Einsteigen - Verstehen - Beherrschen

DM 3,80 öS 30 sfr 3,80

computer kurs

Heft 62

Ein wöchentliches Sammelwerk

Steuerung des Robotarms

Namenserweiterung

Dateizugriffe mit OSFILE

Schüler am Computer

Programmierkurs FORTRAN

computer kurs

Heft 62

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

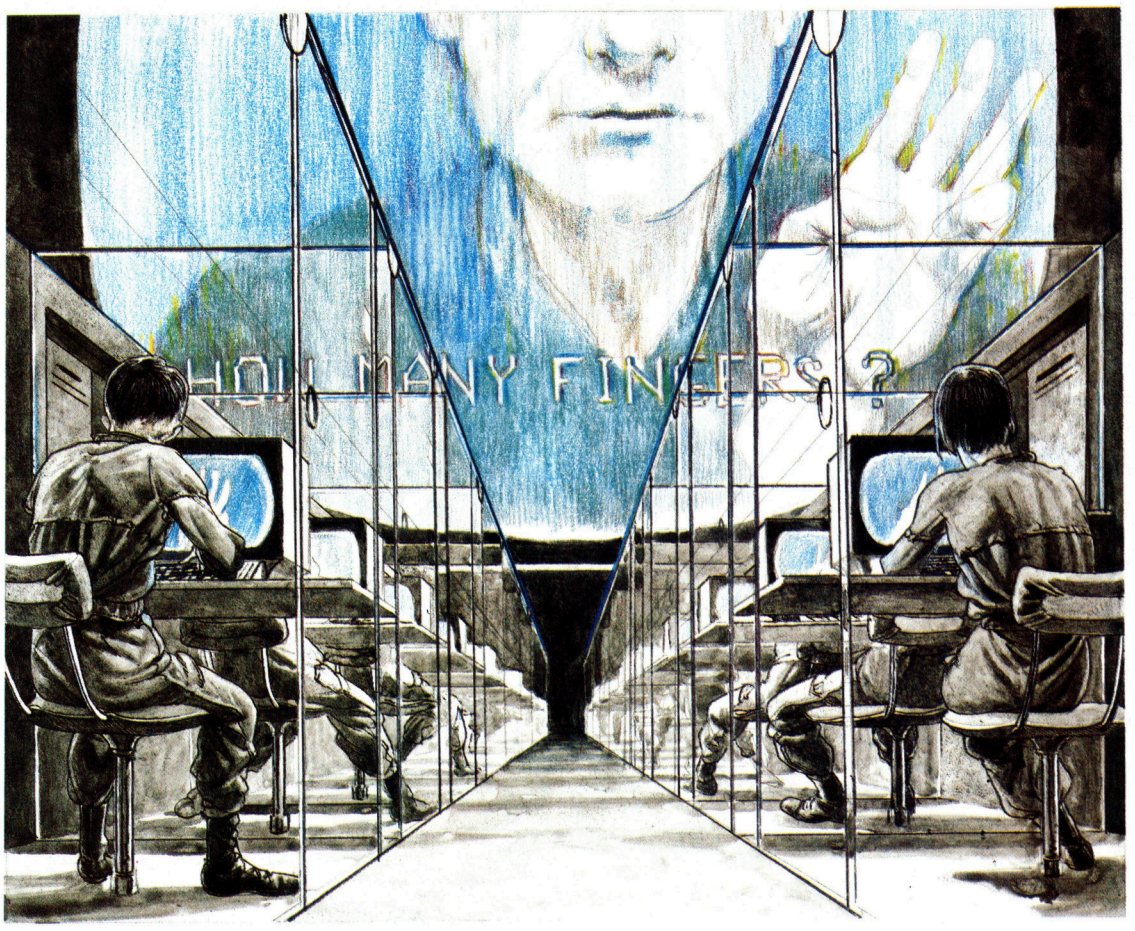
Alle 12 Hefte erscheint ein Teilindex. Die letzte Ausgabe von Computer Kurs enthält den Gesamtindex — darin einbezogen sind Kreuzverweise auf die Artikel, die mit dem gesuchten Stichwort in Verbindung stehen.

Redaktion: Winfried Schmidt (verantw. f. d. Inhalt), Peter Aldick, Holger Neuhaus, 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

Zurück zu den Anfängen?



