den Mittelpunkt des Bereiches.
Führe Schleife aus, bis Zahl gefunden.
  Wenn die Zahl am Mittelpunkt gleich der Zahl ist, die wir suchen, dann ist die Zahl lokalisiert.
  Wenn nicht, prüfe, ob die gesuchte Zahl größer oder kleiner als die Zahl am Mittelpunkt ist.
  Wenn die gesuchte Zahl größer ist, dann finde den Mittelpunkt des oberen Teils des Bereiches.
  Wenn die gesuchte Zahl kleiner ist, dann finde den Mittelpunkt des unteren Teils des Bereiches.
  (Wiederhole dies, bis die Zahl lokalisiert ist.)

Dies kann wie folgt formalisiert werden:

```
BEGIN (STARTE)
  Finde den Bereich mit den Wertungen
  INPUT ZAHL (die gesucht werden soll)
  LOOP (SCHLEIFE) bis die Zahl
    lokalisiert ist
    IF (WENN) ZAHL = (Mittelpunkt)
      THEN (DANN) merke Position des
        Mittelpunktes
      ELSE (SONST)
        IF (WENN) ZAHL > (Mittelpunkt)
          THEN (DANN) finde Mittelpunkt
            obere Hälfte des Bereiches
          ELSE (SONST) finde Mittel-
            punkt untere Hälfte
        ENDIF (WENN FERTIG —
          WEITER)
    ENDIF (WENN FERTIG — WEITER)
  ENDLOOP (ENDE DER SCHLEIFE)
  IF (WENN) ZAHL ist lokalisiert
    THEN (DANN) PRINT Position des
      Mittelpunktes
    ELSE (SONST) PRINT "ZAHL NICHT
      GEFUNDEN"
  ENDIF (WENN FERTIG — WEITER)
END (ENDE)
```

Wenn Sie dieses Programm in unserer Pseudo-Sprache betrachten, werden Sie feststellen, daß es die gesuchte Zahl unfehlbar sucht und finden wird, vorausgesetzt, sie existiert im angegebenen Bereich. Lassen Sie uns diese Pseudo-Sprache weiterentwickeln, bis wir ein fertiges Arbeitsprogramm schreiben können. Diesen Vorgang des Suchens durch wiederholte Unterteilung nennt man übrigens „binäres Suchen".

Nachfolgend finden Sie ein BASIC-Programm, das auf unserem Programm in Pseudo-Sprache basiert. Es erstellt einen Bereich und liest die Wertungen aus einer DATA-Anweisung. Danach fragt es nach der gesuchten Wertung. Wenn es sie findet, gibt es das Element des Bereiches aus, in dem es die Zahl gefunden hat.

```
10 REM PROGRAMM ZUM LOKALISIEREN
   EINER ZAHL IN EINEM BEREICH
20 DIM WERTUNG (20)
30 FOR Z=1 TO 20
40 READ WERTUNG (Z)
50 NEXT Z
60 DATA 0,0,0,1,1,1,1,1,2,2,2,2,2,4,4,5,6,9,11,12
70 LET L=20
80 LET BTM=1
90 LET TP=L
100 INPUT "WERTUNG EINGEBEN"; N
110 FOR Z=0 TO 1 STEP 0
120 LET L=TP-BTM
130 LET MD=BTM+INT (L/2)
140 IF N = WERTUNG(MD) THEN LET Z=X
150 IF N > WERTUNG (MD) THEN LET
```

```
    BTM=MD
160 IF N < WERTUNG (MD) THEN LET
    TP=MD
170 NEXT Z
180 PRINT "DIE WERTUNG BEFINDET SICH
    IN ELEMENT NR. ";MD
190 END
```

Und wieder sollten Sie beachten, daß X entsprechend den Besonderheiten Ihres Computers initialisiert werden muß (siehe BASIC-Dialekte).

Wenn die Daten in einer Datei oder einem Bereich einigermaßen geordnet sind, so wie bei einem Telefonbuch, und wenn die Namen verhältnismäßig gleichmäßig über das Alphabet verteilt sind, dann ist die binäre Suchweise ein effizienter Weg zum Auffinden spezieller Informationen. Trotzdem ist dies nicht die beste Lösung. Es gibt alternative Algorithmen, die gesuchte Daten mit erheblich geringerem Aufwand finden können. Eine dieser anderen Techniken ist die des „Jagens". Hierbei führt das Programm eine grobe Vorausberechnung durch, an welchem Ort sich die gesuchten Daten befinden könnten, und verfeinert diese Berechnungen, bis das Ziel erreicht ist.

Übungen

Wenn Sie das obige Programm starten, werden Sie sehen, daß es wie geplant läuft – vorausgesetzt, daß die gesuchte Wertung im vorgegebenen Bereich vorhanden ist. Wenn Sie eine Wertung von 3 eingeben, die sich nicht in dem Bereich befindet, bricht das Programm nicht ab, und es erscheint keine Fehlermeldung. Wenn Sie die Wertung 12 eingeben, die im Bereich vorkommt, versagt das Programm bei der Lokalisation. Außerdem geht das Programm davon aus, daß jede Zahl im Bereich unterschiedlich ist, was jedoch nicht der Fall ist. Das Programm wird dies niemals bemerken und nur eine der Positionen nennen.

1. Analysieren Sie das Programm, und finden Sie heraus, warum es die Wertung 12 nicht lokalisieren kann.
2. Modifizieren Sie eine Zeile des Programms, um diesen Fehler zu beheben.
3. Finden Sie heraus, weshalb das Programm alle Zahlen, die im Bereich nicht vorkommen, auch nicht verarbeiten kann. Suchen Sie nun nach einem Weg, dieses Problem zu umgehen.

BASIC-DIALEKTE

Im folgenden sehen Sie das Spectrum-Listing für den ersten BASIC-Programmteil:

```
100 DIM F$ (6,4)
110 LET POSITION=0
120 LET F$(1)="MIKE"
130 LET F$(2)="KARIN"
140 LET F$(4)="MARY"
1000 FOR L=0 TO 1 STEP 0
1010 LET POSITION=POSITION+1
1020 IF F$(POSITION)=" " THEN
     LET L=2
1030 NEXT L
1040 PRINT"NR. DES 1TEN FREIEN
     ELEMENTES IST ";POSITION
1050 STOP
```

Beachten Sie, daß die Zeilen 100 bis 140, ebenso wie Zeile 1040, dazu dienen, das Programm-Fragment (Zeilen 1000 bis 1030) in eine lauffähige Testfassung umzuwandeln. Die Werte und Formate dieser zusätzlichen Zeilen können natürlich verändert werden.

Das zweite Programm für den Spectrum sieht wie folgt aus:

```
10 REM PROGRAMM ZUM LOKALI-
   SIEREN VON ZAHLEN IN EINEM
   BEREICH
20 DIM S(20)
30 FOR Z=1 TO 20
40 READ S(Z)
50 NEXT Z
60 DATA 0,0,0,1,1,1,1,1,2,2,2,2,
   2,4,4,5,6,9,11,12
70 LET L=20
80 LET BTM=1
90 LET TP=L
100 INPUT "WERTUNG EIN-
    GEBEN ";N
110 FOR Z=0 TO 1 STEP 0
120 LET L=TP-BTM
130 LET MD=BTM+INT(L/2)
140 IF N = S(MD) THEN LET Z=2
150 IF N > S(MD) THEN LET
    BTM=MD
160 IF N < S(MD) THEN LET
    TP=MD
170 NEXT Z
180 PRINT "DIE WERTUNG BEFIN-
    DET SICH IN ELEMENT
    NR. ";MD
190 STOP
```

Die Besonderheiten von Variablennamen bei unterschiedlichen Geräten lesen Sie bitte im letzten Heft nach.

In beiden Programmen des Haupttextes erscheint folgende Anweisung: „... THEN LET Z=X". Die Werte für die Variable X sind:

ORIC X=1
Dragon X=1
Acorn B X=2
Commodore 64
und VC 20 X=1

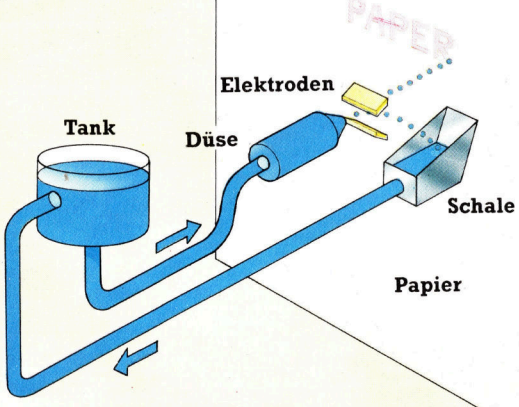
Beide Programme im Haupttext laufen auf dem Acorn B, dem Dragon 32, dem Oric, dem Commodore 64 und dem VC 20, vorausgesetzt, daß die BASIC-Dialekte der Variablen-Namen und die Anweisung STEP 0 beachtet werden. Das Programm für den Spectrum unterscheidet sich bei der DIM-Anweisung in Zeile 100 sowie beim Test in Zeile 1020. Verwenden Sie sie als Hilfe bei der Anpassung für andere Computer.

Mit Düsen-antrieb

Grafik-Ausdruck mit vollem Farbumfang ist jetzt erschwinglich geworden – ein Drucker, der punktweise farbige Tinten auf das Papier sprüht, macht's möglich.

Lenkgeschosse

Die ersten Tintenstrahldrucker waren mit einem sehr aufwendigen Schreibsystem ausgestattet. In der Düse erzeugte ein Piezokristall einen kontinuierlichen Strom geladener Tintentröpfchen. Diese wurden durch ein Elektrodenpaar vertikal abgelenkt, während der Schreibkopf über das Papier lief. Wo das Papier weiß bleiben sollte, ließ man die Tropfen in eine Fangschale fallen, deren Inhalt in den Vorratsbehälter zurückgeführt wurde.



Die verschiedenen Druckersysteme, die es für Heimcomputer gibt, liefern ungleiche Schriftqualitäten. Die besten Ergebnisse werden von den Druckern mit Ganzzeichen-Anschlag erreicht (hervorragend die Typenraddrucker), und am schlechtesten ist die Wiedergabe bei den elektrostatischen und den Thermo-Druckern. Am gängigsten sind im Heimcomputerbereich die Matrixdrucker, trotz Lärm und mäßigem Schriftbild.

Mit dem Auftauchen der ersten „Printerplotter" wurden die Mängel des Nadeldruckverfahrens überdeutlich. Die Printerplotter arbeiten mit kleinen Faser- oder Kugelschreiberpatronen, die Schriftzeichen oder Linien auf das Papier bringen. Manche Geräte ermöglichen einen Ausdruck in vier Farben.

Die größten Chancen gegen die Matrixdrukker hat aber wohl ein Verfahren, bei dem mit feinsten Tintentröpfchen ein Muster auf das Papier geschossen wird – so arbeiten „Tintenstrahl(ink jet)-drucker".

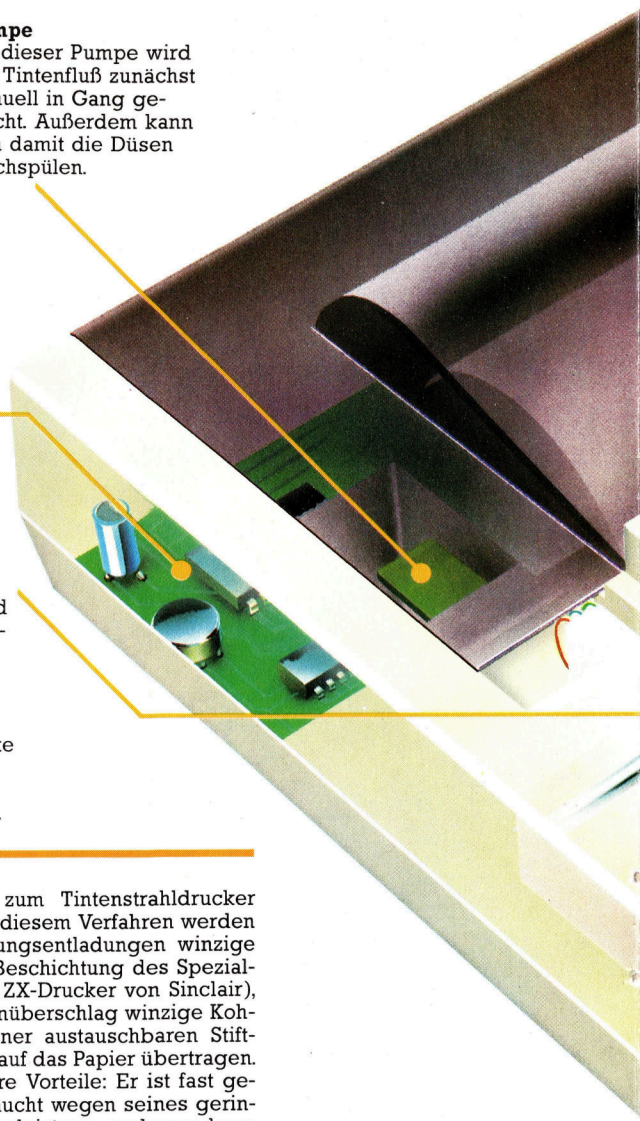
Diese Geräte haben bereits ihren festen Platz im kommerziellen Bereich (wie die ähnlich raffinierten Laserdrucker) und kommen allmählich auch auf den Heimcomputermarkt. Die Tinte wird aus einem Tank zu einer feinen Düsenspitze gepumpt, wo die winzigen Tröpfchen elektrisch geladen austreten und weitergeleitet werden. Eine definierte Tröpfchenbildung wird dabei durch Ultraschallschwingungen in einem piezoelektrischen „Ventil" erreicht.

