Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Ein wöchentliches Sammelwerk

Heft 40

Apricot Portable

Funktionen SIN und COS

Töne mit dem D/A-Wandler

Datenbanken einsetzen

computer kurs

Heft 40

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.

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

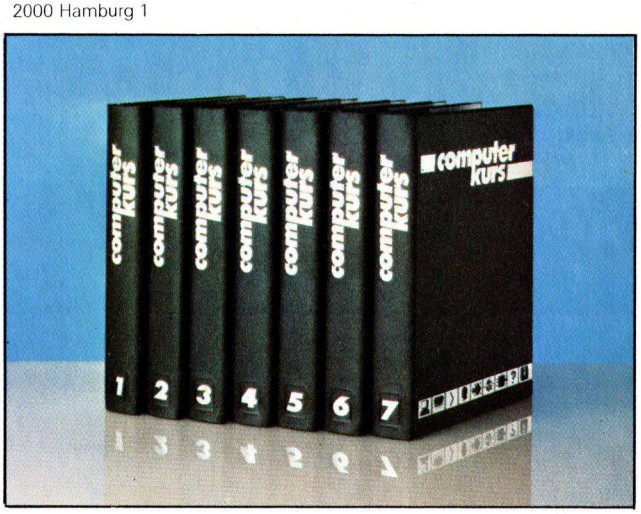
Schweiz: Der Sammelordner kostet sfr 15. Bitte wenden Sie sich an Ihren Kiosk; dort werden Sie jederzeit die gewünschten Exemplare erhalten.

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 Redaktions-Service GmbH, Paulstraße 3, 2000 Hamburg 1

Vertrieb: Marshall Cavendish International Ltd., Heidenkampsweg 74, 2000 Hamburg 1



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

Informationen auf Fingerdruck

Beim Computer dreht sich alles um das Speichern und Verarbeiten von Informationen. Wir zeigen, welche Datenbanken auf Heimcomputern verwendet werden können, welchen Beschränkungen kleinere Rechner unterliegen und wie man Datenbanken am besten nutzt.

Eine „Datenbank" ist eine Sammlung von Daten eines bestimmten Sachgebiets. Eine Datenbank könnte beispielsweise eine der folgenden Informationsgruppen umfassen: Namen und Adressen von Mitgliedern eines Clubs, die bei einem Clubtreffen entstandenen Kosten, Termine und Treffpunkte der Clubveranstaltungen oder die Zahlung der Mitgliedsbeiträge. Ein Datenbank-System könnte allerdings auch einfach alle Datensätze in einem größeren Datensatz zusammengefaßt enthalten.

Ein „File" ist eine Sammlung von Aufzeichnungen, das jeweils eine bestimmte Anzahl von Feldern umfaßt. Bei einigen Applikationen, wie etwa Buchhaltungs-Software, wird die Struktur eines Feldes durch die Software bestimmt. Beim Datenbank-System ist es jedoch üblich, daß die Feldstrukturierung der jeweiligen Aufgabenstellung angepaßt wird. Dazu gehört die Bestimmung der Größe eines jeden Feldes sowie die Definition des darin zu speichernden Inhalts. In einem Namens- und Adreß-File beispielsweise belegen die Informationen über eine Person jeweils einzelne Aufzeichnungen: Der Name wird in einem Zeichenfeld gespeichert und jede Adressenzeile in einzelnen weiteren Feldern.

Alternativ zu Zeichenfeldern gibt es numerische bzw. Nummernfelder, die dem Programm Rechenoperationen mit den gespeicherten Daten erlauben. In unserem Clubmitglieder-File etwa könnte das Programm errechnen, wieviele Mitglieder ihren Beitrag bezahlt haben, wie hoch das Gesamteinkommen des laufenden Jahres bisher war und wieviel Umsatz noch erforderlich ist. Allerdings müssen nicht alle numerischen Daten in Zahlenfeldern gespeichert werden. Telefonnummern sind ein Beispiel für Zahlen, mit denen nie Rechenope-

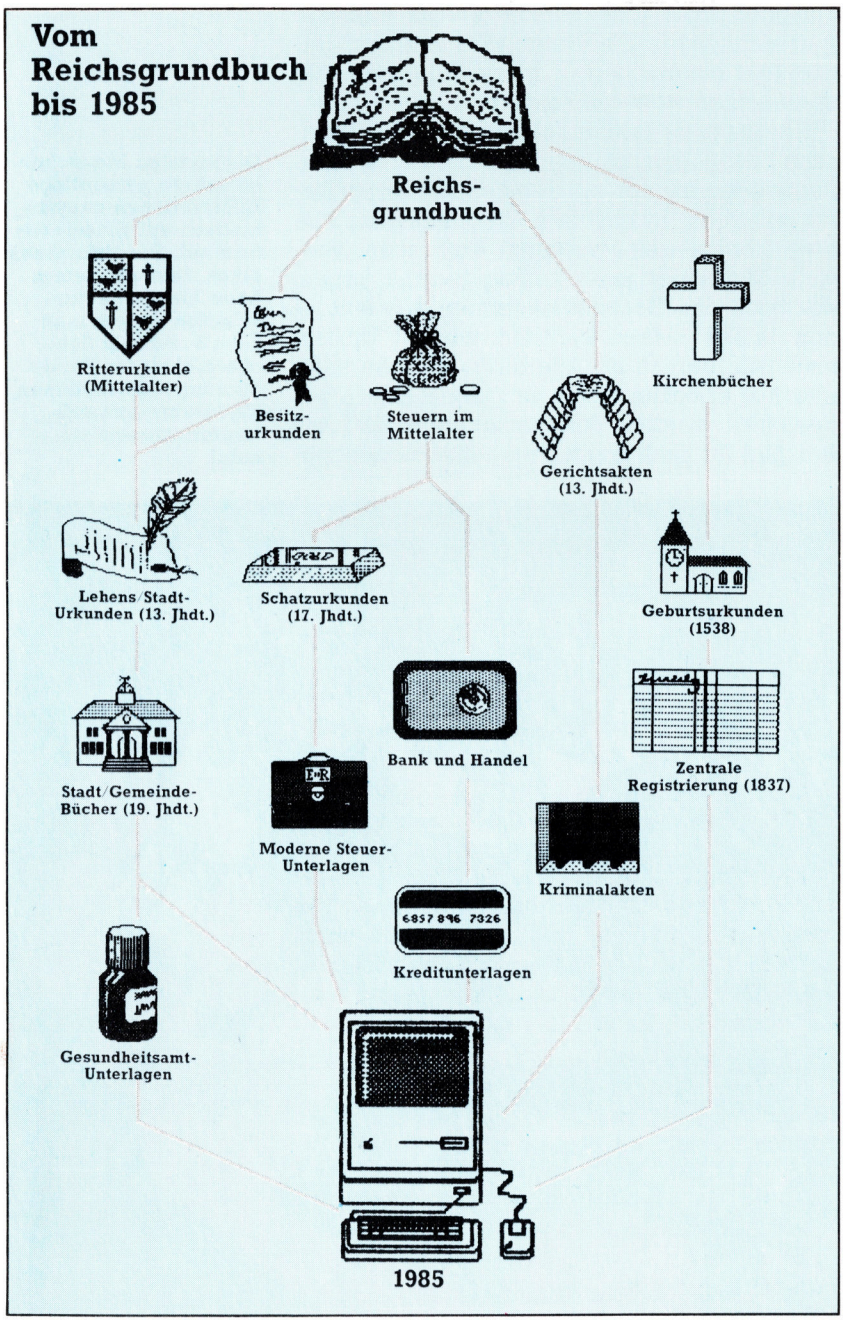
Die meisten Menschen bewahren persönliche Informationen unsystematisch auf Einzelblättern auf. Die Datenbank eines Heimcomputers kann hier Ordnung schaffen, indem man sie als zentrale Speichereinheit für Versicherungsinformationen, Eigentumsnachweis, Kontonummern etc. nutzt.

Datenbanken werden in technologisch orientierten Gesellschaftsformen eingesetzt. In der Vergangenheit waren Datenbanken von andern zeitlich und örtlich getrennt, heute aber stehen so viele private und öffentliche Datenbanken auf Computerbasis zur Verfügung, daß die Möglichkeit des unrechtmäßigen Zugriffs eine Bedrohung darstellen kann.

rationen durchgeführt werden. Sie werden also in Zeichenfeldern gespeichert.

Bevor man die Effektivität eines Files beurteilen kann, ist eine Schätzung der maximalen File-Größe erforderlich (also eine Festlegung, wieviele Aufzeichnungen enthalten sein sollen), ferner eine Berechnung des benötigten Speichers für jede Aufzeichnung und schließlich die Berechnung, ob genügend Kapazität zum Speichern des Files zur Verfügung steht. Eine aus Namen und drei Adressenfeldern von jeweils 30 Zeichen, sowie eine zehn Zeichen umfassenden Telefonnummer bestehende Aufzeichnung, benötigt circa 130 Bytes. Verfügt man über ein System mit 200 KByte Speicher, lassen sich auf jeder Diskette 1500 Daten festhalten. Bei einem System auf Cassettenbasis von 48 KByte hingegen wären das wahrscheinlich nur 300 Daten, da zehn KByte für das Programm und das Betriebssystem benötigt werden. Für die Praxis bedeutet das: Will man das File sortieren oder ähnliche Vorgänge ausführen, benötigt man zusätzliche Kapazität. Es ist deshalb empfehlenswert, nur die Hälfte des verfügbaren Speicherplatzes zu belegen. Die beiden Speichersysteme unterscheiden sich deshalb so grundlegend, weil auf Band gespeicherte Files als Gesamtheit in den Arbeitsspeicher gelesen und auch so verarbeitet werden müssen, wogegen die Geschwindigkeit der Diskette eine Teilverarbeitung bzw. ein Teilladen erlaubt.

Ein Datenbank-Verwaltungs-System bietet die Möglichkeit, Informationen von einem File zu nehmen und diese mit Informationen aus einem anderen File zu vergleichen oder auch zu mischen.



Sicherheitskopien erstellen

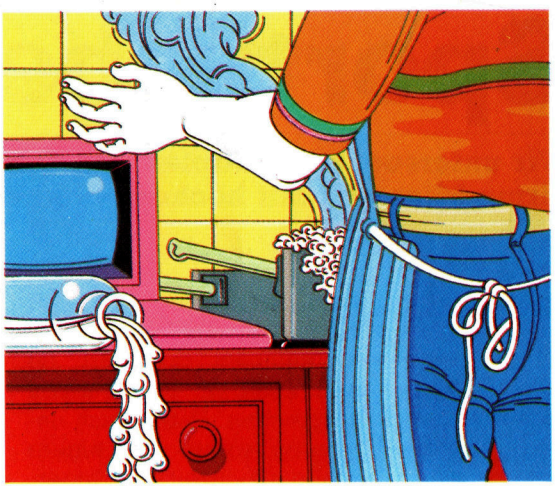
Auf dieser Basis könnte eine Entscheidung über Aktivierung von Mitgliederwerbung oder Verwendung der Einnahmen für einen neuen Club-Computer getroffen werden. Ebenso ist die Erstellung eines Standardbriefes für die Clubmitglieder möglich. Außerdem könnte das Datenbank-System Namen und Adressen verarbeiten, sie auf Umschläge oder Selbstklebeetiketten drucken. Darüber hinaus ist eine Verbindung mit einem Textverarbeitungssystem zwecks Erstellung von Briefen und Berichten durchaus möglich.

Eine große Gefahr bei jeder Datenverarbeitung ist die, daß man wichtige Informationen in einen Rechner gegeben hat und diese durch einen dummen Zufall verliert. Deshalb gilt grundsätzlich, daß man Sicherheitskopien, sogenannte „Back Ups", wenn möglich auch regelmäßig Ausdrucke von Datenfiles macht, damit nichts verlorengeht.

Ein einfaches Datenbank-Programm kann nur ein einzelnes Informations-File verarbeiten. Das System sollte die Dateneingabe in ein File so einfach wie möglich machen und sollte Zugang zu den Informationen sowohl auf dem Bildschirm wie über den Drucker erlauben. Der Anwender sollte auch die Möglichkeit haben, einzelne Aufzeichnungen durch bestimmte Suchbegriffe und Kriterien aufrufen zu können, wie etwa „alle Mitglieder, die noch keinen Jahresbeitrag gezahlt haben und über ein Postfach verfügen". Ferner sollte das Programm zudem bestimmte Felder wie Namen und Adressen für den Postversand ausdrucken können.

Neben den bereits erwähnten Einsatzmöglichkeiten können Datenbank-Programme zum Beispiel für das Erfassen der Schallplattensammlung, Bücher, bibliografischer Hinweise und Briefmarkensammlungen bis hin zu Fußballergebnissen und Bundesligatabellen verwendet werden.

Erich Jedermann hat einen VC 20, von dem er sehr begeistert ist. Durch ihn hat er Computergrundlagen erlernt und sich einige BASIC-Programmierkenntnisse angeeignet. Doch leider kann Erich nicht zwischen ernsthaften Applikationen und Spaß mit dem Computer unterscheiden. Er nutzt ihn zur Erfassung der Telefonnummern seiner (wenigen) Freunde und zur Sammlung von Rezepten. Besucht man ihn, um bei ihm zu essen, darf man bis Mitternacht warten, weil Erich darauf besteht, die Rezeptzutaten auf seinem VC 20 erst zu überprüfen. Er wirft ihn dramatisch erwartungsvoll an, lädt das Programm von Cassette – wozu überlicherweise mehrere Versuche erforderlich sind –, wartet fröhlich, während der Computer sucht und lädt, sieht sich die wenigen Anweisungen auf dem Bildschirm an und eilt dann schließlich in die Küche. Der Besucher stellt inzwischen den Computer ab und schaltet den Fernseher ein, um den Hunger zu vergessen. Es macht also keinen Sinn, ihn zu später Zeit zu besuchen, da Erich unbedingt die Möglichkeiten des Computers demonstrieren will. Erich macht das Spaß, aber er nutzt seinen Computer nicht effizient.



Lieschen Normalverbraucher hat eine große Schallplattensammlung und macht durch einen Disco-Service ein recht gutes Geschäft damit. Aus Erfahrung weiß sie, daß für bestimmte Veranstaltungen die richtigen Songs benötigt werden. Lieschen hat sich kürzlich entschlossen, einen Computer zu kaufen. Da es auf Schnelligkeit und Verarbeitung vieler Daten ankommt, hat sie sich für zwei Diskettenstationen entschieden. Dann hat sie ein Programmprinzip für eine Datenbank ausgearbeitet, um die verschiedenen Musik-Kategorien zusammenfassen zu können. Es gibt zwei Hauptgruppen, die weiter untergliedert sind. Der Hauptunterschied besteht zwischen „schwarzer" und „weißer" Musik. Schwarze Musik ist weiter untergliedert in Reggae, Jazz und Soul, weiße dagegen in „Rock" und „Neue Welle". Der „weiße" Rock gliedert sich in „Heavy Metal", „Progressiv" usw. Lieschen kann mit ihrer Datenbank alle Platten nach bestimmten Suchkriterien abfragen und aussuchen. Beispielsweise, wenn sie die Titel aller Heavy Metal-Platten haben will: Hat sie die betreffenden Stücke gefunden, läßt sie diese ausdrucken und wählt dann die Platten aus.

Ein mit einem Datenbank-Programm ausgestatteter Computer ist für Aufgaben sinnvoll, bei denen Informationen schnell sortiert bzw. durchgesucht werden müssen. Doch für solche Applikationen ist nicht jedes Datenbank-System geeignet. Zudem ist es unrationell, ein Telefunnummern-File jedesmal auf einem Heimcomputer zwecks Auffinden einer bestimmten Rufnummer zu durchsuchen, wenn man anrufen will. In der Zeit, die man fürs Einschalten des Computers, Laden des Programms, Aufruf des Files und den Suchvorgang benötigt – um eine Rufnummer zu finden, kann man mindestens sechsmal angerufen haben, weil man die Nummern im Telefon- oder Notizbuch schneller findet. Das Speichern von Namen für andere Zwecke mag sinnvoller sein, etwa um Anschriften auszudrucken oder Standardbriefe zu schreiben. Doch einen Computer zu verwenden, wenn es zeitsparendere manuelle Möglichkeiten zur Lösung derselben Aufgabe gibt, ist sinnlos.

Höher entwickelte Datenbanksysteme bieten mehr Möglichkeiten, so auch die, Rechenoperationen zur Ermittlung der gesamten Clubeinnahmen durchzuführen oder die Berechnung, welche Clubeinrichtung am häufigsten benutzt wird. Verschiedene Informations-Files lassen sich so miteinander verbinden, daß verwandte Bereiche gemeinsam benutzt werden können.

Die Nutzung von Datenbank-Systemen hat in den vergangenen Jahren merklich zugenommen. Ärzte verwenden sie, um so die Krankheitsgeschichte ihrer Patienten zu erfassen, und Forscher setzen sie zum Sortieren von Daten und zur Erstellung von Kreuzverweisen ein. Geschäftsleute erarbeiten damit Adreßlisten und geben Kundeninformationen. Heute können Datenbanken Informationen aller Art verarbeiten, von einfachen Listen bis zu komplexen Übersichten mit „Mehrfach-Files", die zudem die Möglichkeit von Auszugsberichten bieten und die Umwandlung von Informationen in eine Form erlauben, mit der sie in andere Computerprogramme wie etwa Textverarbeitungsprogramme integriert werden können. Die Kommunikation mit öffentlichen Datenbanken gibt dem Computer-Benutzer Zugang zu zahlreichen Informationsquellen.

Sportliche Daten

Wir untersuchen Multiplan, ein Tabellenkalkulationsprogramm der Firma Microsoft mit hochentwickelten Fähigkeiten.

Microsoft hat viele Eigenschaften früherer Kalkulationssysteme in das Tabellenkalkulationsprogramm Multiplan integriert. So kann Multiplan unter anderem Feldgruppen über die Namen ansprechen, Daten nach unterschiedlichen Kriterien sortieren, mehrere Ausschnitte der Tabelle gleichzeitig anzeigen, die Tabellendaten schnell durchsuchen und den gewünschten Wert anzeigen. Ursprünglich lief Multiplan nur auf kommerziellen Maschinen wie dem IBM PC oder Apple II, seit kurzem gibt es das System jedoch auch für den Commodore 64.

In unserem Modell setzen wir Multiplan zur Vereinfachung statistischer Darstellungen ein. Die gezeigten Daten beziehen sich auf das amerikanische Fußballsystem. Das Beispiel läßt sich jedoch auch für andere Sportarten verwenden.

Nach dem Laden von Multiplan steht eine Tabelle mit dem Standardformat von 63 Spalten und 255 Zeilen zur Verfügung. Zeilen und Spalten sind numeriert, wobei die linke obere Ecke als R1C1 bezeichnet wird – Row (Zeile) 1 Column (Spalte) 1. Im unteren Bildschirmviertel befindet sich das Befehlsmenü, und der Cursor steht auf der ersten Bearbeitungsmöglichkeit (Alpha). Die Befehle lassen sich entweder über die jeweils ersten Buchstaben abrufen oder durch das Setzen des Cursors auf das Kommando und anschließendem Return.

Die ersten beiden Zeilen der Tabelle enthalten die Überschriften. Wir formatieren daher die Felder von R1C1 bis R2C5 für die Texteingabe, so daß wir hier nicht an die Feldgrenzen gebunden sind:

F(ormat) C(ells) R1C1:R2C5

Danach wird der Cursor auf das Wort Cont gestellt und Return gedrückt. Der Doppelpunkt zeigt einen Zellenbereich an. In diesem Modell werden weiterhin einige Spalten verbreitert bzw. verkleinert, um sie den unterschiedlichen Datenformaten anzupassen.

Die Tabelle besteht aus zwei Hauptteilen: Ein Teil enthält die Informationen für ein bestimmtes Team über eine Periode von neun Wochen, während der andere die Gewinne oder Verluste für alle Teams der gleichen „Conference" darstellt (siehe rechte Spalte). Wenn das Gerüst des Modells fertig ist, werden die Wochendaten eingegeben. Anschließend berechnen die Formeln automatisch die Gesamtwerte der Spielsaison.

Die SORT-Funktion

In die erste Tabelle des Modells werden die wöchentlichen Spielergebnisse der Teams eingetragen, wobei Teamnamen, Kategorien und Ergebnisse anfangs in der Reihenfolge ihrer aktuellen Ligaposition stehen. Die Tabelle läßt sich jedoch mit Hilfe der Funktion SORT nach verschiedenen Kriterien sortieren.

Als Beispiel der SORT-Funktion sollen die Namen der Teams in alphabetischer Reihenfolge geordnet werden. Nach der Eingabe von S für SORT gibt Multiplan die folgende Zeile auf dem Bildschirm aus:

SORT by column:__between rows:__and:__
order:><

Hier sollen nun die Zeilen 7 bis 21 nach Kriterium „Spalte 1" in aufsteigender Reihenfolge sortiert werden. Sobald die Eingabe mit Return abgeschlossen wurde, ordnet Multiplan die Namen neu und justiert dabei auch die zugehörigen Daten. So wandern etwa alle zu Miami

Liga-Daten

Team-Daten

Sortierte Tabelle

gehorenden Zahlen ebenfalls in die neue Zeile von Miami. Bei Eingabe einer anderen Spalte als Sortierkriterium können die Teams nun nach der erreichten Punktzahl, der Trefferzahl der Gegner etc. geordnet werden.

Wenn Sie die Bildschirmanzeige mit den Cursortasten „scrollen", sehen Sie den zweiten Teil des Modells, der den Überblick über die Ergebnisse eines einzelnen Teams enthält. Hier wurden zwei Formeln eingesetzt: Zunächst soll die Funktion SUM das Gesamtergebnis der Wochenwerte in der Spalte TOTALS (R24C12) anzeigen. Da diese Spalte die Gesamtwerte aller Kategorien enthalt und den Bereich R25C12 bis R32C12 berechnen soll, muß man relative Feldadressen einsetzen, um die Formel problemlos kopieren zu können. Multiplan verwendet zu diesem Zweck den Cursor als Zeiger auf die aktiven Felder.

