Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 29

Der Apple IIc

Schach auf dem Rechner

Flags im Assembler

Universelles Grafiktablett

Ein wöchentliches Sammelwerk

computer kurs

Heft 29

Inhalt

WIE SIE JEDE WOCHE IHR HEFT BEKOMMEN

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

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

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

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

Schweiz: Das einzelne Heft kostet sfr 3,80. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

Abonnement

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

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

SAMMELORDNER

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

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

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

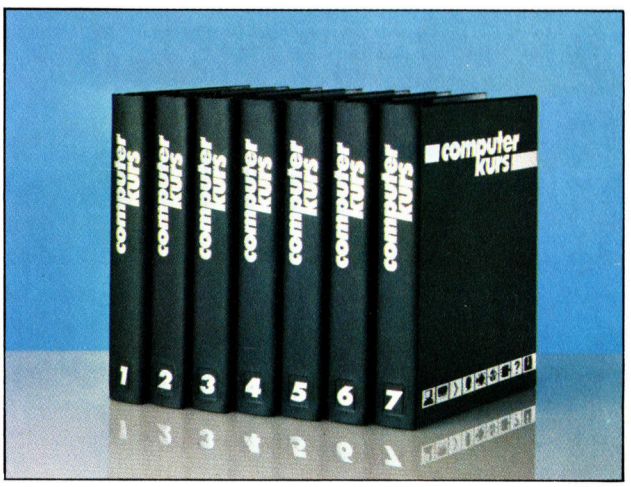
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

INHALTSVERZEICHNIS

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

Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Elke Leibinger, Susanne Brandt, Uta Brandl (Layout), Sammelwerk RedaktionsService 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

David Levy ist internationaler Schachmeister, der seit 1978 aber nicht mehr in Turnieren spielt. 1968 setzte er eine große Summe in einer Wette aus, die beinhaltete, daß ihn kein Schachprogramm in den nächsten zehn Jahren schlagen könne. Die Wettdauer ist seitdem verlängert worden, doch bis heute ist er unbesiegt geblieben. Levy, wohl führende Kapazität im Computerschach, ist Chef des Unternehmens Intelligent Software, einer Gesellschaft, die auf die Entwicklung von Schachprogrammen spezialisiert ist. Levy glaubt, daß auch kleinere Computer allmählich die Schachspielstärke von Großrechnern erlangen. Er schätzt, daß in fünf bis acht Jahren ein Microcomputer den „Belle" (einen Schachcomputer) und den Großrechner „Cray Blitz" schlagen wird.

Schach auf Chips

Bekanntlich werden Heimcomputer hauptsächlich zum Spielen benutzt. So ist es auch nicht verwunderlich, daß Schach, eines der ältesten und beliebtesten unter den strategischen Brettspielen, schon in der zweiten Rechnergeneration für Computer umgesetzt wurde.

Es gibt kaum ein Spiel, das eine ähnliche Faszination wie Schach ausübt. Das „Spiel der Könige" wird weltweit von Millionen Menschen seit mehreren tausend Jahren gespielt, wobei die Regeln, die im siebzehnten Jahrhundert aufgestellt wurden, bis heute fast unverändert blieben. Einige Schachbegeisterte widmen ihr Leben dem Studium und der Beherrschung dieses Spiels und trainieren ihren Intellekt auf die vom Spiel geforderte geistige Beweglichkeit und Präzision. So wurde auch versucht, zusätzliche Schwierigkeitsstufen einzubauen: zum Beispiel ein dreidimensionales Schachspiel, das mit mehreren, übereinanderliegenden Brettern ausgestattet vom Spieler eine noch höhere Konzentration fordert. Eine andere Variante ist das Drei-Personen-Schach, dessen Grundlage ein Y-förmiges Brett bildet. Für das Spiel auf der Diagonalen, wo die drei „Flügel" aufeinandertreffen, gelten spezielle Regeln. Die Idee, auf der diese Version basiert: Zwei Spieler schließen sich zusammen, um gegen den dritten anzutreten. Danach kämpft jeder einzeln um den Sieg. Doch all diese Neuentwicklungen haben der Beliebtheit des ursprünglichen Zweier-Schachs keinen Abbruch getan.

Ein Grund dafür ist, daß das Spiel an sich bereits zahllose Variationsmöglichkeiten zuläßt. Im Jahre 1949 berechnete der Mathematiker Claude Shannon in seiner Studie „Programming a Computer for Playing Chess", daß sich aus 40 Zügen allein 10^{120} Spielmöglichkeiten ergeben. Das bedeutet, daß ein Spieler, der, 24 Stunden am Tag, sieben Tage in der Woche spielt und dabei für jedes Spiel eine Stunde braucht, 10^{17} Jahre benötigt, um alle Variationen auszuprobieren! Natürlich wurde dabei der Faktor „Erfahrung", durch den sich die Anzahl der Möglichkeiten reduzieren läßt, außer acht gelassen.

Es wird deutlich, wie aufwendig es ist, ein Schachprogramm für den Computer zu erstellen. Die ersten Schachspiele liefen auch nur auf Großrechnern, die schon damals genügend Speicherkapazität und eine hohe Ausführungsgeschwindigkeit boten. Seit einigen Jahren gibt es jedoch auch diverse leistungsfähige Programme für kleinere Computer.

Schachprogramme basieren auf numeri-

Software-Turnier

Um einige der populärsten Schachprogramme für Heimcomputer besser bewerten zu können, führte COMPUTER KURS ein Mini-Turnier mit folgenden Programmen durch: „Sargon III" für den Apple IIe (Hayden Software), programmiert von Dan und Kathe Spracklen; „Cyrus IS Chess" für den Spectrum mit 48 K (Intelligent Software); „Colossus 2.0" für den Commodore 64 (CDS Micro Systems, programmiert von Martin Bryant) und „GrandMaster 64", ebenfalls für den Commodore 64 (Audiogenic), ein Programm des deutschen Softwarehauses „Kingsoft".

Obwohl diese Programme bereits bei internationalen Computer-Wettbewerben gegeneinander spielten, wollten wir eine grobe Einschätzung der Leistungsfähigkeit auf der Basis von Besonderheiten, Spielbarkeit und Stärke gewinnen. Unser Turnier wurde wie folgt durchgeführt: Je zwei Spiele für jedes Programm, eines in der niedrigsten Spielstärke und eines in Turnierspielstärke. Ein Gesamtgewinner sollte nicht ermittelt werden.

Eines der Probleme beim Spielen von Schachcomputern gegeneinander resultiert daraus, daß die Spielstärke der Programme nur schwer bestimmbar ist. Die Stärke wird normalerweise durch die Zeitdauer festgelegt, die ein Computer intern benötigt, um den besten Zug zu errechnen. Es gibt aber keine direkte Wechselbeziehung zwischen einer 10-Sekunden-Begrenzung in einem Programm und derselben in einem anderen. So stiehlt zum Beispiel „Sargon III" seinem Gegenspieler Zeit, da das Programm während des gegnerischen Zuges weiterrechnet. Alle anderen Programme unterbrechen die Berechnung in dieser Zeit. Trotzdem versuchten wir, so fair – und vor allem so genau – wie möglich zu sein, um die Programme für Sie zu vergleichen.

Spielqualität

Grundsätzlich spielten alle Programme, falls keine Eingabeänderung erfolgte, in der untersten Spielstärke (pro Zug zehn Sekunden „Bedenkzeit"). Bei allen fiel auf, daß sie seltsame, offensichtlich sinnlose Züge im Endstadium des Mittelspiels machten. – Möglicherweise das Ergebnis einer „stillen" Position. Die Computer versuchten einfach, die „Zeit totzuschlagen" und darauf zu warten, daß etwas Interessantes geschähe. Alle vier Programme demonstrierten ein witziges, in Teilen brillantes taktisches Spiel.
Die Ergebnisse des Turniers sind auf der gegenüberliegenden Seite zusammengefaßt.

schen Berechnungen, wobei die zwei Hauptelemente des Spiels zu beachten sind: das Material und die Beweglichkeit. Das „Material" bezieht sich auf die Anzahl und den Wert der Figuren auf dem Brett. Das Programm weist jeder Figur einen numerischen Wert zu. Der König kann dabei beispielsweise einen hohen Wert von 10 000 erhalten (da der Verlust des Königs das Spiel beendet), die Königin bekommt den Wert neun, der Turm fünf, Läufer und Springer drei und Bauern den Wert eins. Bei der Überlegung, ob es vorteilhafter ist, einen Stein zu opfern, um dafür den des Gegners einzunehmen, vergleicht der Computer die jeweiligen Werte. Die meisten Programme legen hier größte Sorgfalt auf den relativen Wert und vermeiden es, Figuren auszutauschen, wenn dieses einen Materialverlust bewirkt – es sei denn, daß die Aktion offensichtlich zu einer stärkeren Position verhilft.

Die „Beweglichkeit" spielt beim Schach eine wichtige Rolle, da eine Figur, die man nur begrenzt versetzen kann, einen niedrigen Wert erhält. Umgekehrt erhöht sich ihr Wert, wenn sie sich so setzen läßt, daß sie mehrere Positionen gleichzeitig beeinflußt. Das Programm muß daher Beweglichkeit sowie Material berechnen. Weiterhin sollte es in der Lage sein, im voraus zu planen und die optimale Zugfolge in kürzester Zeit zu ermitteln.

Die meisten Schachprogramme nutzen die „Schlagvorteilsberechnung", bei der ermittelt wird, wieviele Züge in einer vorgegebenen Zeitspanne ausführbar sind. Die zur Verfügung stehende Zeit für jeden Zug hängt von der Spielstufe ab, die zu Beginn des Spiels gewählt wurde. Die Zeitspanne, in der der Computer seinen Zug durchführen muß, variiert bei den verschiedenen Stufen und kann bis zu mehrere Stunden dauern. Je mehr Zeit der Computer zum „Überlegen" hat, um so wahrscheinlicher ist die Möglichkeit für einen optimal plazierten Angriff.

Zug um Zug rechnen

Bei jedem Zug überprüft der Rechner, ob der König sich im Schach befindet, anschließend, ob Figuren genommen werden oder geschlagen werden können, ob Schlüsselpositionen zu besetzen sind und dergleichen mehr. Je mehr Kriterien das Programm durchrechnen kann, desto besser wird das Ergebnis sein. Bei der letzten Frage geht es darum, herauszufinden, ob der gegnerische König in eine Schachposition gezwungen werden kann.

Bei Partien zwischen Computer und Mensch haben Computer zwar den Vorteil von Schnelligkeit und Überblick, doch ein guter menschlicher Schachspieler wird ein hervorragendes

Cyrus IS Chess erreichte bei Colossus ein Remis und schlug GrandMaster in der einfachsten Spielstufe. In der Wettkampfstufe erreichte es bei Sargon III ein Remis.

Colossus erreichte bei Cyrus IS Chess in der einfachsten Stufe ein Remis, schlug GrandMaster in der Wettkampfstufe und erreichte in der gleichen Stufe bei Sargon III ein Remis.

Sargon III verlor gegen GrandMaster in der einfachsten Stufe und erreichte in der Wettkampfstufe bei Cyrus und Colossus ein Remis.

GrandMaster schlug Sargon III und verlor gegen Cyrus IS Chess in der einfachsten Stufe. Es verlor gegen Colossus in der Wettkampfstufe.

Merkmal	GrandMaster	Colossus	Cyrus IS	Sargon III
Unsichtbares Spiel	NEIN	JA	NEIN	NEIN
Auflistung möglicher Züge	NEIN	JA	NEIN	NEIN
Spiel des „zweitbesten" Zuges	NEIN	NEIN	JA	NEIN
Spielwiedererholung	NEIN	JA	JA	JA
Listing-Darstellung	NEIN	JA	NEIN	JA
Listing-Ausdruck	NEIN	NEIN	JA	JA
Zurücknahme eines Zuges	JA	JA	JA	JA
Unterricht	JA	NEIN	NEIN	JA
Dame aus dem Spiel nehmen	NEIN	JA	JA	JA
Spiel zwischen Menschen	NEIN	JA	JA	JA
Seitenwechsel	JA	JA	JA	JA
Problemanalyse	NEIN	JA	JA	JA
Zugsuche	EIN PLY	JA	NEIN	JA
Invertiertes Brett	JA	JA	JA	JA
Zug-Angebot	NEIN	NEIN	NEIN	JA
Spielfeld-Ausdruck	NEIN	NEIN	JA	JA
Spiel speichern	NEIN	JA	JA	JA
Programmbibliothek abschaltbar	NEIN	NEIN	NEIN	JA
Autospiel	NEIN	JA	JA	JA
Echtzeit-Uhr	JA	JA	NEIN	JA

Besonderheiten

Alle getesteten Programme beinhalteten die Rochade (König/Turm-Tausch), Wechsel Bauer gegen Königin und das Schlagen „en passant" neben normalen Zug- und Schlagregeln. Dazu kommen bei einigen Programmen interessante Ergänzungen. „Sargon III" ist diesbezüglich am besten ausgestattet. Zum Lieferumfang gehört eine zweite Diskette, auf der 107 klassische Schachpartien sowie 45 Schachprobleme festgehalten sind. Die Dokumentation ist außerordentlich gut. Sie besteht aus 75 Seiten. Da „Sargon III" auf einem Apple IIe läuft, ist das Programm dreimal so teuer wie die anderen. Wie die Übersicht zeigt, bieten „Cyrus IS" und „Colossus" dieselben guten Eigenschaften.

Zusammenfassung

Unter dem Gesichtspunkt „Spielbarkeit" sind „Cyrus IS" und „Colossus" die einfachsten Programme, da die Eingabe mittels Cursor erfolgt, wogegen bei „Sargon III" und „GrandMaster" algebraische Eingaben erforderlich sind. „Colossus" und „Sargon" haben die beste Bildschirmdarstellung. „GrandMaster" bietet ausgezeichnetes Schach für wenig Geld.

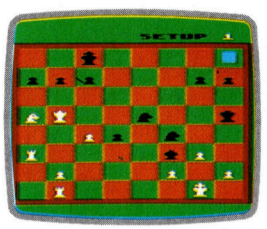
Computerschachprogramm immer deshalb schlagen können, weil der Mensch die Möglichkeit hat, neue Eröffnungen und Positionen zu erkennen bzw. zu schaffen. Computer spielen ein ausgezeichnetes taktisches Schach, doch auch beim Menschen ist es so, daß ein guter Stellungsspieler einen taktischen Spieler schlagen wird. Die Programmierer des Computerschachs haben sich auf das taktische Spiel konzentriert, da es für den Computer lediglich ein Durchrechnen von Zahlenwerten darstellt. Macht dagegen ein menschlicher Gegner einen unkonventionellen Zug, kann der Computer oft nicht den besten Gegenzug errechnen. Deshalb haben sehr viele Schachprogramme auch Probleme mit dem „stillen" (also: Stellungs-)Spiel, bei dem keiner der möglichen Züge einen erkennbaren taktischen Vorteil bietet. In diesen Situationen, die häufig im Endspiel entstehen, schiebt das Programm einfach Figuren umher, statt vorauszuplanen.

Eine unlängst entwickelte Programmiertechnik beinhaltet die „selektive Suche". Bei Verwendung dieser Technik ahmt der Computer einen menschlichen Spieler nach, indem er eine kleinere Anzahl von Zügen mit größerer Genauigkeit durchrechnet. Das deutsche Unternehmen Hegener und Glaser hat auf dieser Grundlage „Mephisto III" entwickelt, das die ersten beiden „ply" genau berechnet.

Herz gegen Kopf

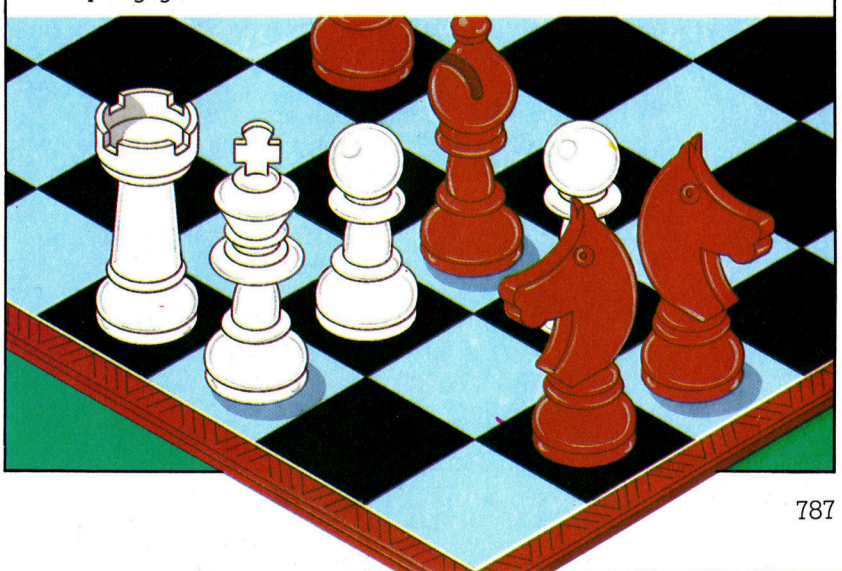
In diesem Beispiel spielt Moritz (schwarz) gegen Emmerich (weiß) im Jahre 1922. Die Position wurde im Film „Night Moves" dargestellt. Schwarz wird durch das Damenopfer und drei anschließende elegante Springerzüge Weiß mattsetzen. Die meisten menschlichen Spieler würden diese Folge von Zügen allen anderen vorziehen. Moritz übersah diese Lösung und bedauerte seinen Fehler anschließend sehr schnell.

Zwar erkannten alle vier Programme die „Matt"-Situation, doch keines machte einen Vorschlag, wie der König zu ziehen sei. Die Unfähigkeit des Computers, dieses Schlußspiel als das „beste" zu ermitteln, scheint des Menschen beste Verteidigung beim Spiel gegen den Rechner zu sein.



Die Züge des Springers

1		H5—H2	Schach
2	G1—H2	E5—G2	Schach
3	H2—G1	F4—H3	Schach
4	G1—F1	G4—H2	Schach-matt



Touchmaster

Für den Entwurf von Grafiken gibt es auf dem Markt eine Vielzahl von Hilfsmitteln. Das hier vorgestellte „Touchmaster"-Grafiktablett zeichnet sich durch universelle Verwendbarkeit für fast alle gängigen Heimcomputer aus und ist – so der Hersteller – auch als einfache Tastatur zu gebrauchen.

Mit hochauflösender Grafik können heute alle erfolgreichen Rechner aufwarten. Wenn aber keine fertige Software verfügbar ist, kostet es viel Zeit und Mühe, die Grafiken zu entwerfen. Die Möglichkeiten werden daher meist nur zum Teil genutzt.

Häufig sind Vorlagen in den Rechner zu übertragen – dann reicht ein Skizzen-Programm zum Freihandzeichnen nicht aus. Dafür wird eine Reihe von Digitalisier-Tabletts angeboten, allerdings meist mit rechnerspezifischen Schnittstellen. Das Touchmaster-Grafiktablett ist dagegen so universell ausgelegt, daß es an die meisten Heimcomputer mit Standard-Schnittstellen angeschlossen werden kann. Das Gerät wird auch als Tastatur-Ersatz angepriesen, es eignet sich aber wegen der einfachen Ausführung nur zur Menüsteuerung. Die Dateneingabe und auch das Laden der Touchmaster-Software erfordern jedoch eine richtige Tastatur.

Das Gerät hat ein solides graues Gehäuse mit den Maßen 350 × 330 × 35 Millimeter in flacher Pultform und ermöglicht eine bequeme Arbeitshaltung. Mitgeliefert wird ein Transformator mit einer Leuchtdiode als Betriebsanzeige, aber ohne Ein/Ausschalter. Zum Anschluß an den Computer sind an der Rückseite serielle und parallele Ausgänge angebracht, außerdem eine – in der Anleitung nicht erwähnte – Buchse für einen Fußschalter. Auch sonst ist die Anleitung eher dürftig: Sie enthält zwar Anschlußhinweise und eine Anzahl einfacher BASIC-Programme zum Auslesen von Koordinaten, aber zu wenige Detailinformationen.

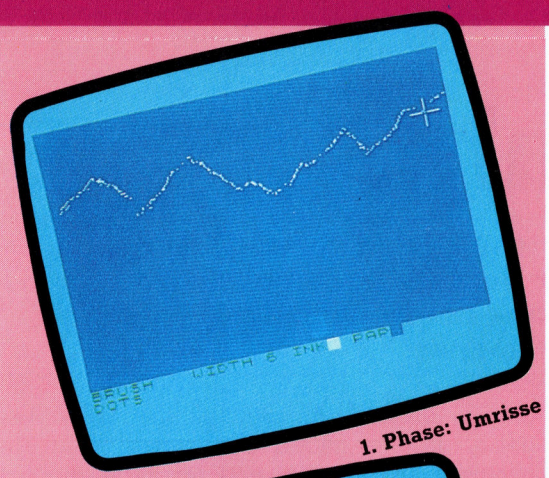
Das Tablett arbeitet nach dem Membranschalter-Prinzip, mit einer Auflösung von 256 × 256 Punkten. Zwischen der Deckfolie und einer leitfähigen Unterlage befindet sich ein isolierendes Gitternetz. Durch Druck auf die Deckfolie wird der Kontakt zur Unterlage hergestellt.