Die austretenden Tröpfchen werden von einem elektrischen Feld aufgefangen und auf das Papier geschossen. Dieses ist über eine Blechplatte gespannt (und nicht über Hartgummi wie beim Anschlagdrucker). Das Blech ist entgegengesetzt zu den Tintentropfen aufgeladen und zieht sie deshalb von vorn auf das Papier. Dieses Verfahren klingt abenteuerlich; es funktioniert aber überraschend sauber – schlimmstenfalls verstopft die Düse, oder die Tropfen werden zu groß.

Pumpe
Mit dieser Pumpe wird der Tintenfluß zunächst manuell in Gang gebracht. Außerdem kann man damit die Düsen durchspülen.

Leiterplatte
Der Drucker enthält einen eigenen 6809-Microprozessor sowie ein ROM und ein RAM. Alle eintreffenden Daten werden in einem Buffer abgelegt, weil bei jedem Durchlauf des Schreibkopfs nur eine Punktreihe erzeugt wird.

Schreibkopf-Deckel
Die Düsenspitzen sind ein kritischer Punkt, und deshalb muß der Sprühkopf bei Nichtgebrauch in der Ruhestellung abgedeckt werden. Nach dem Einschalten des Druckers sind bestimmte Betriebsvorbereitungen zu treffen, um Schäden am Gerät zu vermeiden.



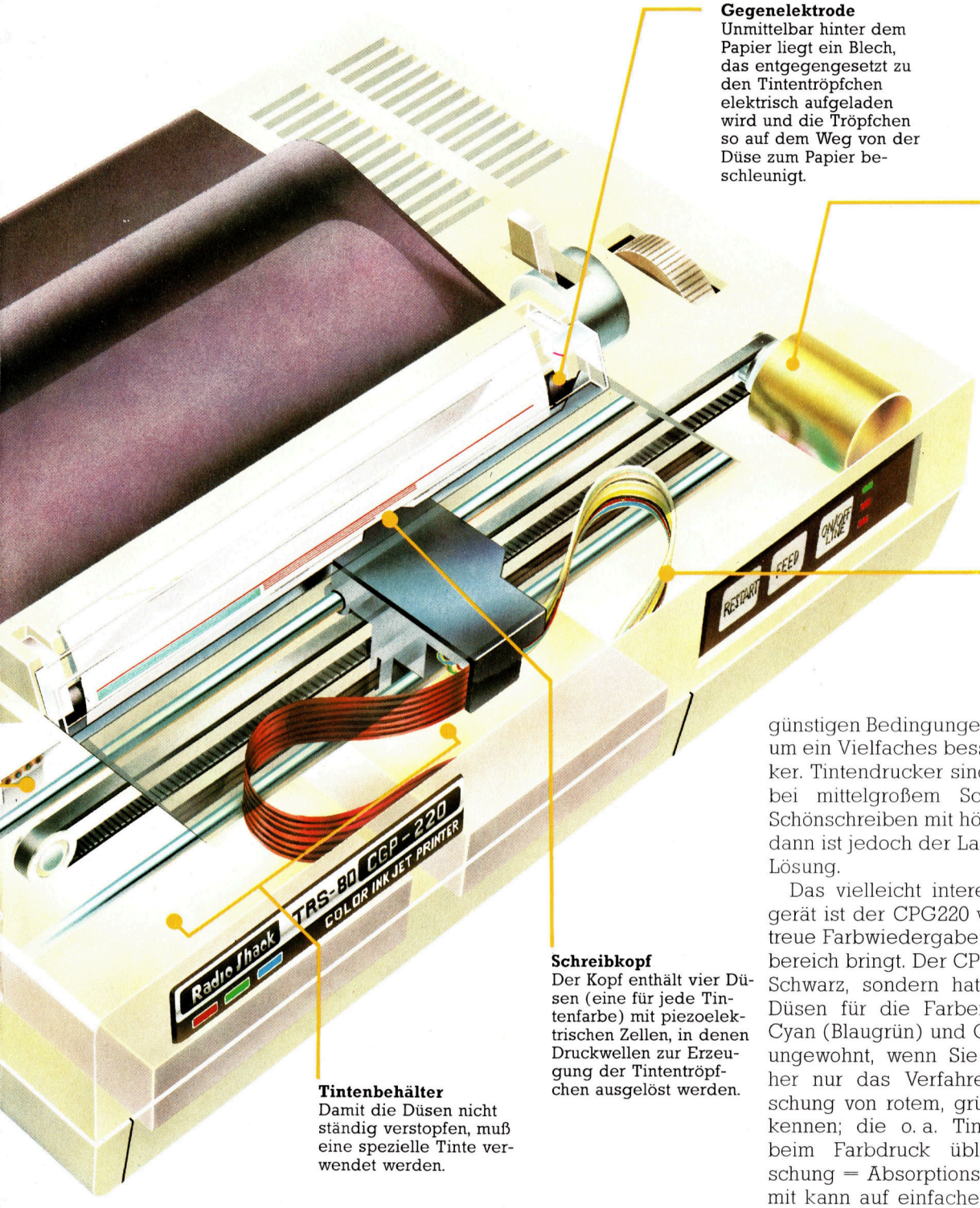
Druckertypen
Eine interessante Variante zum Tintenstrahldrucker sind die Thermodrucker. Bei diesem Verfahren werden entweder durch Hochspannungsentladungen winzige Löcher in eine metallisierte Beschichtung des Spezialpapiers gebrannt (z. B. beim ZX-Drucker von Sinclair), oder es werden durch Funkenüberschlag winzige Kohlepartikel von der Spitze einer austauschbaren Stiftelektrode als „Trockentinte" auf das Papier übertragen.

Dieser Drucker hat mehrere Vorteile: Er ist fast geräuschlos, der Druckkopf braucht wegen seines geringen Gewichts wenig Antriebsleistung, und man kann fast jedes Papier benutzen. Nachteilig ist eigentlich nur die niedrige Druckgeschwindigkeit (weil bei jedem Hin- und Rücklauf des Kopfes nur eine Punktreihe erzeugt wird).

Die Arbeitsweise des Tintenstrahldruckers läßt sich mit der eines Matrixdruckers, der nur mit einer Nadel ausgestattet ist, vergleichen: Eine eintreffende Kette von ASCII-Zeichen wird in einem Buffer abgelegt, bis dieser voll ist oder ein Wagenrücklauf ansteht. Der „Zeichengenerator" nimmt sich die Kette dann zeichenweise vor und erzeugt die zugehörigen Punktmuster (z. B. in einem 8 × 8-Raster), die zu Papier gebracht werden sollen. Für jede Zeichenreihe muß der Schreibkopf achtmal über das Papier laufen, wobei zur Beschleunigung im Hin- und Rücklauf gedruckt werden kann. Während so der eine Buffer abgearbeitet wird, füllt sich ein zweiter Buffer, der nach Leerung des ersten an die Reihe kommt; inzwischen wird der erste wieder geladen.

Professionelle Tintenstrahldrucker (z. T. mit Mehrdüsen-Kopf) schaffen eine Seite in wenigen Sekunden. Die Schriftqualität wird mit vom Papier beeinflußt: Je saugfähiger das Papier, desto geringer die Konturenschärfe. Unter günstigen Bedingungen ist das Schriftbild aber um ein Vielfaches besser als beim Nadeldrukker. Tintendrucker sind das Richtige fürs Büro bei mittelgroßem Schreibaufkommen; wird Schönschreiben mit höchstem Tempo verlangt, dann ist jedoch der Laserdrucker die optimale Lösung.

Das vielleicht interessanteste Tintensprühgerät ist der CPG220 von Tandy, der eine getreue Farbwiedergabe für den Heimcomputerbereich bringt. Der CPG220 druckt nicht nur in Schwarz, sondern hat außerdem Tanks und Düsen für die Farben Magenta (Violettrot), Cyan (Blaugrün) und Gelb. Diese Farben sind ungewohnt, wenn Sie von der Farbbildröhre her nur das Verfahren der „additiven" Mischung von rotem, grünem und blauem Licht kennen; die o. a. Tintenfarben sind jedoch beim Farbdruck üblich („subtraktive" Mischung = Absorptionsüberlagerung, auch damit kann auf einfache Weise jeder beliebige Farbton erzeugt werden).



Gegenelektrode
Unmittelbar hinter dem Papier liegt ein Blech, das entgegengesetzt zu den Tintentröpfchen elektrisch aufgeladen wird und die Tröpfchen so auf dem Weg von der Düse zum Papier beschleunigt.

Antriebsmotoren
Für den Hin- und Rücklauf des Schreibkopfes sorgt ein ganz gewöhnlicher Elektromotor, während für den Papiertransport ein Schrittmotor zuständig ist.

Flexible Leitungen
Wie bei den meisten Druckern dienen auch hier Flachbandkabel zur Verbindung von Schreibkopf und Leiterplatte. Beim Tintenstrahldrucker bereitet die Zuführung der vier verschiedenen Tinten zu dem sich schnell bewegenden Schreibkopf zusätzliche Schwierigkeiten.

Schreibkopf
Der Kopf enthält vier Düsen (eine für jede Tintenfarbe) mit piezoelektrischen Zellen, in denen Druckwellen zur Erzeugung der Tintentröpfchen ausgelöst werden.

Tintenbehälter
Damit die Düsen nicht ständig verstopfen, muß eine spezielle Tinte verwendet werden.

Satter Klang

Tonerzeugung auf dem Acorn B.

Mit seinen ausgezeichneten Möglichkeiten zur Tonerzeugung und einem hervorragend darauf abgestimmten BASIC-Befehlssatz gehört der Acorn B in die Spitzengruppe der Heimcomputer, die sich besonders für die Klanggenerierung eignen. Das Standardgerät enthält drei einzeln ansprechbare Oszillatoren für Rechteckschwingungen, acht Arten von Rauschen und vier Hüllkurven für Tonhöhen und ADSR, die sich unabhängig voneinander steuern lassen. Bis zu drei Stimmen können gleichzeitig gespielt werden.

Am einfachsten läßt sich ein Ton mit dem SOUND-Befehl erzeugen:

```
SOUND C,V,P,D
```

Die Variablen bedeuten dabei:

C = Kanal oder Oszillator (0—3)
V = Lautstärke
P = Tonhöhe
D = Tondauer oder Notenwert

Die Oszillatoren 1 bis 3 erzeugen Töne, der Oszillator 0 das Rauschen. Die Lautstärke läßt sich mit Werten zwischen 0 (aus) und −15 (laut) bestimmen, während die Tonhöhe in Vierteln eines Halbtones angegeben wird. Der tiefste Ton (mit dem Wert 0) ist A# (mit 115,5 Hz) und der höchste (mit dem Wert 253) ist D (mit 4698,64 Hz). Der Acorn B verfügt damit über einen Tonumfang von fünfeinhalb Oktaven, wobei das mittlere C den Wert 53 hat. Es gibt acht Arten von Rauschen, die über P (Tonhöhe) festgelegt werden:

Zahl	Art des Rauschens
0	"Tremolo" hohe Tonlage
1	"Tremolo" mittlere Tonlage
2	"Tremolo" tiefe Tonlage
3	"Tremolo" Die Tonlage wird von der Tonhöhe des Kanals 1 bestimmt.
4	hohe Tonlage
5	mittlere Tonlage
6	tiefe Tonlage
7	Die Tonlage wird von der Tonhöhe des Kanals 1 bestimmt.

Die Dauer eines Tones wird über Zahlen zwischen 1 und 255 festgelegt, wobei die kleinste Zeiteinheit ein Zwanzigstel einer Sekunde ist und die maximale Tondauer 12,75 Sekunden beträgt. Der Befehl, das A über dem mittleren C auf Kanal 1 eine halbe Sekunde lang mit einer Lautstärke von −7 erklingen zu lassen, lautet:

```
SOUND 1,—7,89,10
```

Erhält der Computer während der Ausführung eines SOUND-Befehls einen weiteren Befehl, wird der neue Ton in die Warteschlange des entsprechenden Kanals eingereiht.

Zusätzlich zu diesen einfachen Funktionen läßt sich der SOUND-Befehl aber noch folgendermaßen erweitern:

```
SOUND &HSFC,V,P,D
```

In diesem Fall weist das „&" den Computer an, HSFC als eine vierstellige Hexadezimalzahl zu verstehen. Dabei sind nur die ersten drei Zahlen wichtig, da C wie bei dem normalen SOUND-Befehl den Kanal angibt.

H = 1 läßt einen Ton ausklingen, bevor auf dem gleichen Kanal ein neuer Ton beginnt. Ist H = 0 (nicht aktiv), wird die Ausklingzeit (Release) eines Tones von dem nächsten Ton abrupt unterbrochen.

S gibt die Möglichkeit, einen Akkord zu spielen. S = 0 schaltet diese Funktion aus. Wird S auf 1 gesetzt, kann ein weiterer Kanal zur gleichen Zeit einen zusätzlichen Ton erzeugen, während sich über S = 2 zwei Töne gleichzeitig produzieren lassen. Findet der Computer einen Wert von S, der über 0 liegt, dann wartet er mit dem Spielen dieses Tones, bis er in seinem Programm auf ein oder zwei weitere Töne trifft, die für die anderen Kanäle einen entsprechenden Wert S enthalten, und spielt erst dann alle Töne gemeinsam.

F wird ebenfalls auf 0 oder 1 gesetzt. Null hat keine Wirkung, 1 dagegen bewirkt, daß alle Töne, die sich in der Warteschleife dieses Kanals befinden, gelöscht werden, der augenblickliche Ton unterbrochen und der Ton des SOUND &-Befehls, in dem das F gesetzt wurde, gespielt wird.