Gehen Sie mit dem Cursor auf R24C12, und geben Sie folgende Formel ein.

=SUM(

Drücken Sie nun den Linkspfeil, bis der Cursor auf R25C3 steht. Geben Sie einen Doppelpunkt ein, um dem Programm mitzuteilen, daß Sie einen Bereich kennzeichnen möchten. Der Cursor kehrt nun automatisch zu dem Feld mit der Formel zurück. Drücken Sie einmal den Linkspfeil und dann Return. Die Formel sollte anschließend

=SUM(R[−9]C:R[−1]C)

lauten, während gleichzeitig die Gesamtwerte des angegebenen Bereiches erscheinen. Kopieren Sie die Formel nun in die Zeilen R26C12 bis R32C12, indem Sie den Cursor auf der Formel lassen und den Befehl Copy aufrufen:

C(opy) d(own) 7 rows

Legen Sie mit dem gleichen Vorgang die Gesamtwerte von YDS, RUSH und YDS PASS an. Die SUM-Formel in Feld R27C3 zeigt die gewonnenen Yards an, während R31C3 die an das gegnerische Team verlorenen Yards enthält. Nachdem die Formel auch in die rechten acht Spalten kopiert wurde, enthält unsere Tabelle die gesamte Periode von neun Wochen.

Die zweite Formel ist etwas komplizierter: Multiplan soll mit einer IF-Anweisung entscheiden, ob beim Vergleich der Gesamtpunkte in den Kategorien „Points Scored" (gewonnene Punkte) und „Points Allowed" (verlorene Punkte) ein Spiel gewonnen oder verloren wurde. Der Befehl dazu sieht folgendermaßen aus. If Points Scored > Points Allowed, print WIN, else print LOSS.

Fensteranzeige

Da wir auch hier relative Feldadressen einsetzen müssen, stellen wir den Cursor wiederum auf die Felder mit den zu vergleichenden Werten. Gehen Sie daher mit dem Cursor auf das Feld mit der Überschrift WIN/LOSS (R34C3) und geben Sie ein:

IF(R[−6]C>R[−2]C,"WIN","LOSS")

Beachten Sie, daß Text innerhalb der Formel in doppelte Anführungszeichen eingeschlossen ist und die Bedingungen mit Klammern umgeben wurden. Kopieren Sie diese Formel wie zuvor über die Zeile, um die Daten für neun Wochen darzustellen. Wegen der Bildschirmbegrenzungen können Sie aber weder die Bezeichnungen auf der linken Seite der Tabelle noch die Gesamtwertung auf der rechten Tabellenseite sehen. Der Bildschirm läßt sich jedoch in zwei Fenster unterteilen, die gemeinsam oder unabhängig voneinander verschoben werden können.

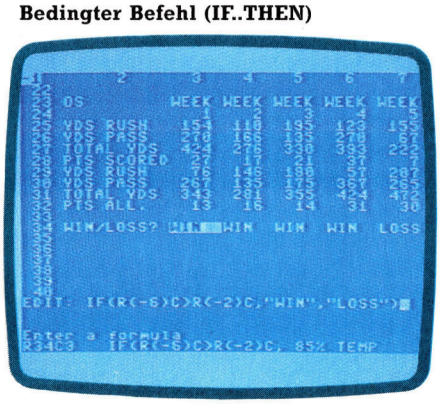
Zur vertikalen Teilung des Bildschirms bei Spalte 3 drücken Sie W(indow) und S(plit), gefolgt von V(ertical). Multiplan stellt dann folgende Zeile dar:

WINDOW SPLIT VERTICAL
at column:_linked YES/NO.

Geben Sie hier Spalte 3 an, setzen Sie den Cursor auf NO und drücken Sie Return. Jetzt lassen sich die Fenster unabhängig voneinander bewegen, wobei die Kategorien ständig angezeigt werden. Zum „Ausschalten" eines Fensters geben Sie W(indow) C(lose) ein, gefolgt von der Fensternummer.

In der nächsten Folge beschäftigen wir uns mit einem der erfolgreichsten Kalkulationssysteme – Lotus 1–2–3.

Bedingter Befehl (IF..THEN)



Fensteranzeige

Hier eine kurze Einführung in die Terminologie des amerikanischen Fußballs. Zwei Teams mit je 11 Spielern versuchen abwechselnd einen länglichen Ball über eine Ziellinie zu befördern. Die beiden Tore sind 100 Yards (ca. 91 Meter) voneinander entfernt. Der Spieler kann den Ball entweder tragen, als Paß werfen oder zwischen die Torpfosten kicken. Ein Kick (Field Goal) wird mit drei Punkten bewertet; sechs Punkte werden gezählt, wenn der Ball über die Linie getragen oder als Paß geworfen wird (Touchdown), während ein Kick nach einem Touchdown einen Punkt bringt (Point after Touchdown).

Jedes Team hat vier Versuche (Downs), um den Ball dem gegnerischen Tor mindestens zehn Yards näherzubringen. Bei Erfolg gibt es weitere vier Versuche. Das Tragen des Balles wird „Rush" genannt. Die Anzahl der Yards, die der Ball per Rush auf das gegnerische Tor hinbewegt wurde, gibt an, wie weit der Ball getragen wurde, während die Yardzahl der Pässe anzeigt, wie weit der Ball geworfen wurde.
In der „National Football League" (der amerikanischen Profiliga) gibt es zwei „Conferences" – die „American Conference" und die „National Conference". Pro Saison bestreiten die Teams 16 Spiele.

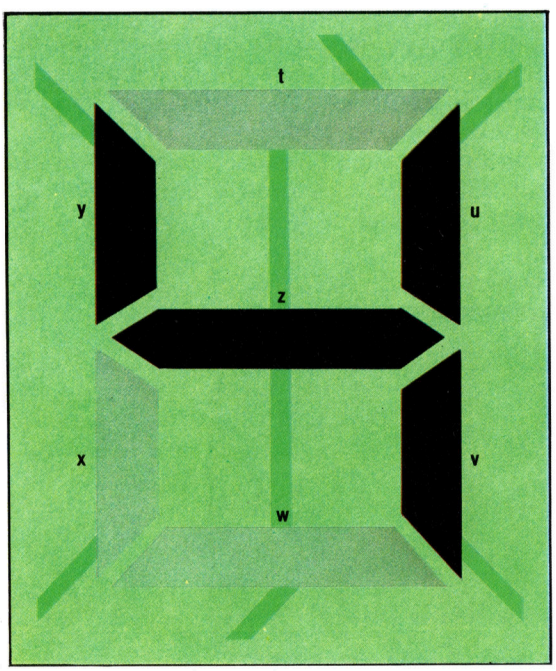
Zahlenspiele

In diesem Teil des Logik-Kurses beschäftigen wir uns mit der Konstruktion eines „Binär-Sieben-Segment-Wandlers". Diese Schaltung setzt die binären Signale des Computers in lesbare Ziffern um.

Unsere Schaltung eignet sich für die Umsetzung aus dem BCD-Code. BCD ist ein Akronym für „Binary Coded Decimal", also „Binär Codierte Dezimalziffern". Dieser Code ordnet jeder Ziffer von 0 bis 9 einen bestimmten Vier-Bit-Wert zu:

BCD	Dezimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010 to 1111	bis ungültig

Eine Sieben-Segmentanzeige zeigt Dezimalziffern durch das Einschalten bestimmter Leuchtdioden, bei Flüssigkristallanzeigen (LCDs) wird die Spannung an einzelnen Streifen umgepolt. In der Abbildung wurden die Einzelsegmente mit den Buchstaben t bis z bezeichnet. Die Tabelle weiter unten zeigt, welche Kombination von Segmenten zur Darstellung einer Dezimalziffer (0 bis 9) eingeschaltet sein muß.



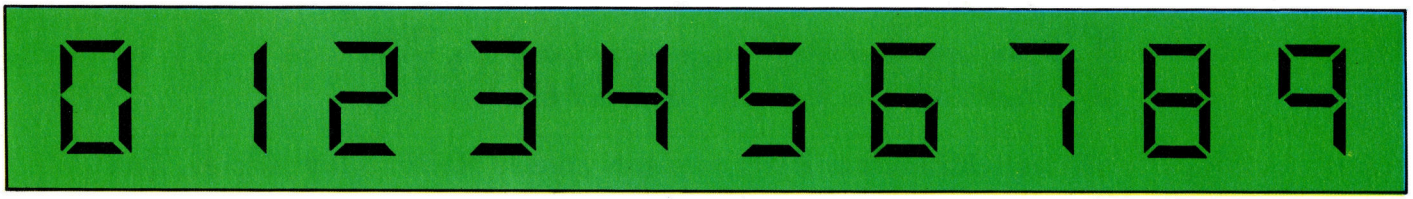
Die LCD-Anzeige besteht aus sieben separat schaltbaren Streifen, die hier mit den Buchstaben t bis z bezeichnet wurden.

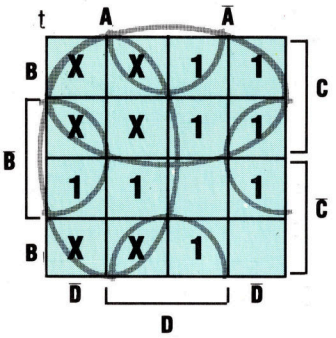
Die vorhandenen Daten genügen bereits zur Aufstellung der Wahrheitstabelle des Wandlers. Das Bild auf der nächsten Seite zeigt, welche Eingabe die Aktivierung einer bestimmten Segmentkombination auslöst. Der binäre Eingabewert 0100 (dezimal 4) schaltet beispielsweise die Segmente u,v,y und z ein, die Eingabe von 1000 (dezimal 8) aktiviert alle Segmente gleichzeitig. Die Wahrheitstabelle:

Dezimal	Eingänge				Ausgänge						
	A	B	C	D	t	u	v	w	x	y	z
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
	1	0	1	0	X	X	X	X	X	X	X
	1	0	1	1	X	X	X	X	X	X	X
	1	1	0	0	X	X	X	X	X	X	X
	1	1	0	1	X	X	X	X	X	X	X
	1	1	1	0	X	X	X	X	X	X	X
	1	1	1	1	X	X	X	X	X	X	X

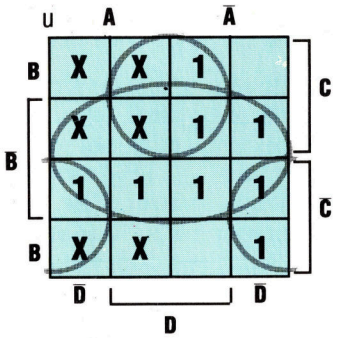
Eine Analyse der Wahrheitstabelle ist leider unmöglich – die Ausgänge t,u,v,w,x,y und z müssen also jeder für sich vereinfacht werden. Dazu brauchen wir sieben verschiedene Karnaugh-Tafeln, die jeweils die Bedingung „Ignorieren" (X) enthalten. Sie sind auf der nächsten Seite abgedruckt. Durch Zusammenfassung der eingekreisten Gruppen lassen sich diese Ausdrücke vereinfachen, und Faktorisierung führt in einigen Fällen zu einer weiteren Reduktion der Ausdrücke.

Die erste k-Tafel enthält alle Ausdrücke, für die die „t"-Balken benötigt werden – es gibt nur zwei Ziffern, in denen „t" nicht benötigt wird. Sind die Formeln für jeden Ausgang vereinfacht, geht es mit dem eigentlichen Schaltungsentwurf weiter.

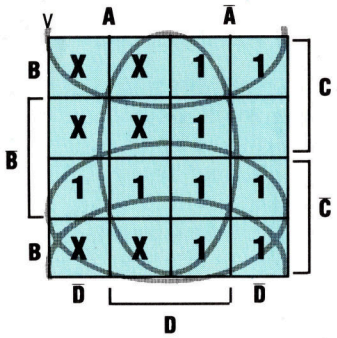




t=C+A+B̄.D̄+B.D

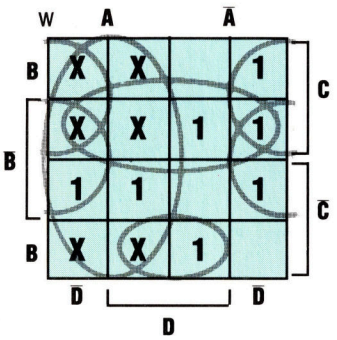


u=B̄+C.D+C̄.D̄

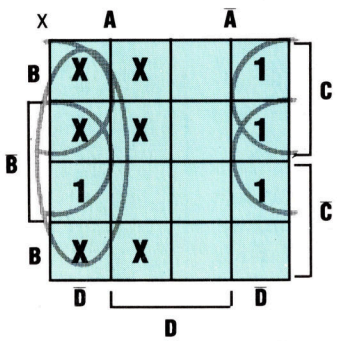


v=D+C̄+B

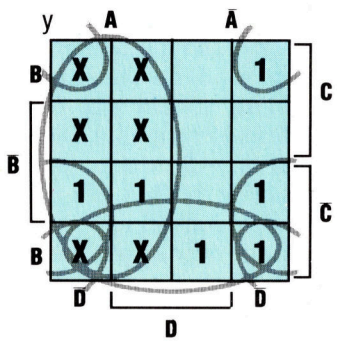
Aus den Tafeln können Sie ersehen, welche der Segmente für die einzelnen Ziffern eingeschaltet sein müssen. Für die „1" werden nur die Segmente „u" und „v" gebraucht.



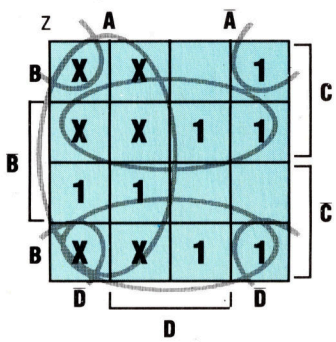
w=A+C.D+B.C+B.D+B.C̄.D̄.



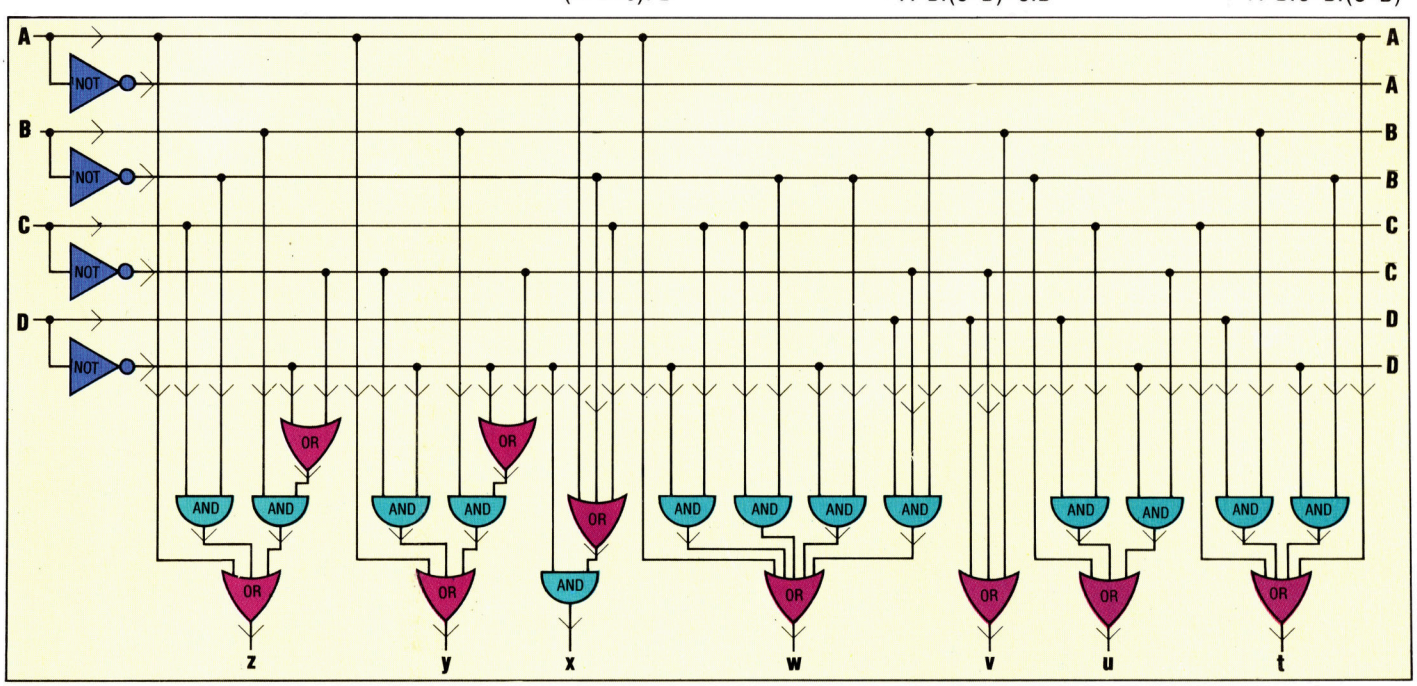
x=C.D+A.D+B.D̄
=(A+B̄+C). D



y=A+B.C̄+B.D+C̄.D̄
=A+B.(C̄+D)+C̄.D̄



z=A+B̄.C+B.C̄+B.D̄
=A+B̄.C+B.(C+D)



Die hier angegebene Schaltung kann nur eine Sieben-Segment-Anzeige steuern, Anzeigefelder bestehen aber meist aus acht bis zehn Einzelziffern. Trotzdem brauchen Sie dank des Multiplex-Verfahrens auch für mehrstellige Anzeigen nur einen Wandler, der die einzelnen Sieben-Segment-Anzeigen nacheinander steuert. Läuft dieser Vorgang schnell genug ab, erscheint die Anzeige völlig stabil – durch die hohe Schaltgeschwindigkeit läßt sich nicht erkennen, daß jede Anzeige immer nur für kurze Zeit aktiv ist.

Reaktionszeit

Mit zunehmendem Alter werden unsere Reflexe langsamer. Das folgende Programm ermöglicht Ihnen, Ihre Reaktionszeit zu messen.

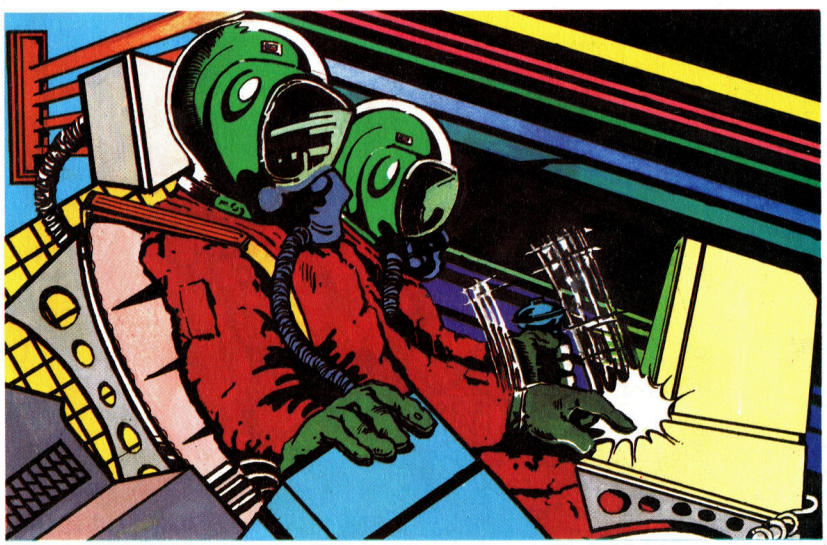
Die meisten Menschen denken, daß sie ein sehr gutes Reaktionsvermögen haben. Jedoch wird dieses durch viele Faktoren beeinflußt. So führt zum Beispiel Alkoholgenuß, Müdigkeit oder Krankheit zu einer Verlangsamung der Reaktionsgeschwindigkeit. Das hier gezeigte Programm kann die Reflexe jeder beliebigen Person testen.

Im Spiel übernehmen Sie die Rolle eines Raumschiff-Kommandanten, der wichtige Medikamente zu einer Kolonie im Asteroiden-Gürtel bringen muß. Geschwindigkeit ist lebenswichtig, wenn die Medikamente rechtzeitig ankommen sollen. Das Raumschiff muß jedoch anhalten, sobald sich eine Kollision mit einem Asteroiden anbahnt. Dies geschieht einfach durch das Drücken der Leertaste. Das Programm stellt die Zeit, die zwischen dem Erscheinen des Asteroiden und dem Drücken der Leertaste vergangen ist, auf dem Bildschirm dar. Wenn Sie das Raumschiff fünfmal gestoppt haben, wird Ihre durchschnittliche Reaktionszeit berechnet.

Mogeln verboten

Um zu gewährleisten, daß nicht „gemogelt" wird, überprüft das Programm, daß die Leertaste nicht konstant gedrückt wird – bei einem derartigen Versuch werden die Maschinen gestoppt, und es ertönt ein Warnton. Gleichzeitig mit dem Erscheinen des Asteroiden auf dem Bildschirm produziert das Radar Ihres Schiffes einen hohen Ton. Als Übung versuchen Sie doch einmal, das Programm in ihrem Rechner so abzuändern, daß Sie entweder eine hörbare oder eine sichtbare Warnung erhalten, aber nicht beides zusammen.

Eine weitere Möglichkeit wäre, zwischen zwei oder mehr Optionen auswählen zu können. Dadurch würde das Programm mehr zu einem Beurteilungstest und nicht nur ein einfacher Indikator von Reaktionszeiten.