Die mitgelieferte Schablone wird auf das Zeichenfeld gelegt. Durch Drücken der entsprechenden Felder am rechten Rand können Sie Farben, Strichstärken und Strukturen wählen – wenn Sie dann den Zeichenstift über die Folie führen, entsteht eine Ihren Fähigkeiten entsprechende Grafik.

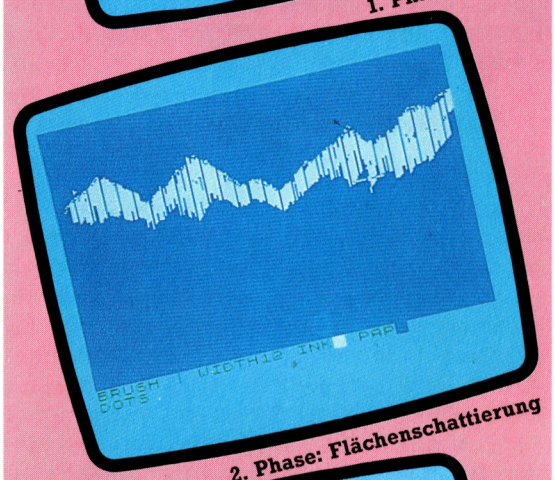
Mäßige Auflösung

Ein eingebauter Microprozessor fragt die Deckfolie in der einen Richtung und die Unterlage senkrecht dazu ab, bis er die Koordinaten des Kontaktpunkts ermittelt hat. Diese gibt er dann über die Serien- und die Parallelschnittstelle aus. Die serielle Übertragung wird für den Acorn B, die parallele für den Commodore 64 und VC20 sowie den ZX-Spectrum und den Dragon verwendet. Die Auflösung des „Touchmaster" liegt unter der vieler hochauflösender Bildschirme, so daß es beim Acorn B im „Mode 0" (640 × 256 Pixel) nicht möglich ist, zwei benachbarte Pixel getrennt anzusprechen.

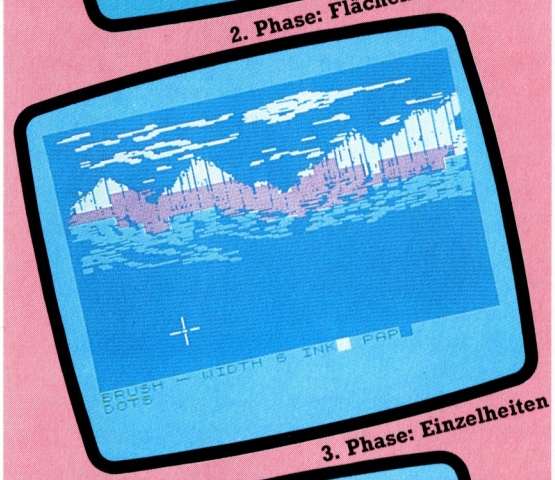
Zum Lieferumfang gehört ein Zeichenprogramm namens „Multipaint". Es dient hauptsächlich zur Demonstration der Anwendungsmöglichkeiten, ist aber als Entwurfshilfe kaum brauchbar. Die Varianten des angebotenen Menüs sind auf einer Plastik-Schablone aufgedruckt, wobei die getroffene Wahl in einem „Status"-Fenster am unteren Bildrand angezeigt wird. Sie können fünf verschiedene Pinseltypen, jeweils mit einer Breite von 2–32 Pi-



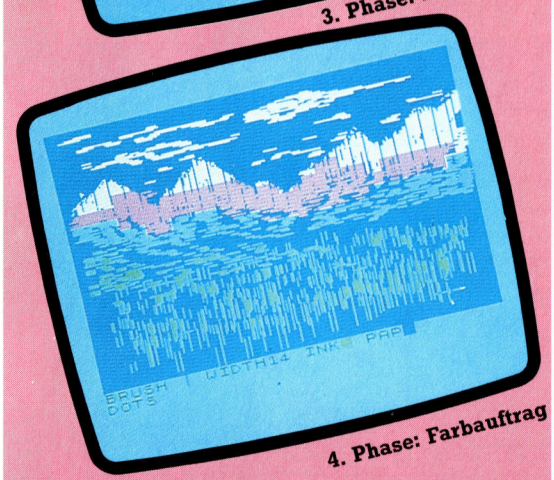
1. Phase: Umrisse



2. Phase: Flächenschattierung



3. Phase: Einzelheiten



4. Phase: Farbauftrag

Der Künstler am Werk
In Verbindung mit der Multipaint-Software und der Schablone, die zum Lieferumfang gehören, können Sie mit dem Touchmaster-Tablett anspruchsvolle Farbgrafiken entwerfen. – Hier sind die Entwicklungsstadien einer Grafik festgehalten.

xeln (in Zweierschritten) vorgeben. Angezeigt werden auch die Betriebsart – „Dots", „Points" oder „Freehand" – und die gewählte Vorder- und Hintergrundfarbe. Zum Wechseln der Farbe wird das entsprechende Schablonenfeld gedrückt, bis die gewünschte Farbe im Status-Fenster erscheint.

Nach Festlegung von Pinseltyp und Farbe stehen Routinen zum Zeichnen von Kästchen, Kreisen, Vielecken und Linienzügen zur Auswahl. Ein Zeichenstift wird mitgeliefert, aber der Zeigefinger tut's auch, und zwar wegen des großen Tablettformats viel besser als etwa beim „Koala-Pad". Die Fingerposition wird recht genau in Koordinaten umgesetzt.

Leider offeriert „Multipaint" nur die elementaren Funktionen. Der „Fill"-Befehl auf der Schablone hat nicht die erwartete Wirkung, jedenfalls nicht beim Spectrum. Es gibt auch keine Bearbeitungsmöglichkeiten wie Detailvergrößerung oder nachträglichen Farbwechsel – ein deutlicher Nachteil zumindest für den Spectrum, bei dem Sie besser schwarzweiß entwerfen und danach Farben auftragen.

Rein hardwaremäßig spricht im Vergleich mit Geräten wie dem „Grafpad" und dem „Koala-Pad" viel für das Touchmaster-Tablett. Der Aufbau ist solide, das Gerät bietet volles A4-Format und kann an die meisten gängigen Heimcomputer angeschlossen werden. Das ist ein wesentlicher Vorteil. Wenn Sie sich irgendwann einen neuen Rechner zulegen, brauchen Sie allenfalls ein neues Interface (und natürlich das passende Programmpaket).

Enttäuschend sind angesichts der Qualität des Gerätes die der Anleitung und mitgelieferten Software. Der Hersteller hat angekündigt, weitere Programme speziell für dieses Tablett herauszubringen. Mehr Erfolg ist eigentlich erst zu erwarten, wenn sich unabhängige Softwarehäuser entschließen, den Touchmaster zu unterstützen.

Touchmaster

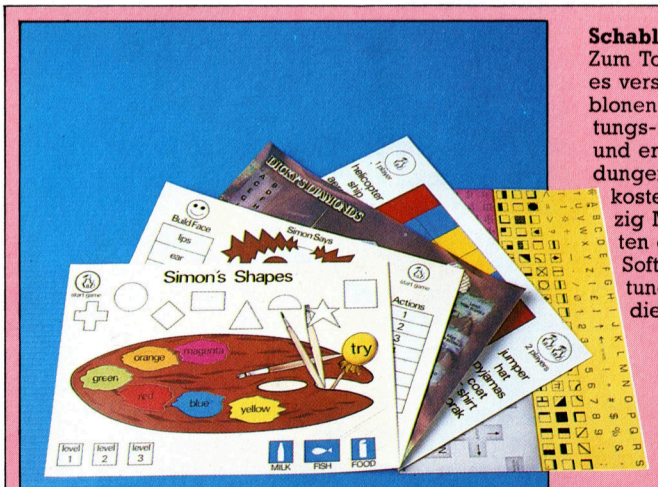
ABMESSUNGEN
350 × 330 × 35 mm

SCHNITTSTELLEN
Serien- und Parallel-Schnittstelle, an die zahlreiche Heimcomputer oder die Touchmaster-eigene Tastatur angeschlossen werden können.

ANLEITUNG
Es werden zwar die erforderlichen Hinweise für den Gebrauch des Tabletts gegeben, aber mehr Detailinformation wäre durchaus wünschenswert.

STÄRKEN
Die vielfältigen Anschlußmöglichkeiten (und Software-Versionen) machen den Touchmaster zu einem äußerst anpassungsfähigen Peripheriegerät.

SCHWÄCHEN
Die Qualität der Software hält einem Vergleich mit der anderer Hersteller nicht stand, und die Anleitung ist etwas dürftig.



Schablonen
Zum Touchmaster gibt es verschiedene Schablonen für Unterhaltungs- und Lernspiele und ernsthafte Anwendungen. Die Packungen kosten je etwa vierzig Mark und enthalten die zugehörige Software, die Anleitungen und natürlich die Schablonen.

Heißer Draht

Der User Port Ihres Heimcomputers ist die Brücke zwischen dem Innenleben des Rechners und seiner Umwelt. In den nächsten Folgen erfahren Sie, wie Sie den User Port zum Registrieren und Verändern physikalischer Vorgänge verwenden können.

Die meisten Heimcomputer verfügen über einen User Port, dessen elektrische Anschlüsse einen direkten Zugang zum Speicher des Rechners ermöglichen. Alle digitalen Geräte arbeiten mit dem Binärsystem aus Einsen und Nullen, die sich durch zwei Spannungszustände symbolisieren lassen.

Jeder Speicherplatz besteht aus einer Gruppe von acht einzelnen Speicherzellen. Jede Zelle liegt entweder auf 0 oder +5 Volt. Das Muster aus hohen und niedrigen Spannungen verkörpert die am betreffenden Platz gespeicherte Zahl. Eine Zellenspannung von +5 Volt wird als „High", eine von 0 Volt als „Low" bezeichnet. Die elektrischen Anschlüsse des User Ports sind mit einem oder mehreren Speicherplätzen im Computer verbunden. Wir können nun die Werte dieser Speicherstellen am User Port ablesen oder neue Spannungen und damit neue Werte eingeben.

Es gibt zwei verschiedene Typen von User Ports: Manche haben acht Anschlußpunkte für die Ein- und acht Anschlußpunkte für die Ausgabe, andere haben nur einmal acht Anschlüsse, die für Ein- und Ausgabe zugleich dienen (Bidirektionaler User Port).

Das Daten-Richtungs-Register

Neben den Speicherstellen für den User Port haben Heimcomputer mit bidirektionalem Eingang zusätzlich einen weiteren Speicherplatz, der als „Daten-Richtungs-Register" (data direction register = DDR) bezeichnet wird. Das Richtungsregister bestimmt, ob die acht Anschlüsse Daten empfangen oder aussenden sollen. Eine 1 im Register stellt den Rechner auf Senden, bei einer 0 werden Daten empfangen. Um alle acht Leitungen des Ports auf Senden zu schalten, muß der Wert im Register auf 255 gesetzt werden (binär 11111111). Umgekehrt ist der User Port auf Einlesen geschaltet, wenn das Richtungsregister den Wert 0 enthält. Sie können aber auch kombinieren, zum Beispiel

Der richtige Stecker

Für das neue Projekt müssen Sie zuerst die richtigen Kabelanschlüsse für den Acorn B bzw. den C 64 herstellen. Zur Verbindung mit der Außenwelt dienen acht Datenleitungen, neben denen auf jeder Seite noch eine Masseleitung geführt wird. Sie brauchen dazu:

Acorn B

- 20-poligen IDC-Stecker
- 20-poliges Flachbandkabel (ca. 1 Meter)
- Lötkolben und Lötzinn

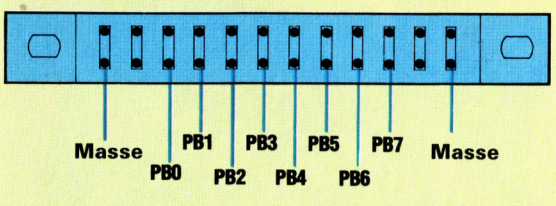
Commodore 64

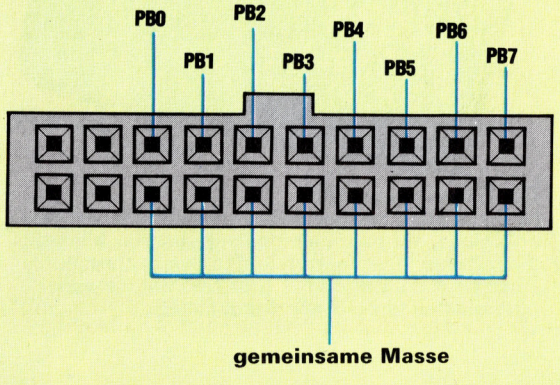
- 24-poligen 0,15-Zoll-Platinenstecker
- 10-poliges Flachbandkabel (ca. 1 Meter)
- Lötkolben und Lötzinn

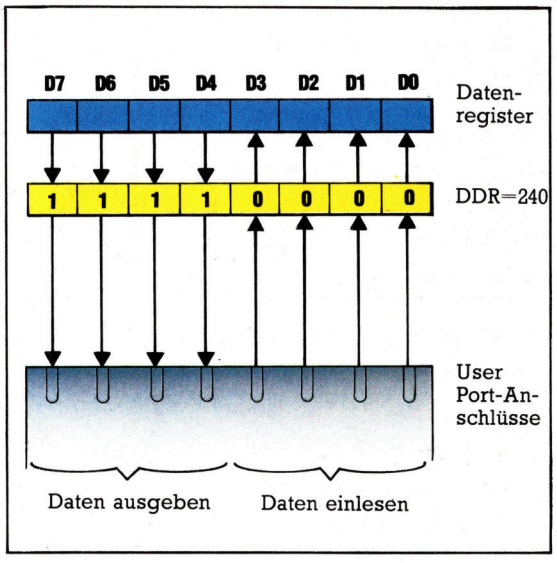
Im Handbuch des Rechners können Sie nachschauen, wo die zehn Anschlußpunkte (zweimal Masse, acht Datenleitungen) liegen. Der Schneidstecker für den Acorn B hat einseitig eine Markierung. Er läßt sich in zwei Teile zerlegen. Halten Sie ihn mit der Markierung von sich abgewandt, die Seite mit den Klammern nach oben. Die rot bezeichnete Seite des Kabels muß nach rechts zeigen, wenn Sie sie aufdrücken. Danach wird die Oberseite des Steckers mit gleichmäßigem Druck aufgesetzt (notfalls Klemme oder Schraubstock verwenden). Am anderen Kabelende wie im Bild die überflüssigen Drähte kürzen. Die restlichen Drähte werden abisoliert und verzinnt.

Bei der Kabelmontage für den Commodore müssen Sie die Oberseite des Steckers eindeutig kennzeichnen und beim Einsetzen immer auf diese Markierung achten. Drähte abisolieren und verzinnen, danach in Übereinstimmung mit der Zeichnung am Platinenstekker festlöten.

Mit den Test-Programmen und einem Multimeter können Sie Ihre Anschlußkabel prüfen.







vier Leitungen des Ports auf Senden und vier Leitungen auf Empfangen einstellen: Dazu muß das Richtungsregister auf den Wert 240 (binär 11110000) gesetzt werden. Die Adressen der Speicherstellen (Datenregister) und des Richtungsregisters können Sie folgender Tabelle entnehmen:

Computer	Datenregister	Daten-Richtungsregister
Acorn B	&FE60 (65120 dez.)	&FE62 (65122 dez.)
Commodore 64	$DD01 (56677 dez.)	$DD03 (56579 dez.)

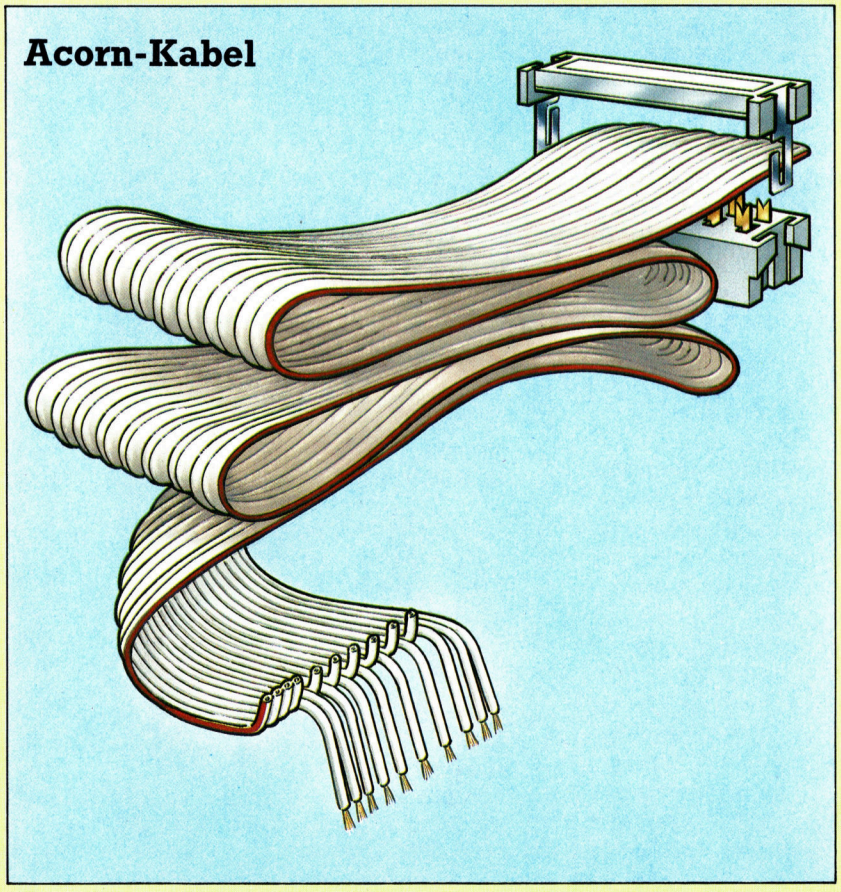
Dieses Programm stellt alle Eingänge des User Ports auf „Empfang" und zeigt den Inhalt der Datenregister:

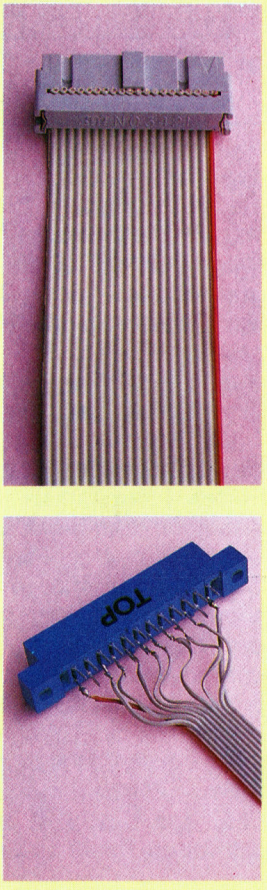
```
  4 REM**********C64***********
  5 REM* DATREG DISPLAY      *
  6 REM**********C64***********
 10 DIM A$(10) :A$(0)="E" :A$(1)="H"
 20 DATREG=56577:DDR=56579
 30 POKE DDR,0: REM = INPUT ONLY
 50 :
100 PE=PEEK(DATREG) :GOSUB 500
150 PRINT "DATREG = " ;PE;" = ";B$
200 GOTO 100
300 :
499 REM*************************
500 REM* BINARY CONVERT S/R *
501 REM*************************
550 B$="":N=PE
600 FOR D=1 TO 8
650 N1=INT(N/2):R=N-2*N1
700 B$=A$(R) + B$:N=N1
750 NEXT D:RETURN
```

Beim Acorn B sind folgende Änderungen zu beachten:

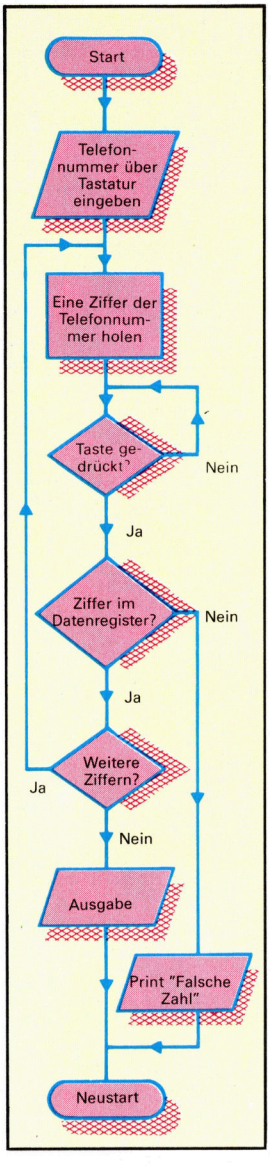
```
 20 DATREG=&FE60:DDR=&FE62
 30 ?DDR=0
100 PE=?(PE):GOSUB 500
```

Beim Starten des Programms mit RUN sehen Sie, daß das Datenregister normalerweise auf





Kabelanschlüsse und Stecker für den User Port des Acorn B und des Commodore 64.