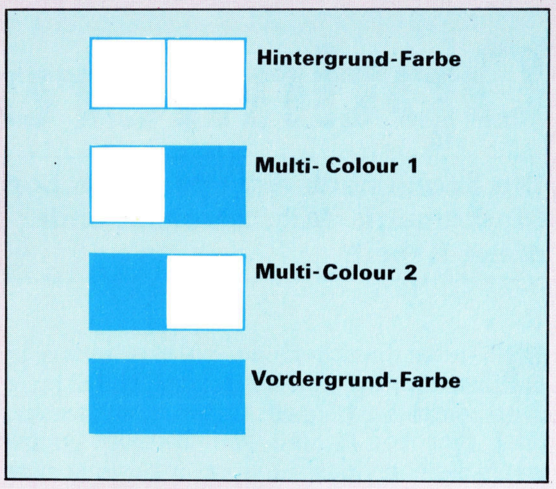
SOUND & läßt sich am besten anhand eines Beispiels darstellen. Das folgende Programm spielt die erste Zeile von „Happy birthday to you" in G#:

```
G# G# A# G# C# C

10 SOUND 1,—7,37,10: REM * 1. G# *
20 SOUND 1,—7,37,10: REM * 2. KURZES
   G# *
30 FOR I=1 TO 3: READ N
40 SOUND 1,—7, N,10: NEXT I: REM
   * A#,G#,C#*
50 SOUND &201, —7,37,15: REM *G#*
60 SOUND &202, —7,53,15: REM *C*
70 SOUND &203,—7,65,15: REM *D#*
80 DATA 45,37,57
90 END
```

Lichte Höhe

Die Grafikmöglichkeiten des Commodore 64.



Der Commodore 64 bietet eine Vielzahl von Grafikmöglichkeiten. Von Nachteil ist dabei allerdings – ähnlich wie bei dem VC 20 – der außerordentlich begrenzte Befehlssatz des mitgelieferten BASIC. Über POKE und PEEK lassen sich zwar alle Möglichkeiten des Gerätes ansprechen, aber diese Methode der Programmierung kann recht kompliziert werden. Auch hier bietet sich ein Ausweg: Eine Anzahl unabhängiger Hersteller bietet Softwarepakete an, die die Verwendung von Sprites oder die Neudefinition von Zeichen einfacher gestalten; Commodore liefert zur Lösung dieses Problems eine Zusatzcartridge mit „Simon's BASIC".

Interessante Darstellungen

Der Commodore 64 besitzt einen Standardzeichensatz mit Groß- und Kleinbuchstaben, die sich normal oder invers darstellen lassen. Weiterhin gibt es spezielle Grafikzeichen, die ursprünglich für den PET entwickelt wurden und auch auf dem VC 20 zur Verfügung stehen. Das Bildschirmformat hat 25 Zeilen mit je 40 Zeichen. Die Zeichen lassen sich in jeder gewünschten Farbe entweder über den PRINT-Befehl auf den Bildschirm bringen oder durch die Belegung der entsprechenden Speicherstellen im Bildschirm- und Farbspeicher (siehe Handbuch) ansprechen. Mit den speziellen Grafikzeichen des Commodore läßt sich in der niedrigen Auflösung eine Vielzahl interessanter Darstellungen programmieren. Es sind mehr als 60 Spezialzeichen vorhanden, die normal oder invers dargestellt werden können. Sie haben damit über 120 Grafikelemente zur Verfügung. Sollten diese einmal nicht ausreichen, können Sie das gewünschte Zeichen selbst definieren. Dieser Vorgang ist leider nicht ganz einfach, da erst die normalen Zeichen vom ROM in den RAM-Bereich kopiert werden müssen, bevor die neuen Zeichen durch entsprechende POKE-Werte definiert werden können.

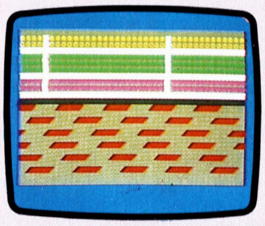
Jedes Zeichenfeld besteht aus einer Matrix von 8 × 8 Pixeln (Bildpunkten), die in binärer Form als Nullen und Einsen dargestellt werden. Normalerweise ist 1 die Farbe des Vordergrundes oder die Zeichenfarbe und 0 die Hintergrund- oder Bildschirmfarbe. Es gibt aber die Möglichkeit, innerhalb eines Zeichenfeldes bis zu vier Farben darzustellen. Dabei wird die Farbe eines Bildpunktes über zwei Bits festgelegt. Diese Darstellungsmethode heißt „Multi-Colour Modus".

Die Anzahl der Farben in einem Zeichenfeld ist auf vier begrenzt, da sich mit zwei Bits nur vier Kombinationen darstellen lassen.

Das Commodore-BASIC enthält keine Befehle für hochauflösende Grafik. Mit Hilfe einer Technik, die „Bit-Mapping" genannt wird, läßt sich das Ziel aber trotzdem erreichen. Der Commodore 64 baut seine Bildschirmdarstellung aus 64 000 Bildpunkten auf, die sich über die Technik des Bit-Mapping einzeln festlegen lassen. Dieser Vorgang ist jedoch recht kompliziert zu programmieren. Zudem baut sich der Bildschirm bei BASIC-Programmen nur sehr langsam auf. Es gibt allerdings zwei Möglichkeiten, eine schnelle hochauflösende Grafik zu erhalten: durch die Cartridge mit Simon's BASIC von Commodore oder durch eine Programmierung im Maschinencode.

Das folgende kurze Programm verwendet die Grafikzeichen des Commodore für die Darstellung eines Regals in einem Supermarkt. Im weiteren Verlauf des Kurses werden wir genauer auf den Einsatz von Sprites und auf den Gebrauch von Simon's BASIC eingehen.

Dieses Bild eines Supermarktes wurde über Grafik mit niedriger Auflösung programmiert. Im weiteren Verlauf dieser Serie werden wir einen Käufer mit dem Einkaufswagen auf den Bildschirm zaubern.



```
3000 REM ** SUPERMARKET **
3010 PRINT"■"
3020 POKE53280,6:POKE53281,12
3030 PRINT"■●●●●■ ■●●●●●●●●●●●●●●●●●●●●●●●●■ ■●●●●●●●●●●●●●●"
3040 PRINT"■OOOO■ ■OOOOOOOOOOOOOOOOOOOOOOOO■ ■OOOOOOOOOOOOOO"
3050 PRINT"■                                              ■"
3060 PRINT"■♦♦♦♦■ ■♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦■ ■♦♦♦♦♦♦♦♦♦♦♦♦♦♦"
3070 PRINT"■♦♦♦♦■ ■♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦■ ■♦♦♦♦♦♦♦♦♦♦♦♦♦♦"
3080 PRINT"■♦♦♦♦■ ■♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦■ ■♦♦♦♦♦♦♦♦♦♦♦♦♦♦"
3090 PRINT"■                                              ■"
3100 PRINT"■♥♥♥♥■ ■♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥■ ■♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
3110 PRINT"■♥♥♥♥■ ■♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥■ ■♥♥♥♥♥♥♥♥♥♥♥♥♥♥"
3120 PRINT"■                                              ■"
3130 PRINT"■                                              ■"
3140 PRINT"■  ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤"
3150 PRINT
3160 PRINT" ■◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤"
3170 PRINT
3180 PRINT"◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤  "
3190 PRINT
3200 PRINT"■ ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤ '
3210 PRINT
3220 PRINT"■  ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤"
3230 PRINT
3240 PRINT" ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤   ◤"
3250 FORI=1063TO2023STEP40
3260 POKEI,160:POKE54272+I,6:NEXT
3270 GOTO3270
```

Optimierungen

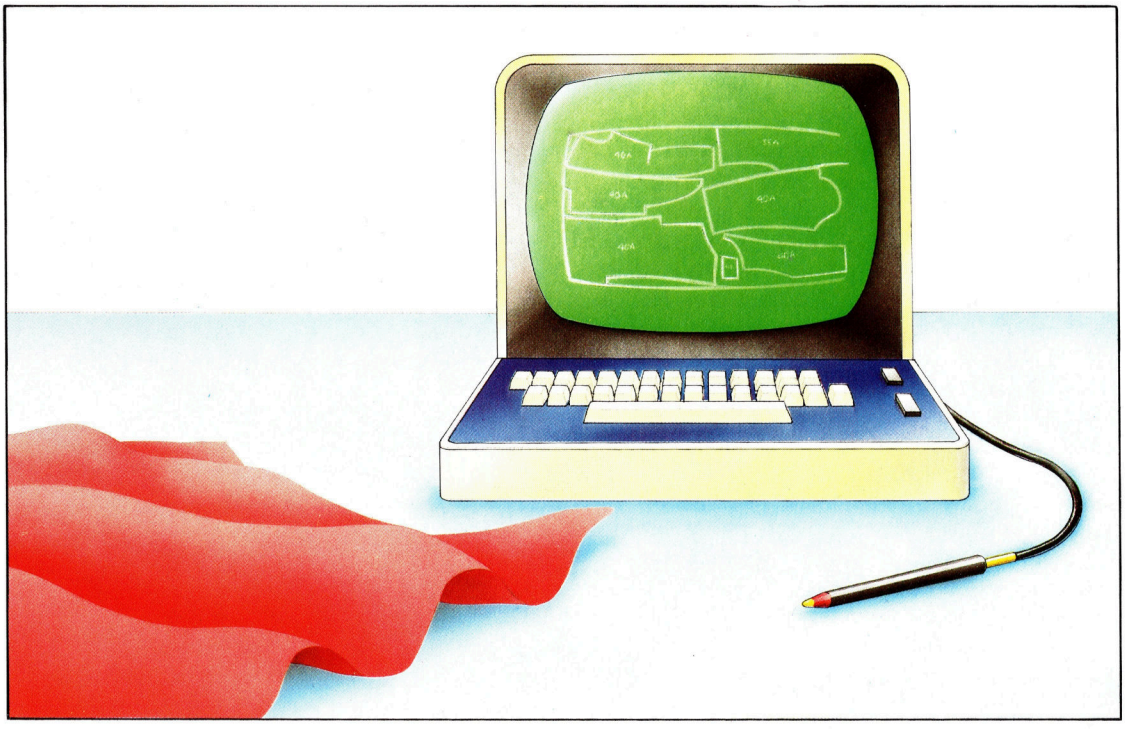
Die Suche nach der optimalen Lösung erfordert häufig komplexe Mathematik. Mit fortschreitender Technik übernehmen Computer diese Arbeit.

Bei jeder Entscheidung, die Sie treffen, ist unweigerlich ein Kompromiß enthalten – zum Beispiel zwischen Kosten und Leistung oder zwischen Kosten und Zeit. Es ist unwahrscheinlich, daß die absolut beste Lösung zu den absolut geringsten Kosten erreicht werden kann. Das „optimale" Ergebnis wird irgendwo in der Mitte zwischen beiden Möglichkeiten liegen.

Folgende Überlegungen würden Sie zum Beispiel bei der Auswahl zwischen zwei verschiedenen Waschmitteln anstellen: „650 Gramm dieses Pulvers kosten 2,16 DM, 1300 Gramm des anderen kosten 3,89 DM. Wenn ich nun aber 20 Prozent mehr vom billigeren Waschmittel benötige, um das gleiche Ergebnis zu erhalten, welches Mittel ist dann preiswerter?" Falls Sie diese Gedanken in Form einer einfachen Gleichung – in diesem Fall durch Prozentdifferenzen – ausdrücken, ist die Antwort leicht zu ermitteln.

Sie könnten aber auch den verschiedenen Faktoren unterschiedliche Bedeutung beimessen. (Vielleicht ist Ihnen die Kostenersparnis wichtiger als das Waschergebnis.) Die Lösung müßte dann natürlich anders aussehen. Das Prinzip, eine Berechnung mit einem konstanten Wert zu „gewichten", funktioniert gut, wenn die Differenzen zwischen gleichen Größen (zum Beispiel Preis und Gewicht) ebenfalls konstant sind. Wenn aber diese Differenzen sich in unterschiedlichem Maß verändern, wird logischerweise auch die erforderliche Mathematik komplizierter. Sind nur wenige Faktoren zu berücksichtigen, dann eignet sich die Methode der Matrizenrechnung. Bei größeren Mengen von veränderlichen Daten wird die lineare Optimierung angewendet. Ein anderer Weg ist, die Lösung abzuschätzen und dann den Wert schrittweise zu verändern, bis alle Bedingungen erfüllt sind (Näherungsverfahren). Je besser die erste Schätzung ist, um so schneller wird dieser Prozeß zum Ziel führen.

Optimierungsstrategien wie die genannten sind für Handel und Industrie außerordentlich wichtig. Sie werden besonders in den Bereichen Serienfertigung und Baugewerbe eingesetzt. Lineare Programmierung, CPA (Critical Path Analysis) und PERT (Program Evaluation Research Technique) sind einige der Namen für solche Methoden. In ihren ursprünglichen Formen tauchten sie etwa 30 Jahre vor Anbruch des Computerzeitalters auf. Die Methoden waren sehr arbeitsaufwendig. Heute können Mi-



Das Anordnen von Schnittmusterteilen auf einer Stoffbahn, so daß möglichst wenig Abfall entsteht, ist ein gutes Beispiel für rechnerunterstützte Optimierung. Hier zeigt der Computer seinen Vorschlag auf dem Schirm, und ein erfahrener Bediener kann Korrekturen mit Hilfe eines Lightpens durchführen.