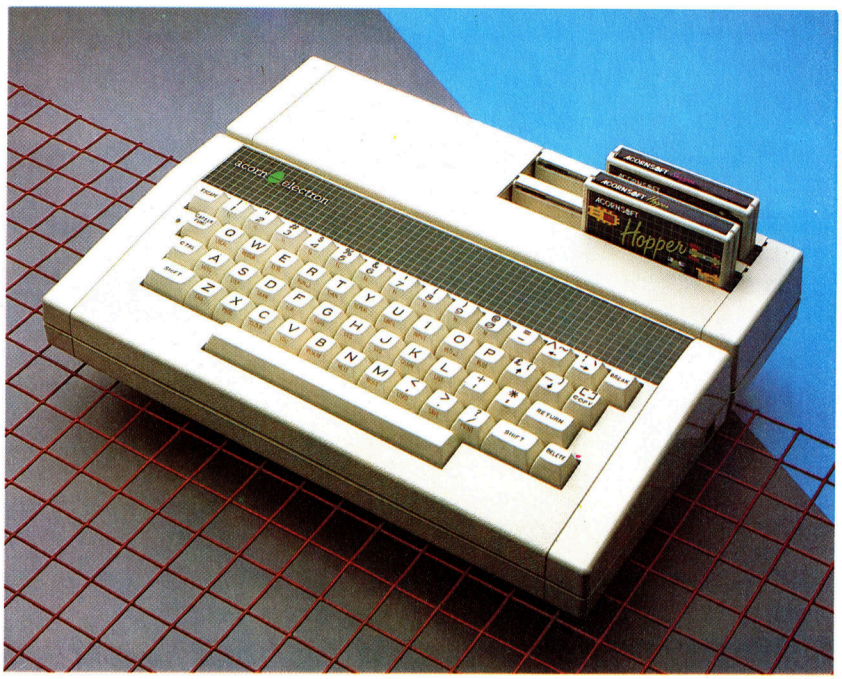
Vollbremsung

```
 10 REM *FOR THE SPECTRUM
 20 INK 7: PAPER 0: BORDER 0
 30 FOR T=0 TO 31: READ A
 40 POKE USR "A"+T,A: NEXT T
 60 DATA 123,231,255,255,245,233,123,100
 70 DATA 34,187,208,49,86,96,178,205
 80 DATA 67,234,45,123,198,255,29,245
 90 DATA 78,100,245,60,50,160,245,189
150 LET M=0: LET A=0: LET V=0: DIM R(5): CLS
220 FOR G=1 TO 5
250 FOR P=1 TO RND*150+50
260 PLOT INT (RND*250),INT (RND*170)
270 IF INKEY$<>" " THEN  GO TO 280
275 BEEP .1,-3: GO TO 265
280 NEXT P
290 FOR A=144 TO 147
320 PRINT AT RND*20,RND*30;CHR$ (A)
330 NEXT A
350 BEEP .05,15: LET C=0
360 LET C=C+1: LET A$=INKEY$
410 IF A$<>" " THEN  GO TO 355
420 LET R(G)=C/66+.05
430 PRINT AT 19,0;"YOU TOOK ";INT (R(G)*100)/100;
440 PRINT " SECONDS TO STOP  "
445 PRINT "THE ENGINES "
450  LET M=M+R(G)
453 FOR J=1 TO 300: NEXT J
455 NEXT G
460 LET A=M/5: CLS
480 PRINT "OVER FIVE TURNS YOUR AVERAGE "
490 PRINT "REACTION TIME WAS ";INT (A*100)/100;
495 PRINT " SECS  "
500 FOR G=1 TO 5: LET V=V+ABS (R(G)-A)
510 NEXT G
520 PRINT : PRINT "YOUR REACTION TIME VARIED BY"
530 PRINT INT (V*20/A);"  PER CENT"
540 PRINT : PRINT "DO YOU WANT ANOTHER GO? (Y/N)"
550 LET R$=INKEY$: IF R$="y" THEN  GO TO 150
560 IF R$<>"n" THEN  GO TO 550
570 INK 0: PAPER 7: BORDER 7: CLS
```

```
 10 REM *FOR THE BBC MICRO AND ELECTRON
 20 MODE 1:CLS:VDU 23;8202;0;0;0:DIM REACT(5)
 30 REM *SET UP CHARACTERS (ASTEROIDS)
 40 FOR J=0 TO 3
 50 VDU 23,237+J,78,100,245,60,50,160,245,189
 60 VDU 23,233+J,89,67,34,156,123,200,256,29
 70 NEXT J
 80 SUM=0:AVERAGE=0:VARY=0:CLS
 90 FOR GO=1 TO 5
100 REM *PLOT STARS
110 FOR DELAY=1 TO 100+RND(300)
120 PLOT 69, RND(1200),RND(1012)
130 IF INKEY(-99)=0THEN 150 ELSE
140 SOUND 1,-15,5,1:GOTO 130
150 NEXT DELAY
160 REM *PLOT ASTEROIDS AROUND SCREEN
170 FOR ASTEROID =233 TO 240
180 X=RND(28):Y=RND(28)
190 PRINT TAB(X,Y);CHR$(ASTEROID)
200 NEXT ASTEROID
210 SOUND 2,-15,150,3
220 REM * TEST SPACE BAR
230 TIME=0
240 IF INKEY(-99)=0 THEN 240
250 REACT(GO)=TIME/100
260 PRINT TAB(3,26);"YOU TOOK ";
270 PRINT;REACT(GO);" SECONDS   "
280 PRINT TAB(3,28);"TO STOP THE ENGINES"
290 FOR I=1 TO 500:NEXT I
300 SUM=SUM+REACT(GO)
310 NEXT GO
320 FORI=1 TO 3000:NEXT I
330 REM *CALCULATE RESPONSE TIME
340 AVERAGE=SUM/5
350 FOR GO=1 TO 5
360 VARY=VARY+ABS(REACT(GO)-AVERAGE)
370 NEXT GO
380 CLS:PRINT:PRINT
390 PRINT "OVER FIVE TURNS YOUR AVERAGE"
400 PRINT"REACTION TIME WAS ";AVERAGE;" SECS."
410 PRINT:PRINT"YOUR REACTION TIME VARIED BY"
420 PRINT INT(VARY*20/AVERAGE);" PER CENT"
430PRINT:PRINT"DO YOU WANT ANOTHER GO Y/N ?"
440 R$=INKEY$(0):IF R$="Y" THEN 80
450 IF R$<>"N" THEN 440 ELSE MODE 4
```

Der Acorn Plus 1

Das Interface Plus 1 verwandelt den Acorn Electron, der nur über wenige Erweiterungsmöglichkeiten verfügt, in eine preiswerte Alternative zum Acorn B.



Ursprünglich war der Electron als reduzierte Acorn-B-Version konzipiert. Acorn produziert jetzt die Plus 1 benannte Erweiterungseinheit, womit der Electron auch über die wichtigsten Schnittstellen verfügt.

Der Electron war als heruntergefahrene Acorn-B-Version entwickelt worden. Seine Konstrukteure behielten das ausgezeichnete Acorn-BASIC und das Betriebssystem bei, ließen aber die meisten Schnittstellen, die den Acorn B so vielseitig verwendbar machen, weg. Der Electron verfügt nur über ein Cassetten-Interface, zwei Monitor-Anschlüsse (RGB- oder Composite-Video), einen Modulator-Ausgang für Fernsehanschluß und den Anschluß für Stromversorgung.

Mit dem Plus-1-Interface ist der Rechner zusätzlich mit einer parallelen Druckerschnittstelle, einem Joystickanschluß sowie zwei Steckschächten für Cartridges ausgerüstet. Acorn setzt darauf, daß diese zusätzlichen Möglichkeiten den Abverkauf verbessern. Mit Modul-Schächten und Joystick-Anschluß kann der Electron nun auch für Action-Spiele verwendet werden, wogegen der Druckeranschluß für jene Anwender sinnvoll ist, die Programm-Listings ausdrucken lassen wollen oder Textverarbeitungs-Programme benutzen.

Der Anschluß des Plus-1-Erweiterungs-Interfaces ist denkbar einfach. Es wird auf die rückseitig angebrachte Steckleiste des Electron geschoben. Stromversorgung und Datenübertragung sind durch die Steckleiste gewährleistet. Die zum Betrieb des Interfaces erforderliche Software ist in einem speziellen ROM-Chip enthalten.

Parallel und analog

Beim Druckeranschluß handelt es sich um eine parallele Centronics-Schnittstelle. Der Drucker wird durch Eingabe von VDU2 aktiviert und durch VDU3 deaktiviert. Diese Befehle sind mit denen für den Acorn B identisch. Ebenso werden auch hier Zeichen durch Eingabe von VDU1 an den Drucker geschickt.

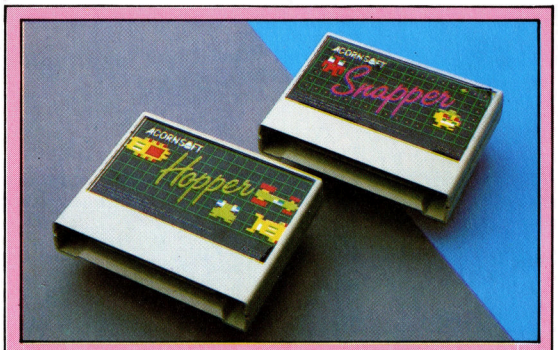
Joysticks werden an eine analoge Schnittstelle angeschlossen. Es handelt sich dabei um einen 15poligen D-Stecker. Das Plus 1 kann bis zu vier analoge Spannungen messen.

Die Umwandlung der analogen in digitale Signale, die der Computer verarbeiten kann, bewirkt ein A/D-Wandler-Chip, in diesem Fall ein ADC0844. Der Chip ist weniger ausgefeilt als das im Acorn B verwendete Gegenstück. Deshalb ist die Joystickbewegung beim Electron auch nicht so präzise wie beim Acorn B. Bei Spielprogrammen, die lediglich Joystickbewegungen in vier Richtungen erforderlich machen, ist das unproblematisch. Wird der Joystick hingegen bei Grafikprogrammen verwendet, etwa beim „freien Zeichnen auf dem Bildschirm", zeigen sich die Schwächen, und die Zeichnungen sind weniger detailliert.

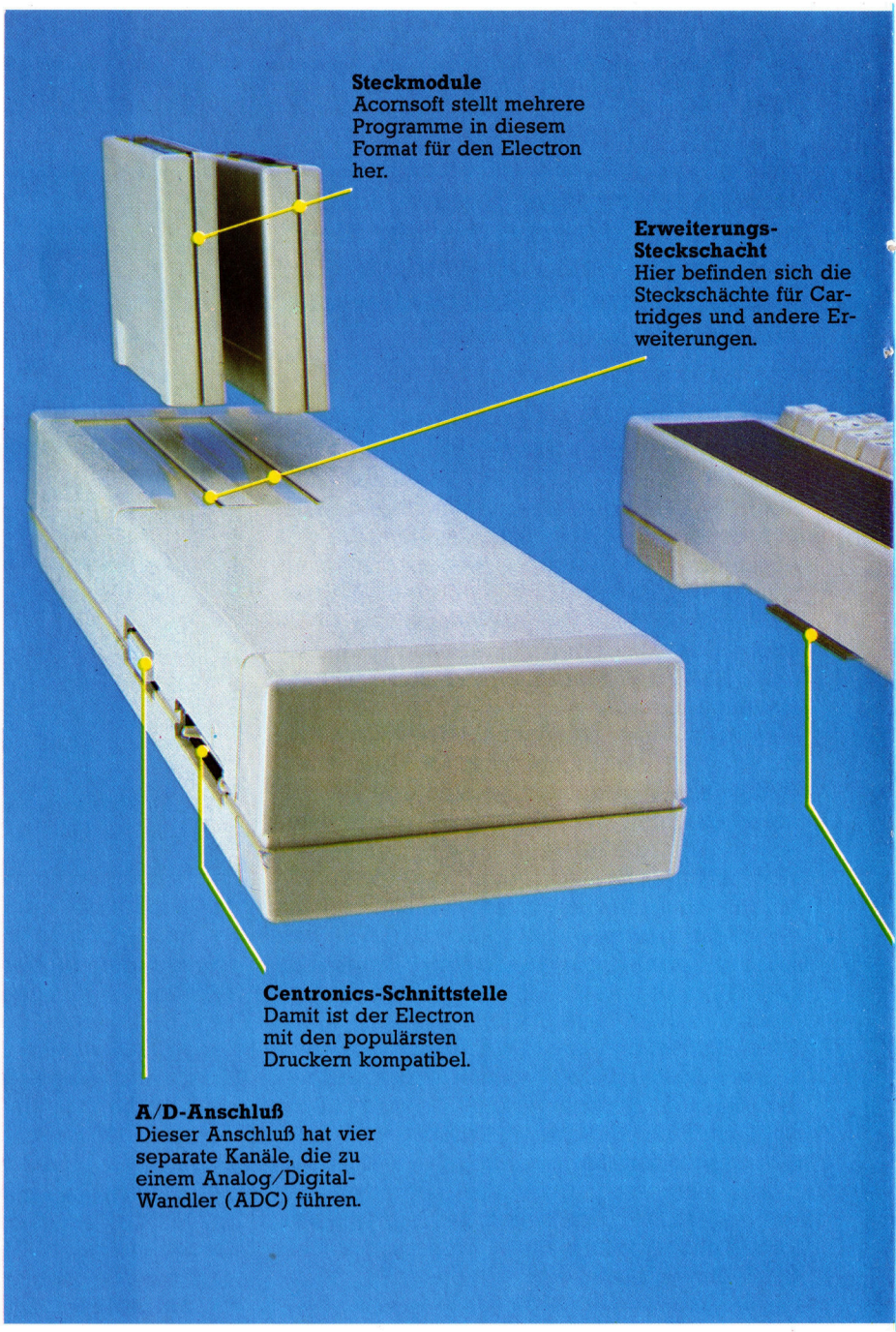
Nach dem Einschalten des Electron Plus 1 aktiviert das Modul im vorderen ROM-Schacht den Autostart, der mit „ESCAPE" gestoppt wird. Das ROM-File-System arbeitet ähnlich wie die Cassettenversion. Will man Cassetten-Software mit der Erweiterung benutzen, ist entweder der Befehl * TAPE einzugeben oder das Steckmodul zu entfernen. Dazu muß zunächst das Gerät ausgeschaltet werden. Neben Spielprogrammen werden auch andere Programmiersprachen in Modulform angeboten.

Wie bei vielen Computer-Peripherien gibt es auch hier eine Reihe unerwünschter Nebeneffekte. Ist das Plus 1 mit dem Electron verbunden, tauchen bei Verwendung von Cassettenprogrammen häufig Ladefehler auf. Das geschieht besonders oft bei Programmen, die Daten-Files enthalten. Allerdings kann man durch eine Reihe von Betriebssystem-Befehlen den Electron dahingehend „täuschen", daß für den Rechner intern das Plus 1 nicht angeschlossen ist. Dieses Verfahren wird im Handbuch lediglich beiläufig erläutert und klingt für den Anfänger völlig verwirrend.

Beim Acorn B erhält man nach Eingabe von



Sechs Software-Titel
Die Steckmodule für den Electron bieten gegenüber Cassetten den Vorteil, daß sie in nur wenigen Sekunden geladen sind. Als Steckmodule stehen nur sechs Programme zur Verfügung: Vier Spiele, ein Lernprogramm sowie die Sprache LISP.



* HELP eine Übersicht der verschiedenen ROMs des Computers. Gibt man diesen Befehl auf einem mit dem Plus 1 erweiterten Electron ein, erhält man die Meldung, daß es sich bei dem ROM-Betriebssystem um die OS 1.00-Version handelt, sowie folgende Information: Expansion 1.00 ADC/Printer/RS423. Interessant ist die letzte Information – RS423. Hierbei handelt es sich um einen Verweis auf eine serielle Standard-Schnittstelle, mit der weder der Electron noch der Plus 1 ausgestattet sind. Acorn will die serielle Schnittstelle zu einem späteren Zeitpunkt anbieten.

Teletext-kompatibel?

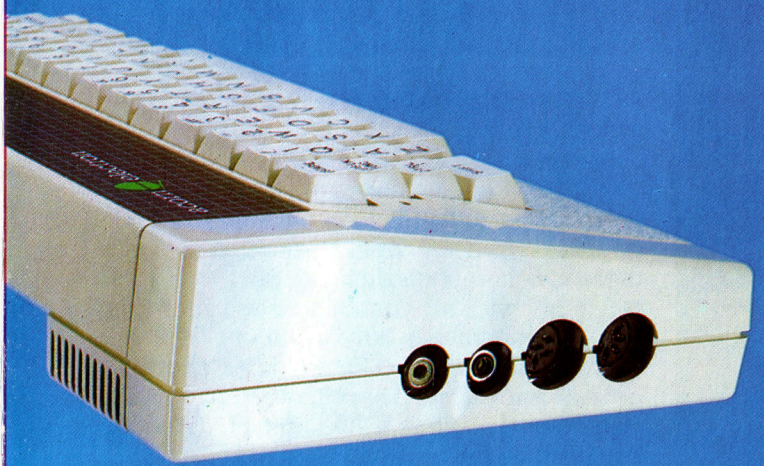
Hauptunterschied zwischen den beiden BASIC-Versionen ist, daß der Electron nicht über den Mode 7 verfügt – den Teletext-kompatiblen Grafik-Mode. Bei Programmen für den Acorn B wird er unter anderem für Titel- und Anweisungs-Seiten benutzt. Für das eigentliche Spielen wird jedoch in fast allen Programmen ein anderer Mode verwendet. Es gibt also keine wirklichen Probleme, wenn Auftaktseite und Anweisungen vom Schirm verschwunden sind.

Der andere wesentliche Unterschied wird bei den Befehlen SOUND und ENVELOPE deutlich. Der Electron verfügt nur über einen Soundkanal gegenüber den vier des Acorn B. Ähnlich beeinflußt der ENVELOPE-Befehl beim Electron lediglich die Tonhöhe, nicht aber die Lautstärke. Allerdings sind die Befehle kompatibel, so daß die entsprechenden Programme auf beiden Rechnern laufen. Der Klang ist auf dem Electron merklich anders.

Die Tastatur des Electron vermittelt ein sicheres „Schreibgefühl", hat aber weniger Tasten als die des Acorn B. Das bedeutet, daß die meisten Electron-Tasten mit drei oder vier verschiedenen Funktionen belegt sind. So erzeugt beispielsweise die L-Taste entweder „l" oder „L", abhängig davon, ob die Shift-Taste gedrückt wurde oder nicht. Wird sie aber in Verbindung mit der „Function"-Taste betätigt, aktiviert sie den BASIC-Befehl LIST. Bei gleichzeitigem Drücken der Control-Taste löscht „L" den Bildschirm. Der Electron verfügt zwar auch über Funktionstasten, doch diese sind lediglich numerische Tasten, die zusammen mit der „Function"-Taste gedrückt werden müssen.

Die beiden Rechner unterscheiden sich deutlich im Hinblick auf die Interface-Ausstattung. Der Acorn B verfügt über mehr Schnittstellen als andere Heimcomputer, so daß alle nur denkbaren Peripherien einfach angesteuert werden können. Hierfür sind zusätzliche Chips und Steckverbindungen erforderlich, die die Kosten beträchtlich in die Höhe treiben.

Snapper
Acornsofts „Snapper" ist eine Version des klassischen „PacMan". Das Steckmodul wird über Tastatur oder Joystick gesteuert. Auf Cassette ist das Programm ebenfalls lieferbar.



Erweiterungs-Steckleiste
Über diese Steckleiste wird das Plus-1-Interface mit dem Electron verbunden.

Acorn B

ABMESSUNGEN
75 × 340 × 410 mm

ZENTRALEINHEIT
6502, 1,8 MHz

SPEICHERKAPAZITÄT
32K RAM, 32K ROM

BILDSCHIRMDARSTELLUNG
8 Darstellungs-Modi. Höchste Auflösung: Text 80 × 32 Zeichen, Grafik 640 × 256 Pixel, bis zu acht Farben, Teletext-Display Mode. Der Anwender hat die Möglichkeit, Zeichen selbst zu definieren.

SCHNITTSTELLEN
UHF für Fernseher, RGB und Composite-Video für Monitore, Cassettenrecorder-Anschluß; RS423- und Centronics-Drukker-Schnittstelle; Analog-Schnittstelle (für Joysticks etc.); ROM-Schacht (für Software); Steckplatz für weitere Prozessoren; 1-MHz-Bus; Diskettenschnittstelle (wahlweise); Econet-Netzwerk-Interface (wahlweise); User Port; Hilfsstromversorgung (für Betrieb von Diskettenstationen etc.).

PROGRAMMIERSPRACHEN
BASIC (mitgeliefert), 6502 Assembler (mitgeliefert), LISP, FORTH, BCPL, PASCAL

TASTATUR
72 Schreibmaschinentasten, einschließlich zehn frei programmierbare Funktionstasten.

HANDBÜCHER
Das Handbuch ist ein ausgezeichneter Führer für den fortgeschrittenen Programmierer, hilft dem Anfänger aber wenig.

STÄRKEN
Eine der besten BASIC-Versionen, viele Schnittstellen, gute Grafik, guter Sound und ausgezeichnete Tastatur.

SCHWÄCHEN
Für den Anfänger ist der Acorn B nicht einfach zu bedienen. In Deutschland ist das Software-Angebot für den Acorn gering.

Acorn Electron Plus 1

ABMESSUNGEN
65 × 260 × 340 (mit aufgesecktem Plus 1)

ZENTRALEINHEIT
6502, 1,8 MHz

SPEICHERKAPAZITÄT
32K RAM, 32K ROM

BILDSCHIRMDARSTELLUNG
Sieben Darstellungsmodi. Höchste Auflösung: Text 80 × 32 Zeichen; Grafik 640 × 256 Pixel. Bis zu acht Farben; vom Anwender definierbare Zeichen.