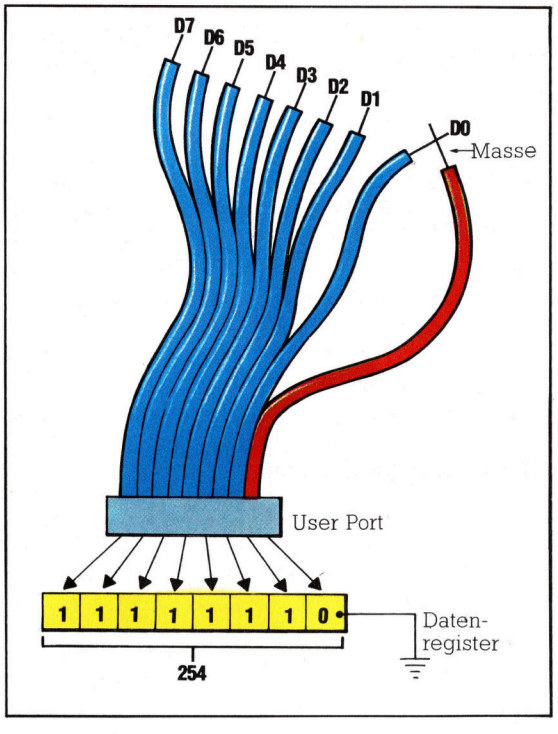


Das Telefonnummern-Programm akzeptiert die Eingabe einer Nummer beliebiger Länge und vergleicht die Nummer Ziffer für Ziffer mit dem Inhalt des zum User Port gehörigen Datenregisters. Leerstellen in der Nummer werden ignoriert. Das Programm wartet auf die Betätigung einer Taste, bevor die eingegebene Zahl mit dem Inhalt des Datenregisters verglichen wird.

255 gesetzt ist (Bildschirmanzeige HHHHHHHH). Das heißt, alle Speicherzellen liegen auf +5 Volt. Mit einem in den User Port eingesteckten Kabel können Sie diese Spannung verändern – damit ändert sich natürlich auch der numerische Inhalt des Datenregisters (DATREG).

0 Volt bei geerdeter Datenleitung

Zum User Port führen zehn Leitungen: Acht davon sind Datenleitungen, für jedes Bit des Datenregisters eine. Die beiden anderen Leitungen liegen auf 0 Volt – dem Masse-Anschluß



des Systems. Wird eine Datenleitung mit Masse verbunden, fällt ihre Spannung auf 0 Volt ab. Gleichzeitig ändert sich der Inhalt des Datenregisters: Wenn Sie während des Programmablaufs etwa D0 erden, ist der Inhalt des Datenregisters DATREG 254 (Bildschirmanzeige HHHHHHHE): Der Anschluß der untersten Speicherzelle liegt auf 0 Volt, alle anderen führen eine Spannung von +5 Volt.

Ein Computer kann Entscheidungen in Abhängigkeit vom Wert bestimmter Variablen treffen, Programmabläufe und Verzweigungen also nach dem Inhalt einer Speicherzelle steuern. Mit dem User Port als Bindeglied kann der Computer auf äußere Ereignisse reagieren – wenn Sie dafür sorgen, daß eine Veränderung der Umwelt über den Port gemeldet wird und ein passendes Programm diese Meldung weiterverarbeitet. Ein einfaches Beispiel ist ein Programm, das Telefonnummern auf ihre Richtigkeit prüft. Die Nummern werden – Ziffer für Ziffer – durch das Erden der Datenleitungen über den User Port in das Datenregister geschrieben. Damit es beim gleichzeitigen Erden mehrerer Datenleitungen nicht zu Fehlern kommt, wartet das Programm auf einen Tastendruck, bevor der Inhalt des Datenregisters analysiert wird. Den einfachen Logikaufbau des Programms können Sie im nebenstehenden Flußdiagramm verfolgen.

```
100 REM**********C64**************
101 REM* TELEPHONE NUMBERS *
102 REM**********C64**************
120 :
130 DATREG=56577:DDR=56579
140 POKE DDR,0:REM = INPUT ONLY
150 :
160 INPUT"TELEPHONE NUMBER";TN$
170 :
180 FOR K=1 TO LEN(TN$)
190 DG$=MID$(TN$,K,1): REM GET DIGIT
200 IF DG$<>" " THEN GOSUB 500
210 NEXT K
250 :
300 PRINT
    "--------SORRY--------"
310 PRINT" NO ONE IS IN AT ";TN$
320 PRINT" PLEASE CALL LATER"
350 PRINT:PRINT:RUN
400 :
500 REM**CONVERT & CHECK S/R**
550 DG=VAL(DG$)
600 PRINT"SET UP DIGIT ON THE LINES"
650 PRINT" AND HIT ANY KEY"
700 GET GT$:IF GT$=" " THEN 700
750 PE=PEEK(DATREG):IF PE=DG
    THEN PRINT DG" OK":RETURN
800 :
850 PRINT"???WRONG NUMBER???"
900 PRINT TN$"<>"LEFT$(TN$,K-1):PE
950 PRINT"??? TRY AGAIN ???"
999 PRINT:PRINT:RUN
```

Beim Acorn B sind folgende Änderungen zu beachten:

```
130 DATREG=&FE60:DDR=&FE62
140 ?DDR=0
700 A=GET
750 PE=?(DATREG):IF PE=DG THEN
PRINT DG" OK" :RETURN
```

Programmier-Übungen

Das Telefonnummern-Programm kann Ihnen bei diesen Vorschlägen als Leitfaden dienen:

1) Schreiben Sie ein Programm, das die Arbeitsweise einer Alarmanlage simuliert.
2) Schreiben Sie ein Programm, das die am User Port ankommenden Impulse während einer festgelegten Zeitdauer zählt.
3) Verändern Sie dieses Zähl-Programm so, daß die ankommenden Impulse für jede der acht Datenleitungen extra gezählt werden.
4) Schreiben Sie ein Programm, das die Funktion eines Zahlenschlosses simuliert.
5) Schreiben Sie ein Programm, mit dem Sie die Hintergrundfarbe Ihres Monitors über den User Port verändern können.

Stille Wasser . . .

Alligatoren, Anakondas und Minen werfende Hubschrauber sind nur einige der Hindernisse, die beim beschwerlichen Weg durch den Bildschirm-Urwald auftauchen.

Dem Wesen nach ist „River Rescue" ein reines Arcadespiel. Höhere Ansprüche werden ohnehin nicht gestellt. Produziert hat dieses Programm Creative Sparks, die Software-Tochter des Medien-Multis Thorn-EMI. „River Rescue" ist für vier Computer-Systeme erhältlich: für den ZX Spectrum (48 K), den Commodore 64, die Atari-Computer und den VC 20 in der Grundversion.

Den verschiedenen Programmversionen sind sehr ausführliche Bedienungsanleitungen beigefügt. Außerdem wird der Spieler eingeladen, dem Creative Sparks Software Club beizutreten. Die Mitgliedschaft ist kostenlos und bietet aktuelle Informationen, Sonderangebote und Wettbewerbe im Heimcomputerbereich.

Das Spiel selbst ist sehr einfach, bietet aber genug interessante Elemente, um Spaß zu machen. Sie steuern das Rettungsboot und haben die Aufgabe, eine Gruppe von Wissenschaftlern, die sich am Oberlauf des Flusses befindet, zu retten. Warum die Wissenschaftler überhaupt Ihre Hilfe nötig haben, bleibt ungeklärt. Die Bedienungsanweisung fordert aber, daß sie in ein Krankenhaus zu bringen sind. Es muß ihnen also ein Unglück zugestoßen sein.

. . . sind gefährlich

Während sie versuchen, die verletzten Wissenschaftler zu bergen, müssen Sie gleichzeitig das Boot, das mit beträchtlicher Geschwindigkeit über das Wasser rauscht, um Inseln und treibende Baumstämme herumsteuern. Und zugleich sind auftauchende Alligatoren aus dem Weg zu räumen. Die VC 20-Version unterscheidet sich von den anderen insofern, als einige Anakondas und Kanus als „Dreingabe" gestiftet wurden. In regelmäßigen Abständen findet man Stege am Ufer, wo die Wissenschaftler aufgenommen werden können. Der erfolgreiche Transport eines Verletzten auf die andere Flußseite oder das Erlegen eines Alligators läßt das Punktekonto beachtlich wachsen.

Bringt man die Wissenschaftler gleich gruppenweise in Sicherheit, gibt's Zusatzpunkte. Allerdings faßt das Boot nur höchstens neun Insassen. Die Sache wird dadurch etwas komplizierter, daß, trifft das Boot auf ein Hindernis, alle Mann über Bord gehen und verloren sind. Es liegt also im Ermessen des Spielers, auf hohe Punktzahl oder auf Nummer Sicher zu gehen, indem man einen Passagier nach dem anderen außer Gefahr bringt. Damit keine Langeweile aufkommt, taucht ein Hubschrauber – in der Spectrum-Version ein Flugzeug – auf und läßt Minen ins Wasser fallen, die beseitigt werden müssen, bevor sie Unheil anrichten können. Die Version für den VC 20 wird als Steckmodul geliefert. In den höheren Schwierigkeitsgraden dürfen Sie dort die pro Leben angesammelten Punkte sogar in das nächste mitnehmen!

River Rescue ist ein totales „Action-Baller-Spiel", bei dem man auf alles schießen muß, was sich bewegt. Solange man es noch nicht allzu gut kennt, macht es viel Spaß. Später wird es jedoch, wie so viele andere Computerspiele auch, recht langweilig.

River Rescue: Für den ZX Spectrum, Atari, C 64 und VC 20 (Grundversion)

Hersteller: Creative Sparks (Thorn-EMI)

Autor: Kevin Buckner

Joysticks: Kempston/Interface 1 (Spectrum), Commodore-kompatible Joysticks, (bei VC 20 und C 64), Atari-kompatible Joysticks

Format: Cassette; Steckmodul (VC-20)

Hier ist die Spectrum-Version im Bild. Die erste Abbildung zeigt den Auftakt-Screen, der einen guten Eindruck der Grafik vermittelt. Im zweiten Bild ist ein fortgeschrittenes Spielstadium zu sehen. Das Boot bringt einen Wissenschaftler in Sicherheit zu seinen bereits geretteten Kollegen.

Trigonometrie

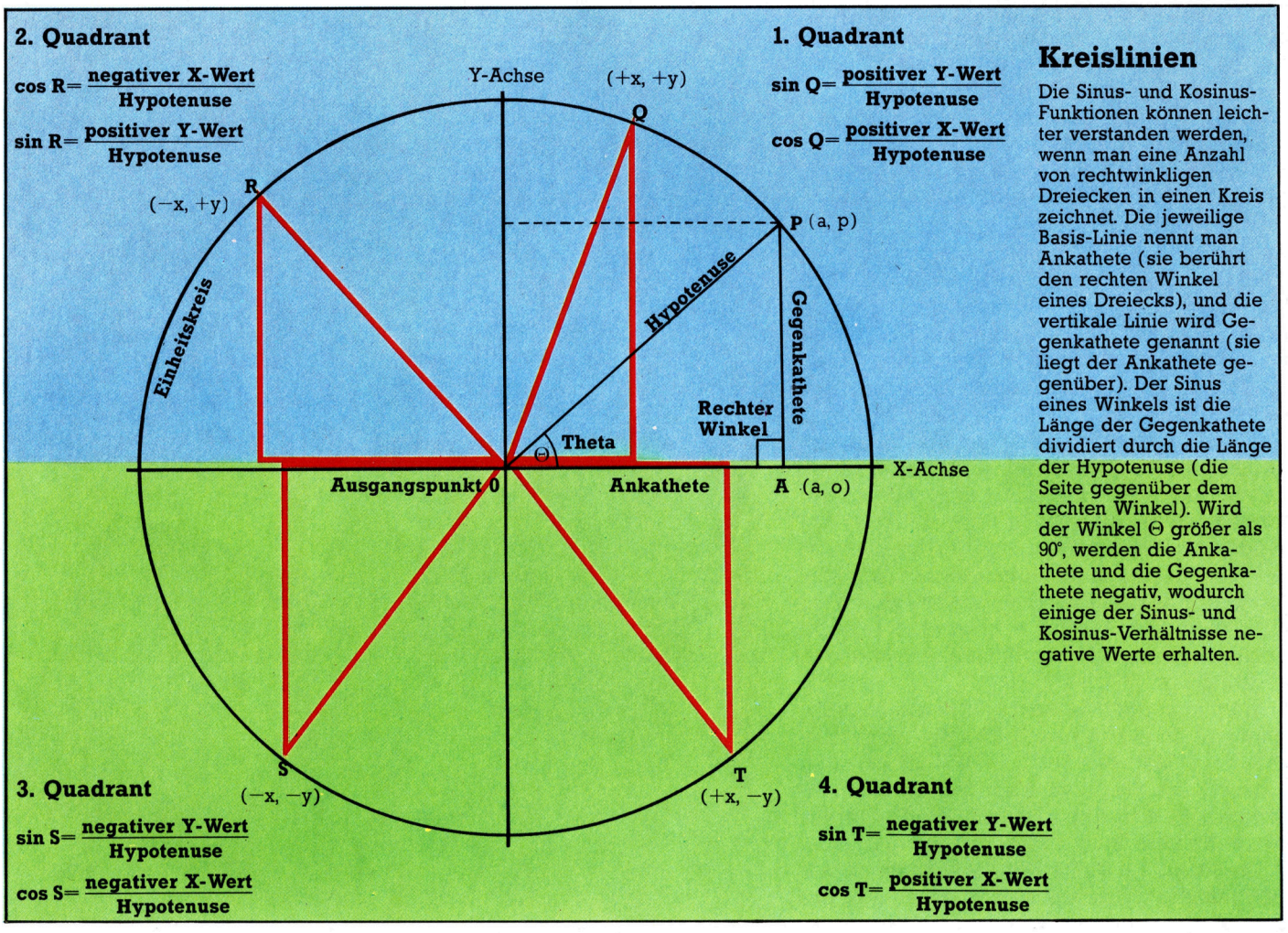
Oft ist es nicht zu vermeiden, mathematische Ausdrücke in den Programmen zu verwenden. Hier werden die Winkelfunktionen Sinus und Kosinus erklärt.

Welche Art der Mathematik müssen Programmierer kennen? Das hängt von der Art des Programms ab, das sie schreiben wollen. Die BASIC-Versionen, die in den meisten Heimcomputern integriert sind, umfassen viele Anweisungen und Funktionen für den Einsatz von Bildschirm-Grafiken – PLOT, CIRCLE, FILL, LINE, COLOUR, INK, PAPER usw. –, so daß beim Verschieben und Bewegen einfacher Figuren auf dem Bildschirm keine Probleme entstehen. Das ist meist sogar dann einfach, wenn trigonometrische Funktionen wie COS, SIN und TAN gebraucht werden, vorausgesetzt, Ihre BASIC-Version ist mit diesen Funktionen ausgerüstet. Wenn Sie die hier verwendeten Ausdrücke zunächst nicht verstehen sollten, haben Sie keine Angst – wir werden alles erklären.

Wie sieht es bei BASIC aus, wenn man statistische Funktionen braucht? Die Antwort lautet: sehr schlecht! Die meisten Versionen dieser Sprache verfügen nicht über eingebaute statistische Funktionen, die bei der Manipulation von Daten helfen könnten. Wenn Ihr Programm vorhersagen soll, wer die nächste Wahl gewinnt oder ob blauäugige Kinder im Examen besser abschneiden, müssen Sie die entsprechenden Funktionen selbst programmieren.

Unterstützung für BASIC

Auch wenn Sie Spiele oder Schreibmaschinenkurse schreiben, wobei es darauf ankommt, die Antwortzeit des Benutzers genau zu erfassen, lassen Sie die meisten BASIC-Versionen im Stich. Sie unterstützen den Program-



Kreislinien

Die Sinus- und Kosinus-Funktionen können leichter verstanden werden, wenn man eine Anzahl von rechtwinkligen Dreiecken in einen Kreis zeichnet. Die jeweilige Basis-Linie nennt man Ankathete (sie berührt den rechten Winkel eines Dreiecks), und die vertikale Linie wird Gegenkathete genannt (sie liegt der Ankathete gegenüber). Der Sinus eines Winkels ist die Länge der Gegenkathete dividiert durch die Länge der Hypotenuse (die Seite gegenüber dem rechten Winkel). Wird der Winkel Θ größer als 90°, werden die Ankathete und die Gegenkathete negativ, wodurch einige der Sinus- und Kosinus-Verhältnisse negative Werte erhalten.

mierer nicht einmal mit den einfachsten Zeitfunktionen. Und das sind dann auch die drei Hauptgebiete, mit denen wir uns in den folgenden Artikeln beschäftigen wollen – Trigonometrie, Statistik und Zeitmessung.

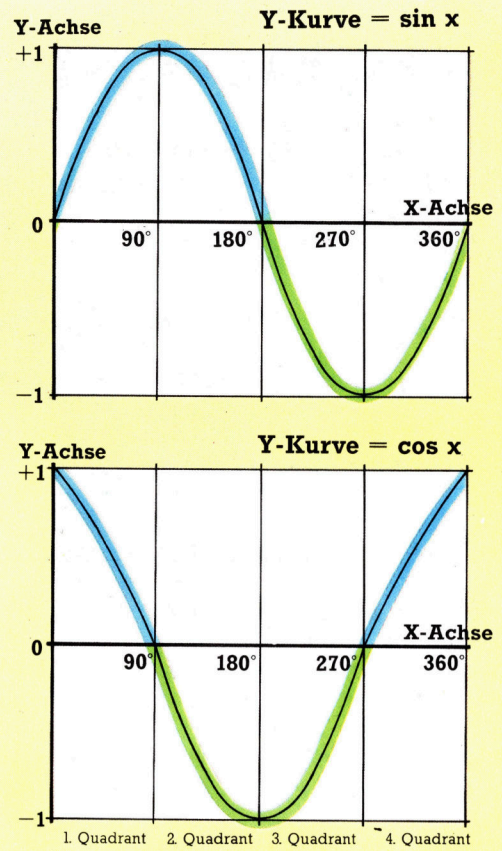
Mathematikschüler fragen sich oft, welche Bedeutung die Trigonometrie für die wirkliche Welt hat. Die Trigonometrie stellt die Verbindung zwischen der euklidschen Geometrie, die sich mit der Manipulation von Punkten, Linien und Kurven beschäftigt, und der Algebra, die mathematische Lösungen durch die Manipulation von Variablen mit unbekannten Werten ermöglicht, dar. Nehmen Sie als Beispiel eine Parabel. Ihre besonderen Eigenschaften lassen sich mit Hilfe von Millimeterpapier, Winkelmesser, Lineal und Bleistift herausfinden. Trotzdem ist es natürlich einfacher, die Kurve durch die algebraische Formel $y=x^2$ auszudrücken.

Diese einfache Formel gestattet, Werte für jeden beliebigen Punkt auf der Kurve zu berechnen, ohne daß man sie zeichnen müßte. Probleme, die algebraisch gelöst werden können, sind auch auf einem Computer erheblich einfacher zu lösen als solche, die gezeichnete Grafiken oder Figuren erfordern.

Kosinus und Sinus sind zwei Wege, das Verhältnis der Seiten eines rechtwinkligen Dreiekkes zueinander darzustellen. Jedes rechtwinklige Dreieck kann so gezeichnet werden, daß es genau in einen Kreis, genannt der Einheitskreis, hineinpaßt. Die Bezeichnung Einheitskreis resultiert daraus, daß er einen Radius von „1 Einheit" hat – das wirkliche Maß spielt dabei keine Rolle. Unser Bild zeigt eine Linie, die um 35° gedreht wurde. Der Startpunkt für eine Drehung ist vereinbarungsgemäß die horizontale Achse, und die Drehung erfolgt gegen den Uhrzeigersinn. Die horizontale Achse nennt man die X-Achse. Der Winkel einer Drehung heißt Theta (Θ). Wenn eine Linie vom Kreisumfang zur X-Achse gezogen wird, erhalten Sie ein rechtwinkliges Dreieck.

Gleichbleibende Verhältnisse

Der Kosinus von Θ ist definiert als das Verhältnis der Länge der am Winkel anliegenden Seite eines Dreiecks (der Teil parallel zur X-Achse, genannt die Ankathete) zur Hypotenuse (der Radius des Einheitskreises). Wenn wir beispielsweise einen Kreis mit einem Radius von 15 mm nehmen und die am Winkel anliegende Seite des Dreiecks messen, erhalten wir als Ergebnis annähernd 12,28728 mm. Dividiert man diesen Wert durch 15, so erhält man als Ergebnis den Wert 0,819152 und somit den Kosinus von 35°. Dieses Verhältnis bleibt bestehen, ganz gleich, welche Größe der Einheitskreis und das in ihn gezeichnete Dreieck haben mögen. Ob der Radius des Kreises nun einen Zentimeter, einen Kilometer oder ein Lichtjahr beträgt, die Seite des Dreiecks, die an der X-Achse liegt, wird immer ungefähr 0,82 des Radius-Wertes betragen.