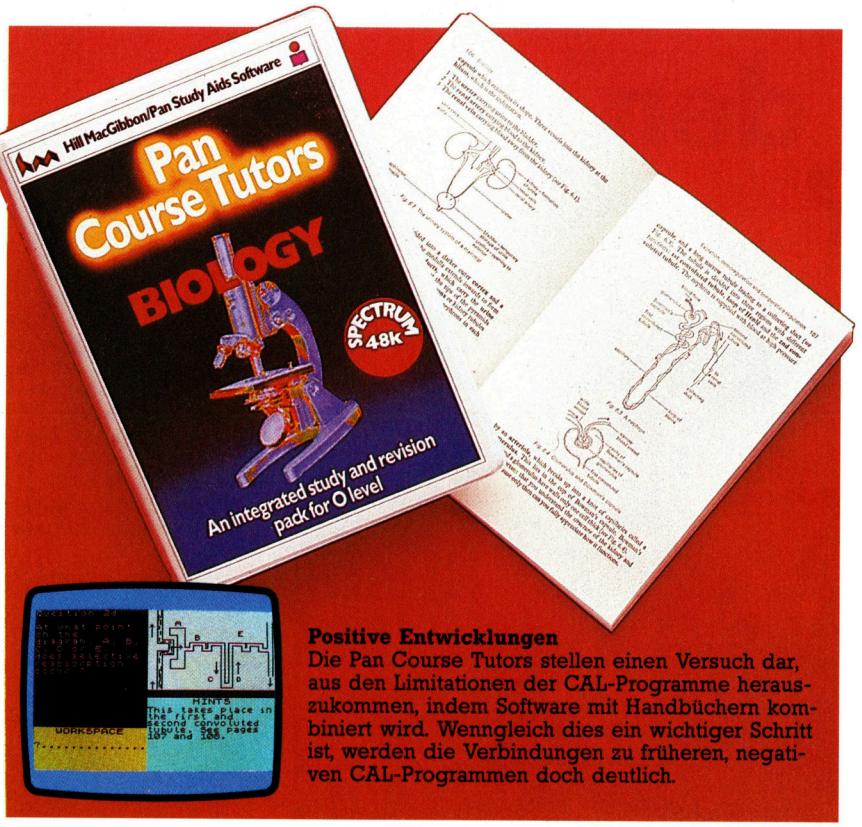
Die Betrachtung von CAL als Vorläufer eines Orwellschen Alptraums ist immer noch weitverbreitet. Compact-Disc-Technologie, interaktives Video und Netzwerke sind jedoch vielversprechende Entwicklungen für künftiges computergestütztes Lernen.

Die Bezeichnung „computer-assisted learning" (CAL) umfaßt nahezu alle Lernprozesse, bei denen ein Rechner im Spiel ist. Wir stellen die computergestützten Lern- und Lehrmethoden vor.

CAL ist ein Akronym für computergestütztes Lernen. Es ist ebenfalls als CAI bekannt (computer-aided instruction), was auf deutsch der „computer-gestützten Instruktion" entspricht oder CBL (computer-based learning). Auf deutsch: Lernen auf Computerbasis. Dieser Begriff wurde in sehr unterschiedlichen Zusammenhängen benutzt. Damit bezeichnete man alle Arten von Software, die in irgendeiner Form mit dem Lernen auf dem Computer zu tun hatten. Bei CAL wird der Computer zur Vermittlung von Informationen benutzt, zum Testen, zur Kontrolle von Fertig- und Fähigkeiten.

In England hat CAL einen geringen Stellenwert. Die Ursache dafür ist in den frühen Tagen des „programmierten Lernens" zu suchen und wurde durch die Veröffentlichung schlechter CAL-Programme im Lauf der Jahre bestärkt. Daraus resultiert die Skepsis der Lehrer. „Die CAL-Programme, die uns an Schulen zur Verfügung standen, waren unbrauchbar", kommentiert Alan Coode, Rezensent der „Sunday Times" für Computerlernprogramme. „Sie basieren auf einer merkwürdigen Ausbildungsphilosophie. Ich brauche gute Argumente, um vom Erwerb eines weiteren CAL-Programms überzeugt zu sein."