SCHNITTSTELLEN
UHF-Ausgang für Fernseher; RGB und Composite-Video für Monitore; Cassettenrecorder-Anschluß; Paralleldrucker-Anschluß, Analog-Anschluß (für Joysticks etc.), zwei Steckschächte für Module.

PROGRAMMIERSPRACHEN
BASIC, 6502 Assembler (mitgeliefert), FORTH, LISP (auch auf ROM-Cartridge), S-PASCAL

TASTATUR
56 Schreibmaschinentasten, Funktionstasten ermöglichen die Eingabe von BASIC-Befehlen. Zehn programmierbare Tasten.

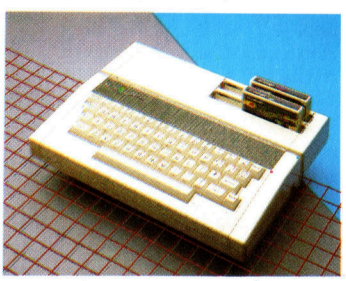
HANDBÜCHER
Gut gestaltetes und leicht verständliches Anwenderhandbuch. Ausführliche Einführung in das BASIC und in die 6502 Assembler-Sprache.

STÄRKEN
Gute Grafik und sehr gutes Bild. Ausgezeichnete Tastatur, sehr gutes BASIC.

SCHWÄCHEN
Verwirrende Mehrfach-Funktionstasten. Nur ein Sound-Kanal. Geringer Speicherplatz, kein Teletext-Mode. Der Acorn Electron Plus 1 ist im Vergleich mit anderen Computern, bei denen diese Schnittstellen integriert sind, relativ teuer.

Läßt man diese Schnittstellen außer acht, wird der Rechner natürlich billiger, da die Kosten für zusätzliche Chips entfallen und auch die Stromversorgung erheblich reduziert wird. Diese Philosophie steckte hinter dem ursprünglichen Electron-Konzept. Die Tatsache, daß Acorn nun das Plus 1 entwickelt hat, verdeutlicht, daß zu viel gespart wurde.

Der Electron Plus 1 ist weniger vielseitig als der Acorn B und bietet weniger Erweiterungsmöglichkeiten. Viele Anwender benötigen diese jedoch nicht, und für den Preis bietet der Electron Plus 1 mit ausgezeichneter Grafik und einigen Klangmöglichkeiten, strukturiertem BASIC und einer ständig wachsenden Software-Bibliothek doch viel fürs Geld.



Einsamer Stern

Texas Instruments gehört zu den Firmen, die an der Heimcomputer-Revolution großen Anteil haben. 1958 erfand ein Ingenieur des Unternehmens die integrierte Schaltung, die Basis der Microcomputer. Zwanzig Jahre später wurde der TI99/4A auf den Markt gebracht.

Der überwiegende Teil der Fertigung von Texas Instruments-Produkten erfolgt in dieser modernen, direkt an der Schnellstraße gelegenen Fabrik in Dallas, Texas.

Mr. J. Fred Bucy, Präsident und Generalmanager von Texas Instruments. Mr. Bucy ist seit April 1976 Präsident des Unternehmens.

Der 1978 eingeführte TI99/4A von Texas Instruments basierte auf dem hauseigenen TMS9900-Microprozessor und verfügte über einen Speicher von 16 KByte RAM. Der Rechner stellt 16 Farben dar und hat drei Klangkanäle. Trotz guter Verkäufe wurde der Computer Opfer des heftigen Preiskampfes, der 1982 und 1983 in den USA stattfand. Rund 2800 Mark kostete das System ursprünglich. Der Preis wurde kontinuierlich reduziert, bis er schließlich bei 320 Mark lag. Ende 1983 stellte TI die Produktion des Rechners ganz ein.

Richard Mann, europäischer PR-Manager von TI, stellt dazu fest: „Der TI99/4A wurde extrem gut verkauft, nur brachte er keinen Profit, und wir verloren mehrere hundert Millionen Dollar."

Die Produktpalette des Unternehmens umfaßt sowohl elektronisches Kinderspielzeug wie „Speak and Spell" als auch Minicomputer und hochentwickeltes elektronisches seismografisches Gerät. Weltweit verfügt TI über fünfzehn Fertigungsbetriebe und macht einen Jahresumsatz von 16 Milliarden Mark.

Texas Instruments wurde 1930 von den beiden Wissenschaftlern John Karchner und Eugen McDermott als „Geophysical Service Incorporated" gegründet. Karchner beschäftigte sich mit der Reflexion von Schallwellen auf geologischen Schichten, um so ihre Tiefenlage zu ermitteln. Um diese Idee an die Ölindustrie zu verkaufen, etablierte man das Unternehmen in Dallas, Texas.

In den dreißiger Jahren wuchs GSI ständig, entwickelte neue Techniken und Ausrüstungen für seismologische Beobachtungen. Während des Zweiten Weltkriegs erwiesen sich diese Geräte als nützlich bei der Ortung von U-Booten, was zur Gründung der Labor- und Fertigungsabteilung der GSI führte. 1951 war dieser Ableger des Mutterunternehmens so gewachsen, daß man beschloß, ihn zu verselbständigen. Das neue Unternehmen wurde Texas Instruments benannt.

Im folgenden Jahr erwarb Texas Instruments eine Lizenz zur Herstellung der gerade erfundenen Transistoren von Bell Laboratories, und 1954 baute das Unternehmen das erste Transistor-Radio der Welt. 1958 erfand einer der Ingenieure von TI, Jack Kilby, die integrierte Schaltung, und seitdem zählt TI zu den Pionieren in diesem Bereich. Kilby war auch an der Entwicklung des ersten elektronischen Taschenrechners der Welt beteiligt.

Größter Chiphersteller

Der im Januar 1983 auf den Markt gebrachte TI Professional Computer basiert auf dem 8088-Prozessor und kann deshalb sowohl mit CP/M-86 als auch mit MS-DOS arbeiten. Zum Jahresende präsentierte TI dann einen tragbaren Business Computer sowie den CC40.

Heute produziert das Unternehmen eine Reihe elektronischer Bauteile, einschließlich 64-KByte-RAM-Chips. TI ist einer der weltweit größten Hersteller dieser Chips. Wie die meisten anderen Chip-Hersteller hat auch TI eine Reihe von 16-Bit-Prozessoren entwickelt, so den im TI99/4A enthaltenen TMS9900. Dieser Prozessor war insofern ungewöhnlich, weil er nicht über interne Register verfügte. Diese befinden sich in einer außerhalb der CPU angeordneten „Scratchpad memory".

Seit einiger Zeit vertreibt Texas Instruments einen als TMS99000 bekannten 16-Bit-Prozessor, der, wie das Unternehmen hofft, erfolgreicher sein wird.

Gesprächsbereit

Der Apricot Portable ist mit der modernsten Technologie ausgerüstet: eine drahtlose Verbindung zwischen den einzelnen Systemkomponenten, sprachgesteuerte Befehlserkennung und ein vielseitig einsetzbares Eingabegerät.

Der Markt für tragbare Maschinen ist einer der wichtigsten in der Microcomputerbranche. Er stützt sich auf der Idee, daß Geschäftsleute ihre Computer mit auf Reisen nehmen, um überall damit arbeiten zu können. In der Praxis werden tragbare Geräte jedoch nur selten auf diese Weise eingesetzt. Oft dienen sie als Zweitgerät, auf dem die Arbeit fertiggestellt wird, die von einem kompatiblen Bürocomputer übernommen wurde.

ACT ist der jüngste einer langen Reihe von Herstellern, deren tragbare Maschinen mit ihrer Bürocomputerreihe kompatibel sind.

Der Apricot Portable befindet sich in einem stabilen Plastikkoffer von der Größe eines Aktenkoffers. Die Tasche läßt sich über zwei Klippschlösser zu beiden Seiten des Griffes öffnen.

Die Tastatur und der Computer sind mit Gurten an den Innenseiten des Koffers befestigt. Mitgeliefert werden weiterhin zwei Handbücher, ein Mikrofon und ein Eingabegerät, das aus Maus und Trackball kombiniert wurde. Weder die Maus noch die Tastatur sind durch Kabel mit dem Computer verbunden. Die Datenübertragung geschieht mittels Infrarotstrahl. Zwei kleine Birnen auf der Rückseite der Tastatur und vorn auf der Maus senden und empfangen die Informationen, die der Computer über ein großes Empfangsfeld aufnimmt. Die Tastatur kann somit mehrere Meter vom Hauptgerät entfernt aufgestellt werden.

Das Infrarotsystem hat jedoch auch Nachteile. Wenn mehrere Portables gleichzeitig in einem Raum betrieben werden, können die Infrarotsignale der einzelnen Geräte sich gegenseitig stören. Um dies zu vermeiden, bietet Apricot ein Glasfaserkabel an, mit dem sich Tastatur und Computer auf herkömmliche Weise verbinden lassen. Ein weiteres Problem tritt bei Verwendung der Maus auf: Wenn die Tastatur direkt vor dem Computer steht, blokkiert sie das Signal der Maus, da das Empfangsfeld des Computers nur wenig über der Tischoberfläche liegt. Wird nun die Tastatur zur Seite gestellt, kann der Computer ihre Signale nicht mehr empfangen.

Bei tragbaren Versionen von Bürocomputern ist es schwierig, alle Funktionen auf kleinerem Raum unterzubringen. Besonders der Aufbau der Tastatur stellt die Hersteller vor größere Probleme. So hat ACT bei der Umsetzung der Apricot-Tastatur in eine kleinere und leichtere Version auf die zusätzliche LCD-Anzeige des Taschenrechners verzichtet und die 87 Tasten sehr eng gepackt. Der Aufbau entspricht der Büroversion des Apricot, wobei die Funktionstasten hier rechts und nicht oberhalb des Tastenfeldes liegen.

Zusätzliche Funktionstaste

Als Ausgleich für den Verlust der LCD-Anzeige, die auf der Büroversion auch die Funktionen der Befehlstasten anzeigt, gibt es eine weitere Funktionstaste (insgesamt sind es neun) und eine Taste für Datum und Zeit. Auf beiden Seiten des Haupttastenfeldes befinden sich die Control-Tasten, während die Cursorsteuerung rechts unten liegt. Zwischen dem alphanumerischen Feld und den Funktionstasten liegt der Zehnerblock des „Taschenrechners", dessen Zahlen in einem helleren Grau gehalten sind als die übrige Tastatur.

Obwohl die Tastatur sehr elegant aussieht, liegen die Tasten zu eng beieinander, um sie bequem bedienen zu können. Statt skulpturier-

Der Apricot Portable verfügt über einige der neuesten Technologien. Die Maschine hat eine LCD-Anzeige für 80×25 Zeichen, die für eine breite Palette von MS-DOS-Anwendungen geeignet ist. Das Bild zeigt die Tastatur und die Kombination Maus/Trackball, die nicht auf herkömmliche Weise mit dem Computer verbunden sind, sondern ihre Daten per Infrarotstrahl übermitteln. Auf der rechten Seite des Computers befindet sich ein Mikrofon, mit dem das Gerät die Signale für sein Spracherkennungssystem empfängt.

ter Oberflächen hat der Portable flache Tasten im QL-Stil. Die Tastatur ist so zwar kompakter, da die Tasten nicht herausragen, doch werden viele Anwender Schwierigkeiten mit der Unterscheidung der einzelnen Tasten haben. Da zudem Haupttastenfeld, Zehnerblock und Funktionstasten nicht voneinander getrennt sind, wird dieses Problem noch weiter verstärkt. Das Design hat jedoch auch Vorteile: Die Tasten für Shift, Stop und Caps Lock sind doppelt so groß wie bei den anderen Apricot-Geräten und einfacher zu bedienen.

Statt der Taschenrechneranzeige hat der Portable vier Tasten für Spezialfunktionen: Die Reset-Taste löst einen Warmstart aus. Mit dem zweiten Knopf läßt sich die automatische Wiederholungsgeschwindigkeit der Tasten ändern. Der dritte Knopf stellt die im Computer gespeicherte Zeit auf Null, während der vierte die Tastatur ausschaltet. Gerade diese letzte Funktion ist sehr praktisch, wenn beim Einsatz der Maus unbeabsichtigte Eingaben über die Tastatur ausgeschlossen werden sollen.

Der LCD-Schirm ist fest mit dem Computer verbunden. Die Anzeige auf dem hellolivfarbenen Hintergrund läßt sich schlechter lesen als bei anderen vergleichbaren Geräten. Rechts vom Bildschirm befindet sich ein Mikrofon, das über ein dünnes Kabel mit einer Micro-Buchse auf der linken Seite des Apricot-Computers verbunden ist.

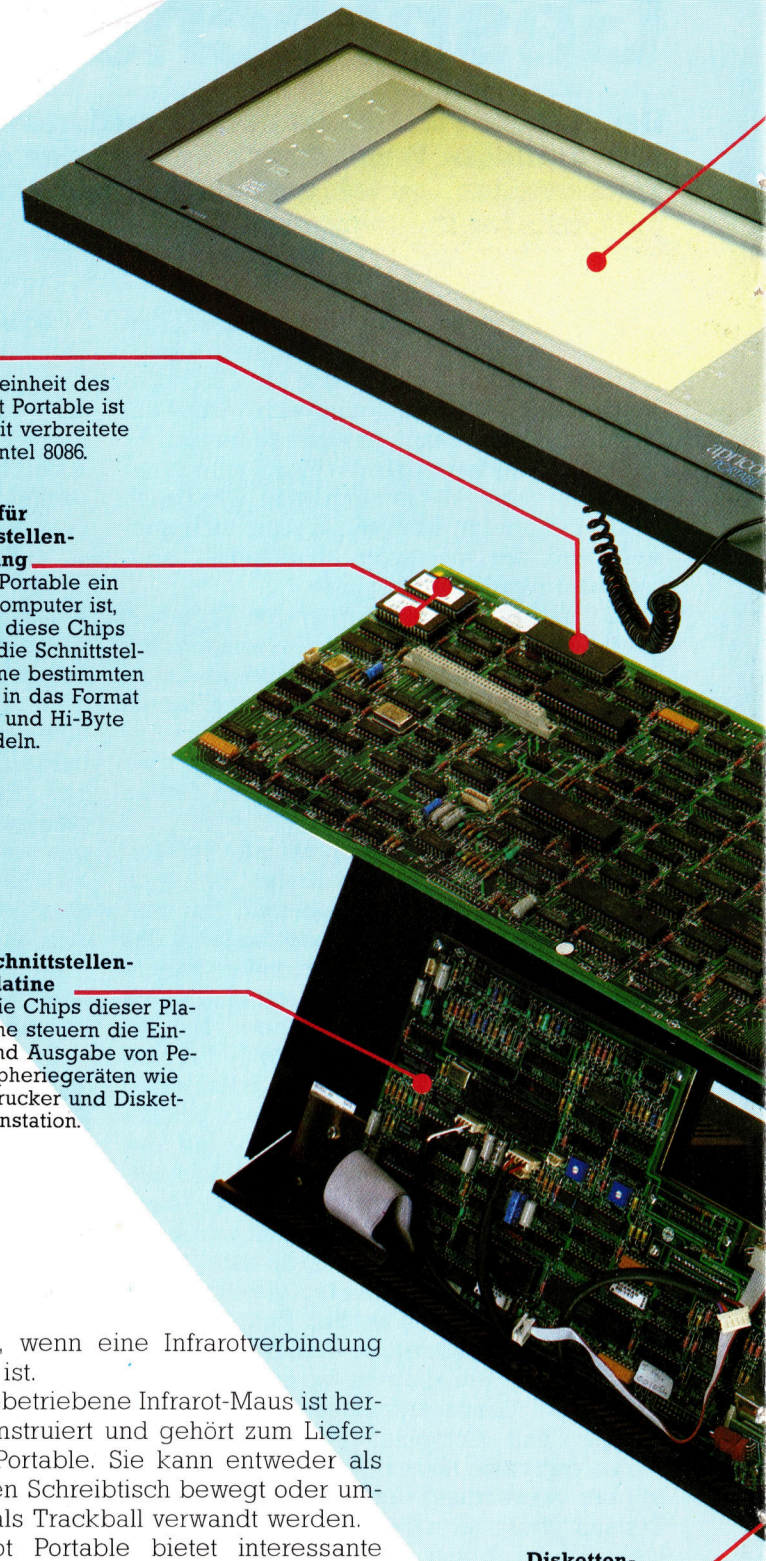
Spracherkennung

Die sicherlich interessanteste Eigenschaft des Portable ist ein Spracherkennungssystem, das gesprochene Befehle erkennt und sie auf normale Weise ausführt. Beim Einschalten der Spracherkennung kann das Gerät bis zu 63 Wörter verstehen und damit Anwendungen wie WordStar oder SuperCalc steuern. Das Spezialprogramm zur Eingabe dieses Vokabulars fragt zunächst, ob die Stimme männlich, weiblich oder von einem Kind ist, und weiterhin, ob viel Hintergrundlärm existiert. Nach mehrfacher Wiederholung eines Befehls vergleicht der Computer die Versionen und nimmt sie in seinen Sprachschatz auf. Da das Gerät Befehle jedoch häufig falsch interpretiert oder überhaupt nicht ausführt, sollten Funktionen wie DELETE oder FORMAT nicht verbal eingegeben werden.

Wie die anderen Geräte der Apricot-Reihe hat auch der Portable ein 3½-Zoll-Sony-Laufwerk, das sich auf der rechten Seite des Computers befindet. Auf der Rückseite sind drei mit einem Plastikschutz abgedeckte Schnittstellen untergebracht: ein paralleler Druckerausgang im Centronics-Format, ein RS232-Interface für externe Modems oder andere serielle Geräte und eine Joystickbuchse im Atari-Standard. Der Joystickanschluß ist nicht für die Spielsteuerung gedacht, sondern für eine „normale" Maus, die dann eingesetzt werden kann, wenn eine Infrarotverbindung nicht möglich ist.

Die batteriebetriebene Infrarot-Maus ist hervorragend konstruiert und gehört zum Lieferumfang des Portable. Sie kann entweder als Maus über den Schreibtisch bewegt oder umgedreht und als Trackball verwandt werden.

Der Apricot Portable bietet interessante technologische Neuerungen. Viele kommerzielle Anwender werden diesen jedoch nur wenig Interesse entgegenbringen, da sie eine verläßliche Maschine brauchen, auf der ihre Anwendungsprogramme problemlos ablaufen. Hier schneidet der Apricot Portable leider nicht gut ab. Die Schwierigkeiten werden viele Anwender veranlassen, sich für Geräte herkömmlicher Bauart zu entscheiden.

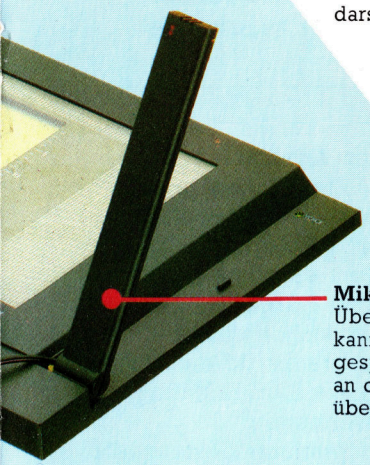


CPU
Zentraleinheit des Apricot Portable ist der weit verbreitete 16-Bit Intel 8086.

ROMs für Schnittstellensteuerung
Da der Portable ein 16-Bit-Computer ist, müssen diese Chips die für die Schnittstellenplatine bestimmten Signale in das Format Lo-Byte und Hi-Byte umwandeln.

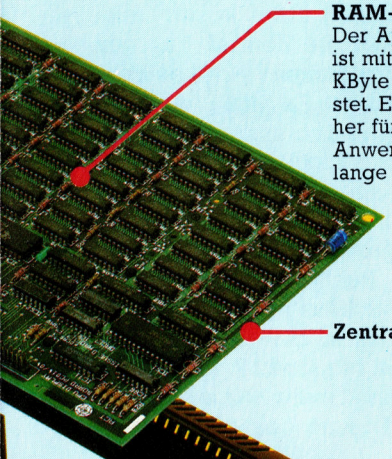
Schnittstellenplatine
Die Chips dieser Platine steuern die Ein- und Ausgabe von Peripheriegeräten wie Drucker und Diskettenstation.

Diskettenstation
Der Computer hat ein 3½-Zoll-Diskettenlaufwerk für doppelseitige Disketten mit doppelter Schreibdichte, das mit anderen Geräten von Apricot kompatibel ist.



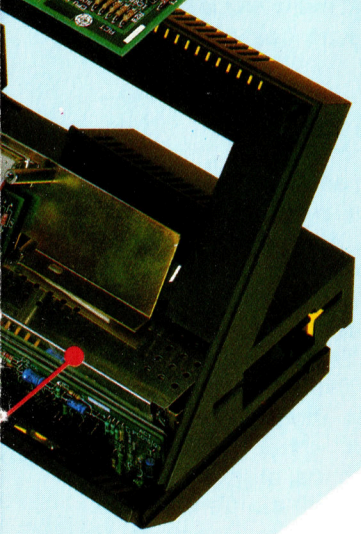
LCD-Anzeige
Dieser Bildschirm kann nicht nur 25 Zeilen × 80 Zeichen anzeigen, sondern auch hochauflösende Grafik darstellen.

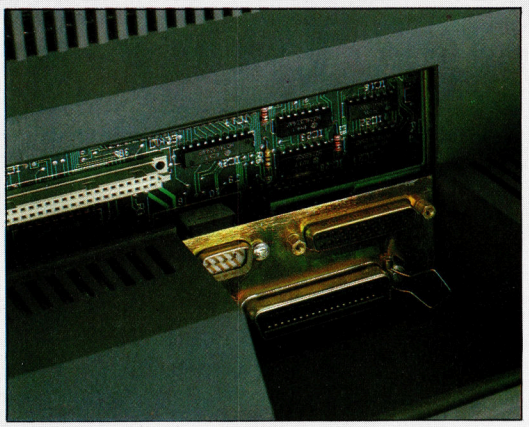
Mikrofon
Über dieses Mikrofon kann der Anwender gesprochene Befehle an den Computer übermitteln.