	Preise in Pfennigen	Eiweiß g	Kohlenhydrate g	Fett g	Kilojoule	Benötigte Menge
400 g Kartoffeln	30	8	—	64	1204	10 kg
200 g Sardinen	150	41	49	1	2554	—
400 g Steak	720	84	28	0	4856	—
1/2 Liter Milch	60	16	18	24	945	6 l
Mindestbedarf pro Woche		360	215	505	36925	

Die optimale Diät

Hier soll die beste Kombination von vier Nahrungsmitteln zu einer vorgegebenen Minimaldiät bei möglichst geringen Kosten gefunden werden. Hierzu geben Sie dem Computer ein: die Nährwerte (rosa) und den Preis jedes Nahrungsmittels (blau) sowie den Minimalbedarf an jedem Nährstoff pro Woche (gelb). Der Rechner bestimmt das kritischste Element und manipuliert den Rest des Rasters entsprechend, um die optimale Verteilung (grün) zu finden. In diesem Beispiel wird der Bedarf ausschließlich mit Milch und Kartoffeln gedeckt, bei knappen Kosten von 14,70 DM.

crocomputer solche Aufgaben übernehmen.

Eine Branche, die beträchtlich von der Entwicklung der Optimierungsmethoden profitiert hat, ist die Textilindustrie. Stoffe werden üblicherweise in standardisierten Breiten geliefert. Und das Problem des Herstellers liegt darin, den Anfall von Verschnitt beim Zuschneiden des Stoffes zu minimieren und gleichzeitig wichtige Faktoren, wie die Laufrichtung des Gewebes oder die Musterung, zu beachten.

Verbesserte Muster

In den modernen Textilfabriken Europas wird die Plazierung von Schnittmustern für Konfektionskleidung auf einem vorgegebenen Stück Stoff mit Optimierungstechniken berechnet, und das vorgeschlagene Ergebnis wird auf einem Bildschirm angezeigt. An diesem Punkt wird der Computerbediener benötigt, um sein Urteil und seine Erfahrung zur Verbesserung der Berechnung des Computers anzubringen. Der Anwender macht durchschnittlich bei einem von fünf Mustern eine Verbesserung.

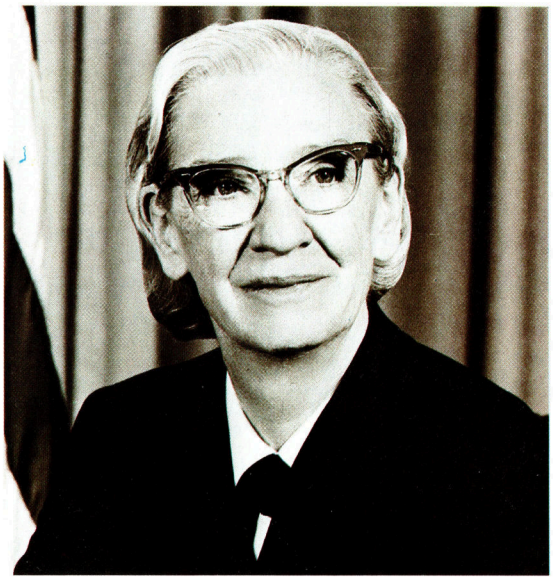
Weil die Anforderungen bei jedem Auftrag und bei jedem Stoff anders sind, ist dies eine sinnvolle Anwendung der mit der Erfahrung des Bedieners kombinierten Computeroptimierung. Noch weitergehende Methoden finden in Werken Anwendung, die wiederholt identische Objekte aus dünnem Material schneiden. Da der Vorgang des Schneidens oder Stanzens ein Teil der Fließbandproduktion ist, wird ein und derselbe Arbeitsgang tausendfach wiederholt. In diesem Fall werden die Kosten des Optimierens durch die Einsparungen an Material bei weitem aufgewogen.

Die CPA ist eine Methode, um die wichtigste Reihenfolge von Tätigkeiten in einem Herstellungs- oder Bauvorhaben herauszufinden. So werden die Arbeiten bestimmt, deren nicht termingerechter Abschluß am ehesten dazu führt, daß alles andere aufgehalten wird. Die Methode ist stark zeitorientiert. Der Wert eines Vorgangs im CPA-Diagramm wird durch die Zeit, die zu seiner Durchführung benötigt wird, bestimmt. Üblicherweise wird die Methode im Planungsstadium von Bauprojekten angewendet, damit die Planer Menschen und Materialien für die verschiedenen Abschnitte des Projektes in der richtigen Reihenfolge einsetzen können – Installationsarbeiten bevor der Fußboden verlegt wird, Verputzen vor dem Streichen. Auch hierfür ist Software für viele Microcomputertypen verfügbar.

Die Art und die Streckenführung von Straßen, ob in der Stadt oder auf dem Land, ist weitgehend von Optimierungstechniken abhängig. Der Architekt wird sich vor allem um Steigungen und Kurvenradien kümmern. Der Landwirt, dessen Boden benötigt wird, gibt völlig andere Punkte zu bedenken. Wenn eine neue Straße geplant wird, muß eine große Datenmenge gesammelt werden, um ein umfassendes Modell der Situation zu erstellen. Anhand des Modells wird die Streckenführung optimiert.

Grace Hopper

Bei der Entwicklung höherer Programmiersprachen erwarb sich Grace Hopper große Verdienste. Sie wirkte, allerdings unfreiwillig, auch bei der Computer-Terminologie mit.



Die Computer-Wissenschaft galt früher als ausschließlich männliche Domäne. In der Entwicklung und Anwendung ziehen die Frauen jedoch zunehmend gleich auf. Grace Hopper hat als eine der ersten weiblichen Computer-Pioniere Großes auf dem Gebiet der Software geleistet – sie erfand den ersten Compiler und wirkte bei der Einführung der Programmiersprache COBOL mit.

Nach ihrer Doktorandenzeit in Yale kehrte Grace Hopper als Dozentin für Mathematik an ihren ersten Studienort Vassar zurück. Mit 39 Jahren wurde sie an ein Computer-Projekt des Verteidigungsministeriums berufen. 1945 ging Frau Hopper als Assistentin Howard Aikens nach Harvard, um ihn bei der Konstruktion eines Computers zu unterstützen. Aiken hatte schon 1937 für IBM einen mechanischen Rechner entworfen. Dieser arbeitete so gut, daß IBM daraus ein verbessertes Modell, den „Harvard Mark II" entwickeln ließ, der schon mit elektromechanischen Relais arbeitete.

Die frühen Computer wurden noch ausschließlich durch Änderungen in der Verdrahtung für unterschiedliche Aufgaben programmiert. Grace Hopper war im Sommer des Jahres 1945 fast ununterbrochen mit dem Verdrahten des Computers beschäftigt: Für den Krieg wurden dringend Möglichkeiten ballistischer Berechnungen gebraucht, und Aiken kam regelmäßig in die Werkstatt, um neue Ergebnisse zu erfragen. Nach einem Totalausfall des Rechners fand man nach langer Suche einen Nachtfalter, der sich in einem der Relais verklemmt hatte. Auf Aikens Drängen nach neuen Zahlen bekam er von Grace Hopper die lakonische Antwort: „Wir müssen erst die Käfer (bugs) aus der Maschine entfernen!" Bis heute werden Computer-Fehler „bugs", also „Käfer" genannt. Den Nachtfalter, der seinen Platz in der Computer-Sprache mit dem Leben bezahlt hat, kann man heute noch sehen: Er ist in die Dokumentation des „Harvard Mark II" eingeklebt worden – neben der Eintragung vom 9. September 1945, 15.45 Uhr. Das Buch ist im Marinemuseum von Virginia ausgestellt.

Im selben Jahr bauten John Mauchly und Presper Eckert den Computer ENIAC, den sie nach dem Krieg für kommerzielle Zwecke weiterentwickelten. Grace Hopper entwarf große Teile der Software für diesen UNIVAC genannten Rechner. Bei den Programmierarbeiten kam sie darauf, daß es nicht unbedingt notwendig sei, bestimmte Unterprogramme immer wieder neu einzugeben. Grace Hopper schuf zusammen mit der Programmiersprache den ersten „Compiler", der das Programm in die Maschinensprache übersetzen konnte.

COBOL erfunden

Grace Hoppers Neuerung erhielt den Namen „A-O". Als der Compiler erstmals vorgestellt wurde, war die Fachwelt überrascht, denn bisher hielt man nur Rechenoperationen oder den Umgang mit Symbolen für möglich. Nun aber sprang der Rechner auf einen einfachen Befehl hin direkt zu einem Unterprogramm im Speicher, und die Befehle ähnelten schon einem normalen englischen Satz.

Im Mai 1959 wurde Captain Grace Hopper vom Pentagon dazu eingeladen, sich an einer Arbeitsgruppe zu beteiligen, die eine einheitliche Computersprache für kommerzielle Anwendungen entwickeln sollte. In weniger als einem Jahr entstand so die „COmmon Business Oriented Language" (COBOL). Grace Hopper übernahm jeweils das Beste aus den schon bekannten Computersprachen und optimierte es so weit, daß COBOL sich in der Geschäftswelt schnell durchsetzen konnte. Ein Beweis für die hohe Qualität der Sprache ist, daß sie bis heute eine der meistgebrauchten Programmiersprachen insbesondere für die kommerzielle Anwendung geblieben ist.

COBOL

COBOL war eine der ersten Programmiersprachen, die besonders für mathematische Laien gedacht waren. Schwierige Codes werden bei COBOL durch das Abrufen allgemeinerer Abläufe mit sprachähnlichen Befehlen ersetzt. Ein COBOL-Programm besteht aus vier Einheiten: Der Programmname, der Name des Autors und Zusatzinformationen bilden die Abteilung „Identifikation". Obwohl COBOL-Programme auf unterschiedlichen Rechnern laufen sollen, werden im Bereich „Environment" (Umgebung) spezielle Daten über den Rechner abgelegt, auf dem das Programm geschrieben wurde. Da oft dieselben Daten in verschiedenen Programmteilen genutzt werden, wird in COBOL außerdem eine spezielle DATA-Abteilung eingerichtet. Im PROCEDURE-Bereich stehen endlich die Operationen, die auf die Daten angewendet werden sollen – also der Programm-Ablauf.

Teile und herrsche!

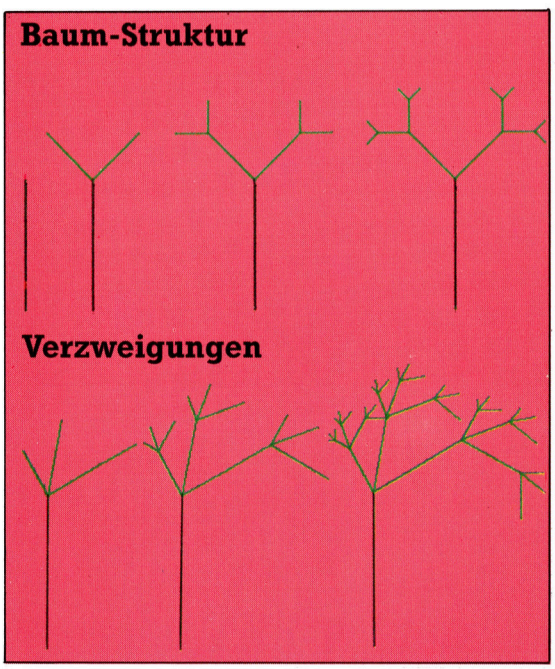
In diesem Teil unseres LOGO-Kurses können Sie lernen, wie mathematische Berechnungen unter Verwendung der Recursion in geometrische Formen umgesetzt werden.

Das erste Programm zeichnet verschiedene Baumformen auf den Bildschirm. Die einfachste Form ist dabei, eine gerade Linie als Stamm zu entwerfen, die in einer linken und rechten Verzweigung endet. Um die Zweige auch als solche erkennen zu können, sollten diese natürlich kürzer sein als der Stamm. Anhand dieses Beispiels, an dem sich die Funktionsweise der Recursion sehr gut darstellen läßt, können Sie nun die unterschiedlichsten Baumstrukturen entwerfen.

Die Prozedur für das Zeichnen eines „Binär"-Baumes erfordert zwei Eingaben: Eine für die Länge des Stammes, die andere für die Berechnung der Zweige.

```
TO VERZW :LAENGE :STUFE
  IF :STUFE = 0 THEN STOP
  FD :LAENGE
  LT 45
  VERZW ( :LAENGE / 2) ( :STUFE —1 )
  RT 90
  VERZW ( :LAENGE / 2) ( :STUFE —1 )
  LT 45
  BK :LAENGE
END
```

Zugegeben, durch diese Prozedur wird ein unrealistischer Baum erzeugt. Um ihn interessanter zu machen, kann das Programm modifiziert werden. Die nachstehende Version zeichnet auf jeder Ebene drei Zweige unterschiedlicher Länge.




```
TO ZWEIGE :LAENGE :STUFE
  IF :STUFE = 0 THEN STOP
  FD :LAENGE
  LT 30
  ZWEIGE ( :LAENGE / 3) ( :STUFE — 1)
  RT 40
  ZWEIGE (:LAENGE / 2) ( : STUFE — 1)
  RT 50
  ZWEIGE (:LAENGE / 1.5) ( :STUFE — 1)
  LT 60
  BK :LAENGE
END
```

Mit weiteren Modifikationen lassen sich andere Bäume schaffen.

Veränderbare Polygone

Die folgende Prozedur zeichnet ein Quadrat, viertelt es, viertelt jedes weitere Teil usw.