Eine Kurve

Die bekannte Sinus-Kurve wird durch die grafische Darstellung der Sinus-Funktion eines kompletten Kreises erzeugt. Entlang der X-Achse reichen die Werte der Winkel von 0° bis 360°, und die Y-Achse repräsentiert den Bereich der Sinus-Werte für diese Winkel. Beachten Sie, daß alle Sinus-Werte in einem Bereich von plus 1 bis minus 1 liegen. Die vier Abschnitte des Diagramms entsprechen den vier Quadranten des in der anderen Abbildung gezeigten Kreises. Der Sinus ist positiv (blaue Linie) für die ersten beiden Quadranten und negativ (grüne Linie) für die anderen Quadranten. Die Kosinus-Kurve hat dieselbe Form wie die Sinus-Kurve, ist jedoch auf der X-Achse um 90° verschoben.

Wir könnten ebenso andere Werte von Θ in den hier gezeigten Einheitskreis einzeichnen. Sie werden feststellen, daß für jeden Wert von Θ der Wert von cos Θ niemals größer als 1 oder kleiner als 0 sein wird. Für Werte von Θ größer als 90° wird cos Θ allerdings einen negativen Wert annehmen. Dies kommt daher, daß cos Θ den Wert der X-Koordinate beeinflußt. Mathematische Definitionen legen fest, daß der Ursprungspunkt einer Drehung bei 0 auf der X-Achse liegen muß. Alle Punkte links dieser Achse haben somit negative Werte. Aus demselben Grund ist der Kosinus von Winkeln zwischen 180° und 270° ebenfalls negativ, wogegen der Kosinus von Winkeln größer als 270° bis zu 360° wieder positive Werte hat.

Die Sinus-Funktion eines Winkels ist der des Kosinus sehr ähnlich, allerdings werden dabei Werte auf der Y-Achse beeinflußt. Wenn Θ den Wert 0 hat, so ist die anliegende Seite des Dreiecks gleich der Länge der Hypotenuse. Die Koordinate von P auf der X-Achse wird immer 1 (da $1/1 = 1$), die Koordinate auf der Y-Achse dagegen 0 sein. Für alle Werte von Θ bis 90° liegt der sin Θ in einem Bereich von 0 bis 1. Im zweiten Quadranten des Kreises ist sin Θ ebenfalls positiv, nimmt jedoch abfallende Werte von 1 bis 0 an, entsprechend der Annäherung von Θ zu 180°. Alle Werte von Θ größer als 180° bis zu (nicht einschließlich) 360° sind dann negativ.

Klein und leistungsstark

Obwohl sich Apple mit der Einführung des Macintosh bereits einen festen Marktanteil gesichert hat, poliert die Firma auch die Modelle der Apple-II-Reihe auf. In diesem Artikel stellen wir Ihnen den Apple IIc näher vor.

Der Erfolg des Macintosh und wachsende Konkurrenz haben die Zukunft der Microcomputerreihe des Apple II in Frage gestellt. Trotz der Beteuerungen von Apple, daß sie dem 6502 und seiner großen Anwendergemeinde treu bleiben werden, sehen viele Händler und Marktstrategen das baldige Ende dieser Modellreihe voraus. Apple hat deshalb nicht nur die Soft- und Hardware der bestehenden Apple-II-Modelle verbessert, sondern auch den Apple IIc auf den Markt gebracht. Diese Produkte sollen die Langlebigkeit des Apple II noch weitere drei Jahre sichern.

Die Apple-Computer II, II+ und IIe haben den Markt für Personal Computer mitgeprägt, wobei der jahrelange Verkaufserfolg dieser Serie der Firma über eine Milliarde Dollar Umsatz einbrachte. Doch obwohl weltweit mehr als zwei Millionen Apple-Computer verkauft wurden, konnte der Apple II wegen verfehlter Preis- und Marktpolitik außerhalb der USA nicht den gleichen Marktanteil erreichen. Mit einem Preis von circa 6000 Mark (einschließlich Monitor und Diskettenlaufwerk) lag die Maschine weit über der Schallgrenze für Heimcomputer. Dennoch hält die vergleichsweise kleine Gruppe der europäischen Apple-Anwender dem Apple II unerschütterlich die Treue.

Vor etwa einem Jahr wurde eine weitere Variante des Apple II vorgestellt, der IIc (das „c" bedeutet „compact"). Obwohl er um die Hälfte kleiner ist als seine Vorgänger, verfügt er über ein eingebautes Laufwerk für 5 1/4-Zoll-Disketten. Mit 3,4 Kilo und einem versenkbaren Griff eignet sich der IIc ausgezeichnet für den Transport. Der Tragegriff läßt sich für einen bequemeren Arbeitswinkel nach unten klappen. Schräggestellt ermöglicht er durch eine bessere Luftzirkulation außerdem die Überhitzung der Maschine.

Im Gegensatz zu seinen Vorgängern ist der IIc ein geschlossenes System ohne Erweiterungssteckleisten. Die Firma entschied sich, die wichtigsten „Erweiterungen" gleich in die Maschine einzubauen. Darunter sind Anschlüsse für Monitor und Fernseher, eine Joystickbuchse, die auch für den Betrieb einer Maus geeignet ist, Modem- und Druckerausgänge, eine Audiobuchse und ein Steckkontakt für ein zweites Diskettenlaufwerk. Die Schnittstellen sind durch symbolische Abbildungen ihrer Funktionen gekennzeichnet. Der IIc verfügt außerdem über eine 80-Zeichen-Darstellung und einen Arbeitsspeicher mit 128 KByte. Die meisten dieser Fähigkeiten sind auf dem IIe nicht vorhanden und lassen sich nur durch Erweiterungskarten erreichen.

Die QWERTZ-Tastatur des Apple IIc ähnelt der des IIe. Die Reset-Taste wurde jedoch auf eine Position links oberhalb des Tastenfeldes verlegt. Neben ihr befinden sich zwei kleine Schalter. Mit dem linken läßt sich die Bildschirmdarstellung von 40 auf 80 Zeichen umstellen. Mit dem zweiten Schalter kann man zwischen dem europäischen Zeichensatz und den amerikanischen Zeichen der Tastatur

Der Apple IIc ist eine verbesserte und tragbare Version des Apple II. Der IIc besitzt 128 K RAM, eine 80-Zeichen-Darstellung, eine Reihe von Schnittstellen und eine eingebaute Diskettenstation. Das Bild zeigt ihn mit dem ebenfalls von Apple angebotenen Bildschirm mit grüner Phosphoranzeige.

wählen. Da einige Symbole nur in einem der beiden Zeichensätze vorhanden sind (das „#" wird zum Beispiel im europäischen Zeichensatz durch „£" ersetzt), ist diese Möglichkeit sehr praktisch. Die Lichter rechts oberhalb der Tastatur zeigen an, ob der Strom eingeschaltet ist und ob das Diskettenlaufwerk angesprochen wird.

Das Einschalten des IIc aktiviert automatisch das Diskettenlaufwerk. Wenn keine Diskette eingelegt ist und die Reset-Taste zusammen mit der Control-Taste betätigt wird, lädt der IIc das im ROM gespeicherte Applesoft-BASIC. Applesoft-BASIC weicht nur geringfügig von dem Standard des Microsoft-BASIC ab. Allerdings fehlen bei dieser BASIC-Version Anweisungen, die die strukturierte Programmierung unterstützen. So besitzt Applesoft keinen RANDOMIZE-Befehl, keine automatische Zeilennumerierung, keine IF. . .THEN. . .ELSE-Struktur und kein WHILE. Auch fehlen in der grafischen Darstellung Möglichkeiten wie CIRCLE und PAINT.

Hierarchischer Dateiaufbau

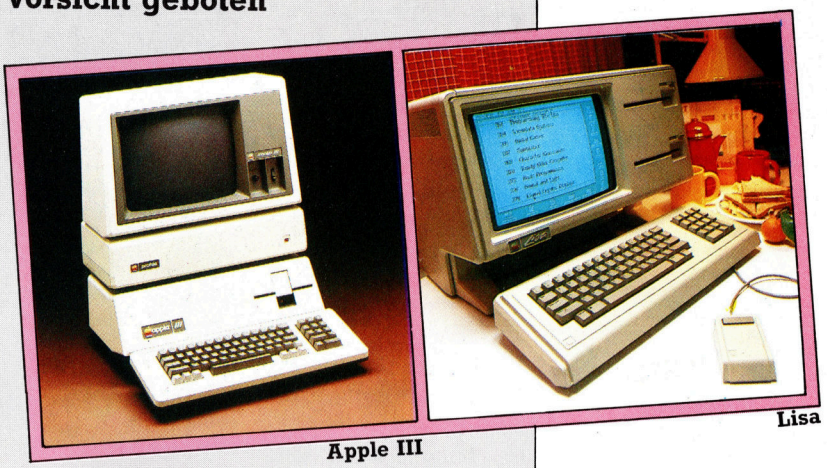
Wenn sich eine Diskette im Laufwerk befindet, arbeitet der IIc entweder mit DOS 3.3 (das Diskettensystem des II+ und des IIe) oder PRODOS, dem neuen Betriebssystem von Apple. PRODOS ist eine Weiterentwicklung des Betriebssystems, das Apple auf dem Apple III, seinem ersten kommerziellen Computer, einsetzte. Es verfügt über einen hierarchischen Dateiaufbau (Baumstruktur), bei dem Dateien wie in einem Karteischrank abgelegt werden. Dabei sind alle Dateien, die sich zum Beispiel auf ein Projekt namens ZED beziehen, auf der Diskette unter der Kennung ZED gespeichert. Dateien von ZED, die sich auf den finanziellen Ablauf beziehen (beispielsweise Kosten, Verkauf, Umsatz), werden in der Untergruppe „Finanzen" gesammelt. Wenn ein manuelles Dateisystem die Datei „Verkauf" ansprechen wollte, müßte zunächst die Hauptdatei ZED eröffnet werden, dann die Datei „Finanzen" und erst dann wäre der Zugang zu der Datei „Verkauf" frei. Bei PRODOS wird der Zugriffsweg über die hierarchischen Dateinamen gesteuert. Der eben beschriebene Vorgang enthält dann die Namen der einzelnen Dateien, die hierarchisch geordnet und durch Querstriche getrennt sind:

```
/ZED/FINANZEN/VERKAUF/
```

Die Angaben der Zugriffswege können bis zu 64 Zeichen enthalten. Baumstrukturen wie PRODOS werden auch von MS/DOS eingesetzt, das unter anderem den IBM PC steuert.

Die Bildschirmdarstellung des IIc ist weitaus besser als die seiner Vorgänger. Neben den beiden Textdarstellungsmodi (24 Zeilen mit je 40 oder 80 Zeichen) verfügt der IIc über 16 Far-

Vorsicht geboten



Apple III

Lisa

Nachdem Apple die Bedeutung des Marktes für kommerzielle Computer erkannt hatte, entwickelte die Firma den Apple III, ein ausgezeichnetes Einplatzgerät mit zwei 6502-Prozessoren und einem Arbeitsspeicher von 512 K. Mit seinem Betriebssystem SOS (Sophisticated Operating System) konnte der Apple III Festplatten ansprechen und Dateien hierarchisch speichern. Die dabei eingesetzte Baumstruktur bildete das Grundkonzept für das Betriebssystem des Lisa, wobei viele seiner Merkmale auch im MS/DOS des IBM PC auftauchen. Unglücklicherweise erhielt der III durch eine Reihe von Hardwareproblemen unmittelbar nach seiner Vorstellung einen schlechten Ruf. Obwohl Apple alle fehlerhaften Maschinen austauschte, erholte sich der III nie von der schlechten Presse. Auch konnte der Eindruck nicht beseitigt werden, daß sein Betriebssystem zu kompliziert war.

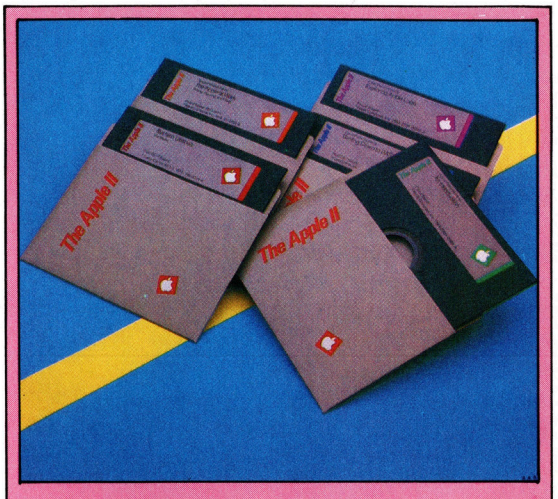
Der 1983 vorgestellte Lisa war der erste Personal Computer mit integrierter Fenstertechnik, dessen Betriebssystem mit Symbolen statt mit Worten arbeitete und der sich über eine Maus steuern ließ. Die durch die Einführung des Lisa geweckten Erwartungen ließen die Börsennotierung der Appleaktien innerhalb weniger Monate von 24 $ auf 60 $ pro Anteil hochschnellen. Wegen der hohen Entwicklungskosten von 50 Millionen Dollar wurde für den Lisa jedoch ein zu hoher Preis angesetzt. Außerdem war er auf eine falsche Käufergruppe ausgerichtet. Ursprünglich für das höhere Management großer Firmen gedacht, gingen die meisten Lisas jedoch an kleine Spezialfirmen hauptsächlich im Bereich von Werbung und Grafik. Da in der Folge die Verkaufszahlen des Lisa nicht den Erwartungen entsprachen, fielen die Appleaktien – wiederum in nur wenigen Monaten – auf 17 $ pro Anteil. Unter der Leitung von John Scully, der von Pepsi Cola zu Apple überwechselte, scheint die kreative Computerfirma jedoch aus der Vergangenheit gelernt zu haben. Der Macintosh ist ein weltweiter Verkaufsschlager, der die Entwicklungskosten des Lisa und des Mac wieder einspielt. Der Lisa wurde kürzlich durch den billigeren Lisa II ersetzt, und der IIc scheint dem Zugpferd Apple II neues Leben zu geben.

ben und drei Grafikarten: 40 × 40 (niedrige Auflösung), 280 × 192 (hohe Auflösung) und 560 × 192 (Ultrahochauflösung). Für die Bildschirmausgabe bietet Apple einen grünen Phosphormonitor und einen flachen LCD-Schirm (24 Zeilen mit je 80 Zeichen) an. Mit dem angekündigten Batteriepack wäre der IIc dann ein vollwertiger tragbarer Computer.

Der größte Vorteil des IIc ist die umfangreiche Programmbibliothek, die für die Apple-II-Modelle zur Verfügung steht. Diese Softwarepalette enthält einige der besten Anwendungsprogramme der Welt. Es gibt eine Vielfalt von Textsystemen, Kalkulationsprogrammen, Datenbanken, Finanzbuchhaltungen, grafischen Darstellungssystemen, wissenschaftlichen Steuermodulen für Laboratorien und Lehrprogrammen, die vom Lesenlernen bis zur Differenzialrechnung reichen.

Außer der existierenden Software des II und IIe stellte Apple speziell auf dem IIc ein Programm namens Appleworks vor, das eine integrierte Textverarbeitung, ein Kalkulationssystem und eine Datenbank mit Fenstertechnik enthält. Appleworks ist ein ausgereiftes, einfach zu bedienendes Programmpaket. Im Lieferumfang des IIc ist eine Diskette enthalten, die den Programm-Bestseller Appleworks vorstellt, aber keine funktionierende Kopie des Programms enthält. Es wird dabei vorausgesetzt, daß der Anwender dieses Programm für 1000 Mark nachkaufen wird. Andere mitgelieferte Disketten enthalten eine ähnlich geartete Vorstellung des Apple-LOGO und BASIC, die interaktive Einführung des Systems („Apple Presents Apple") und eine Diskette mit Hilfsprogrammen für PRODOS. Weiterhin gibt es „MousePaint", ein Maus-gesteuertes Zeichenprogramm auf der Grundlage von MacPaint. MousePaint wird zur Maus mitgeliefert.

Zum Lieferumfang des IIc gehören eine Broschüre, die die Installation des Systems erklärt,

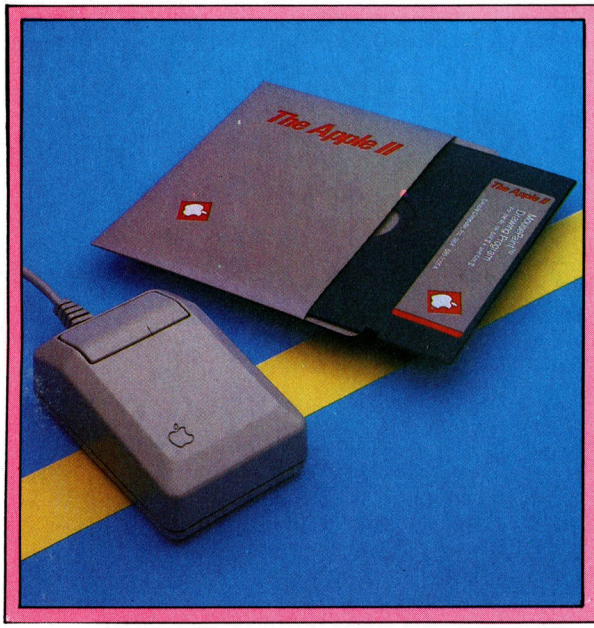


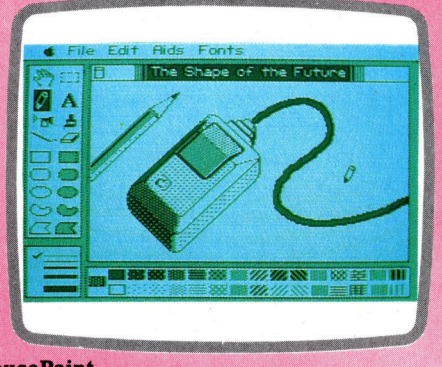
Diskettenpaket
Im Lieferumfang des Apple IIc sind fünf Disketten enthalten. Vier davon geben Einführungen in die Funktionen der Maschine, in die BASIC-Programmierung und in Anwendungsprogramme. Die fünfte Diskette enthält PRODOS.

und ein Benutzerhandbuch mit 142 Seiten, das kurz und klar die Systemvorgänge und den Inhalt der fünf mitgelieferten Disketten beschreibt. Die gut aufgebauten und bunt illustrierten Handbücher sind für den Erstanwender bestimmt.

Auch das Design des Apple IIc ist attraktiv. Apple verzichtete auf das beige Plastik des II, Macintosh und Lisa und entschied sich für ein strahlendes Weiß. Die „Ralleystreifen" im Deckel fördern die Luftzirkulation in der Maschine und schützen vor Überhitzung.

Wie seine Vorgänger ist der Apple IIc ausgezeichnet als Einzelgerät für den Büroeinsatz geeignet. Mit dem LCD-Schirm und dem angekündigten Batteriepack weist er alle Merkmale eines zuverlässigen, tragbaren Arbeitscomputers auf. Mit einem niedrigeren Preis wäre das Gerät auch als Heimcomputer attraktiv.





MousePaint
Zum Zubehör der Maus für den II+, IIe und IIc gehört MousePaint, eine einfachere Version von MacPaint. Mit MousePaint läßt sich die Maus problemlos für Zeichnungen einsetzen. Das Programm ist speziell auf die hohe grafische Auflösung der Bildschirmdarstellung des IIc ausgerichtet. Auf den anderen Computern der Apple-II-Reihe läßt sich die Maus nur mit einer Zusatzplatine einsetzen.