RAM-Chips
Der Apricot Portable ist mit vollen 256 KByte RAM ausgerüstet. Er eignet sich daher für kommerzielle Anwendungen und lange BASIC-Programme.

Zentralplatine





Schnittstellen für Peripheriegeräte
Die Schnittstellen befinden sich auf der Rückseite des Computers unter einem Plastikschutz. Unten liegt die Centronics-Parallelschnittstelle und rechts darüber der RS232-Eingang. An die linke Buchse läßt sich eine herkömmliche Maus oder ein Trackball anschließen.

Trackball mit Infrarotsignal
Die Bewegung des Balles löst auf dem Bildschirm eine entsprechende Bewegung des Cursors aus. Sobald der Cursor auf dem gewünschten Symbol steht, kann man die Anwendung durch Drücken des Knopfes aufrufen. Das Eingabegerät kann auch als Maus eingesetzt werden.

Apricot-Software
Zum Lieferumfang des Apricot Portable gehört eine Anzahl Applikationspakete, die auf 3½-Zoll-Sony-Disketten gespeichert sind. Darunter befinden sich SuperCalc (Kalkulationssystem), SuperPlanner (ein Indexsystem mit Kalender) und SuperWriter (Textsystem).

Apricot Portable

ABMESSUNGEN
450 × 335 × 200 mm

ZENTRALEINHEIT
Intel 8086

SPEICHERKAPAZITÄT
32 K ROM und 256 K RAM, von denen 211 K für Anwendungsprogramme zur Verfügung stehen.

BILDSCHIRMDARSTELLUNG
LCD-Anzeige mit der Auflösung 80 × 25 Zeichen (Text) und 640 × 256 Pixel (Grafik). Bei Anschluß eines externen Monitors können bis zu 16 Farben angezeigt werden.

SCHNITTSTELLEN
RS232 seriell, Centronics parallel und Mausbuchse. Die Tastatur kann auch über ein Glasfaserkabel mit dem Computer verbunden werden. Anschlußbuchse für ein Mikrofon ist vorhanden.

PROGRAMMIERSPRACHEN
Der Portable ist mit drei doppelseitigen 3½-Zoll-Disketten von Sony ausgerüstet, die die Betriebssysteme MS-DOS, CP/M-86 und Concurrent CP/M enthalten.

TASTATUR
IBM-kompatible Tastatur mit 87 Tasten, darunter zehn Funktionstasten. Vier zusätzliche Spezialfunktionstasten.

HANDBÜCHER
Das Betriebshandbuch erklärt den Aufbau des Computers, den Gebrauch der Betriebssysteme und die Funktionsweise der Spracherkennung. Das Handbuch für das Anwendungspaket enthält Einführungen in SuperCalc, SuperPlanner und SuperWriter.

STÄRKEN
Der Apricot Portable enthält viele der neuesten Technologien, die der Markt heute bietet.

SCHWÄCHEN
Die in die Maschine eingebaute neuartige Technologie ist nicht zuverlässig. Die LCD-Anzeige läßt sich nur schwer entziffern.

Tips und Tricks

Die meisten Leute lernen Programmieren aus den Handbüchern, die mit ihrem Computer geliefert werden. Für den Anfang ist das meist ausreichend. Hier beginnt eine Serie von Beiträgen, in der wichtige Techniken vorgestellt werden.

Guter Programmierstil wird durch Experimentieren und Erfahrung entwickelt. Der Neuling, der Probleme häufig nur mit großer Anstrengung löst, entwickelt sich erst allmählich zu einem Techniker und entdeckt Wege, wie er bestimmte Dinge anstellen muß, um zu dem gewünschten Ergebnis zu kommen. Warum aber sollte der Lernprozeß des Computerbesitzers nicht beschleunigt werden, indem er aus den Fehlern anderer lernt, die denselben Weg gegangen sind? In dieser Serie werden hilfreiche Tips gegeben, die für den Anfänger wichtig sind.

Das Programmieren ist ein Problemlösungs-Prozeß. Ein großer Teil dessen kann im Kopf und mit Bleistift und Papier durchgeführt werden, bevor die erste Programmzeile geschrieben wird. Die Stadien dieses Prozesses sind bestens bekannt: eine klare zusammenfassende Aussage des Problems, gefolgt von einer neuerlichen, detaillierteren Darlegung, die immer weiter spezifiziert wird und so genau und detailliert wie möglich darzulegen ist. Diese Beschreibung enthält fast immer die eigentliche Lösung oder gibt zumindest Hinweise für die Vorgehensweise. Beim Programmieren erfolgt die Codierung erst im letzten Stadium. Es sollte eine zügige Realisierung der vorhergegangenen Denkschritte sein. Nimmt das Codieren, also das Programmschreiben selbst, mehr Zeit in Anspruch als die eigentliche Problemlösung, ist das Ergebnis nicht optimal und das Programm an sich meist schlecht.

Lösungen werden häufig als Algorithmen bezeichnet. Das sind Prozesse, die eindeutige Handlungsanweisungen darstellen und in den Logikphasen analysiert wurden. Die Brauchbarkeit eines Programms hängt entscheidend von der „Genauigkeit" und „Vollständigkeit" des Algorithmus ab. Diese Bezeichnungen verweisen auf die theoretische und praktische Fähigkeit des Programms, vorhersehbare Eingaben zu erfassen und zu verarbeiten, sowie auf die Richtigkeit seiner internen Logik. Natürlich ist es immer einfacher, ihr Nichtvorhandensein festzustellen als ihr Vorhandensein zu demonstrieren.

Lösungen müssen verläßlich sein, nicht nur vollständig und richtig. Das bedeutet: Neben dem vorgeschriebenen Problemlösungsbereich müssen sie darüber hinaus Zustände außerhalb des eigentlichen Arbeitsbereichs berücksichtigen können. Ein Programm sollte also mögliche Fehlerbedingungen erkennen, den Rechenprozeß unter Wahrung aller Daten abbrechen und möglichst eine sinnvolle Statusmeldung geben. Es ist schwer zu beurteilen, ob ein Code wirklich sicher ist. Bei einem wenig zuverlässigen Programm ist das einfacher zu erkennen. Auch hier führt Erfahrung zu besserem Beurteilungsvermögen.

Die Gestaltung von Programmen in der zuvor beschriebenen Weise ist ein erstrebenswertes Ziel, das aber fast immer mit einem anderen ebenso erstrebenswerten kollidiert – nämlich dem der Wirtschaftlichkeit. Alles kostet Geld, und selbst wenn es nur die Zeit ist, die man fürs Schreiben eines Programms – sei es auch nur aus Spaß – aufwendet. Es gibt Punkte, an denen man sich entscheiden muß, ob man an einem nahezu „bombensicheren" Programm weiterarbeiten oder abbrechen soll, um mit einem neuen Projekt zu beginnen. Selbst wenn die eigene Zeit unbegrenzt ist, Computer-Speicherkapazität und Rechengeschwindigkeit sind es nicht. Es ist durchaus möglich, daß der eigentliche Algorithmus in soviele Hilfs- und Fehlersuchprogramme eingebettet ist, daß für die „Fehlerverhinderung" mehr Zeit aufgewendet wird als für die Lösung des eigentlichen Problems.

Erst mal testen

Die theoretische Lösung analytischer und logischer Probleme ist sehr wichtig. Programme aber sind dazu da, eine Aufgabe durchzuführen. Sind erst einmal Syntax- und Logik-Fehler ausgemerzt, beginnt die „Testphase". Dieser Begriff ist so vertraut, daß er kaum erläuterungsbedürftig scheint. Tatsache aber ist, daß dieser Prozeß meist mißverstanden wird. Es gibt gewöhnlich in allen Programmen – von ganz simplen abgesehen – viel zuviele mögliche Eingabe-Kombinationen, als daß man sie alle testen könnte. Also muß ein Test so einfach und direkt wie möglich sein. Das heißt, er prüft die Schwachstellen und die starken Teile des Programms. Das Schreiben oder Herstellen umfassender Testbedingungen ist nicht einfach und erfordert viel Zeit.

Effektive Tests zeigen die Unzulänglichkeiten eines Programms anhand einer eigenen

Logik, um den Zeitaufwand für das „Reinigen" des Programms, das Entfernen von Fehlern, so kurz wie möglich zu halten. Wie das Testen ist die Fehlerbeseitigung ein wichtiger Prozeß, der aber oft daran scheitert, weil er dieselben menschlichen Unzulänglichkeiten enthält, die ihn überhaupt erst erforderlich machen.

Sind diese Entwicklungsphasen ausgeführt und hat sich der Programmierer von der Arbeitsfähigkeit des Programms überzeugt, sind keine weiteren Änderungen mehr erforderlich, da die Programmierung als klares Modell steht. Doch kein Programm erklärt sich von selbst, und es gibt immer wieder Gründe, ein lauffähiges Programm zu ändern. Wie alle anderen Mechanismen bedürfen sie der Wartung – und Wartung heißt Dokumentation. Intern sollten Programme durch REM-Zeilen dokumentiert werden, die dem Programmierer die Arbeit erleichtern. Sie sollten zum Nutzen des Anwenders extern dokumentiert sein, mit einem begleitenden Handbuch etwa – auch wenn der Anwender der Programmierer selbst ist. Ausgereifte Programme werden von effektiven Programmierern auf der Basis gewachsener Erfahrung und logischen Denkens geschrieben. Dies zu fördern, ist das Ziel dieser neuen Serie.

Farben und Formen der Balken, aus denen das Diagramm zusammengesetzt ist, werden im Programm durch einfache Wertänderung der Steuervariablen vorgenommen.

```
399 REM**************************
400 REM*   3-D BAR CHARTS       *
401 REM**************************
500 GOSUB 1500:       REM INITIALISE
720 YY=22:XX=2:       REM ORIGIN COORDS
760 GOSUB 3200:       REM DRAW AXES
800 FOR E=LT-1 TO 0 STEP -1
820 OC=XX+E*DB:OL=YY-E*DB
840 GOSUB 2200:       REM INPUT DATA
900 FOR D=1 TO NN
920 HT=DT(D)
940 X0=OC+(D-1)*(BB+LT*DB):Y0=OL-HT-DB
960 GOSUB 4000:       REM PRINT BAR
980 NEXT D
1000 NEXT E
1100 XP=10:YP=23:GOSUB 3500
1120 PRINT"THREE-DIMENSIONAL HISTOGRAM"
1200 A$=INKEY$:IF A$="" THEN GOTO 1200
1400 END
1499 REM**************************
1500 REM*  INITIALISE  S/R      *
1501 REM**************************
1520 CL$=CHR$(147):   REM CLEAR SCREEN
1540 PRINT CL$
1560 PO$=CHR$(19):    REM HOME CRSR
1580 RT$=CHR$(13):    REM <RETURN>
1600 BB=2:DB=1:       REM BAR DIMENSIONS
1620 SW=40:SD=25:     REM SCREEN DIM'S
1640 HB=SD-DB:        REM MAX BAR HEIGHT
1660 DIM B$(HB+DB)
1680 FOR K=1 TO SD:PO$=PO$+RT$:NEXT K
1800 DIM DT(SW)
1900 GOSUB 2400:      REM BUILD BAR
2100 LT=4:            REM DEPTH FACTOR
2190 RETURN
2199 REM**************************
2200 REM* INPUT DATA ARRAY S/R  *
2201 REM**************************
2220 READ NN
2240 FOR Z=1 TO NN:READ DT(Z):NEXT Z
2310 DATA 6,12,10,4,7,8,10
2320 DATA 5,7 ,8 ,8,6,7
2330 DATA 6,7 ,4 ,8,5,3,9
2340 DATA 5,11,6 ,4,11,6
2390 RETURN
2399 REM**************************
2400 REM* BUILD WHOLE BAR S/R   *
2401 REM**************************
2500 TC$=CHR$(158):   REM SIDES=YELLOW
2520 FC$=CHR$(31):    REM FRONT=BLUE
2540 RV$=CHR$(18):    REM REVERSE ON
2560 NL$=CHR$(146):   REM REVERSE OFF
2580 CR$=CHR$(29):    REM CURSOR RIGHT
2600 CH$=CHR$(32):    REM SPACE CHAR.
2620 C1$=CHR$(169):   REM "◤"  CHAR.
2640 C2$=RV$+C1$:     REM "◢"  CHAR.
2680 FOR K=1 TO SW
2700 SP$=SP$+CH$
2720 RC$=RC$+CR$
2740 FF$=FF$+CH$
2760 NEXT K
2800 TL$=SP$+"/"
2820 BL$=SP$+NL$+C1$
2840 FL$=LEFT$(FF$,BB)
2900 L$=TC$+C2$+RV$+RIGHT$(TL$,BB)
2920 FOR K=1 TO DB
2940 B$(K)=LEFT$(RC$,DB-K)+L$+LEFT$(SP$,K-1)
2960 NEXT K
3000 L$=FC$+RV$+FL$+TC$+RIGHT$(TL$,DB)
3020 FOR K=DB+1 TO HB
3040 B$(K)=L$
3060 NEXT K
3100 L$=FC$+RV$+FL$+TC$
3120 FOR K=1 TO DB
3140 B$(HB+K)=L$+RIGHT$(BL$,DB+2-K)
3160 NEXT K
3190 RETURN
3199 REM**************************
3200 REM*    DRAW   AXES    S/R   *
3201 REM**************************
3300 PRINT TC$:           REM SET COLOUR
3320 FOR Y=2 TO YY-1
3340 XP=XX-1:YP=Y:GOSUB 3500:PRINT"!"
3360 XP=XX+YY-Y:GOSUB 3500:PRINT"/"
3380 NEXT Y
3400 YP=YY
3420 FOR X=XX-1 TO SW-1
3440 XP=X:GOSUB 3500:PRINT"-"
3460 NEXT X
3490 RETURN
3499 REM**************************
3500 REM* PUT CRSR  @ XP,YP S/R  *
3501 REM**************************
3600 PRINT LEFT$(PO$,YP)TAB(XP-1);
3620 RETURN
3999 REM**************************
4000 REM* PRINT PARTIAL BAR S/R  *
4001 REM**************************
4100 FOR V=1 TO HT
4120 XP=X0:YP=Y0+V-1
4140 GOSUB 3500:      REM PLACE CRSR.
4160 PRINT B$(V)
4180 NEXT V
4200 FOR V=1 TO DB
4220 XP=X0:YP=Y0+HT+V-1:GOSUB 3500
4240 PRINT B$(HB+V)
4260 NEXT V
4490 RETURN
READY.
```

Struktur
Modular und möglichst selbst erklärend.

Dokumentation
Diese Routine ist offensichtlich entscheidend, aber überhaupt nicht erklärt.

Variablennamen
Gut kommentiert, aber nicht aussagekräftig.

Vollständigkeit/ Genauigkeit
Erfolgt auch in den Fällen, wenn ein Wert kleiner als Null ist, die richtige Ausgabe?

Fehlersuche
Keine Prüfsumme, keine Parameter und keine Hilfe bei Fehlern.

Dokumentation
Gut, da es hier keine „magischen" Zahlen gibt und alles übersetzbar ist.

Mittelmäßig

Dieses Programm zur Darstellung dreidimensionaler Balkendiagramme zeigt guten und schlechten Stil: Die interne Dokumentation ist, soweit vorhanden, gut, und es ist modular strukturiert. Doch das Programm erklärt sich nicht selbst, es gibt keine Unterstützung zur Fehlerprüfung und keine Dokumentation.

BASIC-Dialekte

Das in Microsoft-BASIC geschriebene Programm sollte unmodifiziert auf Rechnern mit einer 40 × 25-Zeichen-Bildschirmdarstellung laufen. Die Bildschirm-Dimensionierung wird mit Zeile 1620 initialisiert. Die Kontrollzeichenwerte, die in den Subroutinen 1500 und 2400 initialisiert werden, gelten für den C 64.

Magische Kreise

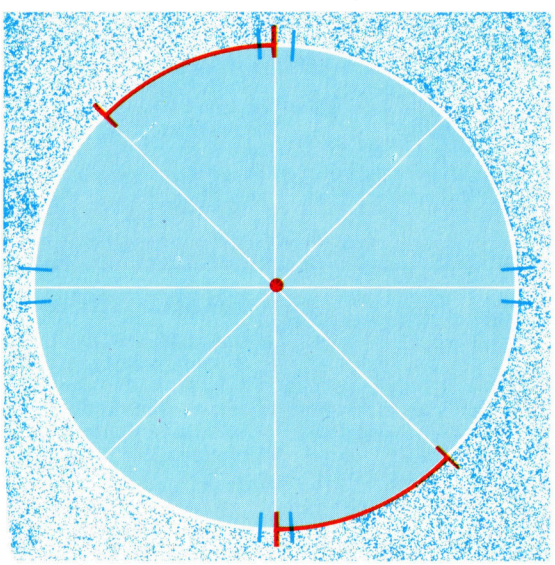
Das BASIC des Acorn B verfügt zwar über eine ganze Reihe von beeindruckenden Grafikmöglichkeiten, doch leider fehlt ein Befehl, der Kreise erzeugt. Mit der folgenden Maschinencoderoutine ist dieses jedoch möglich.

Ein Kreis ist schnell gezeichnet; eine mathematische Gleichung, die Kreise mit annehmbarer Geschwindigkeit punktweise auf den Bildschirm plottet, läßt sich jedoch nicht so einfach programmieren. Die leichteste Methode, Kreise zu erzeugen, arbeitet mit den Funktionen SIN und COS:

```
DEF PROCCIRCLE (XORG,YORG,R)
  MOVE XORG+R, YORG
  FOR THETA =0 TO 2*PI STEP PI/32
    X=R*COS(THETA)
    Y=R*SIN(THETA)
    DRAW X+XORG,Y+YORG
  NEXT THETA
ENDPROC
```

Da für die Funktionen SIN und COS einige Berechnungen durchgeführt werden, dauert es lange, bis der Kreis auf dem Bildschirm erscheint. Mit dem folgenden Algorithmus läßt sich dieser Vorgang jedoch wesentlich beschleunigen.

```
  3 MODE1
  5 PROCCIRCLE(500,600,200)
  7 END
 10 DEF PROCCIRCLE (XORG,YORG,R)
 20 Y=R
 30 FOR X=1 TO Y*.707
 40 Y=Y-X/Y
 50 PROCPOINTS(X,Y)
 60 NEXT
 70 ENDPROC
 80 DEF PROCPOINTS (X,Y)
 90 PLOT 69,XORG+X,YORG+Y
100 PLOT 69,XORG-X,YORG+Y
110 PLOT 69,XORG-X,YORG-Y
120 PLOT 69,XORG+X,YORG-Y
130 PLOT 69,XORG+Y,YORG+X
140 PLOT 69,XORG-Y,YORG+X
150 PLOT 69,XORG-Y,YORG-X
160 PLOT 69,XORG+Y,YORG-X
170 ENDPROC
```



Das Maschinencodeprogramm setzt eine besonders schnelle Formel zum Plotten der Kreise ein. Wenn dabei jeder Kreispunkt einzeln berechnet werden müßte, würde aber auch diese Routine viel Zeit benötigen. Da die obere Hälfte des Kreises jedoch ein Spiegelbild der unteren Hälfte ist, muß nur der obere Teil berechnet und als Spiegelbild in die untere Hälfte projiziert werden. Dieses Prinzip läßt sich auch auf die linke und rechte Hälfte des Kreises anwenden.

Im Endeffekt berechnet das Programm nur ein Achtel des Kreises. Jeder Punkt dieses Achtels wird siebenmal projiziert, so daß schließlich ein vollständiger Kreis entsteht.

Die Routine berechnet acht verschiedene Kreisabschnitte gleichzeitig und plottet schneller als die erste Version. Doch obwohl hier keine Sinus- und Kosinuswerte mehr berechnet werden, ist die Routine immer noch relativ langsam, da für jeden Kreispunkt eine Division auszuführen ist.

Es gibt jedoch ein Verfahren ohne komplizierte mathematische Berechnungen:

```
 10 MODE4
 20 PNUM=69
 30 PROCCIRCLE(500,600,200)
 40 END
 50 :
 60 DEF PROCCIRCLE(X,Y,R)
 70 VDU29,X;Y;:REM SET GRAPHICS ORIGIN
 80 X=0:Y=R:D=3-2*R:REM VARIABLES
 90 REPEAT
100 PROCCPLOT
110 IFD<0:D=D+4*X+6:ELSED=D+4*(X-Y)+10:Y
    =Y-4
120 X=X+4
130 UNTIL X>Y:ENDPROC
140 :
150 DEF PROCCPLOT
160 PLOT PNUM,X,Y
170 PLOT PNUM,Y,X
180 PLOT PNUM,Y,-X
190 PLOT PNUM,-X,Y
200 PLOT PNUM,-X,-Y
210 PLOT PNUM,-Y,-X
220 PLOT PNUM,-Y,X
230 PLOT PNUM,X,-Y
240 ENDPROC
```

Dieser Ablauf ist als der „Breshen Algorithmus" bekannt. Er ist sehr schnell, da er nur Additionen, Subtraktionen und Multiplikationen mit Zwei und Vier enthält, die sich leicht mit Bit-Verschiebungen ausführen lassen.