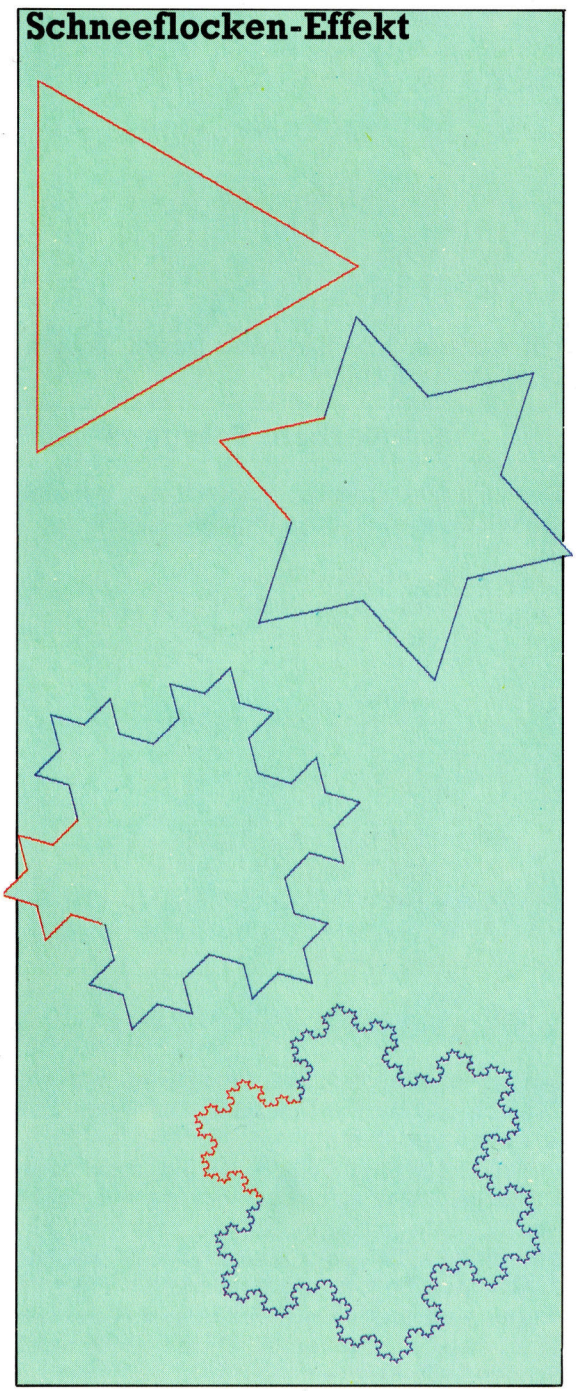
```
TO GEO :LAENGE :STUFE
  IF :STUFE = 0 THEN REPEAT 4
    (FD :LAENGE RT 90) STOP
  GEO ( :LAENGE / 2) ( :STUFE — 1)
  FD ( :LAENGE / 2)
  GEO ( :LAENGE / 2) ( :STUFE — 1)
  RT 90
  FD ( :LAENGE / 2)
  LT 90
  GEO ( :LAENGE / 2) ( :STUFE — 1)
  BK ( :LAENGE / 2)
  GEO ( :LAENGE / 2) ( :STUFE — 1)
  LT 90
  FD ( :LAENGE / 2)
  RT 90
END
```

Mit einem ähnlichen Programm kann man ein Dreieck in kleinere Dreiecke unterteilen und wie oben verfahren.

Nach Zeichnen eines gleichschenkligen Dreiecks drittelt man die Schenkel und zeichnet auf dem mittleren Teil ein neues gleichschenkliges Dreieck. Entfernen Sie die gemeinsamen Seiten und wiederholen Sie den Vorgang an jeder Seite der neuen Form. Die sich daraus ergebende Form nennt man Schneeflockenkurve.

```
TO SCHNEE :GROESSE :STUFE
  REPEAT 3 [SEITE :GROESSE :STUFE RT 120]
END
TO SEITE :GROESSE :STUFE
  IF :STUFE = 0 THEN FD :GROESSE STOP
  SEITE ( :GROESSE / 3) ( STUFE — 1)
  LT 60
  SEITE ( :GROESSE / 3) ( :STUFE —1)
  RT 120
  SEITE ( :GROESSE / 3) ( :STUFE — 1)
  LT 60
  SEITE ( :GROESSE / 3) ( :STUFE — 1)
END
```

Beachten Sie, daß die Variable SEITE in einer separaten Prozedur definiert wurde, um die Turtle zum Zeichnen der nächsten Seite in der richtigen Position zu haben.



Wenn dieser Teilungsprozeß unendlich fortgesetzt wird, ist das Ergebnis eine Kurve von unendlicher Länge, die doch einen endlichen (festgelegten) Raum umgibt. Eine ähnliche Kurve kann erzeugt werden, wenn man mit einem Quadrat beginnt, seine Seiten drittelt und auf den mittleren Teilen jeweils weitere Quadrate konstruiert.

Die Reihe der hier gezeigten Kurven wurde von dem Mathematiker Sierpinski entwickelt. Wird der Prozeß begrenzt fortgesetzt, ist das Ergebnis eine Kurve (eine eindimensionale Linie), die jeden Punkt des umgebenden Quadrates (eine zweidimensionale Form) schneidet. Es gibt viele andere raumfüllende Kurven, die dieses eigenartige Verhalten darstellen.

Die Prozedur für das Zeichnen dieser Kurve ist sehr komplex. Die Kurve auf Stufe 1 wird aus vier Seiten (dargestellt in Blau) konstruiert, die durch vier Diagonalen (in Rot) ergänzt werden. Die Hauptprozedur SIERP teilt den Zeichenvorgang in vier Abschnitte und ruft EINE.SEITE entsprechend einzeln auf.

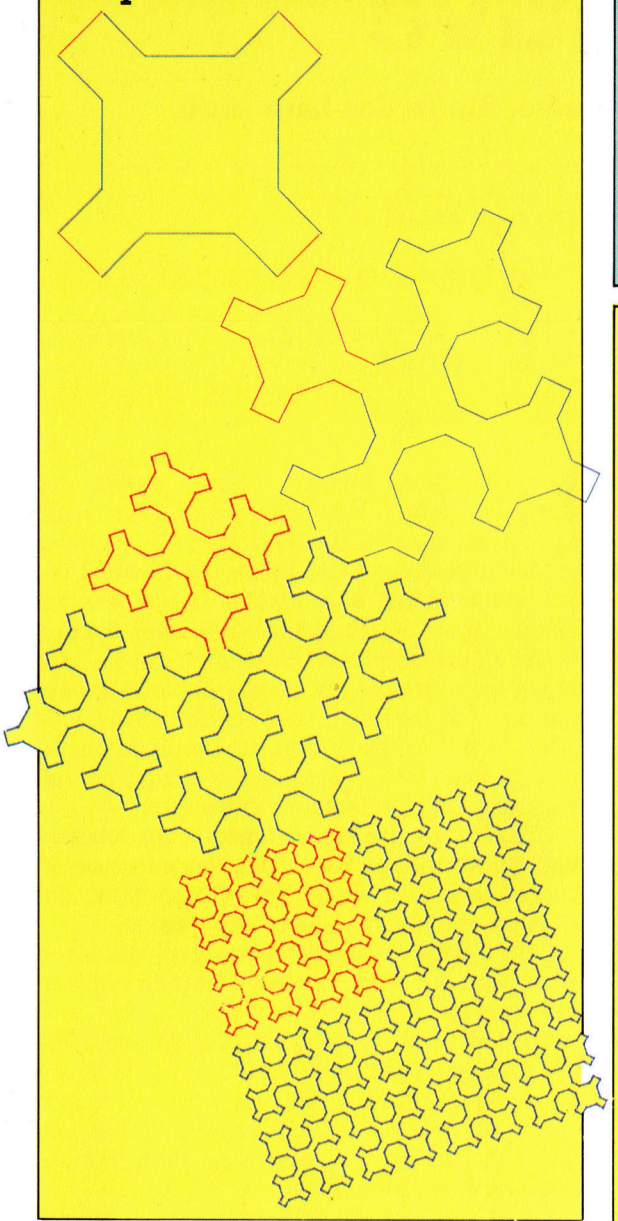
Betrachten wir einmal nur eine Seite. Sie besteht aus drei Linien – einer diagonalen, einer horizontalen oder vertikalen Linie sowie einer weiteren Diagonalen. In der zweiten Schrittfolge wird jede Diagonale durch eine andere, kleinere Linienschar ersetzt. Die Horizontale oder Vertikale wird mit zwei ähnlichen, um jeweils eine Linie ergänzten Dreiergruppen ausgetauscht. Diese Schrittfolge wiederholt sich für jede Ebene.

Nachstehend die Prozeduren zum Zeichnen der Kurven, wobei zu beachten ist, wie der Befehl MAKE zum Initialisieren der Variablen DIAG verwendet wird:

```
TO SIERP :SEITE :STUFE
  MAKE "DIAG :SEITE / SQRT (2)
  REPEAT 4 [EINE.SEITE :STUFE RT 45 FD
    :DIAG RT 45]
END

TO EINE.SEITE :STUFE
  IF :STUFE = 0 STOP
  EINE.SEITE (:STUFE — 1)
  RT 45
  FD :DIAG
  RT 45
  EINE.SEITE ( :STUFE — 1)
  LT 90
  FD :SEITE
  LT 90
  FD :SEITE
  LT 90
  EINE.SEITE ( :STUFE — 1)
  RT 45
  FD :DIAG
  RT 45
  EINE.SEITE ( :STUFE — 1)
END
```

Sierpinski-Kurve



LOGO-Dialekte

Einige LOGO-Versionen verwenden den Befehl SETPOS, um die Position der Turtle festzulegen. Dafür müssen zwei Eingaben (Koordinaten) in Verbindung mit einer Liste angegeben werden. Zum Beispiel:

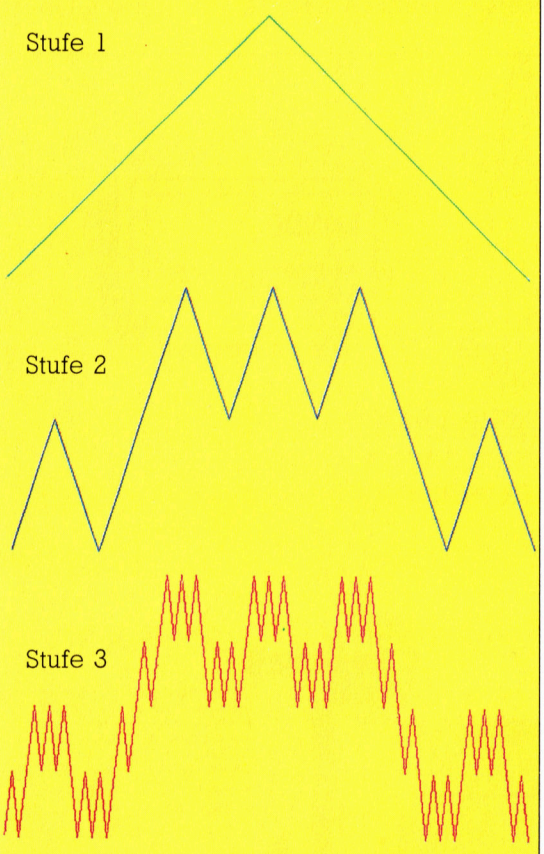
```
SETPOS LIST 45 67
```

Abweichungen bestehen auch bei der Bedingungsabfrage, wie zum Beispiel:

```
IF STUFE = 0 [STOP]
```

Übungen

Die Abbildung zeigt eine Reihe von unterschiedlichen Formen, mit denen Kurven ohne Gradienten definiert werden. Die erste Kurve besteht aus einer aufwärts und einer abwärts führenden Linie. Um die nächste Ebene zu zeichnen, werden die aufsteigende wie die absteigende durch unterbrochene fünfteilige Linien ersetzt. Beachten Sie dabei die unterschiedliche Länge der einzelnen Geraden.



Entwickeln Sie Prozeduren, die diese Kurvenfolgen zeichnen. Verwenden Sie dabei SETXY statt FD und RT. Die Hauptprozedur sollte die Aufgabe in zwei Abschnitte unterteilen. Ferner sind zwei separate Prozeduren für die Berechnung dieser Abschnitte erforderlich. Setzen Sie dabei wieder die Recursion ein.

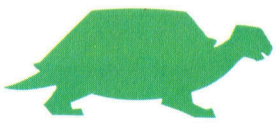
Lösungen

So könnte Ihr Turm-Programm aussehen:

```
TO TOWER :SIZE
   IF :SIZE < 5 THEN STOP
   SQUARE :SIZE
   MOVE :SIZE
   TOWER ( :SIZE / 2)
END

TO SQUARE :SIZE
   REPEAT 4 [FD :SIZE RT 90]
END

TO MOVE :SIZE
   FD :SIZE
   RT 90
   FD ( :SIZE / 4 )
   LT 90
END
```



Eigeninitiative

Gibt es wirklich intelligente Programme, die in der Lage sind, eigene Routinen zu entwickeln?

Der Computer ist doch so schlau. Warum braucht er überhaupt noch den Menschen, der ihn programmiert?" Diese Frage, vom Laien aufgeworfen, löst beim erfahrenen Computeranwender kaum mehr als ein Achselzucken aus. Sie ist aber gar nicht so töricht, wie es den Anschein hat. Tatsächlich wird die Entwicklung von Programmen, die selbst Programme generieren und Codierfehler in „handgeschriebenen" Programmen korrigieren können, mit viel Aufwand vorangetrieben.