Ausgang für Farbfernseher/Monitor

RS232-Druckeranschluß

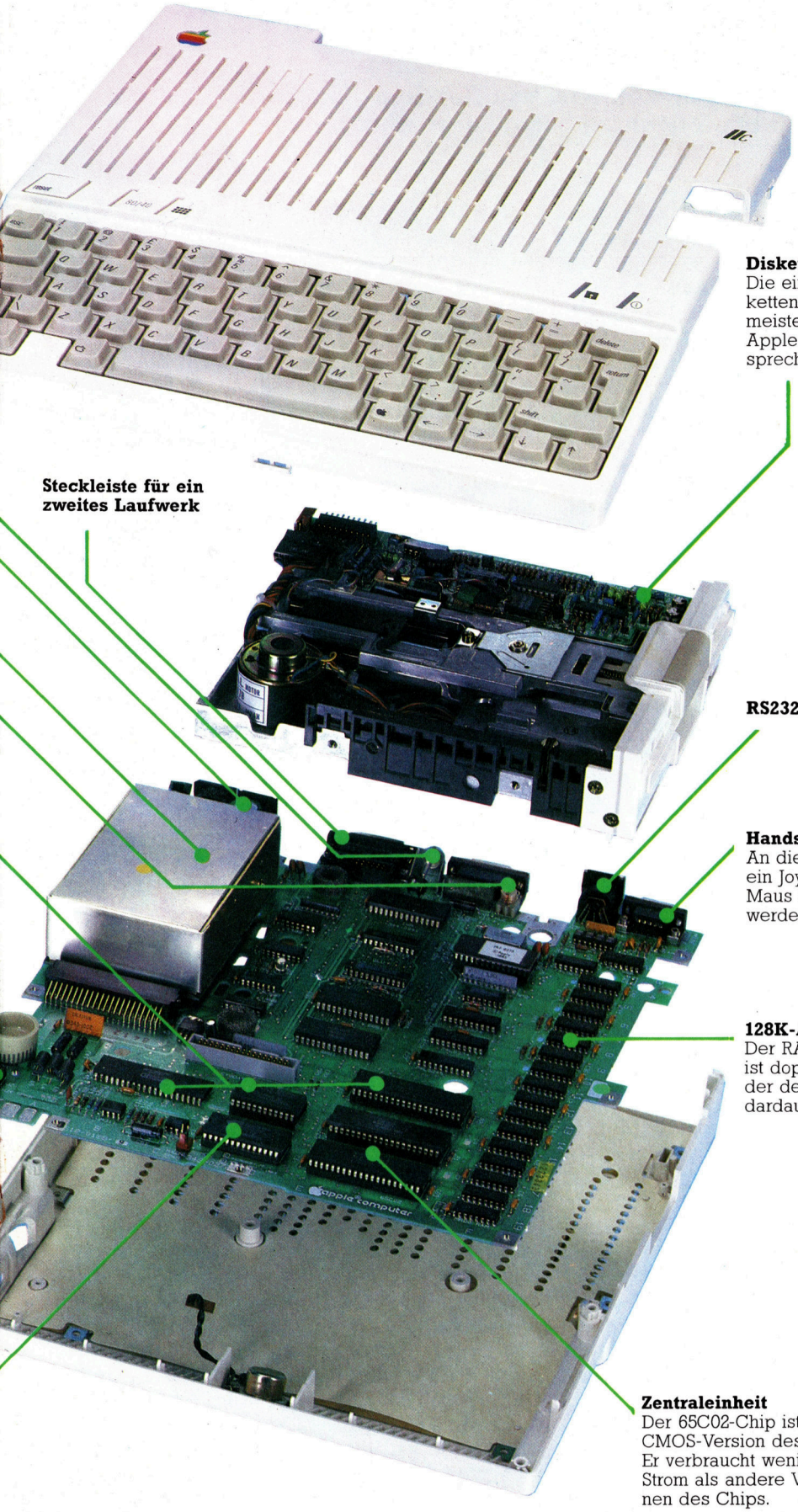
Stromversorgung
Der IIc hat ein eingebautes Netzgerät für 12 Volt, benötigt aber außerdem ein Teil für die Umformung der Netzspannung.

RGB-Ausgang
Mit einem kleinen PAL-Adapter läßt sich der IIc an einen normalen Fernseher anschließen.

Ein-/Ausgabe-Steuerung
Diese Chips steuern die Tastatur und die Anschlüsse für Ein- und Ausgabe.

Audio-Buchse
Hier kann der IIc an einen HiFi-Verstärker angeschlossen werden.

ROM
Das ROM enthält das Applesoft-BASIC und die Verwaltungsroutinen.



Apple IIc

ABMESSUNGEN

305 × 292 × 64 mm

ZENTRALEINHEIT

65C02 mit 1 MHz.

SPEICHERKAPAZITÄT

128K RAM, 16K ROM.

BILDSCHIRM-DARSTELLUNG

24 Zeilen mit entweder 40 oder 80 Zeichen pro Zeile. Drei Grafikarten mit einer maximalen Auflösung von 560 × 192 Pixeln und 16 Farben.

SCHNITTSTELLEN

Joystickbuchse mit 9 Kontakten, an die auch eine Maus angeschlossen werden kann; RS232-Modemausgang; RGB (oder PAL-TV)-Ausgang; Composite-Video-Ausgang; Anschlußmöglichkeit für ein externes Laufwerk; RS232-Druckerschnittstelle.

PROGRAMMIERSPRACHEN

Applesoft-BASIC im ROM, LOGO, PASCAL, FORTRAN.

TASTATUR

63 Schreibmaschinentasten mit vier Cursortasten, internationale Zeichensätze.

HANDBÜCHER

Ein farbenfrohes und leicht verständliches Anwenderhandbuch ist im Lieferumfang enthalten. Das Handbuch ist auf den Erstanwender ausgerichtet. Zwei Broschüren geben Hilfestellung beim Aufbauen des IIc und bei dem Einsatz der Hilfsprogramme.

STÄRKEN

Der IIc profitiert hauptsächlich von seiner Kompatibilität mit dem Apple II. Dem Anwender stehen zahlreiche Programme zur Verfügung, die auf anderen Modellen dieser Reihe erstellt wurden.

SCHWÄCHEN

Das Applesoft-BASIC wurde in den letzten sechs Jahren nicht wesentlich verändert. Ihm fehlt Flexibilität. Der Preis des IIc ist für viele Besitzer von Heimcomputern zu hoch.

Schlüsselposition

In diesem Artikel untersuchen wir, wie Random-Access-Dateien mit Indizes und dem Hash-Verfahren organisiert werden können.

Wer je mit Random-Access-Dateien gearbeitet hat, weiß, wie einfach sich mit dieser Technik Dateien anlegen und abrufen lassen. Jeder Datensatz kann direkt angesprochen werden. Jedoch sind die Methoden zum Einfügen und Löschen von Datensätzen, die in der vorigen Folge erwähnt wurden, nicht sehr praktisch. Ein Index trägt dazu bei, daß die Aufgabe viel einfacher erledigt werden kann.

Beim Aufbau eines Indexes wird ein bestimmtes Feld der Datensatzstruktur als Schlüssel definiert. Jeder Schlüssel bezieht sich dabei auf einen bestimmten Datensatz. Ein Index besteht daher aus dem Wert des Schlüsselfelds und der Nummer des dem Schlüssel zugeordneten Datensatzes.

Damit Indizes schnell zugänglich sind, werden sie normalerweise im RAM untergebracht. Das Anlegen eines Indexes ist einfach: Eine Routine liest alle Datensätze einer Datei und schreibt den Inhalt der Schlüsselfelder in eine Matrix, worin sie dann sortiert werden. Da der Lesevorgang jedoch viel Zeit erfordert, werden Indexdateien häufig, zusätzlich zu den Hauptdateien, auf Disketten gespeichert. Dabei lassen sich beliebig viele Indexdateien für eine beliebige Anzahl Schlüsselfelder anlegen, sortieren und speichern. Der Zugriff auf die Hauptdatei kann nach den unterschiedlichsten Kriterien erfolgen: zum Beispiel alphabetisch (A–Z oder Z–A), nach Datum etc.

Die Indexdatei wird vollständig in den Arbeitsspeicher eingelesen und nur dann zurückgeschrieben, wenn Datensätze verändert oder hinzugefügt wurden. Da die Daten hierfür nur in der gespeicherten Reihenfolge benötigt werden, sind sequentielle Dateien für diese Aufgabe ausgezeichnet geeignet.

Markierung gelöschter Daten

Sollen einzelne Datensätze aus einer indizierten Datei gelöscht werden, brauchen sie nur als gelöscht markiert zu werden, beispielsweise mit einem Stern am Anfang des ersten Feldes. Bei einer anderen Technik wird die Nummer des Datensatzes auf einen bestimmten Wert (beispielsweise −1) gesetzt. Dieser Wert zeigt ebenfalls an, daß der Satz gelöscht ist. Dieser kann wieder aktiviert werden, solange er nicht überschrieben wurde.

Doch welche Methode auch gewählt wird, es muß festgehalten werden, welche Datensätze gelöscht sind. Neue Datensätze können dann gelöschte Sätze überschreiben. Der ursprüngliche Indexeintrag muß dabei gegen den neuen ausgetauscht und der Index sortiert werden, damit sich der neue Datensatz an der korrekten Position innerhalb der umfangreichen Datei befindet.

Index- und Hauptdateien sollten öfter neu geordnet werden. Da bei dieser Speichermethode die Datensätze nicht in einer bestimmten Reihenfolge abgelegt sind, enthält die Datei nach einiger Zeit viele Leersätze. Das Dateisystem arbeitet zwar ohne Fehler, die Zugriffszeit wird jedoch langsamer. Eine „Reorganisationsroutine" bringt die Datensätze in eine praktische Ordnung und beseitigt alle gelöschten Sätze.

Der Hashing-Algorithmus

Ein Index ist aber nicht die einzige Methode, in großen Dateien einzelne Datensätze schnell anzusprechen. Das „Hash-Verfahren" (oder „Hashing") eignet sich besonders für extrem große Dateien. Es wird hauptsächlich auf Systemen mit Festplatten oder großen Diskettenkapazitäten eingesetzt. Viele Betriebssysteme und Programme verwenden internes Hashing, um größere Verarbeitungsgeschwindigkeiten zu erzielen.

Hashing ersetzt den Index durch eine Formel oder einen Hashing-Algorithmus. Die Formel errechnet eine Satznummer („Hash") aus dem Wert des Schlüsselfeldes. Der entsprechende Datensatz wird dann auf dieser Dateiposition gespeichert. Enthält ein Schlüsselfeld zum Beispiel ein Datum, dann multipliziert der Algorithmus die Monatszahl mit den letzten zwei Ziffern der Jahreszahl und addiert das Ergebnis zu der Tageszahl; Namensfelder können über die ASCII-Codes der einzelnen Buchstaben gehashed werden.

Hashing unterscheidet sich erheblich von Indexsystemen. Denn so ist nur ein Schlüsselfeld (und ein Hashing-Algorithmus) je Datei möglich, das beim Beginn des Dateiaufbaus eingesetzt werden muß. Ein Indexsystem kann jedoch eine beliebige Anzahl von Indizes haben, die jederzeit während oder nach dem Aufbau der Datei angelegt werden können.

Hashing ist zwar weniger flexibel als ein Index, aber weitaus schneller. Um einen bestimmten Datensatz zu finden, nimmt das Programm den Schlüssel, wendet die Hash-Formel an und greift auf den errechneten Satz zu.

Es wird daher keine Zeit für das Durchsuchen (und den Aufbau) eines Indexes benötigt.

Beim Hashing gibt es nur dann ein Problem, wenn zwei Datensätze den gleichen Hash-Code ergeben und daher die gleiche Position in der Datei einnehmen sollen. Um dies zu vermeiden, wird bei dem Entwurf von Hashing-Algorithmen große Sorgfalt darauf verwandt, keinen Fall zuzulassen (außer bei völlig identischen Schlüsseln), bei dem zwei gleiche Hash-Codes entstehen können. Auch befinden sich zwischen den einzelnen Datensätzen große Zwischenräume, so daß zwei scheinbar nebeneinander liegende Hashes oft durch fünf oder mehr unbenutzte Datensätze deutlich getrennt werden.

Der Ablauf bei einem Hashing-System sieht folgendermaßen aus: Beim Speichern eines Datensatzes wird aus dem Schlüssel eine Satznummer errechnet. Wenn dieser Platz bereits besetzt ist, sucht das System den nächsten Datensatz. Diese sequentielle Suche kann sich auf die nächsten fünf Datensätze ausdehnen, die mit diesem bestimmten Hash angesprochen werden. Beim Abruf eines Datensatzes wird der Schlüssel gehashed und die angesprochene Gruppe von Datensätzen sequentiell nach dem exakten Schlüssel durchsucht. Auf den ersten Blick scheint die sequentielle Suche den Geschwindigkeitsvorteil wieder aufzuheben. Bei großen Dateien verringert diese Methode jedoch die Anzahl der Datensätze, die durchsucht werden müssen, von beispielsweise dreitausend auf fünf oder sechs.

Datei voll?

Was geschieht, wenn alle fünf Datensätze eines bestimmten Hash-Codes belegt sind? Dieses Problem kann auf mehrere Arten gelöst werden. Die einfachste Methode ist die Anzeige: „Datei voll". Eine andere Methode schreibt Datensätze, für die kein Platz vorhanden ist, in eine Spezialdatei mit eigenem Index, die dann möglichst in die Hauptdatei integriert wird. Die meisten Hashing-Systeme umgehen dieses Problem jedoch dadurch, daß sie die Datei nur zu 80 Prozent oder weniger füllen. Hier wird ein weiteres Merkmal dieser Speichertechnik deutlich: Mit der Hash-Technik angelegte Dateien belegen mehr Platz als Systeme mit Indizes.

Auch das Löschen einzelner Datensätze läßt sich mit dem Hash-Verfahren beschleunigen. Dabei wird der Schlüssel in den Hash-Code umgewandelt, der Datensatz nach kurzer Suche angesprochen und die Position als gelöscht markiert. Der nächste neue Datensatz mit identischem Hash überschreibt dann den alten Satz.

In der letzten Folge dieser Serie werden wir uns mit den Befehlen beschäftigen, die den Aufbau und Zugriff auf serielle Cassettendateien steuern.

Zugriff per Index

Ein Index ist die einfachste Methode, auf eine Random-Access-Datei zuzugreifen. Die RAM-Liste zeigt die Werte für ein bestimmtes Schlüsselfeld und die Nummer des entsprechenden Datensatzes. Beim Zugriff auf einen Datensatz wird der Schlüssel im Index gesucht und der entsprechende Datensatz in den Speicher geladen. Gelöschte Datensätze bleiben markiert in der Datei.

Finde „David"

Schlüssel	Nr.
Andreas	1
Bach	−1
Braun	5
Christian	−1
David	7
Daxel	23
Fisch	15
Georg	28
Harms	37
Johnes	25
Klaus	11
Marks	10

Indexdatei

Hauptdatei

	Name	Tel. Büro	Tel. priv.	Beruf
1	Andreas	242 0791	727 0942	Designer
2	Philips	636 2418	221 3940	Buchhalter
3	Schmidt	631 0836	286 8170	Redakteur
4		Datensatz löschen		
5	Braun	729 8213	236 2190	Zahnarzt
6	Peter	836 6622	298 4310	Dekorateur
7	David	743 7216	450 6926	Gärtnerin
8		Datensatz löschen		
9		Datensatz löschen		
10	Marks	730 6321	429 7592	Mechaniker
11	Klaus	493 9899	455 8431	Rechtsanwalt
12	Walter	736 7700	693 0452	Friseurin

Die Hash-Technik

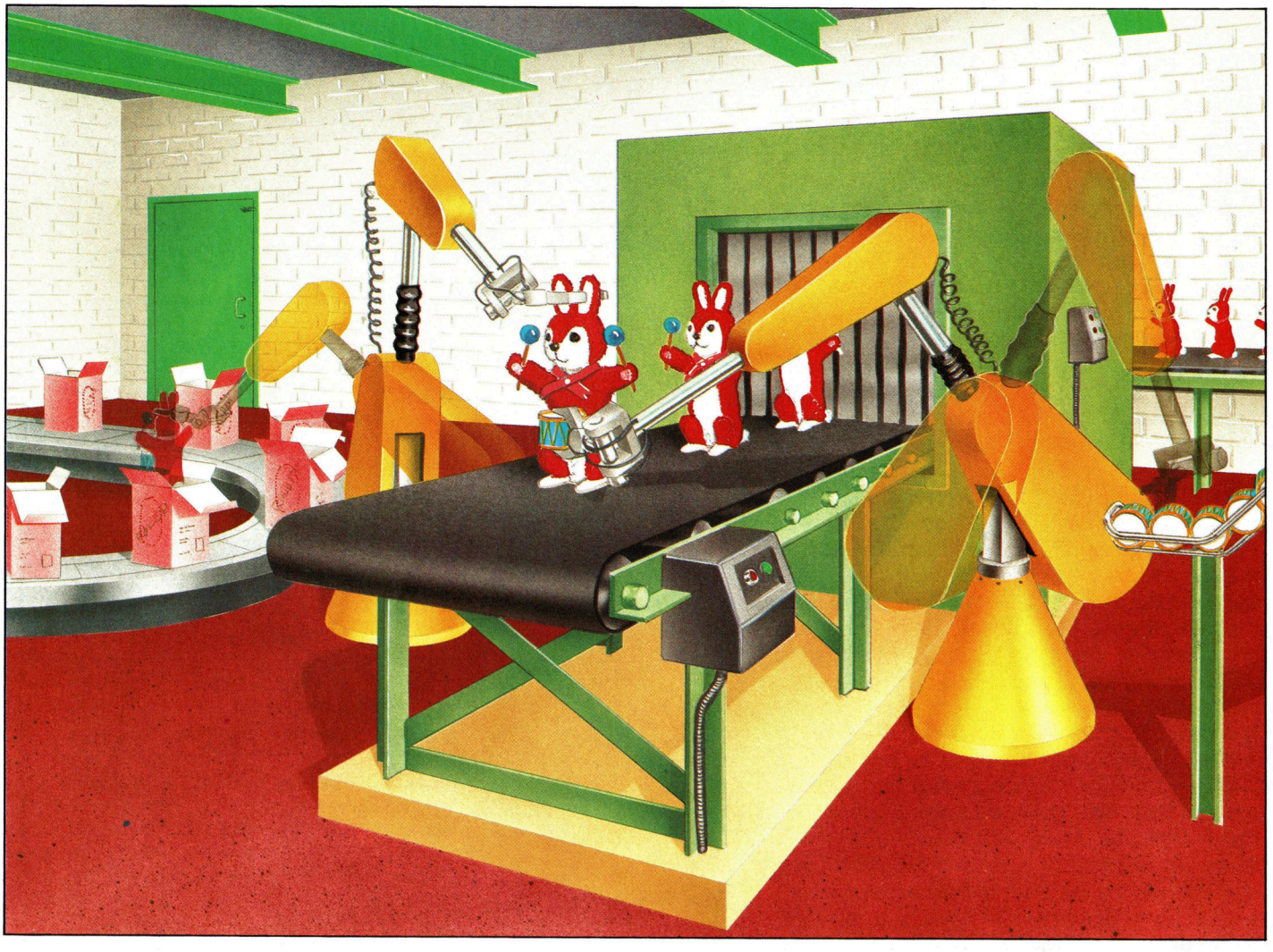
Finde „David"

Hashing-Algorithmus

Der Hashing-Algorithmus wandelt den Schlüssel um, so daß er sich auf einen bestimmten Datensatzblock bezieht. Datensätze mit identischen Hash-Codes werden als Gruppe gespeichert. Zwischen den einzelnen Blöcken befinden sich freie Datensätze, in denen neue Sätze gespeichert werden können.

Name	Tel. Büro	Tel. priv.	Beruf
Davidson	629 0491	430 0592	Klempner
Dagobert	436 2488	362 0066	Direktorin
Danzer	730 0021	626 9191	Pfleger
Dattrichs	439 9933	630 4918	Schriftsteller
David	743 7216	450 6926	Gärtner
Daxel	830 0123	340 9924	Schwester
Eckert	731 6666	458 0021	Designerin
Ebert	831 8294	450 6218	Händler

Nach dem Hash-Verfahren angelegte Daten bieten besonders bei großen Dateien schnellen Zugriff auf einzelne Datensätze. Die Technik bringt jedoch zahlreiche Einschränkungen mit sich und muß umsichtig programmiert werden. Aus dem Schlüssel wird mit einem Hashing-Algorithmus die Nummer eines Datensatzes errechnet. Jeder mögliche Hash bezieht sich dabei auf einen Block von Datensätzen, der sequentiell nach dem gewünschten Satz durchsucht wird.



Wenn Roboterarme gemeinsam eine Aufgabe zu erfüllen haben, ist eine „Choreographie" zur Steuerung erforderlich, damit sie einander nicht stören. Hier muß ein Arm das Spielzeug aufnehmen und festhalten, während der andere eine Trommel aufnimmt und befestigt. Dann befördert der erste Arm das fertige Spielzeug in eine Schachtel. Ist die Bewegung der Arme richtig aufeinander abgestimmt, können Fließbänder installiert werden, die eine Serienproduktion erlauben. Die Montage der Hasen durch Menschen würde einen ganz anderen Aufbau der Fertigungsstraße erfordern.

Perfekte Kopie

Der typische Computerbesitzer wird kaum Zugriff zu einem Roboter haben und kann deshalb die bisher geschilderten Ideen nicht umsetzen. Dagegen hat er die Möglichkeit, das Verhalten eines Roboters mit seinem Rechner zu simulieren.

Die Fortentwicklung der Computertechnologie hat zu einer steigenden Nutzung der Simulationsmöglichkeiten geführt. Man kann „Modelle" schaffen, die alles simulieren, was in der wirklichen Welt geschieht. Flugsimulatoren sind vielen Menschen bereits vertraut. Sie wissen, daß es sehr komplizierte Geräte gibt, mit denen Flugschüler Erfahrungen sammeln können, ohne ein richtiges Flugzeug steuern zu müssen. Doch auch viele andere Arbeiten lassen sich mit Hilfe der Computer-Simulation durchführen – so beispielsweise Geschäftsplanungen, Ingenieurtätigkeiten und physikalische Versuche verschiedenster Art.

In einigen Fällen dienen Computer-Modelle dazu, Experimente auszuführen, die in Wirklichkeit zu gefährlich wären. Zum Beispiel ist es lebenswichtig zu wissen, was in einem Atomreaktor geschieht, wenn er leck ist. Naheliegenderweise kann man das nicht in Wirklichkeit ausprobieren, also nutzt man die Computer-Simulation. Ist das Modell ausreichend mit allen wichtigen Einzelheiten erstellt worden, kann man genau feststellen, was passiert, wenn ein Leck vorhanden ist.