Kreiserzeugung

```
 30PNUM=69
 40S%=2
 500SWRCH=&FFEE
 60DIM CODE% 600
 70DIM D% 12
 80X=D%+2
 90Y=D%+4
100D=D%+6
110N1=D%+8
120N2=D%+10
130PROCCOMPILE
140MODE4
150PROC_OLYMPIC
160END
170#
180DEF PROCCOMPILE
190FOR I% =0 TO S% STEP S%
200K%=P%
210P%=CODE%
220[OPT I%
230.CIRCLE
240JSR INIT
250:
260.LOOP
270JSR COMPXY:BMI DOIT:JSR CPLOT:RTS
280.DOIT
290JSR CPLOT
300LDA D+1:BPL D_IS_POS
310:
320.D_IS_NEG
330JSR DNEG
340JSR ADD_4_TO_X
350JMP LOOP
360:
370.D_IS_POS
380JSR DPOS
390JSR ADD_4_TO_X
400JMP LOOP
410:
420.INIT
430LDY #8
440.L7
450LDA &601,Y:STA &80,Y
460DEY:BPL L7
470INY
480LDA (&80),Y:STA X
490LDA (&83),Y:STA Y
500LDA (&86),Y:STA D
510INY
520LDA (&80),Y:STA X+1
530LDA (&83),Y:STA Y+1
540LDA (&86),Y:STA D+1
550LDA #29:STA D%+1
560LDA #0:STA D%
570JSR PSTR
580LDA #25:STA D%
590LDA #PNUM:STA D%+1
600JSR SETD
610RTS
620:
630.COMPXY
640LDA X:STA N1:LDA X+1:STA N1+1
650LDA Y:STA N2:LDA Y+1:STA N2+1
660JSR SUB
670LDA N1+1
680RTS
690:
700.CPLOT
710LDX #4
720.L2
730JSR P2
740DEX:BNE L2
750RTS
760:
770.DNEG
780LDA X:STA N1:LDA X+1:STA N1+1
790JSR TIMES4
800LDA #6:STA N2:LDA #0:STA N2+1
810JSR ADD
820LDA D:STA N2:LDA D+1:STA N2+1
830JSR ADD
840LDA N1:STA D:LDA N1+1:STA D+1
850RTS
860:
870.DPOS
880LDA X:STA N1:LDA X+1:STA N1+1
890LDA Y:STA N2:LDA Y+1:STA N2+1
900JSR SUB
910JSR TIMES4
920LDA #10:STA N2:LDA*#0:STA N2+1
930JSR ADD
940LDA D:STA N2:LDA D+1:STA N2+1
950JSR ADD
960LDA N1:STA D:LDA N1+1:STA D+1
970JSR SUB_4_FROM_Y
980RTS
990:
1000.ADD_4_TO_X
1010LDA #4:STA N1:LDA #0:STA N1+1
1020LDA X:STA N2:LDA X+1:STA N2+1
1030JSR ADD
1040LDA N1:STA X:LDA N1+1:STA X+1
1050RTS
1060:
1070.SUB_4_FROM_Y
1080LDA#4:STA N2:LDA #0:STA N2+1
1090LDA Y:STA N1:LDA Y+1:STA N1+1
1100JSR SUB
1110LDA N1:STA Y:LDA N1+1:STA Y+1
1120RTS
1130.SETD
1140LDA #0:STA X:STA X+1
1150LDA D:STA Y:LDA D+1:6TA Y+1
1160ASL D:ROL D+1
1170LDA #3:STA N1:LDA #0:STA N1+1
1180LDA D:STA N2:LDA D+1:STA N2+1
1190JSR SUB
1200LDA N1:STA D:LDA N1+1:STA D+1
1210RTS
1220:
1230.P2
1240JSR PSTR
1250JSR SWAPXY
1260JSR PSTR
1270JSR NEGY
1280RTS
1290:
1300.TIMES4
1310ASL N1:ROL N1+1
1320ASL N1:ROL N1+1
1330RTS
1340:
1350.ADD
1360CLC
1370LDA N1:ADC N2:STA N1
1380LDA N1+1:ADC N2+1:STA N1+1
1390RTS
1400:
1410.SUB:\ (N1=N1-N2)
1420SEC
1430LDA N1:SBC N2:STA N1
1440LDA N1+1:SBC N2+1:STA N1+1
1450RTS
1460:
1470.PSTR
1480LDY #250
1490.L1
1500LDA D%-250,Y
1510JSR OSWRCH
1520INY
1530BNE L1
1540RTS
1550:
1560.SWAPXY
1570LDA X:PHA:LDA X+1:PHA
1580LDA Y:STA X:LDA Y+1:STA X+1
1590PLA:STA Y+1:PLA:STA Y
1600RTS
1610:
1620.NEGY
1630LDA #0:STA N1:STA N1+1
1640LDA Y:STA N2
1650LDA Y+1:STA N2+1
1660JSR SUB
1670LDA N1:STA Y
1680LDA N1+1:STA Y+1
1690RTS
1700]
1710NEXT
1720ENDPROC
1730#
1740DEF PROCCIRCLE(P1%,P2%,P3%)
1750CALL CIRCLE,P1%,P2%,P3%
1760ENDPROC
1770#
1780DEF PROC_OLYMPIC
1790PROCCIRCLE(300,600,150)
1800PROCCIRCLE(650,600,150)
1810PROCCIRCLE(1000,600,150)
1820PROCCIRCLE(475,450,150)
1830PROCCIRCLE(825,450,150)
1840VDU29,0;0;
1850MOVE100,250
1860DRAW100,800
1870DRAW1200,800
1880DRAW1200,250
1890DRAW100,250
1900ENDPROC
```

Mit diesem Maschinenprogramm lassen sich auf dem Acorn Kreise zeichnen. Der Einsatz der Routine ist einfach: Den Ganzzahlvariablen (X%, Y% und R%) werden Werte zugeordnet und der Code mit CALL CIRCLE X%,Y%,R% aufgerufen. Die Routine zeichnet einen Kreis mit dem Radius R% um die von den Koordinaten X% und Y% festgelegte Mitte. Beachten Sie dabei, daß die Routine den Ausgangspunkt in die Mitte des Kreises verlegt. Mit VDU29,0;0; läßt er sich jedoch wieder in die linke obere Ecke setzen. Leider kann der CALL-Befehl als Parameter nur X%,Y% und R%, aber keine Formeln enthalten. Mit dem Einsatz von PROCCIRCLE (siehe Listing) läßt sich diese Einschränkung jedoch beseitigen.

Da die Routine in allen Grafikarten funktionieren soll, muß man für das Plotten der Kreispunkte den Befehl VDU 25 einsetzen, der den Vorgang erheblich verlangsamt. Zum besseren Verständnis wurden nur eigenständige Routinen mit einfachen Befehlsfolgen verwandt. Dadurch wird der Ablauf zwar nochmals verlangsamt, doch ist der Maschinencode bei einem Kreis (Radius = 300 Einheiten) mit 0,52 Sekunden immer noch schneller als die erste BASIC-Version mit 1,9 Sekunden.

Wenn der Wert von PNUM in Zeile 30 von 69 auf 5 gesetzt wird, plottet das Programm mit Linien und zeichnet statt eines Kreises eine farbige Scheibe. Als Nebenwirkung dieser Änderung erscheint in jedem Kreis eine überflüssige Linie, die mit folgendem Befehl beseitigt werden kann:

```
1745 VDU29,0;0;:MOVE P1%,P2%n
```

Dies kann auch im Assembler mit den folgenden zusätzlichen Befehlen ausgeführt werden:

```
575 LDA#25:JSR OSWRCH:LDA#4:JSR
    OSWRCH:LDA#0:JSR OSWRCH:LDA#0:JSR
    OSWRCH:LDA#0:JSROSWRCH:LDA#0:JSR
    OSWRCH
```

Durch weitere Veränderungen kann das Programm auch Kreisbögen und Ellipsen berechnen und zeichnen.

Mondlandung

In diesem Artikel zeigen wir eine vereinfachte BASIC-Version des Spiel-Klassikers Lunar Lander.

Bei Lunar Lander besteht die Aufgabe darin, eine Landefähre sicher auf dem Mond (oder einem anderen Planeten) zu landen. Da der Bordcomputer ausgefallen ist, müssen Sie die Fähre vorsichtig mit kurzen Treibstoffschüben für den Raketenantrieb steuern. Dabei sind neben der Aufsetzgeschwindigkeit auch die Sinkrate sowie der Treibstoffvorrat zu beachten.

Die Idee bei diesem Programm ist, das Verhalten eines Raumschiffes unter vorgegebenen Bedingungen so real wie möglich zu simulieren. Es ist klar, daß die mathematischen Formeln hierzu sehr kompliziert sind, und das hier gezeigte Programm ist eine stark vereinfachte Version, wenngleich es viele Details einer richtigen Simulation enthält.

Lassen Sie uns einen näheren Blick auf das Problem der Landung werfen:

– Der Planet hat eine bestimmte Anziehungskraft. Dies bewirkt, daß das Raumschiff während des Landemanövers beschleunigt.

– Das Raumschiff verfügt über einen Raketenantrieb, der diesem Effekt durch Gegenschub entgegenwirkt.

– Das Raumschiff hat eine Masse (oder Gewicht). Je größer diese Masse ist, desto weniger zeigt der Gegenschub des Raketenantriebes Wirkung. Die Masse des Raumschiffes setzt sich aus seinem Eigengewicht und dem Gewicht des transportierten Treibstoffes zusammen.

Mathematische Simulation

Um das Verhalten der Landefähre zu simulieren, braucht man Gleichungen und Formeln für die Beschleunigung, die Masse, die Geschwindigkeit etc. Diese können sehr einfach oder sehr kompliziert sein, je nachdem, wie genau die Simulation sein soll. Bei dem hier gezeigten Spiel haben wir diese Gleichungen sehr einfach gehalten.

Der wichtigste Faktor ist die gegenwärtige Höhe des Raumschiffes, die sich, bedingt durch die Anziehungskraft bzw. durch den Ge-

Das Bild zeigt die ungefähre Anziehungskraft der Sonne, des Mondes und der anderen Planeten unseres Sonnensytems in Metern pro Sekunde. Diese Werte für die Variable g im hier gezeigten Programm müssen in Zeile 20 eingegeben werden.

genschub, ständig ändert. Damit wir zu jedem Zeitpunkt seine Position bestimmen können, unterteilen wir die „Zeit" in Intervalle.

In jedem Zeitintervall können wir berechnen, wie weit sich das Raumschiff bewegt hat, welche Veränderungen sich in Hinsicht auf die Geschwindigkeit, die Masse usw. ergeben haben. Diese Intervalle können beliebig lang sein – je kürzer sie jedoch sind, desto genauer wird die Simulation. Mit Hilfe der Zeitintervalle ist die Entwicklung der Gleichungen nicht mehr schwer.

Geschwindigkeit wird mit einer Anzahl von Wegeinheiten pro Stunde gemessen. Ein Auto, das mit einer Geschwindigkeit von zehn km/h fährt, legt in zwei Stunden eine Strecke von 20 Kilometern zurück. In drei Stunden sind es 30 Kilometer usw. Daraus ergibt sich die folgende Formel:

Entfernung = Zeit × Geschwindigkeit

Somit können wir in jedem Intervall berechnen, wie weit sich die Landefähre nach oben oder unten bewegt hat, indem wir die Geschwindigkeit mit der Länge des Zeitintervalls multiplizieren. Danach können Sie die Geschwindigkeit durch Beschleunigung (durch die Anziehungskraft) oder Verzögerungen (durch den Raketenantrieb) regulieren.

Die Beschleunigung ist aufgrund der Schwerkraft immer konstant (die Variable g innerhalb des Programms) und hängt von der Beschaffenheit des Planeten ab, dem Sie sich nähern. Das Bild auf der linken Seite zeigt die einzelnen Werte für die Planeten unseres Sonnensystems. Sie können jedoch auch mit anderen Werten experimentieren oder den Wert für g vom Programm per Zufall auswählen lassen.

Die Simulation des Raketenantriebs ist erheblich schwieriger. In der hier gezeigten Version kann der Spieler bis zu neun Treibstoffeinheiten je Zeitintervall verwenden, woraus das Programm unter Berücksichtigung der Masse der Landefähre die Beschleunigung berechnet. Die exakte Formel ist von der Kraft des Antriebs und dem verwendeten Treibstoff abhängig. In diesem Programm wurden diese Faktoren so gewählt, daß sie während des Spieles nicht ohne weiteres „durchschaut" werden können.

Echtzeit-Simulation

Eine weitere Schwierigkeit, die integriert werden kann, ist die „Real Time"-Simulation. Dies ist ein inzwischen sehr geläufiger Begriff, der nicht mehr bedeutet, als daß das Programm kontinuierlich weiterläuft und der Anwender ständig Daten oder Befehle eingeben muß. Oft besteht bei der Programmierung der einzige Unterschied darin, daß statt der INPUT-Anweisung die Abfragen INKEY$ oder GET verwendet werden müssen.

Im Landeprogramm ist eine Zeitschleife eingebaut, die für jedes Zeitintervall einmal aktiviert wird. Verändert man das Zeitintervall derart, daß es der Zeit entspricht, die zur Ausführung der Schleife benötigt wird, arbeitet die Simulation in Echtzeit. Das bedeutet, daß die Landung des Raumschiffes in der Simulation genausoviel Zeit in Anspruch nimmt, wie es in der Realität auch dauern würde. Obwohl dies für eine Simulation sehr wünschenswert ist, wird ein Spiel dadurch oft uninteressant, da es zu lange dauert.

Es gibt viele Dinge, die Sie an dem Programm verändern können. Die naheliegendste Möglichkeit ist die Unterstützung durch Grafik. Hierbei ist Ihrer Phantasie keine Grenze gesetzt. Außerdem konnten Sie auch eine Seitwärtsbewegung einbauen, so daß die Landefähre horizontal gesteuert werden muß.

Im Weltraum würde sich die Landefähre normalerweise nicht seitwärts bewegen, da es keine Kraft gibt, die sie nach rechts oder links ziehen könnte. Doch wenn Sie beispielsweise auf einem Planeten mit einer Atmosphäre landen, könnten Sie das Problem eines Oberflächenwindes einarbeiten. Einige Versionen von Lunar Lander beinhalten verschiedenartigste Landegebiete, zum Beispiel in Höhlen oder Kratern, so daß eine Landung sehr viel Feingefühl erfordert. Versuchen Sie, diese Ideen in Ihr Programm einzubauen.

Lunar Lander

```
 10  REM Lunar Lander Game
 20  LET g=-1.6
 30  LET t=1
 40  LET f=1000
 50  LET v=0
 60  LET h=2000
 70  LET m=2000+f
 80  LET g=g*t
100  REM Update screen
110  PRINT AT 0,0
120  PRINT "              Lunar Lander"
130  PRINT : PRINT "Height....";INT h;"   "
140  PRINT : PRINT "Speed.....";INT v;"   "
150  PRINT : PRINT "Fuel......";f;"   "
160  PRINT
165  IF h<0 THEN  GO TO 400
170  IF f<=0  THEN  LET f=0: PRINT "*** OUT
OF FUEL       ": GO TO 190
180  PRINT "Key rocket burn 0-9 "
190  LET b=0: IF f>0 THEN  LET a$=INKEY$: IF
a$<>"" THEN  LET b=VAL a$
200  IF b>f THEN  LET b=0
210  LET h=h+v*t
220  LET v=v+g
230  LET v=v+(b*3000)/m
240  LET f=f-b: LET m=m-b
250  FOR i=1 TO 50: NEXT i
300  GO TO 110
400  REM On planet surface
410  IF v>-10 THEN  PRINT "*** Safe Landing
... Well done": GO TO 500
420  IF v>-20 THEN  PRINT "*** CRUNCH! ... Y
ou wrecked the lander but the crew survived!"
: GO TO 500
430  PRINT "*** SMASH!  ... Lander destroyed
... no survivors"
440  PRINT : PRINT "You've just blasted a ne
w        crater ";INT (-v*2.1);" Km wide"
500  PRINT : PRINT "Play again (Y/N) ? ";
510  LET a$=INKEY$: IF a$="" THEN  GO TO 510
520  IF a$="y"  OR a$="Y" THEN  RUN
530  IF a$<>"n" AND a$<>"N"  THEN  GO TO 510
540  CLS : STOP
```

BASIC-Dialekte

Das nebenstehende Programm ist für den Spectrum geschrieben. Bei anderen Computern ist die Verwendung von LET nicht unbedingt notwendig. Beim Acorn B ersetzen Sie bitte Zeile 110 wie folgt:

110 PRINT TAB(0,0)

Ersetzen Sie ferner INKEY$ in den Zeilen 190 und 510 durch INKEY$(0). Ersetzen Sie VAL a$ in Zeile 190 gegen VAL(a$), sowie INT(h) und INT(v) in den Zeilen 130 und 140 durch INT(h) und INT(v).

Beim Commodore 64 und VC 20 ersetzen Sie Zeile 110 durch:

110 PRINT CHR$(19)

Ersetzen Sie LET a$= INKEY$ in den Zeilen 190 und 510 gegen GET a$. Ersetzen Sie VAL a$ in Zeile 190 gegen VAL(a$), sowie INT h und INT v in den Zeilen 130 und 140 durch INT(h) und INT(v).

Beim Oric/Atmos ersetzen Sie Zeile 110 gegen:

110 PRINT@ 0,0

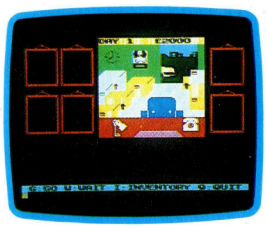
Ersetzen Sie INKEY$ in Zeile 190 und 510 gegen KEY$. Ersetzen Sie weiterhin VAL a$ in Zeile 190 gegen VAL(a$), und ersetzen Sie INT h und INT v in den Zeilen 130 und 140 durch INT(h) und INT(v).

Die schnelle Mark

Hier werden vier Szenen des Spiels „Minder“ gezeigt. Aufgabe ist es, seinen Warenbestand an Händler zu verkaufen. Die Ware befindet sich im „Lager“. Hat man keine Ware mehr, kann man neue bei den Kunden der Winchester Bar beziehen. Natürlich geht es darum, beim Kaufen wie Verkaufen den bestmöglichen Preis auszuhandeln und so das Beste aus dem vorhandenen Geld zu machen.

Die Winchester Bar

Verhandlung



Das Warenlager

Warenbestand

Arthur Daley und Terry McCann wurden durch die englische Fernsehserie „Minder“ populär. Das neue Programm der „Euston Films“ gibt Ihnen jetzt die Möglichkeit, an ihren Heldentaten teilzunehmen.

Eines der populärsten englischen Fernsehprogramme der letzten Jahre war „Minder“, und seine Umsetzung in ein Computerprogramm war längst überfällig (die Serie startete Anfang der siebziger Jahre).

„Minder“ basiert auf den etwas zwielichtigen Geschäften des Arthur Daley, der mit Waren jeder Art und Herkunft handelt, unterstützt von Terry McCann, seinem getreuen Handlanger und „Tipgeber“. Der Spieler übernimmt im Programm die Rolle von Arthur und hat die Aufgabe, innerhalb von vierzehn Tagen soviel Geld wie möglich durch An- und Verkauf von Waren zu verdienen.

Für diese Aufgabe muß man Kontakt zu anderen Händlern aufnehmen, die die eigenen Waren kaufen. Dabei stehen oft sehr seltsame Waren zum Verkauf an, wie etwa Kanister mit tierischen Abfallprodukten oder „atomare Düngemittel“. Sehr häufig bieten die Händler zu niedrige Preise, und folglich sind Fälschereien erforderlich, um mit ihnen einig zu werden.

Arthurs schwierigster Widersacher ist Detektiv Sergeant Chisholm. Der Kriminalist macht ihm sehr zu schaffen! So beginnt sein Einfluß bereits, wenn der Händler ihnen beim Betreten seiner Räumlichkeiten mitteilt, daß Detektiv Chisholm auf der Suche nach gestohlenen Computern ist. Dies ausgerechnet zu einem Zeitpunkt, da sie gekommen sind, um bereits bestellte Computer weiterzuverkaufen. Die Nachricht senkt natürlich den Preis.

Um Geschäfte zu machen und Terry zu finden, der die Ware transportieren soll, begibt man sich in die „Winchester“-Bar. In dieser Szene, wie in jeder anderen des Programms, kann man sich mit bis zu sechs Charakteren unterhalten. Ihre Gesichter erscheinen in einem der „Bilderrahmen“ auf dem Bildschirm.

Begegnung mit Leuten im Winchester ist direkt nicht erforderlich, es sei denn, man möchte mit jemandem speziell sprechen, so etwa mit dem Barkeeper Dave oder Terry.

Die Strategie beim Ankauf ist ähnlich wie beim Verkauf. Zunächst muß festgestellt werden, für welchen Preis der Geschäftspartner verkaufen will und wie groß die Menge ist. Dann kann man mit dem Feilschen beginnen. Während man den Preis aushandelt, schreitet die „Bildschirmuhr“ unaufhaltsam voran – die Verbindung wird unterbrochen, wenn es nicht in einer bestimmten Zeit zum Abschluß kommt. Will man nicht mehr weiterhandeln, drückt man einfach BYE und die Person verschwindet.

Knallhartes Feilschen

Terry ist ein zuweilen sehr schwer faßbarer Typ, und oftmals gelingt es nicht, mit ihm Kontakt aufzunehmen. Hat er eine Arbeit getan, erwartet er eine Belohnung – entweder einen Drink oder einen Bonus. Beim Preisaushandeln sollte deshalb daran gedacht werden, daß auch Terry sein Geld haben soll.

Die Preisverhandlung ist der zweifelsfrei amüsanteste Teil des Programms. Während man mit seinem Gegenspieler feilscht, erscheinen kommentierende Sprüche wie „Du langst ganz schön zu“ oder „beste Qualität“ als Gag auf dem Bildschirm. Bei einem Gebot ist die korrekte Eingabe wichtig, so etwa „Ich biete 20 Pfund“. Andernfalls versteht der Computer einen nicht.