Benutzer von Heimcomputern kennen die Fehleranzeige SYNTAX ERROR? wohl zur Genüge, ebenso wie den Ärger darüber, daß sie sowenige Informationen enthält. Der Compiler eines Hauptrechners liefert da viel mehr Hinweise über die Art des Fehlers. Seine Meldung könnte so aussehen:

```
1090 LET A=(C*2+F$)*((FG-C)*TH+1))
                  ↑              ↑
FEHLER: 1) UNPASSEND — STRINGVARIA-
          BLE F$ NICHT ZULÄSSIG
        2) LETZTE SCHLUSSKLAMMER
          NICHT ERWARTET
```

Es gibt keinen triftigen Grund, warum der Compiler eines Heimcomputers dies nicht auch können sollte, da die Kosten für das dafür notwendige Extra-ROM kaum ins Gewicht fallen. Doch selbst so einfache Dinge wie die Überprüfung auf Positionsfehler oder Syntaxfehler beim Eingeben finden sich nur bei wenigen Heimcomputern. Es besteht jedoch häufig die Möglichkeit, zusätzliche ROM-Chips oder Cartridges zu kaufen, die den Bereich der verfügbaren BASIC-Befehle für Programmentwicklung und Fehlersuche erweitern:

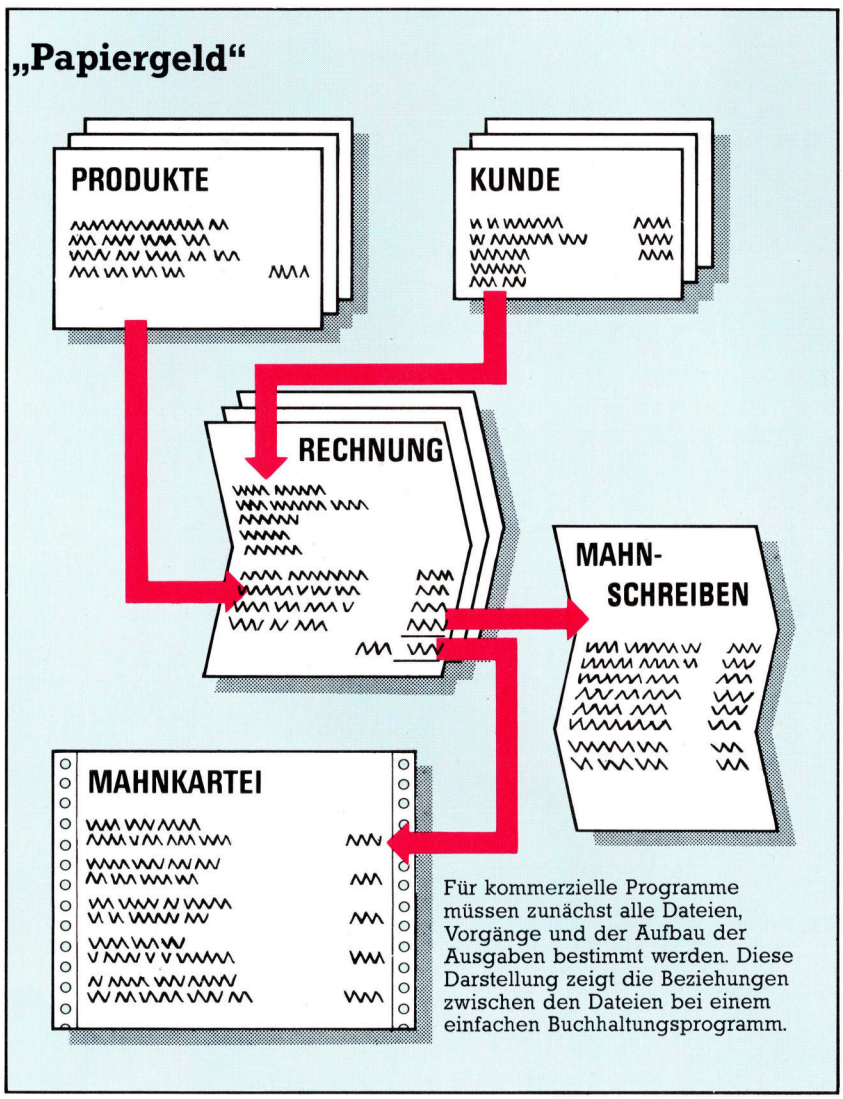
HELP bringt die Programmzeile zur Anzeige und weist auf die Stelle der Programmunterbrechung hin. In vielen Fällen wird damit die Ursache des Syntaxfehlers aufgedeckt.

DUMP listet diejenigen Variablen und deren Inhalt auf, die vom Programm gerade verwendet werden.

TRACE zeigt die Nummer derjenigen Zeile oder Zeilen an, die vom Programm gerade abgearbeitet werden. Damit kann der Bediener den Programmlauf verfolgen und sicherstellen, daß Subroutinen in der gewünschten Reihenfolge ausgeführt werden.

Es ist in der Regel schwierig, ein Programm zu schreiben, das Codierungsfehler berichtigt. Bei einigen Fehlerarten ist dies jedoch ziemlich einfach. So ist zum Beispiel bekannt, daß Programmzeilen, von wenigen Ausnahmen abgesehen, mit einem BASIC-Schlüsselwort beginnen müssen. Beginnt nun eine Zeile mit PRUNT oder PRONT, dürfte es keine Schwierigkeiten bereiten, daraus PRINT zu machen. Eine einfache Möglichkeit wäre, die Eingaben anhand einer Interpreter-Liste, die die am häufigsten auftretenden Eingabefehler und die entsprechende korrekte Schreibweise enthält, zu vergleichen. Aus Sicherheitsgründen sollte jede Änderung überprüft werden.

Von diesen einfachen Verfahren abgesehen, ist die automatische Fehlerbeseitigung sehr viel schwieriger. Im obigen Beispiel wird F$ als Fehler ausgewiesen. Dies kann ein Tippfehler bei der Eingabe von F oder FS oder F4 oder aber auch für etwas ganz anderes sein.



Für kommerzielle Programme müssen zunächst alle Dateien, Vorgänge und der Aufbau der Ausgaben bestimmt werden. Diese Darstellung zeigt die Beziehungen zwischen den Dateien bei einem einfachen Buchhaltungsprogramm.

Ein erfahrener Programmierer kann die Ursache des Fehlers schnell aufspüren, weil er nach den beiden Kriterien „inhaltlicher Zusammenhang" und „Erfahrung" vorgeht.

Diese Technik der Fehlersuche wird jedoch mehr zum Korrigieren von Texten als zum Beseitigen von Programmierfehlern angewendet. Es gibt Buchstabier-Prüfroutinen, die den Text überprüfen und jedes Wort herausziehen, das nicht im gespeicherten „Wörterbuch" mit seinen möglicherweise 50 000 Wörtern enthalten ist. Bei den meisten dieser Routinen ist es möglich, neue Wörter, z. B. Firmen- oder Eigennamen, in das Wörterbuch aufzunehmen. Höher entwickelte Prüfroutinen können, wenn der Orthographiefehler nicht zu kraß ist, auch die korrekte Schreibweise anzeigen. Experimentelle Textsysteme, die das gleiche Verfahren auf Grammatik und Stil anwenden, weisen sogar auf falsche Zeichensetzung, Wortwiederholungen innerhalb eines Absatzes und nicht anwendbare Adjektive hin.

In die Entwicklung von Systemen, die neue Programme erzeugen, wurde bisher jedoch viel mehr Mühe investiert. 1981 tauchte ein Software-Erzeugnis auf, das einen der härtesten Kämpfe auslöste, der je innerhalb der Microcomputer-Industrie ausgetragen wurde. „Die Letzte" (The Last One), so war der Name dieser Software, sollte den letzten notwendigen Software-Kauf darstellen, weil sie die Fähigkeit für sich beanspruchte, jedes gewünschte Programm schreiben zu können. Dieser Anspruch war natürlich nicht gerechtfertigt, obwohl diese Software sich bei der Entwicklung von kommerziellen Programmarten als sehr nützlich erwies. Inzwischen gibt es mehrere solcher Erzeugnisse für geschäftlich genutzte Microcomputer. Sie werden als „Programmgeneratoren" bezeichnet.

Programmgeneratoren

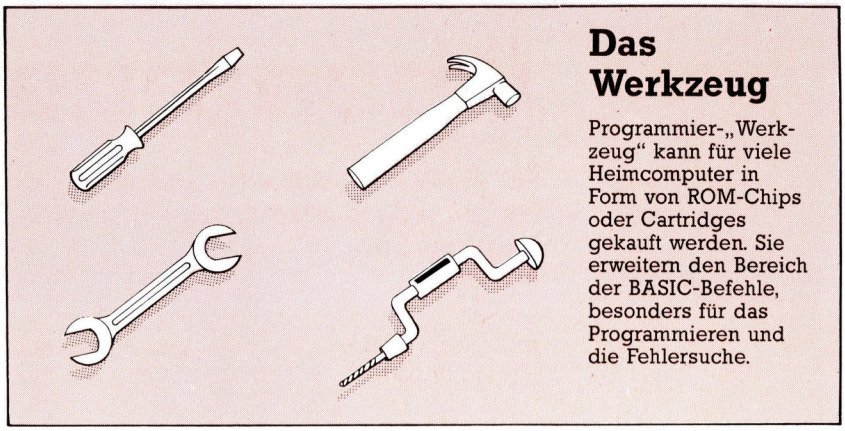
Das Konzept, nach dem diese Programmgeneratoren arbeiten, wird an folgendem Beispiel deutlich:

```
10 PRINT "WAS SOLL DAS PROGRAMM AUF
   DEM BILDSCHIRM ANZEIGEN?"
20 GEBEN SIE A$ EIN
30 PRINT "DAS PROGRAMM IST:"
40 PRINT "10 PRINT";CHR$(34);A$;CHR$(34)
```

Antworten Sie mit HALLO, sollte das Programm folgendes ausgeben:

```
DAS PROGRAMM IST
10 PRINT "HALLO"
```

Wird diese Technik auch auf die Eingabe-, Kalkulations- und Ausgabephasen eines bestimmten Anwendungsfalles angewendet, so können Sie selbst einen sehr einfachen Programmgenerator schreiben. Sind alle Fragen, die das Programm stellt, einfach ausgedrückt, so sollte es auch ohne Computer-Erfahrungen möglich sein, mit Hilfe dieses selbst geschriebenen Programmgenerators ein einfaches Programm zu entwickeln.



Das Werkzeug

Programmier-„Werkzeug" kann für viele Heimcomputer in Form von ROM-Chips oder Cartridges gekauft werden. Sie erweitern den Bereich der BASIC-Befehle, besonders für das Programmieren und die Fehlersuche.

Kommerzielle Programmgeneratoren arbeiten nach der gleichen Methode. Die meisten Geschäftsvorgänge bestehen aus einer Kombination der folgenden fünf Prozesse: Dateneingabe, Datenausgabe auf Bildschirm oder über Drucker, Abspeichern in einer Datei, Suchen von Daten, Kalkulation. Für jeden dieser Prozesse hat der Generator standardmäßige und auch sehr flexible Subroutinen. Mit der Aufforderung an den Bediener, Datenstruktur, Rechenoperationen und Ausgabestruktur genau zu spezifizieren, beschafft er sich die notwendigen Informationen, um die Werte bestimmter Variablen in den Subroutinen zu verändern und sie zur Programmerzeugung zu verketten.

Programmgeneratoren werden zwar immer anspruchsvoller, können aber, zumindest in der nächsten Zukunft, den Mensch als Programmierer nicht ersetzen, weil sie folgenden Einschränkungen unterliegen: Erstens ist die beschreibende Technik zwar sehr gut für geschäftliche Abwicklungsvorgänge wie Buchhaltung, Lagerführung usw. geeignet, kann jedoch kaum auf Text- oder Spielprogramme angewendet werden. Zweitens ist der Programmgenerator auf Standardroutinen angewiesen und kann deshalb, was Schnelligkeit und Speicherbedarf angeht, kein annähernd so leistungsfähiges Programm erstellen wie ein Programmierer. Drittens sind vom Generator erzeugte Programme meist nicht so anwenderfreundlich wie die „manuellen" Systeme.

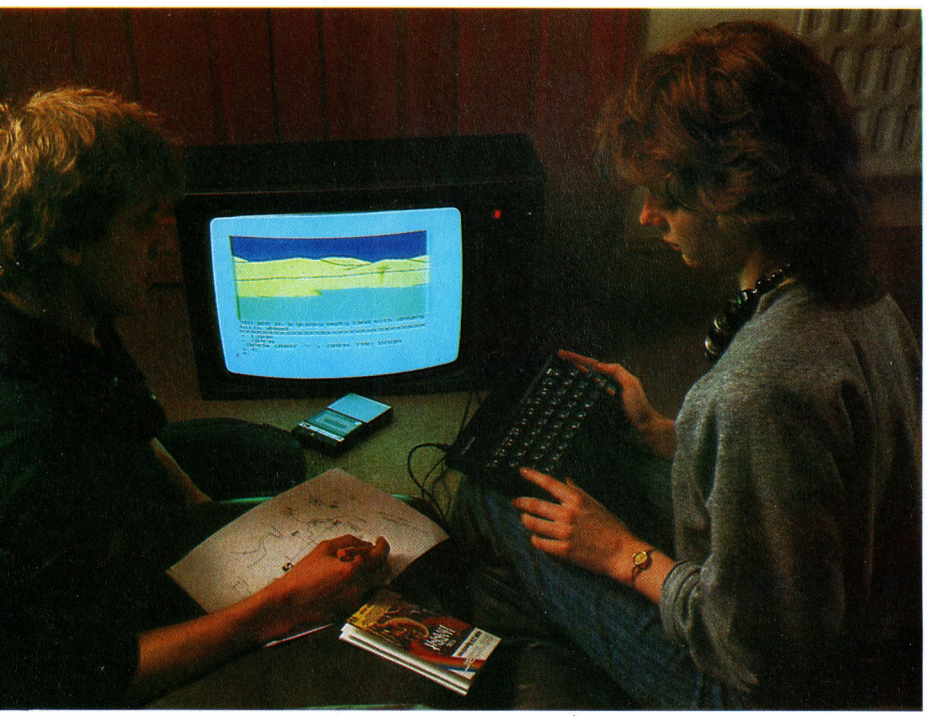
Letztlich können heutige Programmgeneratoren nur die Schlußphase übernehmen, das Schreiben des Code. Es obliegt immer noch dem Anwender, sich die genaue Form der Daten, der Ein- und Ausgabe auszudenken. Aber gerade diese frühen Programmierphasen sind die schwierigeren, da sie besondere Fertigkeiten voraussetzen. In den meisten großen Firmen sind es die Systemanalytiker, die sich die Programme „ausdenken" und das Umsetzen den Programmierern überlassen.