Ähnlich ist es bei der Robotik: Simulationen werden zur Konstruktion neuer Roboter benutzt. Natürlich kann man auch einfach bauen und ausprobieren, – also einen Roboter fertigstellen, ihn testen und dann entsprechende Änderungen vornehmen. Doch diese Verfahrensweise ist zeitraubend und kostspielig.

Nehmen wir als Beispiel eine Autofertigungsstraße, an der mehrere Roboter arbeiten

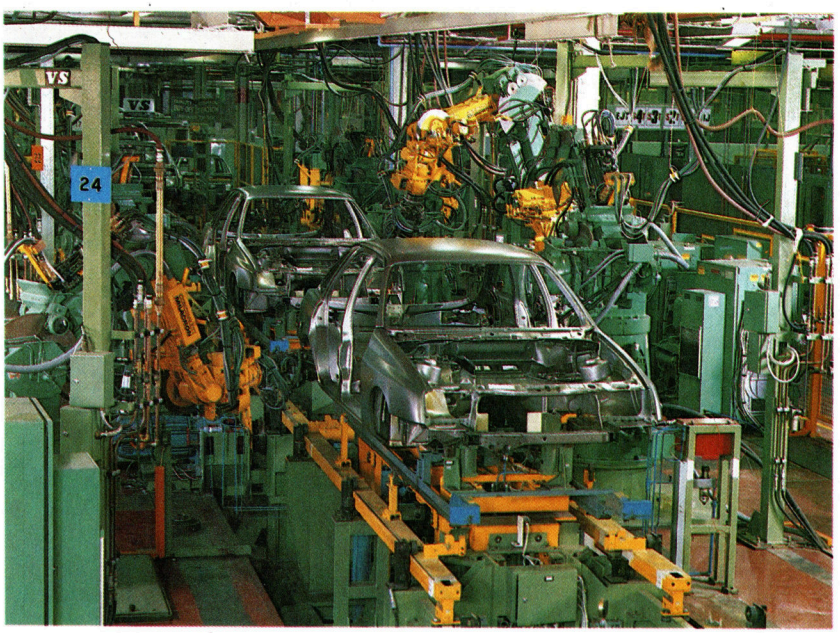
sollen. Die Roboter müssen so programmiert werden, daß sie die Autos in der richtigen Reihenfolge zusammenbauen. Das Programmieren der Roboter erfordert aber Zeit, und jede Produktionsunterbrechung kostet Geld. Man könnte eine Fertigungsstraße nachbauen, mit neuen Robotern ausstatten und so ein neues Programm entwickeln. Doch auch das ist teuer und umständlich, und es kann zu Fehlern führen – etwa bei der Roboter-„Choreographie". Wichtig ist, daß die Roboter zusammenarbeiten, ohne einander zu stören. Denn große Industrieroboter, die schwere Lasten bewegen, sind imstande, andere Roboter zu beschädigen. Und nicht nur diese könnten Schaden davontragen – ein falsch programmierter Roboter könnte beispielsweise seine Zeit damit verbringen, Autotüren zuzuschweißen, was einen späteren Verkauf des Wagens natürlich unmöglich machen würde.

Die einzig mögliche Lösung ist also eine Computer-Simulation, bei der festgelegt werden kann, wie welcher Roboter reagiert. Nach erfolgreicher Beendigung der Simulation kann das Ergebnis mit Hilfe des Programms auf die wirklichen Roboter übertragen werden.

Roboter-Simulation

Wir wollen hier das Prinzip einer Roboter-Simulation anhand eines gedachten Programms mit Namen „Roboterarm" verdeutlichen. Darin nimmt dieser Arm (mit zwei Freiheitsgraden) Gegenstände auf und stellt sie ab. Er hat keine Sensoren, muß also geführt werden. Schulter- und Ellenbogen-Gelenke sind zu kontrollieren, ferner der Greifmechanismus des End-Effektors, um Dinge aufzunehmen und abzusetzen. Es sollte die Bewegungen eines Roboters imitieren, der mit einem einfachen Berührungssensor ausgestattet ist, um sein Ziel zu erreichen. Der Roboter findet den richtigen Weg, indem er sich so lange bewegt, bis er in eine „Sackgasse" gelangt, worauf er zum zuletzt erreichten Punkt zurückkehrt und alles von Neuem versucht. Das ist zwar kein sehr detailliertes Modell, zeigt aber, wie ein Computerprogramm zur Simulation von Bewegungen benutzt werden kann. Der „Roboter" im Programm gehorcht einer vorgegebenen Anzahl von Regeln und fertigt eine „Karte" der Umgebung. Hätte der Roboter innerhalb des Programms direkten und sofortigen Zugriff zu Informationen über alle Teile des Labyrinths, könnte er sich direkt zum Ziel bewegen.

Ähnlich ist es mit dem Roboterarm, der keinen Sensor hat. Das Programm müßte ein Modell der Umgebung des Roboters sowie ein Modell des Arms selbst enthalten. Diese beiden Modelle müssen nun so zusammenwirken, wie es in Wirklichkeit auch wäre. Also kann auch kein Gegenstand aufgenommen werden, wenn der Arm nicht richtig positioniert ist. Obwohl das Programm mit Computergrafik arbeiten würde, in der eine Linie (der „Arm") eine andere (den „Boden") kreuzen kann, ist für eine wirklichkeitsnahe Simulation erforderlich, daß sie sich nicht kreuzen. Läßt der Roboter einen Gegenstand los, darf dieser nicht in seiner gegenwärtigen Position bleiben. Auch die Schwerkrafteinwirkungen müssen berücksichtigt werden. Ließe man das außer acht, wäre es nicht möglich, die Simulation eines Roboters zu schaffen, der ein Ei sicher aufnimmt und anderswo hinlegt.



Die Bandbreite von Bewegung und Synchronisation, die bei echten Robotern erforderlich ist, wird angesichts der extrem nah beieinander stehenden Montageroboter bei der Autofertigung vorstellbar.

Den Möglichkeiten der Computersimulation sind kaum Grenzen gesetzt. Und je umfassender die Simulation ist, desto faszinierender wird sie. Sie kann sogar unterhaltender sein als das Spiel mit „echten" Robotern – aus dem einfachen Grunde, da man sich einen Roboter nach eigener Wahl „erschaffen" kann. Das Programmieren der genauen Einzelheiten eines Roboters und seiner Umgebung führt zu einem besseren Verständnis der Roboter und ihrem Wirken in der räumlichen Welt. Das Programm „Roboter-Arm" müßte feststellen, daß das Objekt auf den Boden fällt und dort liegen bleibt, wenn der Roboter es losläßt. Um das Ganze realistischer zu machen, könnte man das Programm so ändern, daß sich der Gegenstand im Fall dreht, den Gesetzen der Schwerkraft gehorchend. Er könnte auch beim Aufprall zerbrechen, es gibt viele Möglichkeiten. Das Programm müßte so angelegt sein, daß man es nach Belieben ergänzen kann, um die Simulation so lebensecht wie möglich zu machen.

Die Entwicklung von Computer-Simulationen ist dem Schreiben von Software sehr ähnlich. Der Unterschied besteht darin, daß die Simulation die Echtwelt so realistisch wie möglich darzustellen hat. Dies zu erreichen mag schwer sein, doch hat man es erst einmal geschafft, ist die Simulation befriedigender als die passive Beschäftigung mit einem noch so spannenden Computerspiel.

Assemblerschleifen

Schleifen und bedingte Verzweigungen werden in der Assemblersprache über die relativen Sprungbefehle und die Flags des Prozessor-Status-Registers gesteuert. Mit diesen Strukturen und der indizierten Adressierung lassen sich auch Datentabellen erstellen und bearbeiten.

Bevor wir uns den verschiedenen Adressierarten der CPU (und den indizierten Adressen) zuwenden, müssen Sie wissen, wie Schleifen funktionieren. In der Assemblersprache gibt es keine automatischen Strukturen wie beispielsweise FOR. . .NEXT in BASIC (ein Befehl des Z80 ist dieser Struktur jedoch sehr ähnlich). Der Ablauf läßt sich jedoch aus Befehlen zusammensetzen, die Entscheidungen fällen oder Bedingungen enthalten und so den Ablauf des Programms beeinflussen können.

Entscheidungsprozesse der Assemblersprache drehen sich hauptsächlich um die Flags des Prozessor-Status-Registers. Diese Bedingungsflags zeigen an, welche Auswirkungen der letzte Befehl auf den Akkumulator hatte. Der Zustand des Nullflags (Z) und des Übertragsflags (C) bestimmt, ob der nächste Befehl ausgeführt oder zu einer anderen Stelle des Programms verzweigt wird. Dabei „entscheidet" der Prozessor, ob er die in seinem Programmzähler gespeicherte Adresse annimmt oder verändert. Der Programmzähler enthält immer die Adresse des Maschinencodes, der als nächstes ausgeführt werden soll, wobei der Prozessor den Op-code aus dem Byte lädt, auf das der Programmzähler zeigt. Die im Programmzähler enthaltene Adresse wird nach dem Laden der Instruktion um die Anzahl der Befehlsbytes erhöht, so daß sie immer auf den Op-code des nächsten Befehls zeigt. Wenn diese Adresse durch einen Befehl verändert wird, entsteht ein Programmsprung.

Verzweigungsbefehle

Der 6502-Befehl BEQ veranlaßt eine Veränderung des Programmzählers, wenn das Nullflag gesetzt ist. BCS ist die entsprechende Instruktion für das Übertragsflag (auf dem Z80 JR Z und JR C). Diese vier Op-codes werden Verzweigungsbefehle genannt. Ihr Operand – eine Ein-Byte-Zahl – wird auf die Adresse des Programmzählers addiert und erzeugt so eine neue Adresse. Sehen wir uns diesen Vorgang genauer an:

ORG $5E00			
6502		Z80	
5E00	ADC #$34	ADC	A,$34
5E02	BEQ $03	JR	Z,$03
5E04	STA $5E20	LD	($5E20),A
5E07	RTS	RET	

Wenn der in $5E00 gespeicherte ADC-Befehl im Akkumulator das Ergebnis Null erzeugt, dann verursachen die Befehle BEQ und JR Z in $5E02, daß $03 zu dem Inhalt des Programmzählers addiert wird. Der nächste auszuführende Befehl ist dann der Rücksprung zu $5E07, und die Instruktion in $5E04 wird dabei übersprungen.

Symbolische Adressen

Auf den ersten Blick scheint dies ein Fehler zu sein, da der Befehl in $5E02 $03 auf den Programmzähler addiert und die dort gespeicherte Adresse daher $5E05 sein müßte. Hier ist es wichtig zu berücksichtigen, daß der Programmzähler immer auf den **nächsten** Befehl zeigt und nicht auf den, der augenblicklich ausgeführt wird. Wenn also die Instruktion in $5E02 mit der Ausführung beginnt, dann enthält der Programmzähler schon die Adresse $5E04 – die Speicherstelle des darauffolgenden Befehls. So ergibt die Addition von $03 auf $5E04 das Ergebnis $5E07, also die Adresse des nächsten Befehls.

Interessant ist hierbei auch, daß der Prozessor nicht überprüfen kann, ob die Adressen korrekt sind. Wenn wir aus Versehen die zu addierende Zahl mit $02 angegeben haben, erhöht sich der Programmzähler um $02 (vorausgesetzt, der Akkumulator enthält Null), und der Prozessor nimmt $5E06 als die Adresse des nächsten Op-Codes. In unserem Programm enthält $5E06 den Wert $5E, das höherwertige Byte des Operanden der Instruktion in $5E04. Der Prozessor kann jedoch nicht beurteilen, ob dieser Befehl richtig ist oder nicht. Für ihn ist $5E ein gültiger Op-Code, wobei er die darauffolgenden Bytes als Operanden ansieht. Das Programm wird vermutlich abbrechen. Die falsche Berechnung von Abständen ist einer der häufigsten Fehler der Maschinencodeprogrammierung.

In der Assemblersprache ist die Abstandsberechnung jedoch kein Problem, da der Assembler diese Aufgabe übernehmen kann. Statt eine Hexadezimalzahl als Operanden des Verzweigungsbefehls anzugeben, können wir eine symbolische Adresse als Zieladresse des Sprungs angeben. Der Assembler wandelt die symbolische Adresse in eine absolute um, berechnet den Abstand zu dieser Adresse und

schreibt ihn in den Maschinencode. Die symbolische Adresse wird „Label" genannt; sie entspricht etwa der Zeilennummer eines BASIC-Programms.

Labels sind alphanumerische Strings, die am Anfang des Assemblerprogramms stehen. Das Assemblerprogramm interpretiert sie als Zwei-Byte-Symbole, die die Adresse des ersten Befehlsbytes repräsentieren. Wir können daher das Programm folgendermaßen umschreiben:

ORG $5E00		
	6502	Z80
5E00	ADC #$34	ADC A,$34
5E02	BEQ EXIT	JR Z,EXIT
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

Der Befehl in $5E02 kann jetzt folgendermaßen verstanden werden: „Wenn (IF) der Wert des Akkumulators Null ist, dann springe (THEN GOTO) auf die Adresse, die von dem Label EXIT dargestellt wird".

Mit Labels und Verzweigungsbefehlen können wir jetzt eine Schleife aufbauen:

ORG $5E00		
	6502	Z80
5E00 START	ADC #$34	ADC A,$34
5E02	BNE START	JR NZ,START
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

Hier treten das Label START und ein neuer Verzweigungsbefehl auf: BNE bedeutet „Verzweige, falls der Akkumulator ungleich Null ist"; JR NZ bedeutet „Springe, wenn der Akkumulator nicht Null ist". Dieser Code bedeutet: Auf den Akkumulator wird $34 addiert. Falls das Ergebnis nicht Null ist, verzweigt das Programm auf $5E00 – die Adresse des Labels START. $34 wird nochmals auf den Akkumulator addiert, wobei das Ergebnis wiederum entscheidet, ob eine weitere Verzweigung ausgeführt werden soll. Diese „Schleife" setzt sich so lange fort, bis die Verzweigungsbedingung erfüllt ist. Wenn der Inhalt des Akkumulators nach dem ADC-Befehl Null ist, dann findet in $5E02 keine Verzweigung statt, und der Befehl in $5E04 wird ausgeführt.

Es stellt sich die Frage, ob der Akkumulator jemals Null sein kann, da er bei jedem Schleifendurchlauf um weitere $34 erhöht wird. Die Antwort ist einfach: Der Akkumulator ist ein Ein-Byte-Register. Bei einem Zwei-Byte-Ergebnis wird das Übertragsflag des Prozessor-Status-Registers gesetzt, wobei der Akkumulator das niederwertige Byte der Summe enthält. Wenn im Akkumulator beispielsweise $CC gespeichert ist und darauf $34 addiert wird, ist das Ergebnis $0100. Das Übertragsflag wird gesetzt, und der Akkumulator enthält $00 als niederwertiges Byte des Ergebnisses. Da jetzt der Inhalt des Akkumulators Null ist, wird auch das Nullflag gesetzt.

Statt des Nullflags kann auch der Status des Übertragsflags abgefragt werden:

ORG $5E00		
	6502	Z80
5E00 START	ADC #$34	ADC A,$34
5E02	BCC START	JR NC,START
5E04	STA $5E20	LD ($5E20),A
5E07 EXIT	RTS	RET

In dieser Version bedeutet der Befehl in $5E02 „Verzweige nach START, wenn das Übertragsflag nicht gesetzt ist". Sobald das Additionsergebnis größer als $FF ist, wird das Übertragsflag gesetzt, und die Verzweigung nach START wird nicht durchgeführt.

Im Augenblick fehlt jedoch noch ein Schleifenzähler, durch den man beispielsweise die Information erhält, wie oft eine Schleife ausgeführt wurde, bevor die Endbedingung erfüllt war. Dieses läßt sich leicht erreichen, wenn das Indexregister der CPU den Zähler enthält und ein Befehl den Zähler nach jeder Schleife aktualisiert:

6502		
0000	ORG	$5DFD
5DFD	LDX	#$00
5DFF START	INX	
5E00	ADC	#$34
5E02	BCC	START
5E04	STX	$5E20
5E07 EXIT	RTS	

Z80		
0000	ORG	$5DFA
5DFA	LD	IX,$0000
5DFE START	INC	IX
5E00	ADC	A,$34
5E02	JR	NC,START
5E04	LD	($5E20),IX
5E08 EXIT	RET	

Schleifenzähler

Die neue Struktur hat in dem Programm mehrere Veränderungen bewirkt. Zunächst enthalten die Befehle am Anfang des Programms eine neue ORG-Adresse. Diese Instruktionen haben auf dem 6502 und dem Z80 zwar die gleichen Auswirkungen, die Längen und Adressen sind jedoch verschieden. Weiterhin haben neue Versionen der Lade- (LDX, LD IX) und Speicherbefehle (STX, LD ( ), IX) den Anfangswert $00 in das Indexregister der CPU geladen. Das X-Register des 6502 ist ein Ein-Byte-Register, das IX-Register des Z80 hat jedoch ein Zwei-Byte-Format. Die Indexregister haben zwar Spezialfunktionen, sind aber wie der Akkumulator im wesentlichen RAM-Speicher. Wir verwenden sie hier als zusätzliche Akkumulatoren, in denen wir den Schleifenzähler unterbringen. Nach Ende der Schleife

wird der Inhalt des X-Registers (6502) in $5E20 gespeichert, während in der Z80-Version das niederwertige Byte des (Zwei-Byte-)IX-Registers in $5E20 untergebracht wird und das höherwertige Byte in $5E21.

Auch an die Stelle von ADC am Anfang (START) der Schleife sind neue Instruktionen getreten: INX und INC IX sind „Inkrementierbefehle", die den Inhalt des Indexregisters um $02 erhöhen (inkrementieren). Sie aktualisieren den Wert des Schleifenzählers bei jedem Durchlauf.

Berechnungsbeispiele

Das Programm lautet jetzt folgendermaßen: "Stelle den Schleifenzähler auf Null, beginne die Schleife mit der Inkrementierung des Zählers und addiere $34 auf den Akkumulator. Verzweige an den Anfang der Schleife, wenn das Übertragsflag Null ist, andernfalls speichere den Schleifenzähler in $5E20".

6502

```
0000          ORG   $5DFA
5DFA          LDX   #$00
5DFC START    STA   $5E22,X
5DFF          INX
5E00          ADC   #$34
5E02          BCC   START
5E04          STX   $5E20
5E07 EXIT     RTS
```

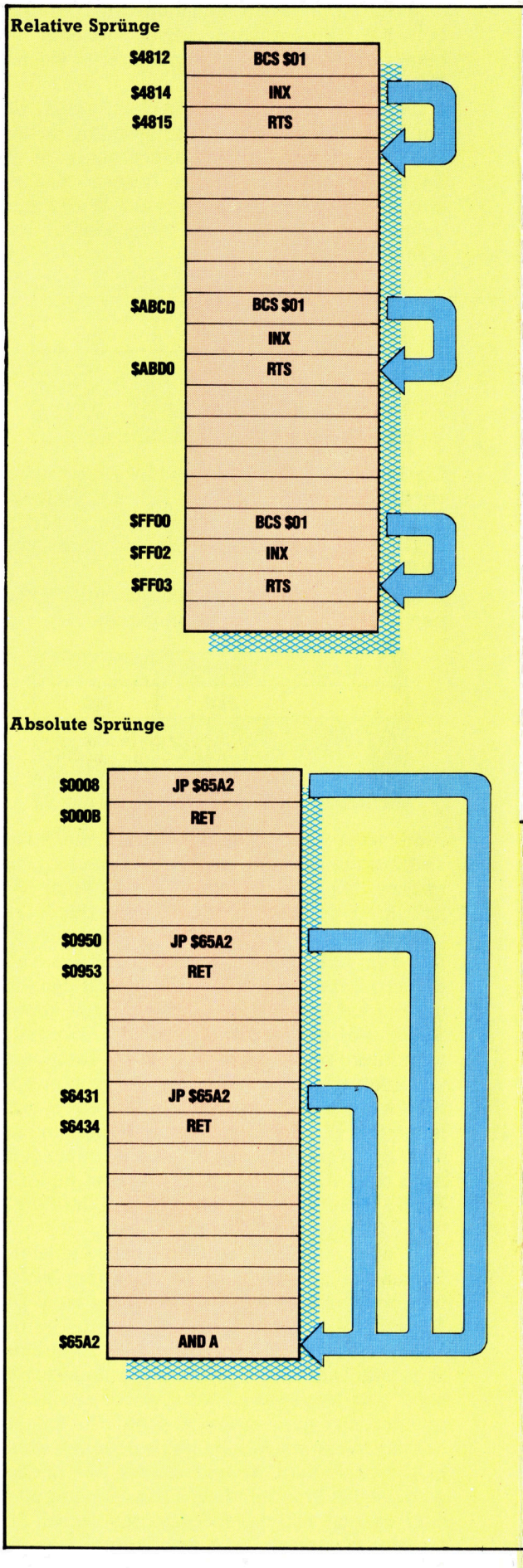
Z80

```
0000          ORG   $5DF7
5DF7          LD    IX,$5E00
5DFB START    LD    (IX+$22),A
5DFE          INC   IX
5E00          ADC   A,$34
5E02          JR    NC,START
5E04          LD    ($5E20),IX
5E08 EXIT     RET
```

Hier haben die Versionen des 6502 und des Z80 die gleiche Wirkung: Sie legen an der Adresse $5E22 eine Tabelle der Akkumulatorwerte an, die bei der Ausführung des Programms entstehen, und speichern den Endwert des Schleifenzählers in $5E20. Dieser Endwert entspricht auch der Anzahl der Tabellenbytes (beginnend bei $5E22).