Anders als die meisten Spiele dieser Art ist „Minder“ ebenso amüsant und interessant wie die englische Fernsehserie. Hier ist es gelungen, die Ausstrahlung des Fernsehprogramms richtig einzufangen und wiederzugeben.

Minder: Für Sinclair Spectrum, MSX, Memotech, Schneider und C 64
Hersteller: DK'tronics, Shire Hill Industrial Estate, Saffron Waldon, Essex CB11 3 AQ
Autor: Don Priestly
Programm: Cassette
Joystick: Nicht erforderlich

Arrays ohne Grenzen

In dieser Folge untersuchen wir, für welche Aufgaben sich PASCAL-Arrays am besten eignen, wie sie „gepackt" und indiziert werden, und wieviele Dimensionen sie haben können.

In vielen Programmiersprachen lassen sich reale Daten – Listen, Tabellen, Matrizen etc – am einfachsten mit Arrays darstellen. Da Arrays außerdem leicht auf einzelne Elemente zugreifen können, werden sie häufig verwendet. Hinzu kommt, daß die frühen Programmiersprachen nur diese eine Methode hatten, um Daten auf Computerebene einigermaßen übersichtlich anordnen zu können.

In diesem Kurs haben wir jedoch erfahren, wie PASCAL durch bedingungsgesteuerte Schleifen (WHILE und REPEAT) zusätzliche Flexibilität gewinnt und wie leicht sich Datenstrukturen wie Sets und Records für komplizierte Aufgaben einsetzen lassen. Daher spielen Arrays in PASCAL nicht die universelle Rolle, die sie in anderen Programmiersprachen einnehmen.

Die meisten Programmierer werden in den PASCAL-Arrays viele vertraute Strukturen wiedererkennen. Es gibt dabei jedoch zwei wichtige Punkte zu beachten. PASCALs Datenstrukturen können Programmieraufgaben oft besser und eleganter lösen als Arrays. Arrays sind in PASCAL jedoch nur wenigen Einschränkungen unterworfen. Sie lassen sich daher weitaus flexibler einsetzen als in vielen anderen Programmiersprachen.

Die Definition eines Array-Typs reserviert einen bestimmten Speicherbereich für das Array und legt den Typ der Array-Elemente und den Basistyp des Indices fest. So reserviert

```
TYPE
  CharZahl = ARRAY ['A'. .'Z'] OF integer;
VAR
  Liste : CharZahl;
```

einen Speicher für 26 Integer, die über den Array-Namen gefolgt von einer – in eckige Klammern gestellten – Indexangabe angesprochen werden können. Arrays werden in PASCAL immer mit eckigen Klammern angegeben. Ursprünglich galt diese Schreibweise auch für BASIC; da einige Zeichensätze jedoch nur runde Klammern zur Verfügung hatten, entschied man sich für die runden Klammern. Ein Integer der oben definierten Liste läßt sich folgendermaßen ansprechen:

Liste ['M'] oder Liste [pred (Symbol)]

Im zweiten Beispiel muß der Ausdruck „pred" (Symbol) natürlich einem „char"-Wert im Unterbereich von „A" bis „Z" entsprechen. Der Index einer Array-Dimension kann jeder echte Skalartyp sein. Reals oder strukturierte Typen sind jedoch nicht zugelassen, damit Eingaben wie Liste ['Zweiten'] oder Flag [3.75] vermieden werden. Die einzelnen Elemente des Array-Typs „Char-Zahl" (wie oben definiert) lassen sich folgendermaßen mit dem Wert Null initialisieren:

```
VAR
  Buchstabe : char;
  Zaehler   : CharZahl;
BEGIN
  FOR Buchstabe := 'A' TO 'Z' DO
    Zaehler [ Buchstabe ] := 0
```

Zaehler [1] wäre in dieser Struktur natürlich illegal (falscher Indextyp) und auch Zaehler ['a'] ist nicht möglich (der Index liegt außerhalb des definierten Unterbereiches). Um Fehler zu vermeiden, sollten Sie daher für alle Indexvariablen entsprechende Typennamen definieren. Beim Schreiben von eigenen Prozeduren und Funktionen ist dies sowieso kaum zu vermeiden.

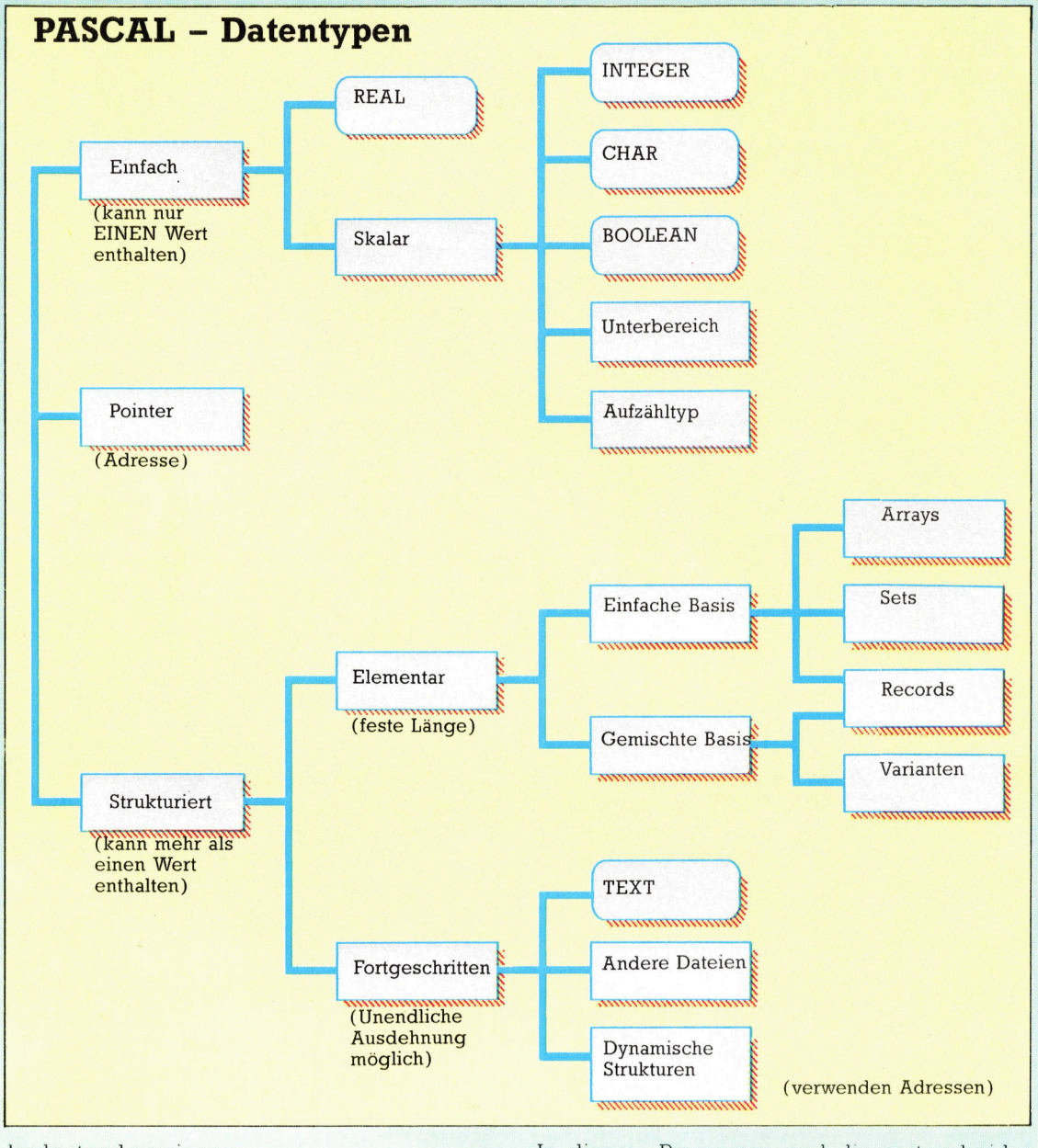
Der String-Typ

Wenn Unterbereiche von Werten bestimmt werden, die über CONST definiert wurden, können ganze Programme oft durch die Änderung einer einzigen Zeile auf andere Größen umgestellt werden. So ließe sich der Typ „String" (der in PASCAL nicht vordefiniert ist) auf folgende Weise einrichten:

```
CONST
  StringLaenge = 25;
TYPE
  StringGroesse = 1. .StringLaenge;
  String = PACKED ARRAY [StringGroesse]
    OF char;
```

Beachten Sie, daß das reservierte Wort „PAKKED" jedem reservierten Wort der Strukturtypen (SET, ARRAY, RECORD oder FILE) vorangehen darf. Durch das „Packen" von Datenstrukturen läßt sich die Speicherplatzbelegung

Die disziplinierte Datenstruktur von PASCAL zwingt den Programmierer, beim Aufbau seiner Programme sehr systematisch vorzugehen. Nur einer der hier gezeigten drei Grundtypen läßt sich nicht in weitere Kategorien unterteilen. Es muß daher für alle Daten eines Programms exakt definiert werden, zu welchem Datentyp sie gehören. Dieser zusätzliche Aufwand schafft jedoch die Voraussetzungen für elegante Lösungen, die in vielen anderen Programmiersprachen nicht möglich sind.



bedeutend verringern.

Beim Packen von Daten müssen Sie nur die String-Konstanten besonders beachten. Ein in Anführungszeichen eingeschlossener String wird von Eins (nicht von Null) an indiziert und muß daher ausdrücklich als

PACKED ARRAY [1. .N] OF char

deklariert werden (N ist die Anzahl „char" innerhalb des Strings). Sehen Sie sich folgendes Programm an:

```
PROGRAM PackString;
CONST
  Pascal = 'Pascal';
TYPE
  String  = PACKED ARRAY [ 1. .10 ] OF char;
VAR
  S       : String;
Begin
  S := PASCAL;
  S := 'Zu viele Zeichen';
  S := 'Pascal
END.
```

In diesem Programm sind die ersten beiden Anweisungen illegal, da die zugewiesenen Strings nicht die geforderte Länge haben. Die dritte Anweisung funktioniert, da der String mit vier Leerzeichen auf die erforderliche Länge gebracht wurde.

Bildschirmausgabe

Wäre String nicht als „PACKED" definiert, ließe sich die Zeichenkette nicht zuordnen. Obwohl PASCAL „read" und „write" im Normalfall nur dann für ganze Strukturen zuläßt, wenn diese auf ein Speichermedium abgelegt oder von dort eingelesen werden sollen, können die Zeichenstrings oder Stringvariablen dieses Typs dennoch direkt auf den Bildschirm ausgegeben werden. Ändern Sie zur Übung das Programm so um, daß es folgende Anweisungen unterstützt:

```
REPEAT
    write ( 'Eingabe String (Q = Ende):' ),
```

```
    ReadLn (S),
    WriteLn ( 'Der String ist :"',S,'"')
  UNTIL S [1] IN ['Q','q']
```

Testen Sie mit dem laufenden Programm dann folgende Punkte:

- Werden vorangehende Leerzeichen oder TABS ignoriert?
- Was passiert, wenn die Zeile länger ist als die Länge des Strings?
- Was enthält S, wenn nur ein Zeichen eingegeben wurde?
- Was passiert, wenn nur RETURN gedrückt wurde?
- Können Sie Elemente, die zu wenig Zeichen enthalten, mit Leerzeichen auffüllen?

Speichergrenzen

In PASCAL können Arrays beliebig viele Dimensionen mit beliebig vielen Elementen enthalten. Folgende Typendefinitionen mögen Computer allerdings überhaupt nicht.

```
  RiesenTyp = ARRAY [integer] OF SET OF char,
    (* mindestens 64K × 128 Bytes!*)
  NochGroesser = ARRAY [ 1 .1000 ] OF
    RECORD
      Nachname,
      Vorname : string;
      Adresse : ARRAY [1. .5] OF String;
      Teilname : SET OF Anwesend;
      (* etc. *)
    END: (NochGroesser *)
```

Hier hat der Arbeitsspeicher des Computers den vom Programm angeforderten Bereich natürlich nicht mehr zur Verfügung.

In den nächsten Folgen werden wir uns mit dem Aufbau und dem Einsatz von Files beschäftigen. Da sich bei PASCAL nur jeweils ein oder zwei Datensätze im Arbeitsspeicher zu befinden brauchen, entfallen viele Beschränkungen die durch die Größe des verfügbaren Speichers bedingt sind.

Das Sieb des Eratosthenes

Dieses bekannte Primzahlenprogramm wird oft für Geschwindigkeitstests eingesetzt. Die Ausführzeit ist optimiert, da alle geraden Zahlen ignoriert und ein Array mit Flags (8192 Elemente für Primzahlen bis 16 384) eingesetzt wurde. Das Programm belegt jedoch mindestens acht KByte, in Programmiersprachen ohne boolesche Variablen (Ein-Byte) und Integer (Vier-Byte) sogar bis zu 32K. Da wir für die PASCAL-Version jedoch nur 16 384 Bit (2K) benötigen, können wir den ursprünglichen Algorithmus von Eratosthenes exakt umsetzen:

Alle Zahlen (1. .Maximum) befinden sich in einem Sieb.
Die Zahl Eins wird herausgenommen und (falls gewünscht) angezeigt.
REPEAT
  Die kleinste Zahl wird aus dem Sieb herausgenommen und angezeigt.
  Alle Vielfache dieser Primzahl werden aus dem Sieb herausgenommen.
UNTIL das Sieb leer ist.

Ein Set dieser Größe ist nur auf Großcomputern möglich, ein Array mit kleineren Sets kann den Ablauf jedoch simulieren. Der Array-Index jedes Sets wird über die Ganzzahlenteilung der bearbeiteten Zahl gefunden, wobei ihre Eigenschaft als Set-Element von N modulo 100 oder 1000 dargestellt ist. Diese Programmversion ist langsamer als die Array-Version.

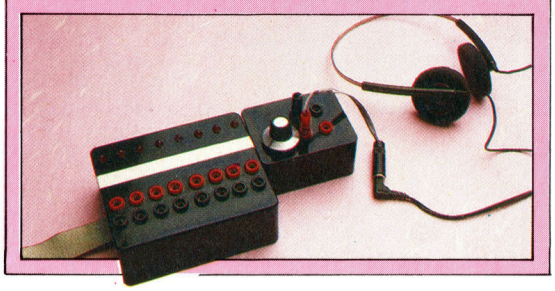
```
PROGRAMM      EratosthenesSieb      ( output );
(* Primzahlengenerierung durch den Algorithmus des
Eratosthenes *)

CONST
  SetGroesse     =   100; (* vom Compiler abhaengig *)
  PredSetGroesse =    99; (* SetGroesse - 1 *)
  MaxPrim        = 16383; (* fuer MaxInt = 32767 *)
  ListMax        =   163; (* MaxPrim DIV SetGroesse *)

TYPE
  PrimBereich    = 1 .. MaxPrim;
  Dimension      = 0 .. ListMax;
  SetBereich     = 0 .. PredSetGroesse;
  Si             = SET OF SetBereich;
  Eratosthenes   ARRAY [ Dimension ] OF Si;
  Kardinal       = 0 .. MaxInt;

VAR
  Sieb           : Eratosthenes;
  Zaehler,
  Vielfaches     : Kardinal;
  N              : Dimension;
  Index          : PrimBereich;
  Nummer         : 0 .. PredSetGroesse;

BEGIN
  WriteLn;
  WriteLn ( 'Sieb des Eratosthenes' : 50 );
  WriteLn ( '======================' : 50 );
  WriteLn;
  WriteLn;

  For Index := 0 TO ListMax DO
    Sieb [ Index ] := [0 .. PredSetGroesse ];
      (* Alle Zahlen in das Sieb laden *)

  Sieb [ 0 ] := [ 2 .. PredSetGroesse ];
(* WriteLn ( 1 ); *) (* Primzahl per Definition *)
  Zaehler := 1;

  FOR N := 2 TO MaxPrim DO
    BEGIN
      Index := N DIV SetGroesse;
      Nummer := N MOD SetGroesse;

      IF Nummer IN Sieb [ Index ] THEN
        BEGIN
          Zaehler := succ (Zaehler );
        (* WriteLn ( N ); *)
          Sieb [ Index ] := Sieb [ Index ] - [ Nummer ];

          Vielfaches := N + N;

          WHILE Vielfaches <= MaxPrim DO
            BEGIN
              Index := Vielfaches DIV SetGroesse;
              Sieb [ Index ] := Sieb [ Index ] -
                    [ Vielfaches MOD SetGroesse ];
              Vielfaches := Vielfaches + N
            END
        END
    END;

  WriteLn;
  WriteLn (Zaehler : 25, ' Primzahlen gefunden.' )

END.
```

Wellenformen

Im letzten Abschnitt des Selbstbau-Kurses haben wir das User-Port-System um einen Digital-Analogwandler erweitert. Jetzt sollen Programme entstehen, die zusammen mit dem neuen Wandler eine digitale Tonerzeugung möglich machen.

So wird's gemacht:

Mit zwei Ausgangsbuchsen am Potentiometer des Wandlermoduls können Sie die Tonerzeugung entweder mit Kopfhörern oder über die Stereoanlage verfolgen. Dazu sind kleine Vorarbeiten nötig: Wenn Sie Kopfhörer benutzen, brauchen Sie eine zum Anschlußstecker passende Buchse. Die zwei positiven Anschlüsse der Buchse werden mit den roten Buchsen des Wandlermoduls verbunden. Wer die Stereoanlage einsetzen möchte, muß in der Gebrauchsanweisung nachsehen, wo der AUX-Anschluß liegt. Den richtigen Stecker dafür bekommen Sie im Fachgeschäft. Nun wird der Ausgangsbuffer mit dem Wandlermodul und dem User Port verbunden. Als nächstes Kopfhörer bzw. Stereo-Anlage am D/A-Wandler anschließen, das Potentiometer nach links drehen und die Stromversorgung einschalten.



```
        LDX #0          2
LOOP1
        LDA TABLE,X     4
        STA PORT        4
        INX             2
        CMP #STEPS      2
        BNE LOOP1       3
  2 falls Schleife versagt
```

Jetzt können Sie das System mit einem kurzen BASIC-Programm testen. Elektrische Klangerzeugung entsteht durch eine wellenförmige Spannung. Eine angenähert wellenförmige Spannung können Sie dem D/A-Wandler entlocken, wenn Sie den Inhalt des Datenregisters von 0 auf 255 und wieder zurück auf 0 setzen. Nach dem Starten des Programms müssen Sie das Potentiometer nach rechts drehen, bis Sie den Ton gut hören.

```
10 REM **** CBM BASIC SOUND GENERATOR ****
20 DDR=56579:DATREG=56577
30 POKEDDR,255
35 N=1
40 POKE DATREG,0:FORI=1TON:NEXT:POKE DATREG,255:GOTO40

10 REM **** BBC BASIC SOUND GENERATOR ****
20 DDR=&FE62:DATREG=&FE60
30 ?DDR=255
35 N=1
40 ?DATREG=0:FOR I=1TON:NEXT:?DATREG=255:GOTO40
```

Damit die wiederholten Teile des BASIC-Programms möglichst schnell ablaufen, sind sie in einer einzigen Zeile untergebracht. Zwischen dem Setzen des Datenregisters auf 255 bzw. auf 0 ist eine Verzögerungsschleife eingebaut, deren Dauer sich durch den Wert von N in Zeile 35 beliebig variieren läßt. Je höher der Wert von N ist, um so tiefer ist der wiedergegebene Ton.

Der höchste Ton entsteht, wenn Sie die Verzögerungsschleife völlig weglassen. Bereits das einmalige Durchlaufen der Schleife hat hörbare Auswirkungen auf die Tonhöhe.

BASIC-Programme laufen relativ langsam – auch eine kleine Erhöhung von N macht den Ton sehr viel tiefer, so daß eine exakte Festlegung der Frequenz nicht möglich ist. Für diese Aufgaben eignen sich Programme in Maschinensprache.

Klangvariationen

Im nächsten Kursabschnitt werden die exakten Tonhöhen- und Lautstärkeregelungen durch Maschinensprache behandelt. Diesmal wollen wir uns auf ein Programm zur Erzeugung unterschiedlicher Wellenformen beschränken. Es gibt eine Vielzahl von Wellenformen, die alle zu unterschiedlichen Klängen führen. So läßt sich eine Sinus- und Sägezahnwelle erzeugen, die vom Rechner durch eine schnell aufeinanderfolgende Ausgabe einiger auf der Wellenform gelegener Punkte simuliert wird. Das für die Ausgabe einer Folge von Wellenpunkten nötige Maschinenprogramm ist recht einfach. Auf der nächsten Seite sind drei Wellenformen mit den entsprechenden Tabellen für die Einzelpunkte abgebildet. Die Programmschleife (siehe Kasten) gibt die zu jeweils einem Wellendurchlauf gehörigen Werte an den User Port aus.

Das größte Problem bei der Tonerzeugung ist das richtige „Timing". Man muß genau wissen, wieviele Taktzyklen der Rechner braucht, um eine Anweisung auszuführen. Bei der Berechnung der Anzahl von Taktzyklen hilft eine Formel. Sie gibt an, wieviele Zyklen der Rechner zur Ausgabe einer vollständigen Welle benötigt: Anzahl=2+(4+4+2+2+3)*Wellenpunkte−1=1+15*Wellenpunkte. Wenn die Welle in 80 Punkte zerlegt wird, braucht der Rechner also für einen Durchlauf 1201 Zyklen.

Der 6502-Prozessor braucht pro Zyklus etwa eine Millionstelsekunde. Die Anzahl der Wellendurchläufe pro Sekunde – die Tonfrequenz oder Tonhöhe – ergibt sich also aus: Fre-

quenz= 1.000.000/1201=832 Hz. Das mittlere C hat die Frequenz 512 Hz. Unser Ton sollte also um einige Noten höher liegen.