Abenteuerspielplatz

Bei Abenteuerspielen auf dem Computer schlüpft der Spieler in die Rolle einer bestimmten Figur und muß gefährliche Situationen durchstehen.

Beim Wort „Abenteuer" denken die meisten Menschen an Bücher, Filme, das Fernsehprogramm oder auch an persönliche Erlebnisse. Das englische Wort „Adventure" hat für viele Computer-Besitzer noch eine Zusatz-Bedeutung: Es steht für eine ganz spezielle Kategorie von Computerspielen.

Der Unterschied zwischen Computer-Abenteuern und entsprechender Literatur liegt darin, daß Sie im Buch zwar über die vielen Gefahren und unheimlichen Abenteuer lesen, selbst aber nur Zuschauer sind. Anders beim Computer-Abenteuer: Aus dem passiven Zuschauer wird ein Teilnehmer, er selbst ist die Hauptperson, die alle Gefahren bestehen muß.



Abenteuerspiele gibt es zwar schon seit Jahren – aber erst durch den Computer sind noch mehr Spieler auf den Adventure-Geschmack gekommen. Die Spiele machen den konventionellen Kriegs- und Weltraumkämpfen zunehmend Konkurrenz. Obwohl man sie fast ausschließlich allein spielen kann, können sie doch jedes Familienmitglied unterhalten. Der Spieler schlüpft in die Heldenrolle und sucht das Glück – sei es Gold oder die Prinzessin . . .

Im Computer-Abenteuer bestimmt allein Ihre Entscheidung oder Tätigkeit die Weiterentwicklung der Handlung. Die Variationsmöglichkeiten sind fast unendlich, und niemand weiß vorher, ob es ein Happy-End geben wird. Der Spieler übernimmt einen aktiven Part – trotzdem sitzt er sicher zu Hause.

Abenteuerspiele können an jedem denkbaren Platz stattfinden, in einer fremden, unterirdischen Welt, einer Geisterstadt, auf anderen Planeten und in sagenumwobenen Ländern, in der Vergangenheit oder in der Zukunft.

Im allgemeinen haben diese Abenteuer ein feststehendes Ziel – und die entsprechenden Hindernisse: Das kann die Flucht von einem Nachbarplaneten, die Vernichtung eines bösen Zauberers, die Rettung einer Prinzessin, Schatzsuche oder auch die Aufklärung eines Verbrechens sein.

Das Lösen von Rätseln bildet neben den erwähnten Schwierigkeiten die eigentlich interessante Aufgabe und ist aus keinem Abenteuer wegzudenken. Auf Rätsel und verwirrende Aufgaben trifft man ständig – meist kann das Spiel erst weitergehen, wenn sie gelöst sind – etwa angesichts einer gefährlich verfallenen Brücke. Oft sind die Rätsel aber auch nur Ablenkungsmanöver: Sie überwinden eine Schlucht, um den Fremden auf der anderen Seite zu treffen, und finden – einen riesigen Spiegel. Nicht immer ist die Lösung für das erfolgreiche Ende des Abenteuers nötig, sondern hilft nur dabei – wie die Entdeckung eines geheimen Weges, der am gefährlichen Troll vorbeiführt. Wenn Sie sich ohne Essen und Trinken in einem Schacht verirrt haben, kann das Lösen eines Rätsels aber auch über Leben oder Tod entscheiden.

Alle Rätsel lassen sich mit dem gesunden Menschenverstand lösen und erfordern kein besonderes Wissen oder spezielle Übung. Der Abenteurer muß allerdings aufmerksam sein, denn der Text gibt – oft versteckte – Hinweise auf die Lösung.

Während des Spielablaufes werden Sie öfter auf Gegenstände oder Nachrichten stoßen, die ohne Zusammenhang erscheinen. Lassen Sie sich davon nicht täuschen, fast alles im Spiel dient einem bestimmten Zweck – der natürlich auch darin bestehen kann, Sie vom rechten Weg abzubringen.

Viele Abenteuerspiele enthalten ein kleines Labyrinth, in dem jeder Raum gleich aussieht. Sie finden nur dann den richtigen Weg, wenn Sie in jedem Raum etwas zurücklassen und ihn damit kennzeichnen. Die Methode ist jedoch schon so bekannt, daß die Spiel-Autoren zusätzliche Schwierigkeiten einbauen, etwa jemanden hinter Ihnen gehen lassen, der die Gegenstände aufhebt und neu verteilt . . .

Abenteuerspiele sind nicht nach wenigen Minuten zu Ende: Manchmal dauert es Tage, ja sogar Wochen oder Monate, bis alle Geheimnisse entschlüsselt sind. Die meisten Spiele ermöglichen die Speicherung des aktuellen

Standes auf Band oder Diskette, so daß Sie es unterbrechen und später wieder beginnen können.

Wie geht nun die Kommunikation mit dem Programm vor sich? Sie werden entweder direkt angesprochen oder durch eine Figur dargestellt, die auf Kommandos reagiert. Der Computer übernimmt die Doppelfunktion als Ausführender Ihrer Wünsche sowie als Erzähler. Die Befehle des Spielers werden über die Tastatur eingegeben, der Rechner antwortet via Bildschirm.

Manche Abenteuerspiele zeigen ausschließlich Text, manche nur Grafik, einige auch eine Mischung aus beidem auf dem Monitor. Geräusche sind meist auf die reinen Grafik-Spiele beschränkt. Reine Textspiele können als Bücher ohne Bilder gedacht werden, in denen Plätze, Gegenstände und Geschehnisse mit Worten dargestellt werden. In Spielen mit Text und Grafik dienen die meist statischen Szenen zur Unterstützung des Textes. Meist zeigen sie vereinfachte Landkarten oder das Innere von Gebäuden. Personen und Gegenstände werden durch Symbole und Figuren dargestellt. Dem Spieler steht eine unterschiedliche Anzahl von Befehlen zur Verfügung, mit der er seine Figur steuern und kontrollieren kann.

Standardisierte Befehle

Textdarstellung in Abenteuerspielen bezieht sich oft auf drei Punkte: Wo man ist, was man sieht, und wohin man gehen kann. Es könnte dort etwa heißen: „Du bist in einem dunklen Wald. Dichtes Blattwerk verdeckt den Himmel. Ein ausgetretener Pfad führt von West nach Ost. Direkt vor dir, im Norden, ist ein tiefer Abgrund, auf dessen Sohle du ein Schwert erblickst, um das sich eine grüne Schlange windet." Also eine Umgebungsbeschreibung, in der auch die Richtungen angegeben sind, in die man weitergehen kann.

Die Befehle (meist in Englisch) setzen sich aus einem Verb und einem Substantiv zusammen. Standardworte sind GET (NIMM), PULL (ZIEH), THROW (WIRF), KILL (TÖTE), EAT (ISS). GO NORTH (GEH NACH NORDEN) würde also den weiteren Weg angeben. Oft genügt dafür jedoch die Abkürzung – für GO WEST nur W.

EXAMINE (UNTERSUCHEN) ist ein besonders wichtiges Wort – es kann für zusätzliche Informationen sorgen. EXAMINE SNAKE (SCHLANGE UNTERSUCHEN) kann zur Mitteilung führen, daß die Schlange harmlos ist, aber auch dazu, daß die Schlange die „Untersuchung" spürt und den Spieler beißt.

INVENTORY (INVENTAR) sagt dem Spieler, welche Gegenstände er bei sich hat. Dabei können einige auch verpackt sein – Wasser in einer Flasche, ein Beil in einem Sack – andere gehören zur Kleidung – etwa ein Mantel.

Schatzsuche

Die Handlung von „The Hobbit" ist dem gleichnamigen Roman von J. R. R. Tolkiens entlehnt. Das Buch enthält die für den Spielablauf erforderlichen Karten sowie wertvolle Tips. Als Bilbo reist der Abenteurer durch Mittelerde und trifft auf seiner Suche nach dem Drachen und seinem Schatz viele Personen aus dem Buch wieder.

Falsche oder unverständliche Befehlskombinationen werden vom Spiel mit „I DON'T UNDERSTAND YOU" – (ICH VERSTEHE DICH NICHT) quittiert.

Zu den meisten Abenteuerspielen gibt es ein Begleitbuch mit Tips für Spieler, die irgendwo festsitzen. Darin finden sich oft nützliche Tricks, wie man einer Falle doch noch entkommen kann.

Inzwischen sind fast für jeden Computer verschiedene Abenteuerspiele im Handel. Sicher finden auch Sie ein interessantes Spiel, das der Anfang eines dauernden Hobbys werden kann! Also – Viel Glück und Spaß am Abenteuer „vom Sessel aus".

Das Spiel „Deadline" variiert das Abenteuerthema. Als Detektiv muß der Spieler einen Mörder entlarven. Als Hilfe erhält er die vorhandenen Indizien und Informationen. Jede Handlung wird als Zeitverlust von der auf (scheinbare) 12 Stunden beschränkten Zeit des Detektivs abgezogen . . .

Variabler Chip

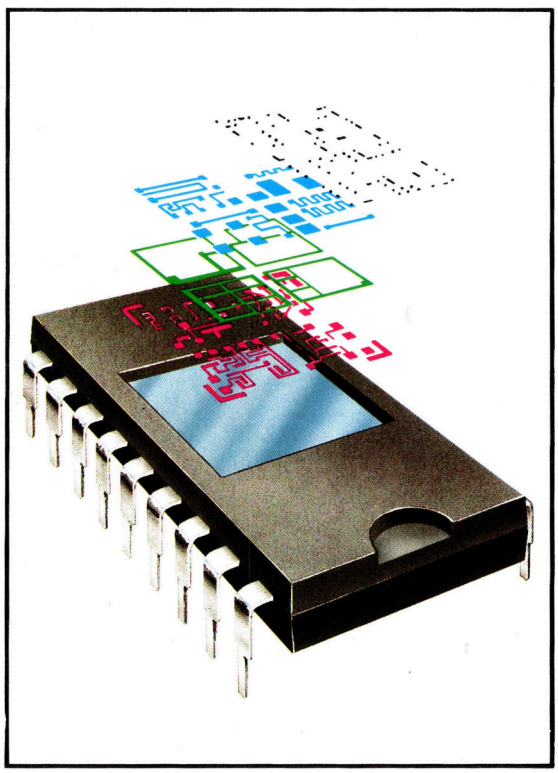
„Ungebundene" Logikschaltungen, kurz ULA genannt, steuern als Schaltzentrale zahlreiche Funktionen des Computers.

Die Microcomputer-Schwemme hat zahlreiche Fortschritte auf dem Gebiet der Elektronik hervorgebracht. Eine der wichtigsten Entwicklungen ist der ULA (Uncommitted Logic Array), ein Chip, der nicht an einen bestimmten Zweck gebunden ist. Diese Entwicklung hat inzwischen einen Stand erreicht, der es möglich macht, hochentwickelte Computer und andere Einrichtungen mit nicht mehr als vier Hauptbestandteilen zu bauen: CPU, RAM, ROM und ULA, der diese Komponenten miteinander verbindet.

Ein ULA besteht aus einer Vielzahl von Logikgattern, die ursprünglich nicht an einen bestimmten Zweck gebunden sind, sich aber so miteinander verbinden lassen, daß sie nahezu jede gewünschte Operation durchführen können. Der ULA ist eine Art ROM; sein Inhalt kann, wie beim ROM, nur vom Hersteller, nicht aber vom Anwender festgelegt werden.

Vor dem Programmieren ist ein ULA nichts weiter als eine Anhäufung einfachster elektronischer Schaltungen, die nicht miteinander verknüpft sind. Sie können darum auch keine Tätigkeiten ausüben. Der ULA besteht aus Halbleiterschichten mit einer Deckschicht aus leitendem Material, die die Verbindungen zwischen den Einzelschaltungen herstellt. Obwohl jede Einzelschaltung ganz einfach ist und vielleicht nur aus einigen Transistoren oder einem einzigen Widerstand bestehen mag, ergibt sich eine große Zahl möglicher Verbindungen, die dem ULA eine außergewöhnliche Flexibilität verleihen und selbst so komplexe Schaltkreise wie ein Flip-Flop entstehen lassen. Solche Schaltkreise, auch „Module" genannt, lassen sich meist aus weniger als einem halben Dutzend Einzelschaltungen bilden.

Ein ULA kann auf ein außergewöhnlich breittes Spektrum von Funktionen programmiert werden. Er kann als Tongenerator arbeiten, Verschlußzeit, Schärfe und Motor in einer Kamera steuern oder den größten Anteil der Arbeit in einem Digitalthermometer leisten. Dabei benötigt der ULA keine Außenverdrahtung – ausgenommen die Anschlüsse für Batterien, Schalter oder Sensoren.