Die Version des 6502 setzt für diese Aufgabe den Befehl STA $5E22,X ein. Er bedeutet: „Addiere den Inhalt des X-Registers auf die Basisadresse $5E22, und speichere dann den Inhalt des Akkumulators auf die neue Adresse". Der STA-Befehl ist hier absolut und direkt adressiert. Das heißt, das X-Register wird als Index eingesetzt, um die Basisadresse $5E22 zu verändern. Da das X-Register anfänglich mit $00 geladen und bei jeder Schleife erhöht wird, ist der Anfangswert des Akkumulators in $5E22 gespeichert, der nächste Wert in $5E23 etc. Nach Beendigung der Schleife legt STX den Endwert des Schleifenzählers in $5E20 ab.

Die Z80-Version verwendet das IX-Register



als Zeiger, der die aktuelle Speicheradresse angibt, während das niederwertige Byte von IX als Schleifenzähler dient. Der Befehl LD IX, $5E00 lädt die Basisadresse $5E00 in das IX-Register, so daß das niederwertige Byte von IX $00 enthält. Die seltsam aussehende Instruktion LD (IX+$22), A bedeutet „Addiere die in IX enthaltene Adresse auf $22, und speichere den Inhalt des Akkumulators auf die dadurch entstehende Adresse". Da IX mit $5E00 initialisiert ist und in der Folge bei jeder Schleife erhöht wird, ist damit der Anfangswert des Akkumulators in $5E22 gespeichert, der nächste Wert in $5E23 usw. Das niederwertige Byte von IX enthält dabei die Anzahl der Durchgänge, die nach Ende der Schleife in $5E20 untergebracht wird. LD (IX+$22),A ist hier eine absolute indirekte und indizierte Adressierung, die zwar komplizierter aber flexibler als der Befehl des 6502 ist.

Relative Sprünge
Die meisten Verzweigungsbefehle wie BCS („Verzweige, wenn das Übertragungsflag gesetzt ist") und JR NZ („Verzweige, wenn der Akkumulator nicht Null ist") arbeiten mit dem Inhalt des Prozessor-Status-Registers, wenn durch relative Sprünge der Fluß des Programms umdirigiert werden soll. Die Alternative ist der absolute Sprung. In dem nebenstehenden Beispiel bewirkt der Befehl BCS $01 ständig einen relativen Vorwärtssprung von einem Byte (falls die entsprechende Bedingung erfüllt ist) – unabhängig von der Adresse, an der der Maschinencode gespeichert ist. Dem Befehl BCS $01 folgt der Ein-Byte-Befehl INX. Ist das Übertragsflag gesetzt, dann verursacht BCS das Überspringen von INX.

Absoluter Sprung
In diesem Beispiel löst der Befehl JP $65A2 in jedem Fall einen absoluten Sprung aus. Die Programmsteuerung wird dabei auf die Adresse seines Operanden geleitet – hier $65A2. Es wird kein Test ausgeführt, und auch die Adresse, von dem aus der Sprung ausgeführt wurde, ist nicht entscheidend. Die Programmausführung setzt sich immer von der angegebenen Adresse an fort.

Beide Sprungarten haben Vor- und Nachteile. Das wichtigste Kriterium bei der Wahl zwischen einem relativen und einem absoluten Sprung ist die Plazierung: Assemblerroutinen werden oft mit einer bestimmten ORG-Adresse assembliert und später mit einer anderen ORG-Adresse nochmals verwandt. Wenn die Routine nur relative Sprünge enthält, funktioniert sie ohne Änderung der Speicheradressen. Absolute Sprünge können bei einer Änderung der ORG-Adresse die Steuerung jedoch zu Adressen senden, die für die Routine keine Bedeutung haben. Relative Sprünge lassen sich versetzen, absolute nicht.

Übungen

In dieser Folge sind wir auf viele wichtige Punkte eingegangen, die Sie zunächst vielleicht irritieren werden. Die neuen Adressierarten und Befehlsstrukturen lassen sich nur verstehen, wenn sie angewandt werden.

Assemblieren und speichern Sie die verschiedenen Programmteile dieser Folge mit dem „CHAMP"-Assemblerpaket. Untersuchen Sie die bei der Ausführung eines Programms angesprochenen Speicherbytes mit der <DEBUG>-Schaltung. Dabei lohnt es sich, diese Bytes vor dem Programmaufruf mit einer leicht erkennbaren Konstanten – zum Beispiel $FF – zu initialisieren, damit Sie nach dem Ablauf erkennen können, wie sich der Speicherinhalt verändert hat. Nehmen Sie dafür entweder den „<DEBUG> Alter"- oder den „<DEBUG> Move"-Befehl.

Bitte beachten Sie, daß die in den Programmen angegebenen Adressen nur Beispiele sind. Die Programme funktionieren nur, wenn Sie die für Ihre Maschine geeigneten Adressen verwenden.

Laden und Speichern von CHAMP
Bitte legen Sie auf einer anderen Cassette zunächst eine Sicherheitskopie von CHAMP an. Bei den folgenden Instruktionen bezieht sich der LOAD-Befehl auf die CHAMP-Cassette und SAVE auf die Kopie:

Acorn B
1) LOAD"CHAMP"
2) SAVE"CHAMP":RUN: Auf BASIC schalten
3) *SAVE"CHAMP M/C" 1000, 4600

Commodore 64
1) LOAD"CHAMP"
2) SAVE"CHAMP":RUN: eingeben <DEBUG> Modus
3) [w] [ret] drücken, gefolgt von s für SAVE
4) Anfangsadresse 1000; Endadresse 4600; Dateiname "CHAMP M/C"

Spectrum
1) LOAD"CHAMP"
2) Auf BASIC schalten: SAVE"CHAMP" LINE 1
3) SAVE"CHAMP M/C"CODE 27000, 9231

Sie haben gesehen, daß die bedingte Verzweigung vom Inhalt des Prozessor-Status-Registers abhängt. Mit der Binäranzeige des Monitors können Sie den Inhalt des PSR vor und nach der Ausführung eines Befehls ansehen und beobachten, wie sich die Flags verändern. Da es in der Assemblersprache des 6502 und des Z80 keinen Befehl gibt, der den Inhalt des PSR speichert, müssen wir folgende Codes einsetzen:

Z80	
3E00 F5	PUSH AF
3E01 F5	PUSH AF
3E02 E1	POP HL
3E03 22 lo hi	LD (STORE1),HL
3E06 F1	POP AF
6502	
3E00 48	PHA
3E01 08	PHP
3E02 48	PHA
3E03 08	PHP
3E04 68	PLA
3E05 8D lo hi	STA STORE1
3E08 68	PLA
3E09 8D lo' hi'	STA (1+STORE1)
3E0C 28	PLP
3E0D 68	PLA

Diese Befehlsfolge verursacht, daß der augenblickliche Inhalt des PSR in einem Byte gespeichert wird, das mit STORE1 addressiert wird, während der Inhalt des Akkumulators in (1+STORE1) untergebracht ist. Sie brauchen diese Befehle nur als Block vor oder hinter den Programmteil zu setzen, den Sie beobachten wollen. Beachten Sie aber, daß Sie für jeden Einsatz dieses Blocks die Zahl Zwei auf den Wert von STORE1 addieren müssen. Bei der Ausführung eines Programms können Sie sich nun mit dem Monitor den Teil des Speichers ansehen, in dem die Inhalte des PSR und des Akkumulators abgelegt wurden.

Es liegt nahe, diesen Block als Unterroutine zu behandeln, statt ihn mehrfach zu speichern. In der Assemblersprache gibt es einen Befehl, der dem GOSUB in BASIC entspricht. Sein Gebrauch würde den Ablauf jedoch komplizierter machen, da er den Stack anspricht, der auch von einigen Befehlen des Programmblocks eingesetzt wird (PLA, PUSH und PHP verändern den Stack, auf den wir später genauer eingehen werden). Für die unterschiedlichen Codelängen des Z80 und 6502 sind die Zwei-Byte-Register des Z80 und die zugehörigen Befehle verantwortlich.

Chip-Geplauder

My Talking Computer soll Kinder an die Computer-Technik heranführen. Deshalb wurde er so einfach wie möglich konstruiert. Die in einem Ringbuch aufbewahrten Überleger werden auf eine berührungsempfindliche Fläche gelegt und durch einen Spannrahmen fixiert. Erweiterungsmodule lassen sich einfach in den entsprechenden Steckschacht auf der linken Seite des Computers einführen.

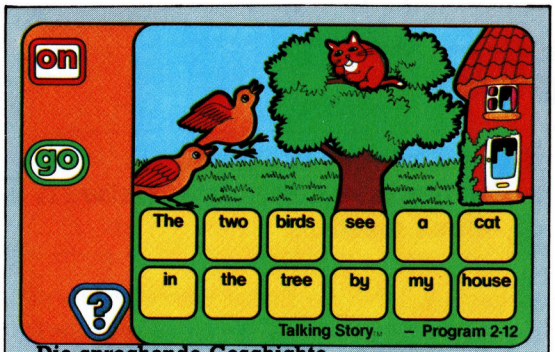
„My Talking Computer" ist eine auf Microprozessortechnik basierende Lernhilfe für Kinder im Vorschulalter. Dabei wird das Problem, wie Kinderhände mit der „Tastatur" eines Computers klarkommen sollen, gut gelöst. Bleibt abzuwägen, wie groß der erzieherische Wert des Gerätes ist, und zu fragen, ob es sich dabei tatsächlich um einen Computer handelt.

My Talking Computer" (Mein sprechender Computer) von Microspeech ist für Kinder ab drei Jahren gedacht. Das Unternehmen behauptet, daß das Gerät bereits überall in England an Grundschulen verwendet wird. Der „Computer" befindet sich in einem leichten Plastikgehäuse mit den Maßen 23 × 25 cm. Überleger für die einzelnen Programme werden auf die mit Berührungssensoren ausgestattete Tafel gelegt und durch einen Rahmen fixiert. Unter diesem Gesichtspunkt ist das Gerät dem „Touchmaster" ähnlich. Allerdings weist der Talking Computer weniger Kontaktpunkte auf. Im Lieferumfang ist „My Talking Clock" (eine sprechende Uhr) enthalten, die den Kindern die Uhrzeit beibringt. Sie ist Bestandteil eines der eingebauten Programme.

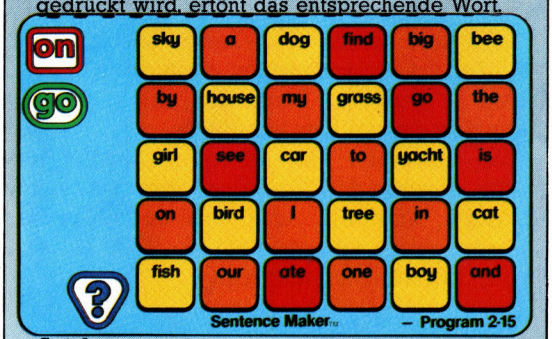
An einer Seite des Gerätes befindet sich ein Schacht, der Programmodule aufnimmt. Wenngleich My Talking Computer eine interessante Lernhilfe für Kinder darstellt, sind die Programme doch in ihren Anforderungen an die Kinder sehr begrenzt. Damit ist die Einsatzdauer der Geräte, bedingt durch die Weiterentwicklung der Kinder, ebenfalls beschränkt. Der Hersteller versucht, dieses Problem durch zusätzliche ROM-Module zu lösen, die in den Erweiterungsschacht gesteckt werden. Das erste dieser ROMs, Expansion Module One (Erweiterungsmodul Eins), weist nur geringfügig höhere Ansprüche auf. Man scheint hier zu beabsichtigen, eine Serie zu schaffen, die sich schrittweise mit der Entwicklung des Kindes steigert.

Über dem Erweiterungsschacht befindet sich ein kleiner Lautsprecher. Das Gerät kann entweder mit fünf 1,5-Volt-Batterien oder über Netzteil mit Strom versorgt werden. Die Programme sind auf einem einzigen ROM-Chip gespeichert, der nach Angabe von Microspeech über 120 KByte enthält, die allerdings nicht gleichzeitig in den Computer geladen werden.

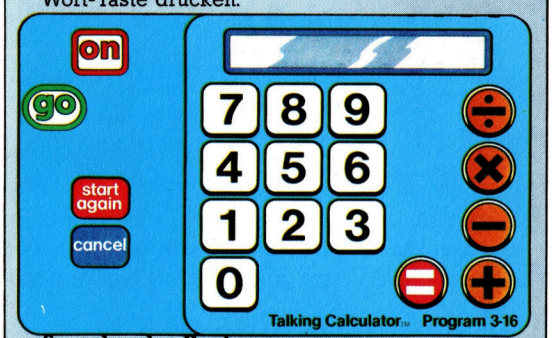
Die ROM-Software ist in fünf Hauptprogramme unterteilt, die wiederum in sich untergliedert sind. Das erste Hauptprogramm beschäftigt sich mit Arithmetik und bietet Übungen wie Zahlenerkennen, einfache Addition, Subtraktion, Multiplikation und Division. Im zweiten Programm werden Zusammenhänge zwischen Bildern und Wörtern erläutert. Beim dritten Programm handelt es sich um einen



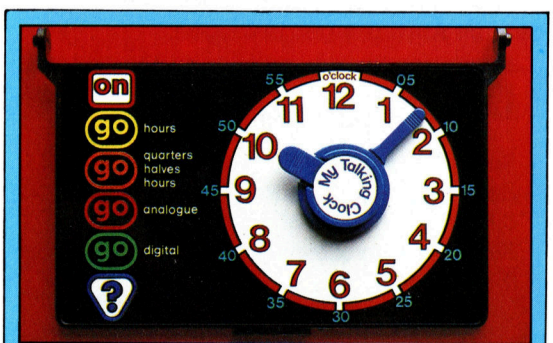
Die sprechende Geschichte
Dieses Programm ist für Zwei- bis Zwölfjährige entwickelt worden. Der Computer erkennt, welcher Überleger benutzt wird, durch die Positionen der „ON"- und „GO"-Knöpfe. Sobald ein Wortquadrat gedrückt wird, ertönt das entsprechende Wort.



Satzbauer
Das Programm lehrt Wörtererkennen. Drückt ein Kind auf das Fragezeichen, spricht der Computer ein Wort, und das Kind muß die entsprechende Wort-Taste drücken.



Sprechender Rechner
Eines der interessantesten Programme ist dieser sprechende Rechner. Durch Drücken von Zahlen und arithmetischen Symbolen können alle einfachen Rechenoperationen ausgeführt werden.



Zeitangabe
Über dem sogenannten „Touch Pad" befindet sich die sprechende Uhr. An der Unterseite der Uhr ragen Zahnräder auf die Sensortafel, die den Eingabefeldern des Computers entsprechen. Dreht man die Zeiger der Uhr, so sagt der Computer exakt die jeweils dargestellte Zeit.

„sprechenden Rechner", der die eingegebenen Zahlen und Operationen nennt und das Ergebnis mitteilt. Das vierte Programm besteht aus „sprechenden" Spielen, die die Fähigkeit des Kindes, Ähnlichkeiten bei Gegenständen festzustellen, testen sollen. Das fünfte, bereits erwähnte Programm, ist die sprechende Uhr.

Programme werden benutzt, indem das Kind den gewünschten Überleger aus einem Ringbuch auswählt und unter den Halterahmen klemmt. Nach Druck auf das „ON"-Feld ist das Gerät eingeschaltet, und eine Frauenstimme sagt „Hallo". Dann fordert das Programm das Kind zum Drücken der „GO"-Taste auf. Der Rechner „weiß", welches Programm gewählt wurde, weil sich „ON"- und „GO"-Taste auf den verschiedenen Überlegern an unterschiedlichen Positionen befinden. Besonders für kleinere Kinder ist dieses Bedienungsverfahren sehr geeignet.

Hat ein Kind gelernt, daß der Computer mit „ON" gestartet und das Programm mit „GO" aktiviert wird, kann es ohne die Hilfe eines Erwachsenen mit dem Gerät arbeiten. Die Überleger sind plastikbeschichtet, können also weder durch Schokolade noch Fruchtsaft beschädigt werden. Die Module können ganz einfach ausgetauscht werden.

Sprache mit Problemen

Das wohl beste Element des Computers ist der integrierte Sprach-Synthesizer, trotz seiner Einschränkungen. Die Sprache ist in ganzen Worten gespeichert, aus denen dann Sätze zusammengefügt werden. Die sich daraus ergebenden Satzformen wirken zuweilen merkwürdig. Das eigentliche Problem aber ist die Undeutlichkeit vieler Wörter. So können „by", „my" und „sky" (der Computer spricht englisch) nur schwer unterschieden werden. Selbst nach der mehrfachen Wiederholung bestimmter Sätze ist schwer zu verstehen, was gesprochen wurde.

Die beim Talking Computer angewandte Lehrmethode ist direkter als bei anderen Systemen, in denen Wort und Bild einander zuzuordnen sind. Der Unterhaltungswert von My Talking Computer ist ebenfalls beachtlich. Aber bei dem System handelt es sich nicht um einen Computer im eigentlichen Sinne des Wortes. Denn das Gerät kann nicht vom Anwender programmiert werden. „Texas Instruments" hat ein ähnliches System entwickelt, das auch auf dem deutschen Markt erhältlich ist (in Fachgeschäften bzw. den entsprechenden Abteilungen der Kaufhäuser). Ob die künstliche Sprache inzwischen Deutsch ist, war bei Redaktionsschluß nicht in Erfahrung zu bringen. Ein weiteres, ebenso funktionierendes System hat „Koala" (in Deutschland Vertrieb bei „harman Deutschland") entwickelt. Auch hier war nicht festzustellen, ob eine deutschsprachige Version lieferbar ist.

My Talking Computer

ABMESSUNGEN
250 × 230 × 70 mm

SOFTWARE
In ROM-Form in den Computer integriert. Ergänzende Software wird als Expansion Module 1 geliefert und in den Steckschacht eingeführt.

STÄRKEN
My Talking Computer ist sehr einfach zu bedienen. Da der Ladevorgang und die Sprachfähigkeit nicht ausführlich erklärt werden müssen, kann ein Kind, ohne das Lesen zu beherrschen, schnell mit dem Gerät umgehen.

SCHWÄCHEN
Die Sprachausgabe ist oft sehr undeutlich. Der Rechner ist nicht programmierbar und kann deshalb nicht als „echter" Computer bezeichnet werden.

Rastermuster

In diesem Teil des LOGO-Kurses wird die Umwandlung von zweidimensionalen Figuren erläutert. Wir beginnen mit dem Entwurf eines Rasters, auf dem die Muster aufgebaut werden.

Anstatt die Figuren an einer Geraden zu spiegeln, wird in dieser Folge gezeigt, wie zweidimensionale Muster durch gleichzeitiges, nicht-paralleles Verschieben entstehen. Das erste Beispiel demonstriert dieses anhand eines einzelnen Punktes:

```
TO PUNKT
  PD FD 1 BK 1 PU
END
```

Die Prozedur, die die Simultan-Verschiebungen vornimmt, sieht wie folgt aus:

```
TO RASTER :STARTX :STARTY :XSCHRITT
  :YSCHRITT :WINKEL
  DRAW HT PU
  SETXY :STARTX :STARTY SETH 0
  REPEAT 3 [ZEILE :XSCHRITT UNTEN
    :YSCHRITT :WINKEL]
END
```

Dieses Programm zeichnet ein Raster aus neun Punkten. Die Eingaben bestimmen die Koordinaten des Ausgangspunktes, die Größe der X- und Y-Schritte und die Richtung, in der die Bezugspunkte auf der nächsten Zeile dargestellt werden sollen.

```
TO ZEILE :X
  REPEAT 3 [OBJEKT SETX XCOR + :X]
  SETX XCOR — 3 * :X
END
```

Die ZEILE-Prozedur zeichnet eine Reihe mit drei Objekten und setzt die Turtle auf den Ausgangspunkt. Zu diesem Zeitpunkt ist das OBJEKT noch ein einzelner Punkt:

```
TO OBJEKT
  PUNKT
END

TO UNTEN :Y :A
  SETH :A
  FD :Y
  SETH 0
END
```

Die Prozedur UNTEN führt die Turtle in die nächste Zeile (in diesem Beispiel in die unterhalb liegende Zeile) und setzt die Richtung (Heading) auf Null.