Sie sehen aus diesen Berechnungen, daß die Zahl der Einzelpunkte, in die wir die Welle zerlegen, sich auf die maximal erreichbare Tonhöhe auswirkt – eine Verdoppelung der Einzelpunkte halbiert die Tonfrequenz. Viele Wellenpunkte machen andererseits die Klangfarbe reiner. Um ein Abwägen zwischen höchster gewünschter Tonhöhe und möglichst großer Klangreinheit kommen wir nicht herum.

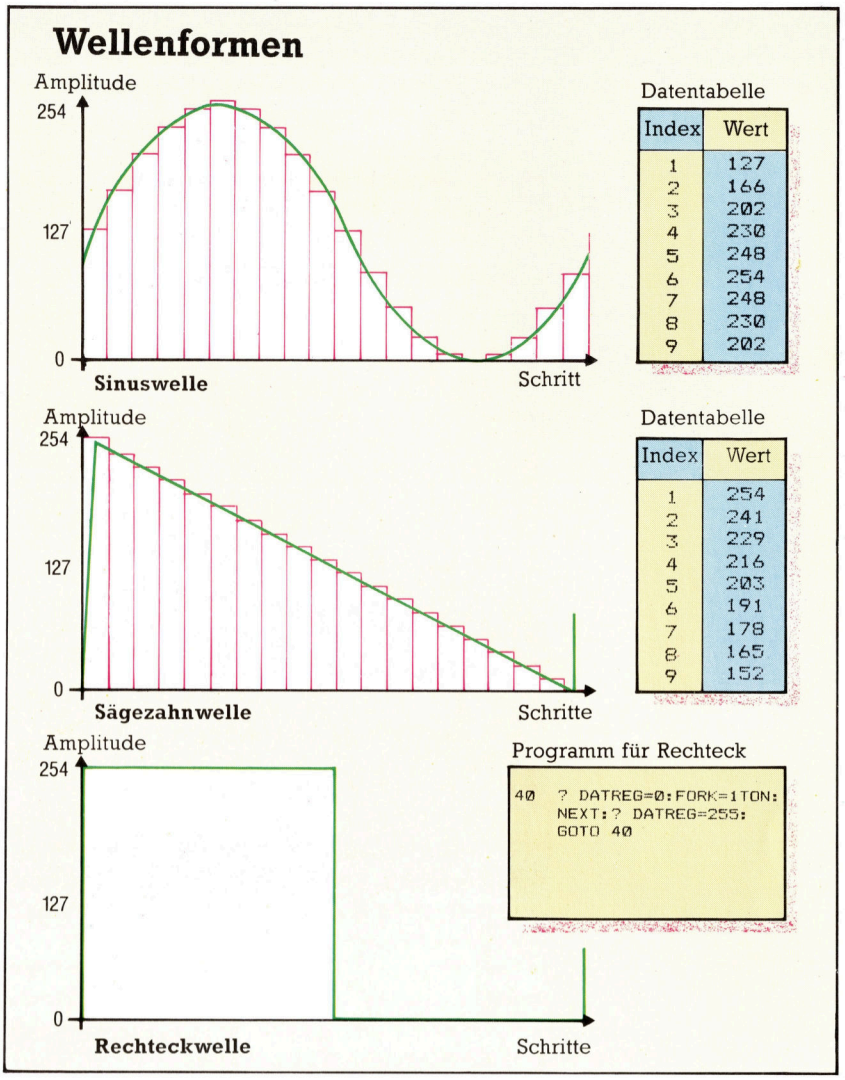
Der einzelne Durchlauf einer Welle ist zu kurz, um überhaupt hörbar zu sein. Das Programm muß daher den Teil mit den tonerzeugenden Informationen mehrfach durchlaufen. Die Häufigkeit der Wiederholung läßt sich durch einen Zähler festsetzen, der nach jedem Durchlauf vermindert wird. Damit die Spanne möglichst groß wird, haben wir dafür eine 16-Bit-Zahl vorgesehen, die auf zwei nebeneinanderliegende Speicherplätze verteilt ist. Zu Beginn des Programms werden mit SEI mögliche Interrupts unterbunden und erst am Programmschluß durch CLI wieder zugelassen – sie könnten andernfalls das exakte Timing des Programms stören.

Die Daten der Wellenform müssen sich aus dem Speicher des Rechners abrufen lassen, wobei jede Wellenform 80 aufeinanderfolgende Plätze belegt. In der Commodore-Version beginnen die Werte für Sinuskurven im Speicher bei $C000; die Daten der Sägezahnkurve haben ihren Platz ab $C050, und die Rechteckwelle fängt bei $C0A0 an. Das Maschinenprogramm kann die Daten der Sinuswelle nicht durch indirekte Adressierung abrufen, es läßt sich aber direkt aus dem BASIC mit POKE auf eine andere Wellenform umstellen. Der LDA-Teil von LDA SINE,X steht auf Speicherplatz $C103. Das LO-Byte der zu ladenden Daten steht auf Speicherplatz $C104, das HI-Byte auf $C105. Um andere Daten zu laden, muß nur der Wert in $C104 geändert werden. Beim Wert 80 in $C104 werden die Daten der Sägezahnwelle geladen, bei 160 in $C104 die Daten der Rechteckform.

In der Acorn-B-Version des Programms stehen die Daten am Anfang einer neuen Speicherseite. Das HI-Byte der Daten-Startadressen ist daher für alle Wellenformen gleich, nur das LO-Byte muß verändert werden. Beim Acorn B in Mode 7 liegt HIMEM bei &7C00. Wenn wir das Ende des Speicherbereichs um drei Seiten niedriger setzen, gewinnen wir ausreichend Platz für den Maschinencode und Wellenform-Daten.

Für den Commodore 64

```
;++++++++++++++++++++
;++++++++++++++++++++
;++                  ++
;++ CBM 64  SOUND   ++
;++    GENERATOR    ++
;++                  ++
;++++++++++++++++++++
;++++++++++++++++++++
```




```
                ;
PORT    = 56577         ;DATA REGISTER ADDRESS
STEPS   = 80            ;NO. OF STEPS IN WAVE
*=$C000
;
;++++ SET UP DATA TABLE AREA ++++
SINE    *=*+STEPS
SAW     *=*+STEPS
SQUARE  *=*+STEPS
NUMBER  *=*+2
COUNT   *=*+2
;
;++++ MAIN PROGRAM ++++
;
78              SEI
AD F0 C0        LDA NUMBER
8D F2 C0        STA COUNT       ;SET COUNT VALUE
AD F1 C0        LDA NUMBER+1
8D F3 C0        STA COUNT+1
;
LOOP2
A2 00           LDX #$00
LOOP1
BD 00 C0        LDA SINE,X      ;GET DATA
8D 01 DD        STA PORT        ;PUSH TO USER PORT
E8              INX
E0 50           CPX #STEPS
D0 F5           BNE LOOP1       ;END OF ONE CYCLE
;
;++++ DECREMENT COUNT ++++
AD F2 C0        LDA COUNT
38              SEC
E9 01           SBC #$01
8D F2 C0        STA COUNT
AD F3 C0        LDA COUNT+1
E9 00           SBC #$00
8D F3 C0        STA COUNT+1
D0 E0           BNE LOOP2       ;IF HIBYTE>0
A9 00           LDA #$00
CD F2 C0        CMP COUNT
D0 D9           BNE LOOP2       ;IF LOBYTE>0
58              CLI
60              RTS
```

Für Sinus- und Sägezahnwellen muß zuerst entschieden werden, aus wievielen Einzelschritten ein Wellendurchlauf zusammengesetzt sein soll. Die jeweils unterschiedlichen Wellenhöhen (Amplituden) werden nun berechnet und gespeichert. Vom Speicher des Computers gelangen die Amplitudenwerte nacheinander in das Datenregister des User Ports und von dort zum Digital-Analog-Wandler, der die Binärwerte in Spannungen umsetzt. Die Speicherung der Wellenform in einer Tabelle macht zeitraubende Berechnungen während des Programmablaufs überflüssig. Dadurch erstreckt sich der Frequenzbereich möglicher Töne über mehrere Oktaven.

Rechteckwellen können auch durch ein BASIC-Programm erzeugt werden.

Sie können das Maschinenprogramm über das abgedruckte Quellprogramm eingeben und es zu einem Hex-File assemblieren. Die Datenlisten für die Wellenform werden durch dieses Programm erstellt:

```
900 REM **** CBM SOUND CALLING PROGRAM ****
910 :
915 DN=8:REM FOR CASSETTE DN=1
920 IFA=0 THENA=1:LOAD"SSOUND.HEX",DN,1
999 :
1000 REM **** SET UP DATA VALUES ****
1005 S=80         :REM NUMBER OF STEPS
1007 TB=12*4096 :REM START OF DATA AREA
1008 :
1010 REM **SINE WAVE **
1020 FOR I=0 TO S-1
1030 Y=127*SIN(X)+127
1040 POKE TB+I,Y
1045 X=X+2/S
1050 NEXT I
1060 :
1065 REM **** SAW WAVE ****
1070 Y=255:TB=TB+S
1080 FOR I=0 TO S-1
1090 POKE TB+I,Y
1100 Y=Y-255/S
1110 NEXT I
1120 :
1125 REM **** SQUARE WAVE ****
1130 Y=255:TB=TB+S
1140 FOR I=0 TO S/2-1
1150 POKE TB+I,Y
1160 NEXT I
1165 Y=0
1170 FOR I=S/2 TO S-1
1180 POKE TB+I,Y
1190 NEXT I
1999 :
2000 REM **** DISPLAY DATA TABLES ****
2005 TB=12*4096
2010 FORI=TB TO TB+3*S-1
2020 PRINTI,I-TB ,PEEK(I)
2030 NEXT
```

Nach dem Programmlauf geben Sie NEW und danach dieses Prüfprogramm ein. Es zeigt die SYS- und POKE-Adressen, die für das Ansprechen des Maschinenprogramms vom BASIC aus erforderlich sind. Die gewünschte Wellenform wird abgefragt, und die Töne werden durch Tastendruck ausgelöst.

```
10 REM ****  CBM 64 SOUND  ****
20 REM **** SAMPLE PROGRAM ****
30 :
40 DDR=56579:POKE DDR,255: REM ALL OUTPUT
65 CL=49392 :REM COUNTER LOBYTE LOCATION
67 TL=49412 :REM TYPE LOBYTE LOCATION
70 SOUND=49396:REM PROGRAM START ADDRESS
75 REM ** SET COUNTER VALUE **
80 NUM=80:NHI=INT(NUM/256):NLO=NUM-256*NHI
82 POKE CL,NLO:POKE CL+1,NHI
83 :
85 PRINTCHR$(147):REM CLEAR SCREEN
86 INPUT"WAVE TYPE (0)SINE (1)SAW (2)SQUARE";WT
87 POKE TL,WT*S
88 PRINT:PRINT"PRESS ANY KEY (RUN/STOP TO END)"
90 GET A$:IFA$=""THEN90 :REM WAIT FOR KEY
100 SYS SOUND:REM CALL MACHINE CODE
110 IF A$="X" THEN 85
120 GOTO 90
```

Wenn Sie keinen Assembler besitzen oder Maschinensprache nicht verstehen, können Sie das Programm trotzdem mit Hilfe dieses BASIC-Loaders eingeben und starten. Lassen Sie in diesem Fall Zeile 920 aus.

```
10 REM **** BASIC LOADER FOR CBM SOUND ****
20 REM ****        MACHINE CODE          ****
30 FOR I=49396 TO 49447
40 READ A:POKE I,A
50 CC=CC+A
60 NEXT I
70 READ CS:IF CC<>CS THEN PRINT"CHECKSUM ERROR":END
100 DATA120,173,240,192,141,242,192
110 DATA173,241,192,141,243,192,162,0
120 DATA189,0,192,141,1,221,232,224,80
130 DATA208,245,173,242,192,56,233,1
140 DATA141,242,192,173,243,192,233,0
150 DATA141,243,192,208,224,169,0,205
160 DATA242,192,208,217,88,96
170 DATA9115:REM*CHECKSUM*
```

Für den Acorn B

Die Kombination von BASIC und Assembler ist beim Acorn B einfacher, weil das Gerät über einen eingebauten Assembler verfügt.

```
  5 REM **** BBC SOUND PROGRAM ****
  8 MODE 7
 10 HIMEM=HIMEM-&0301
 20 MC%=HIMEM+1
 30 DDR=&FE62:? DDR=255:REM ALL OUTPUT
 40 port=&FE60:REM USER PORT DATA REG
 50 steps=80  :REM NO. OF STEPS IN A WAVE
 60 table_start=MC%
 70 PROCset_up_tables
 80 PROCmachine_code
 90 PROCsample_program
999 END
1000 DEF PROCmachine_code
1010 :      -
1020 FOR opt%=1 TO 3 STEP 3
1030 P%=MC%
1060 sine=P%:  P%=P%+steps
1070 saw=P%:   P%=P%+steps
1080 square=P%:P%=P%+steps
1090 number=P%:P%=P%+2
1100 count=P%: P%=P%+2
1110 [
1120 OPT opt%
1130 \**** MAIN PROGRAM STARTS HERE ****
1150 .sound
1160       SEI
1170       LDA number
1180       STA count
1190       LDA number+1
1200       STA count+1
1220  .loop2
1230       LDX #&00
1240 .loop1
1250       LDA sine,X
1260       STA port
1270       INX
1280       CPX #steps
1290       BNE loop1
1310 \**** DECREMENT COUNT ****
1320 \
1330       LDA count
1340       SEC
1350       SBC #&01
1360       STA count
1370       LDA count+1
1380       SBC #&00
1390       STA count+1
1400       BNE loop2
1410       LDA #&00
1420       CMP count
1430       BNE loop2
1440       CLI
1450       RTS
1455 ]
1460 NEXT opt%
1480 ENDPROC
2000 DEF PROCset_up_tables
2020 REM **** SINE WAVE ****
2025 x=0
2030 FOR I=0 TO steps-1
2040 y=127*SIN(x)+127
2050 ?(table_start+I)=y
2060 x=x+2*PI/steps
2070 NEXT I
2090 REM **** SAW WAVE ****
2100 y=255:table_start=table_start+steps
2110 FOR I=0 TO steps-1
2120 ?(table_start+I)=y
2130 y=y-255/steps
2140 NEXT I
2160 REM **** SQUARE WAVE ****
2170 y=255:table_start=table_start+steps
2180 FOR I=0 TO steps/2-1
2190 ?(table_start+I)=y
2200 NEXT I
2220 y=0
2230 FOR I=steps/2 TO steps-1
2240 ?(table_start+I)=y
2250 NEXT I
2270 REM **** DISPLAY DATA TABLES ****
2280 table_start=MC%
2290 FOR I=table_start TO table_start+3*steps-1
2300 PRINT ~I,~(I-table_start),? I
2310 NEXT I
2330 ENDPROC
3000 DEF PROCsample_program
3020 counter=MC%+3*steps: REM COUNTER LOBYTE LOCATION
3030 type=loop1+1: REM TYPE LOBYTE LOCATION
3040 count_value=80
3050 count_hi=count_value DIV 256
3060 count_lo=count_value MOD 256
3070 ?counter=count_lo
3080 counter?1=count_hi
3090 CLS
3100 INPUT"WAVE TYPE (0) SINE (1) SAW (2) SQUARE":wave
3110 ?type=wave*steps
3120 REPEAT
3125 PRINT"PRESS ANY KEY (X TO EXIT)"
3130 A$=GET$
3140 CALL sound
3150 UNTIL A$="X"
3160 GOTO 3090
```

Fachwörter von A bis Z

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

Expert Systems = Expertensysteme
Im Bereich der Künstlichen Intelligenz bezeichnet man Programme für spezielle Wissensgebiete als Expertensysteme. Ein solches System kann unter Rückgriff auf das im Rechner gespeicherte Fachwissen und dessen Auswertung nach vorgegebenen Regeln fachmännischen Rat geben und damit Spezialisten entlasten oder sogar eingeschränkt „vertreten". Ansätze dazu gibt es in der medizinischen Diagnostik und bei der Fehlersuche in mechanischen und elektronischen Systemen und beim CAD.

Exponent = Exponent
Mathematische Gleichungen können oft außerordentlich umständlich werden; deshalb verwendet man bei langen Formeln häufig eine Art Kurzform. Eine Vereinbarung zur Darstellung von Produkten wie

$7x7x7x7x7 = 16\,807$

ist die Exponenten-Schreibweise. Dabei wird das fünfmalige Auftreten des Faktors 7 einfach dadurch ausgedrückt, daß ein Exponent als kleine Ziffer oben rechts neben den Faktor gesetzt wird:

$7x7x7x7x7 = 7^5 = 16\,807$

In der Potenz 7^5 ist 5 der Exponent zur Basis oder Mantisse 7. Bei der Multiplikation ergibt sich das Produkt zweier Zahlen, die als Potenzen zur gleichen Basis darstellbar sind, durch Addition der Exponenten:

$49x343 = 7^2x7^3 = 7^5 = 16\,807.$

Facsimile Transmission = Faksimileübertragung
Die Übertragung von Dokumenten und Bildern spielt in der Bürokommunikation eine wichtige Rolle. Für diese Faksimileübertragung oder Fernkopie gibt es seit vielen Jahren spezielle Geräte. Früher waren das Analogsysteme, bei denen das zu übertragende Schriftstück auf einer rotierenden Trommel befestigt und dann mit einer Fotozelle abgetastet wurde; die Übermittlung einer A4-Seite dauerte damit etwa fünf Minuten. Das elektrische Ausgangssignal der Fotozelle wurde dann über das Telefonnetz an die Empfangsstation gesendet, wo mit dem Thermodruckverfahren aus dem Bildsignal die Vorlage rekonstruiert wurde. Bei neueren Systemen wird das Original mit einem Laser abgetastet und digital an die Empfangsstation übertragen, die mit Hilfe eines Laserdruckers eine Kopie von hoher Qualität erzeugt.

Telefax-Gerät

Factorial = Fakultät
Die Fakultät einer positiven ganzen Zahl(n) ist das Produkt aller ganzen Zahlen von 1 bis n. Als Symbol dafür wird ein Ausrufezeichen hinter die Zahl gesetzt, zum Beispiel:

$1! = 1$
$2! = 1x2 = 2$
$3! = 1x2x3 = 6$
$4! = 1x2x3x4 = 24$

Von der Fakultät wird unter anderem Gebrauch gemacht, um die Anzahl von „Kombinationen" C_n^r und „Variationen" V_n^r zu berechnen:

$C_n^r = \frac{n!}{(n-r)!r!}$ bzw. $V_n^r = \frac{n!}{(n-r)!}$

Wenn Sie etwa aus einem Packen von 52 Karten an jeden Spieler fünf Blätter austeilen, dann haben Sie C_{52}^{5} verschiedene Kombinationsmöglichkeiten, und insgesamt sind (unter Berücksichtigung der Reihenfolge) genau $_{52}^{5}$ Variationen (= 311.875.200) beim Auslegen von Fünfergruppen denkbar.

Fail-Safe = Ausfallsicher
Rechner für störungsempfindliche oder risikoreiche Aufgaben werden so ausgelegt, daß sie ausfallsicher sind. Ein vollkommener Schutz gegen alle erdenklichen Fehler ist praktisch unmöglich, aber durch Ersatzsysteme und andere Sicherheitsvorkehrungen kann man gewährleisten, daß etwaige Ausfälle nur eine minimale Auswirkung auf das Gesamtsystem haben.

Ein Beispiel für ein ausfallsicheres Konzept sind die Lokomotiven mit „Totmannknopf": Wenn der Lokführer eine gewisse Zeit nicht auf diesen Hebel gedrückt hat, wird automatisch die Bremse ausgelöst.

Fan-In = Eingangslastfaktor
Der Fan-In gibt an, welche relative Belastung ein angeschlossener Eingang innerhalb eines Schaltkreises darstellt.

Fan-Out = Ausgangslastfaktor
Als Fan-Out wird die Maximalzahl von Gattereingängen bezeichnet, die von einem einzigen Ausgang zuverlässig gespeist werden können. Wird diese Zahl überschritten, so verringert sich die Differenz zwischen den Spannungspegeln für die logische 1 und die logische 0 und damit auch der Störabstand.

Bildnachweise
1093: Mike Brownlow
1094, 1096, 1109, 1118: Ian McKinnell
1095: Roy Ingram
1097, 1098: Kevin Jones
1099, 1110, 1116, 1119: Liz Dixon
1100: Mike Cloews
1101, 1103, 1105, 1106, 1107: Chris Stevens
1112: Adrian Morgan
1114: Liz Heaney

Der Wren Executive ist ein tragbarer Computer mit zwei Diskettenstationen, eingebautem Selbstwählmodem und integriertem BASIC. Das Gerät bietet so vielfältige Anwendungsmöglichkeiten. Allerdings ist der Wren nur mit einem Acht-Bit-Prozessor ausgerüstet, dafür liegt er im Preis zwischen Home- und Personal-Computern.

computer kurs

Heft 41

Gemeinsamer Nenner

Die verschiedenen BASIC-Dialekte der Homecomputer machen den Austausch von Listings schwer. BASICODE will das Problem lösen.

Voller Sound

Diese Folge von Tips für die Praxis erklärt, wie sich die mit dem D/A-Wandler erzeugten Töne regeln lassen.

Computer-Analyse

Im BASIC-Kurs geht es diesmal um ein Programm, das ein Therapie-Gespräch simuliert.

Necromancer

Das Spiel läßt sich mit einem Schauspiel vergleichen: Es ist in Akte unterteilt.

Banana-Interface

Das Banana-Interface ist eine solide gebaute Ergänzung im Bereich computergesteuerter Geräte. Für Acorn B und C 64.