Jeder Halbleiter-Chip besteht aus Schichten, die individuell so geätzt sind, daß Schaltelemente entstehen. Die oberste Schicht bestimmt die Verbindung zwischen den einzelnen Schaltelementen. Ein ULA besteht aus vielen Logikelementen, die so kombiniert werden können, daß komplexe Schaltungen entstehen.

Große Möglichkeiten

Bei der Konstruktion der Deckschicht, die die Einzelschaltungen verbindet, werden Rechner eingesetzt. Der Mini-Computer eines CAD-Systems, z. B. der DEC PCP11/23, zeichnet zunächst ein Diagramm der gewünschten Logik auf und entwickelt dann daraus das „Layout", die gedachte Anordnung der Einzelschaltungen. Der Konstrukteur führt dabei das System von einem Grafik-Terminal aus und kann sich das Ergebnis vom Plotter ausdrucken lassen.

Ist der Schaltplan fertig, wird er auf einen größeren Computer übertragen. Dieser prüft ihn auf Übereinstimmung mit der ursprünglichen Logik, forscht nach gravierenden Fehlern und stellt auf diese Weise fest, ob der Schaltplan akzeptabel ist. Als nächstes folgt die Simulation der Schaltung mit Hilfe eines vom Anwender zur Verfügung gestellten Testprogramms. Verläuft die Simulation zufriedenstellend, kann der Computer die Vorlagen für die zum Herstellen der Deckschicht erforderlichen Masken produzieren.

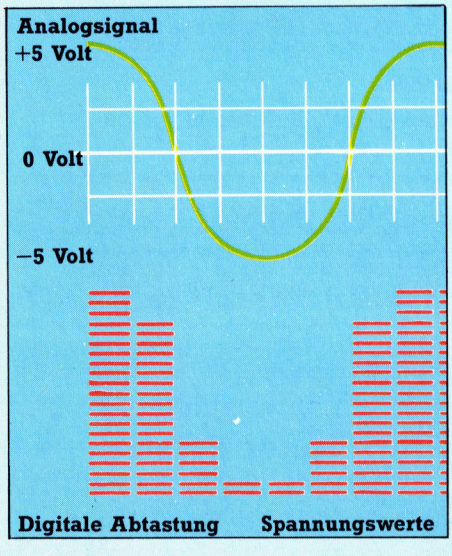
Wie weit gehen die Möglichkeiten eines ULA? Der Gedanke, eine Reihe einfacher Schaltungen auf Silizium festzulegen und es dem Anwender zu überlassen, seine Schaltverbindungen selbst herzustellen, ist verlokkend. Dies könnte die bevorzugte Methode zum Herstellen von Schaltkreisen werden. Beim derzeitigen Stand der Technik sind ULAs jedoch nur dann wirtschaftlich, wenn einige tausend identischer Schaltkreise in einem Gerät benötigt werden.

Fachwörter von A bis Z

A/D-Convertor =
A/D-Wandler
Ein Analog/Digital-Wandler oder ADC (für A/D-Convertor) ist eine Schaltung, die analoge elektrische Signale in Digitalwerte umsetzt. Umgekehrt arbeitet ein Digital/Analog-Wandler (DAC), wozu natürlich eine ganz andere Schaltung nötig ist.

Um zu verstehen, wofür man solche Wandler braucht, müssen Sie die Begriffe „Analog" und „Digital" sorgfältig gegeneinander abgrenzen. Ein digitales Signal hat zwei Eigenschaften: Erstens ist es „diskret", d. h., es kann nur die Werte einer vorgegebenen Stufenskala annehmen – in Ihrem Computer z. B. meist nur die Werte 0 Volt und 5 Volt, entsprechend der logischen Null bzw. der logischen Eins. Zweitens ist ein Digitalsignal fast immer verschlüsselt: Es wird durch eine Folge von diskreten Werten gebildet. Acht Bits stellen im Binärsystem bekanntlich die Zahlen 0 bis 255 dar.

Ein Analogsignal ist dagegen keine diskrete, sondern eine „kontinuierliche" Variable: Sie kann innerhalb vorgegebener Grenzen unendlich viele beliebige Zwischenwerte annehmen. Außerdem ist ein solches Signal immer einer Meßgröße „analog" (daher der Name); ein Thermoelement gibt z. B. eine Spannung ab, die der gemessenen Temperatur proportional ist, und ein Mikrofon liefert eine Wechselspannung.



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.

Adder =
Addierer
Ein Addierer ist eine Logikschaltung aus einfachen Verknüpfungselementen (UND-, ODER- und NICHT-Gattern), mit denen die Summe aus zwei Dualzahlen gebildet wird.

Die einfachste Version des Addierers ist der „Halbaddierer": Er hat zwei Ein-Bit-Eingänge (A und B) und zwei Ein-Bit-Ausgänge, nämlich Summe und Übertrag (carry). Ein Übertrag tritt zum Beispiel auf, wenn zugleich A und B den Binärwert 1 darstellen. In diesem Fall trägt die Summe den Wert 0; der Übertrag lautet 1.

Ein „Volladdierer" verfügt über einen zusätzlichen Eingang. Daher kann man aus Volladdierern eine Kette bilden, indem man den Übertrag eines jeden mit dem dritten Eingang (Carry-Eingang) seines linken Nachbarn verbindet. Mit acht Volladdierern lassen sich dann z. B. zwei Acht-Bit-Binärzahlen summieren (wobei das Ergebnis infolge des letzten Übertrags neunstellig werden kann).

Address =
Adresse
Speicheradressen brauchen Sie kaum, wenn Sie nur in BASIC programmieren, für den Maschinencode jedoch ist die Kenntnis der exakten Adresse Voraussetzung. Die Rechner-Zentraleinheit (CPU) kann mit einer endlichen Anzahl von Speicherplätzen (je ein Byte) kommunizieren, genannt „Adreßbereich" (beim 8 Bit-Rechner reicht er von 0 bis 65535).

Jedes Byte hat eine eigene „Rufnummer" oder Adresse, die von der CPU angewählt wird, wenn der Speicherinhalt gelesen oder überschrieben werden soll. Wird etwa das Byte Nr. 47339 benötigt, gibt die CPU diese Adresse in Binärform auf 16 parallele Leitungen, den „Adreß-Bus". Das gewünschte Byte wird dann auf den „Daten-Bus" (8 weitere Parallel-Leitungen) geschaltet, von dem die CPU die Information in ihre Register übernimmt.

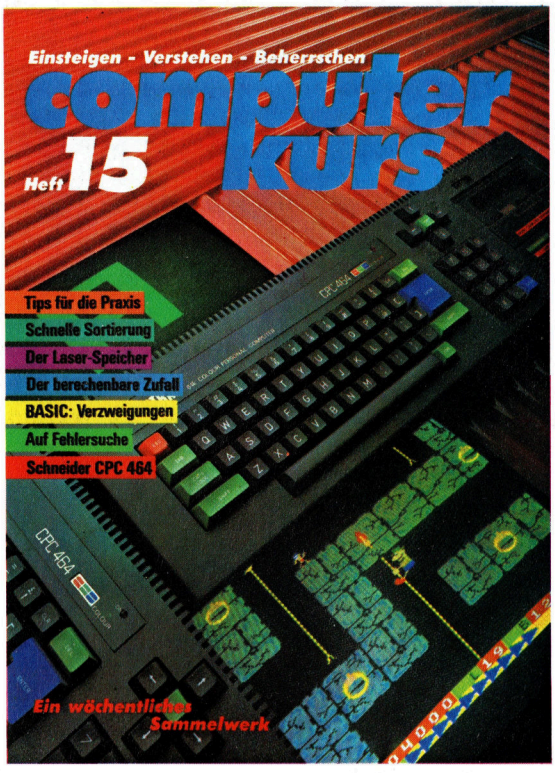
ADSR = ADSR
Schon bei den ersten Heimcomputern gab es Möglichkeiten zur Tonerzeugung, bei etwas besseren Geräten auch eine softwaremäßige Steuerung der Lautstärke. Mit ADSR-Hüllkurvengeneratoren, die in anspruchsvollen Heimcomputern zu finden sind, kann der Programmierer neben Frequenz und Lautstärke auch den Klangcharakter der erzeugten Töne beeinflussen.

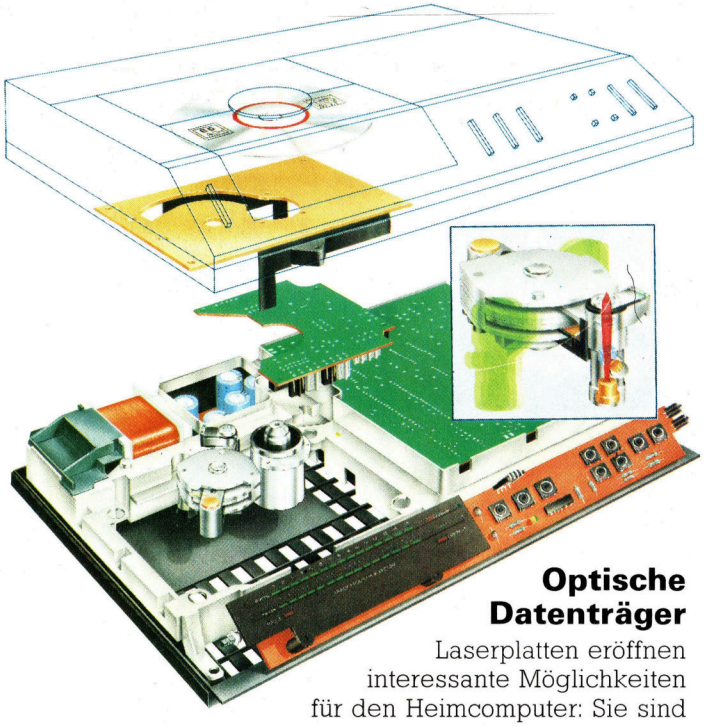
Vollständig heißt es „Attack-Decay-Sustain-Release-Hüllkurvensteuerung", und gemeint ist damit, daß das Klangvolumen eines Tones während seiner Erzeugung verändert werden kann. Die Struktur der Hüllkurve wird mit vier Werten definiert: Einschwingphase (Attack = Anschlag), Abklingen (= Decay), Haltephase (= Sustain; konstante Lautstärke) und Ausklingen (Release = Loslassen).

Bildnachweise
365: Ian McKinnell
366: Ian McKinnell, IO Research
367: Siggraph, Applicon, Intergraph
368, 369: Chris Stevens
370, 371: Ian McKinnell
372: Liz Dixon
378, 379: David Weeks
381: Ian McKinnell
382, 383: Kevin Jones, Ministry of Transport
384: Sperry LTD.
388: Liz Dixon
389: Kevin Jones
390: Ian Dobbie
391: Ian McKinnell, Ian Dobbie
392: Steve Cross

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

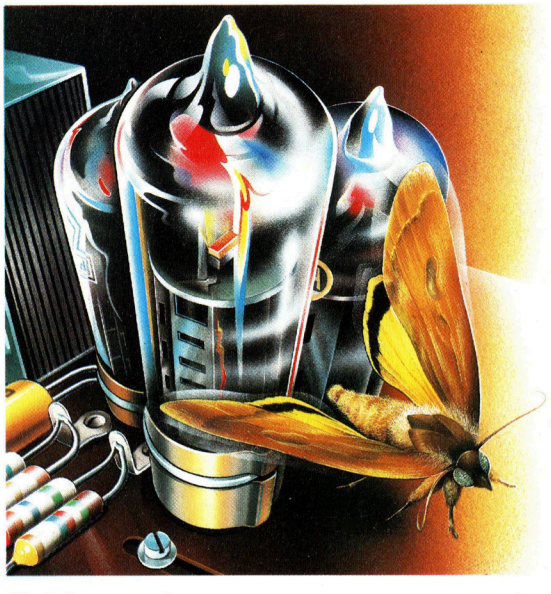
computer kurs Heft 15





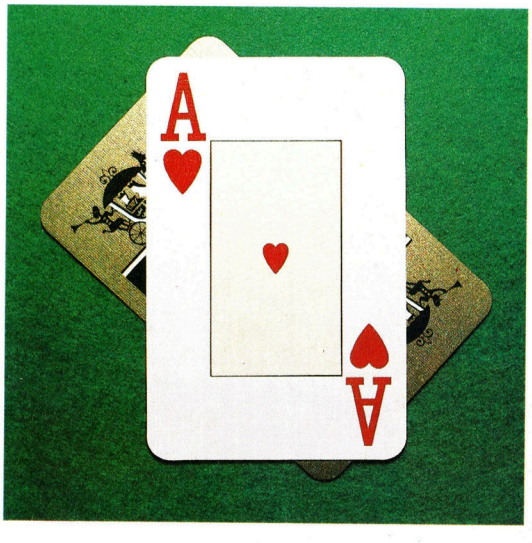
Optische Datenträger

Laserplatten eröffnen interessante Möglichkeiten für den Heimcomputer: Sie sind als Massenspeicher einsetzbar und unempfindlich gegen Magnetfelder.



Fehlersuche

Ein Nachtfalter („Bug") lieferte den Fachausdruck für Fehlersuche: „debugging".



Glücksspiele

Mit dem Computer kann zwar nicht das Glück im Spiel herbeigeführt werden, doch lassen sich die Chancen vergrößern.

+++ Der Schneider CPC 464 +++ BASIC: Verzweigungen +++ Grafiksystem Robo 1000 +++ Sortierverfahren +++ Tips für die Praxis +++ Der Tupel-Effekt +++ LOGOmotion +++ Einhand-Schreiber +++