Die fünf verschiedenen Planraster sowie die dazugehörigen Prozeduren sind im Diagramm dargestellt. Mit Hilfe dieser Raster lassen sich nun zahlreiche andere Musterkombinationen darstellen, zum Beispiel indem Sie den Linienwinkel verändern.

Interessanter wird es, wenn man um die einzelnen Punkte, die die Raster bilden, symmetrische Figuren zeichnet. Im anderen Diagramm sind siebzehn dieser Muster abgebildet. Um diese zu erstellen, muß der OBJEKT-Befehl in der ZEILE-Prozedur durch eine neue Definition ersetzt werden.

Die einzelnen Objektformen werden aus einem Grundmuster gebildet, das sich nach Belieben spiegeln und drehen läßt. Das Grundmuster, wir nennen es HAKEN, darf keine Unterroutinen aufrufen.

```
TO HAKEN
  PD
  FD 15
  RT 90
  FD 5
  BK 5
  LT 90
  BK 15
  PU
END
```

Wir verwenden die Prozeduren, die im letzten Teil des Kurses entwickelt wurden, um die Routinen MOTIV und R.MOTIV zu definieren:

```
DEFINE "MOTIV TEXT "HAKEN
DEFINE "R.MOTIV UMSCHR "HAKEN
```

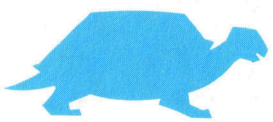
Nun werden die Objektformen festgelegt. Die Objekte für Muster 7 und 17 lassen sich mit folgenden Anweisungen erzeugen:

LOGO-Dialekte

Bei einigen LOGO-Versionen muß DRAW durch CS, ANYOF durch OR und SETXY durch SETPOS ersetzt werden. Beachten Sie die unterschiedliche Schreibweise der IF-Abfrage:

```
IF :WORD = MOTIV [OUTPUT "R.MOTIV]
```

Das Atari-LOGO beinhaltet weder TEXT noch DEFINE. Entsprechende Hinweise finden Sie im Handbuch.



```
TO OBJEKT7
  LT 90 MOTIV RT 90
END

TO OBJEKT17
  RT 30
  REPEAT 6 [MOTIV R.MOTIV RT 60]
  LT 30
END
```

Um diese Muster ausführen zu lassen, ist die ZEILE-Prozedur, die noch immer OBJEKT aufruft, zu ändern. Soll etwa MUSTER7 aufgerufen werden, ist die Prozedur OBJEKT in OBJEKT7 umzuschreiben. Am schnellsten geht das mit DEFINE.OBJEKT7:

```
TO DEFINE.OBJEKT :NUMB
  DEFINE "OBJEKT TEXT WORD "OBJEKT
    :NUMB
END
```

Die folgende Routine veranlaßt, daß Raster und Objekt gleichzeitig gezeichnet werden:

```
TO DEMO :RASTER :NUMB :PROZ
  DEFINE "MOTIV TEXT :PROZ
  DEFINE :R.MOTIV UMSCHR :PROZ
  DEFINE.OBJEKT :NUMB
  RUN (LIST :RASTER)
  ERASE MOTIV
  ERASE R.MOTIV
  ERASE OBJEKT
END
```

Die Eingabe für Muster 17 lautet:

```
DEMO "HEX 17 "HAKEN
```

Nun wird ein sechseckiges Raster gezeichnet, und die einzelnen Punkte werden mit OBJEKT17 versehen.

Diese Methode läßt sich bei fast allen gezeigten Mustern anwenden (ausgenommen sind die Muster 4, 6, 7 und 12, da hier die Objektformen an den einzelnen Punkten nicht identisch sind). Um diese „ungleichen" Muster zu erzeugen, müssen die Prozeduren ZEILE und UNTEN so geändert werden, daß sie die notwendigen Spiegelungen bzw. Drehungen der Objekte vornehmen können. Die Routinen TRANX und TRANY drehen die Objekte in vertikaler bzw. horizontaler Richtung.

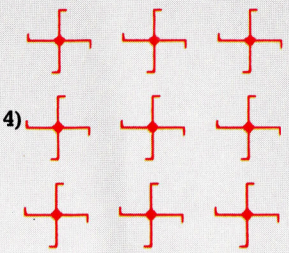
```
TO ZEILE :X
  REPEAT 3 [OBJEKT SETX XCOR XCOR +
    :X TRANX]
  SETX XCOR — 3 * X
END

TO UNTEN :Y :A
  SETH :A
  FD :Y
  SETH 0
```

Siebzehn Rasterfiguren

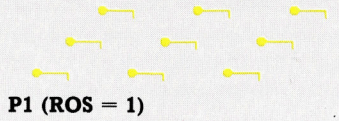
Schlüssel:
P = Parallelogramm-Raster
R = Rechteckiges Raster
C = Rhombisches Raster
S = Quadratisches Raster
H = Sechseckiges Raster
ROS = Drehsymmetrie
M = Spiegelung
G = Gleitspiegelung

Quadrat



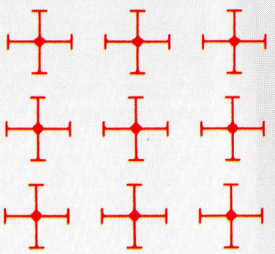
S4 (ROS = 4)

Parallelogramm



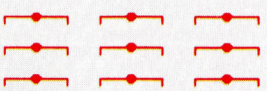
P1 (ROS = 1)

P2 (ROS = 2)

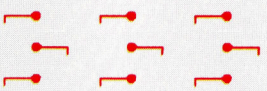


S4M (ROS = 4)

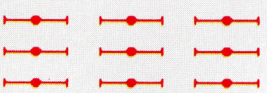
Rechteck



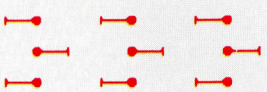
RM (ROS = 1)



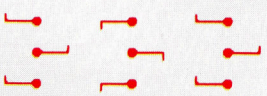
RG (ROS = 1)



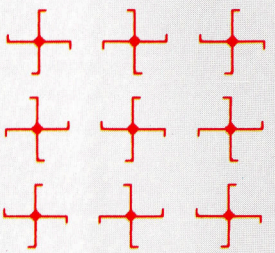
RMM (ROS = 2)



RMG (ROS = 2)

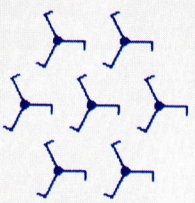


RGG (ROS = 2)

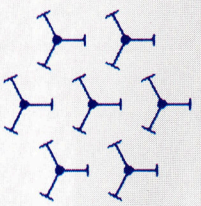


S4G (ROS = 4)

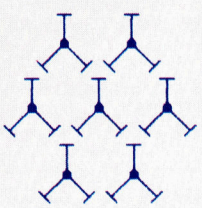
Sechseck



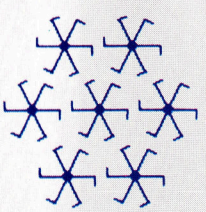
H3 (ROS = 3)



H3M1 (ROS = 3)

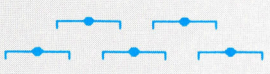


H3M2 (ROS = 3)

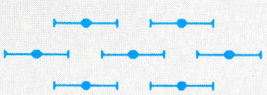


H6 (ROS = 6)

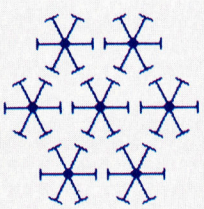
Rhombus



CM (ROS = 1)



CMM (ROS = 2)



H6M (ROS = 6)

Rasteraufbau
Die siebzehn Muster sind in Gruppen unterteilt. Die Zuordnung richtet sich nach der jeweiligen Form der Basisraster. H3M1 und H3M2 zum Beispiel basieren auf sechseckigen Rastern, und jeder Punkt ist mit drei symmetrischen Objekten ausgestattet. H3M1 ist eine horizontale, H3M2 eine vertikale Spiegelung.

```
    TRANY
  END
```

Muster 7 wird neu definiert als:

```
  TO MUSTER7 :PROZ
    DEFINE "TRANX [[] [SPIEGELN RT 180]]
    DEFINE "TRANY [[] []]
    DEMO "RECT 7 :PROZ
    ERASE TRANX
    ERASE TRANY
  END
```

Um die Prozedur zu starten, geben Sie MUSTER7 "BEIN ein. Die SPIEGELN-Routine in TRANX wird durch das Umschreiben der OBJEKT-Prozedur definiert:

```
  TO SPIEGELN
    DEFINE "OBJEKT UMSCHR "OBJEKT
  END
```

Die geänderte Prozedur wandelt alle RT in LT um und umgekehrt und berücksichtigt ferner MOTIV sowie sein Spiegelbild R.MOTIV. Die in der letzten Folge gezeigte Prozedur konnte lediglich RT und LT umwandeln. Hier die verbesserte AENDER.WORD-Prozedur:

```
TO AENDER.WORD :WORD
  IF (ANYOF :WORD = "RT :WORD =
    "RIGHT) THEN OUTPUT "LEFT
  IF (ANYOF :WORD = "LT :WORD =
    "LEFT) THEN OUTPUT "RIGHT
  IF :WORD = "MOTIV THEN OUTPUT
    "R.MOTIV
  IF :WORD = "R.MOTIV THEN OUTPUT
    "MOTIV
  OUTPUT :WORD
END
```

Eine andere Möglichkeit wäre, alle Unterroutinen der Eingabeprozedur mit Hilfe von UMSCHR neu zu definieren (siehe Lösungen).

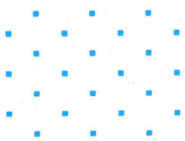
Da bei den übrigen Mustern weder Spiegelungen noch Drehungen erforderlich sind, haben TRANX und TRANY dort keine Funktion, zum Beispiel bei MUSTER17:

```
TO MUSTER17 :PROZ
  DEFINE "TRANX [[] []]
  DEFINE "TRANY [[] []]
  DEMO "HEX 17 :PROZ
  ERASE TRANX
  ERASE TRANY
END
```

Fünf Basisraster

Parallelogramm

```
TO PARALLEL
  RASTER (—60) 90 80 50
    205
END
```

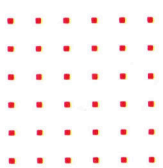


Rhombus

```
TO RHOMB
  RASTER (—30) 90 80 80
    225
END
```

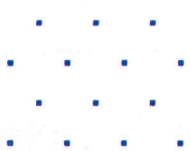
Rechteck

```
TO RECT
  RASTER (—80) 90 80 50
    180
END
```



Quadrat

```
TO QUAD
  RASTER (—80) 90 80 80
    180
END
```



Sechseck

```
TO HEX
  RASTER (—30) 90 80 80
    210
END
```

Lösungen

1. Prozedur zum Drehen einer Figur um den X,Y-Punkt. Drehwinkel=A

```
  TO DREHEN :X :Y :A
    PU
    MAKE "H HEADING
    MAKE "XALT XCOR
    MAKE "YALT YCOR
    MAKE "R QUAT (:XALT — :X) *
      (:XALT — :X) + (YALT — :Y) *
      (:YALT — :Y)
    PU
    SETXY :X :Y
    SETH TOWARDS :XALT :YALT
    RT :A FD :R
    SETH :H + :A
    PD
  END
```

2. Prozedur zum Umschreiben der Unterroutinen:

```
  MAKE "PROZE.DONE []
  TO UMSCHR :PROZ
    MAKE "PROZE: DONE FPUT :PROZ
      :PROZE.DONE
    OUTPUT UMSCHR. PROZ TEXT :PROZ
  END
  TO UMSCHR.PROZ :TEXT
    IF :TEXT = [] THEN OUTPUT []
    OUTPUT FPUT UMSCHR.ZEILE FIRST
      :TEXT
    UMSCHR.PROZ BUTFIRST :TEXT
  END
  TO UMSCHR.ZEILE :ZEILE
    IF :ZEILE = [] THEN OUTPUT []
    IF LIST? FIRST :ZEILE THEN OUTPUT
      FPUT
    UMSCHR.ZEILE FIRST :ZEILE
      UMSCHR.ZEILE BUTFIRST :ZEILE
    OUTPUT FPUT AENDER.WORD
      FIRST :ZEILE
    UMSCHR.ZEILE BUTFIRST :ZEILE
  END
  TO AENDER.WORD :WORD
    IF (ANYOF :WORD = "RT :WORD =
      "RIGHT) THEN OUTPUT "LEFT
    IF (ANYOF :WORD = "LT :WORD =
      "LEFT) THEN OUTPUT "RIGHT
    IF PROCEDURE? :WORD THEN
      SUBPROCEDURE :WORD OUTPUT
      WORD "£ :WORD
    OUTPUT :WORD
  END
  TO PROCEDURE? :NAME
    IF NUMBER? :NAME OUTPUT "FALSE
    IF LIST? :NAME OUTPUT "FALSE
    TEST WORD? :NAME
    IFTRUE IF WORD? TEXT :NAME
      OUTPUT
    "FALSE ELSE IF NOT (TEXT
      :NAME =[])
    OUTPUT "TRUE
    OUTPUT "FALSE
  END
  TO SUBPROCEDURE :WORD
    IF MEMBER? :WORD :PROCE.DONE
      THEN STOP
    DEFINE (WORD "£ :WORD) UMSCHR
      :WORD
  END
```

Fachwörter von A bis Z

Complement = Komplement
Das Komplement einer einstelligen Zahl ist die Ziffer, die sie zur Basis des Zahlensystems ergänzt. Zum Beispiel ist im Dezimalsystem (Basis 10) die 7 das Komplement zur 3. Mit Hilfe des Komplements läßt sich die Subtraktion auf eine Addition zurückführen. Die Aufgabe (8 − 3 = ?) läßt sich auch so formulieren:
(8 + (10 − 3)) − 10 = ?
Das ist einfacher als es aussieht:
10 − 3 = 7 ist das Komplement von 3, und statt 10 von der Summe 8 + 7 = 15 zu subtrahieren, brauchen Sie nur den bei der Addition entstehenden Zehner-Übertrag zu unterschlagen.

Im Binärsystem ist zur Bildung des erforderlichen Komplements lediglich eine Vertauschung von Nullen und Einsen vorzunehmen (beispielsweise entsteht aus 1011 das „Einerkomplement" 0100) und dann eine Eins zu addieren (damit wird aus 0100 folglich das „Zweierkomplement" 0101).

Concatenate = Verketten
Der Begriff „Verkettung" wird bei der Programmierung meist im Zusammenhang mit alphanumerischen Strings oder mit Diskettendateien verwendet. Die Strings A$ und B$ können zum Beispiel durch die Anweisung C$ = A$ + B$ zu einem neuen String C$ verkettet werden. Wenn zwei oder mehr Dateien miteinander verbunden werden, entsteht daraus unter einem neuen Namen eine große Gesamtdatei.

Concurrency = Mehrprogrammbetrieb
Ein „Concurrent"-Betriebssystem ermöglicht die gleichzeitige Abwicklung mehrerer Programme mit nur einem Rechner. Die Bearbeitung erfolgt dabei nicht simultan, sondern „verzahnt" – der Prozessor schaltet ständig zwischen den verschiedenen Aufgaben hin und her. Dieses „Multiprogramming"-Verfahren wird bei Großrechnern seit Jahren angewendet, während es bei kleineren Computern, aufgrund der niedrigen Verarbeitungsgeschwindigkeit, noch nicht weiter verbreitet ist. Unter dem Betriebssystem „Concurrent CP/M" von Digital Research laufen bis zu vier CP/M-86-Programme parallel.

Bei nur einer Tastatur, einem Bildschirm und einem Operateur müssen drei der Programme selbständig ablaufen. Über das Tastenfeld ist allerdings jedes der Programme zu starten, abzubrechen oder auf dem Bildschirm darzustellen. Beim Concurrent CP/M können mit Hilfe des „Piping"-Verfahrens auch die Ergebnisse des einen Programms automatisch als Eingabewerte dem nächsten zugeleitet werden. So kann ein Datenbankprogramm bestimmte Informationen heraussuchen und an ein Textverarbeitungssystem weitergeben, von dem ein drittes Programm den Bericht zum Ausdrucken im Hintergrund übernimmt.

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.

Das Betriebssystem „Concurrent CP/M" ist für viele Bürorechner verfügbar, so auch für den IBM-PC. Ein 16-Bit-Prozessor, ein großer Arbeitsspeicher und schnelle Diskettenlaufwerke sind Voraussetzung für die parallele Abwicklung von mehreren Programmen.

Constant = Konstante
Unter dem Namen einer Variablen werden im Rechner Speicherplätze verwaltet, deren Inhalt sich während des Programmlaufs ändern kann, während eine numerische oder eine String-Konstante unveränderlich ist. In der Anweisung:
A = B* 6.5731
sind A und B Variablen, 6.5731 ist eine Konstante.

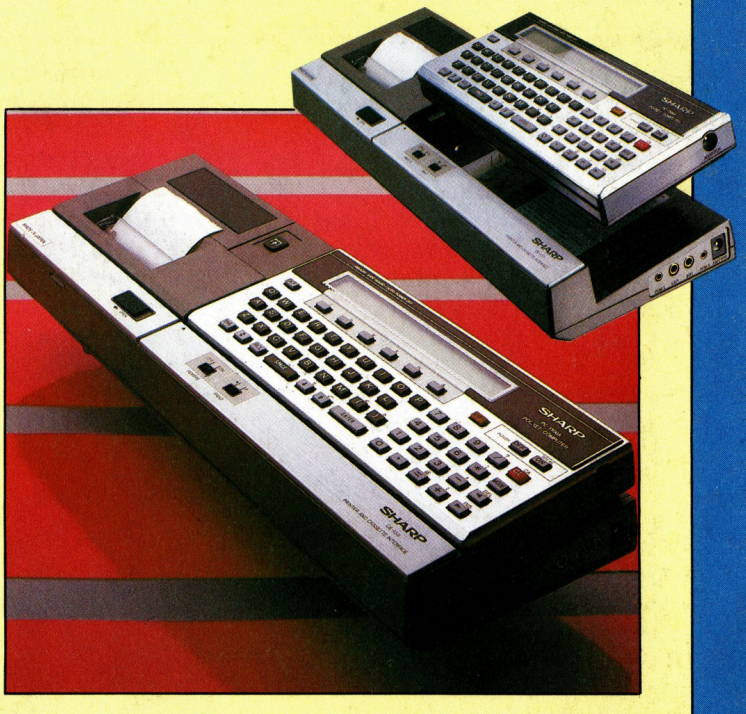
Konstanten weisen gegenüber Variablen einige Nachteile auf. Erstens kosten Konstanten unnötige Rechenzeit, weil fast alle Rechner mit Gleitkommaarithmetik arbeiten und nicht im BCD-System. In einer Programmschleife wird daher in der obigen Anweisung die Konstante bei jedem Durchlauf vom BASIC-Interpreter erneut in Gleitkommadarstellung umgewandelt, während Variable bereits richtig gespeichert vorliegen.

Außerdem belegt die mehrfache Verwendung einer Konstanten in einem Programm unnötig viel Speicherplatz. Vorteilhafter ist die Verwendung von Variablen, denen Sie zu Beginn des Programms den Wert der Konstanten zuweisen. Wenn Sie alle Konstanten-Zuweisungen im Programmkopf als Block zusammenfassen, sind auch Korrekturen leichter durchzuführen.

contents addressable = inhaltsadressierbar
Inhaltsadressierbar sind Speicher, bei denen das Aufsuchen von Speicherplätzen über deren Inhalte erfolgen kann. Besonders nützlich ist das bei Datenbanken, weil es effiziente Suchverfahren ermöglicht. Der Rechner könnte zum Beispiel eine Kundendatei mit Namen, Adressen und Telefonnummern nach der Vorwahl 040 absuchen; dabei käme dann eine Adressenliste der Hamburger Kunden heraus.

Bildnachweise
787: Roy Ingram
788, 808, 809, 811: Ian McKinnell
791, 794, 801: Kevin Jones
793: Liz Heaney
796, 797: Chris Stevens
802: Steve Cross
803: Ford
806: Liz Dixon

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



computer kurs Heft 30

Die japanische Firma Sharp stellt Rechner her, die in die Tasche passen – trotz der Minimal-Maße handelt es sich um „richtige" Computer mit vielen Möglichkeiten.

Kleinroboter
Auf dem Markt gibt es einige Erzeugnisse, die als „Roboter" verkauft werden. In der Serie wird untersucht, welche Geräte tatsächlich die Erwartungen erfüllen.

Sektoren-Grenzen
Eine optimale Software ist ohne Disketten nicht denkbar. Wir beschreiben, wie die Floppys innerhalb des Laufwerks durch Anweisungen des DOS formatiert werden.

Algebraische Strukturen
Die theoretische Basis jedes Computers ist die Boolesche Algebra. Eine Einführung.

Cassetten-Tricks
Dateistrukturen sind abhängig vom Computersystem. Diese Folge zeigt, wie man sie auf Cassetten speichert.

Tele-Kommunikation
Was soll ein Modem leisten? Für welche Zwecke ist welches Gerät geeignet? Antworten im nächsten Heft.