Die Geschichte der CAL-Entwicklung ist nicht sehr ermutigend. Anfang 1980, als es kaum mehr als 100 Microcomputer an britischen Schulen gab, existierte noch kein Markt für Lern-Software. Lehrer, die mit Computern arbeiten wollten, mußten ihre eigenen Programme in BASIC schreiben. Da sie auf diesem



Positive Entwicklungen
Die Pan Course Tutors stellen einen Versuch dar, aus den Limitationen der CAL-Programme herauszukommen, indem Software mit Handbüchern kombiniert wird. Wenngleich dies ein wichtiger Schritt ist, werden die Verbindungen zu früheren, negativen CAL-Programmen doch deutlich.

Gebiet keine Erfahrung hatten, war die Software wenig hilfreich. Ein Lehrer brauchte Tage, um ein Programm zu schreiben, das Addition, Subtraktion, Multiplikation und Division ausführen konnte. Fanden Kinder die richtige Lösung, erschien ein großer Haken auf dem Bildschirm. War die Lösung falsch, wurde ein Kreuz dargestellt. Kinder, die mit Mathematik sonst nicht zurechtkamen, fanden plötzlich Gefallen daran. Ein Programm wurde als Erfolg gefeiert, egal wie unlogisch und fehlerhaft es auch aufgebaut war.

Um auf dem laufenden zu bleiben, akzeptierten viele Lehrer CAL-Programme, die sie normalerweise abgelehnt hätten. Als Folge dieses fehlgeleiteten Enthusiasmus und kommerzieller Manipulation gab es Anfang der 80er Jahre eine Schwemme von CAL-Software. Fast 80 Prozent des amerikanischen Lern-Software-Katalogs „School Micro Directory" bestanden 1982 aus reiner „Pauk"-Software. Die Nachteile dieser Software wurden bald offensichtlich. So schrieb David Chandler in seinem Buch „Junge Schüler und Microcomputer": „Der Microcomputer bietet ein erschreckendes Potential. Er ermöglicht die Rückversetzung in Ausbildungspraktiken des 19. Jahrhunderts. Er ist die ultimative Waffe derer, die ‚zu den Anfängen' zurückkehren wollen, indem sie Kinder mit ‚Input' und ‚Output' dazu erziehen, sich ‚richtig zu verhalten', und anschließend findet eine ‚Qualitätskontrolle' statt . . ."

Traurig genug, daß viele CAL-Programme diese Auffassung stützen. In einer Reihe von Software-Produkten werden schlechte Lehrgewohnheiten gefördert, erwartet man unkreative Antworten oder nicht mehr als Ja oder Nein auf Fragen. Obwohl behauptet wird, daß Kinder so nach ihren Vorstellungen lernten, ist die Feststellung wohl richtiger, daß Kinder nach „Computer-Vorschriften" lernen.

Es gibt allerdings Anzeichen für eine neue Betrachtungsweise in Sachen CAL. Die meisten Heimcomputer werden von 11- bis 18jährigen zum Spielen benutzt. Dieser Markt hat dazu beigetragen, eine breite Hardware-Basis zu etablieren, die von vielen Leuten zu Lernzwecken verwendet wird. Dazu kommen die zahlreichen Veröffentlichungen von CAL-Software in jüngerer Zeit, die diesen Trend unterstützen. Eines der führenden Ausbildungs-Software-Häuser, Hill MacGibbon, hat den neuen Markt erkannt und viel investiert, um anwendbare CAL-Programme zu entwickeln, die den Lehrplan-Anforderungen genügen.

Gemeinsam mit anderen Software-Häusern legte Hill MacGibbon die beiden Hauptanwendungsbereiche für Lern-Software fest. Zum einen werden die Programme für einen bestimmten Zweck geschrieben, um so dem Schüler zu ermöglichen, sich auf Prüfungen vorzubereiten. Zum anderen geht es um Ausbildung im weitesten Sinne, wobei der Computer dem Schüler dabei hilft, Wissensgebiete zu erforschen, die ihm sonst verschlossen blieben.

Ein gutes Beispiel für die erste Kategorie ist die „Pan Course Tutor"-Serie, die gemeinsam von Pan Books und Hill MacGibbon veröffentlicht wurde. Die Serie umfaßt sechs Titel – Mathematik, Biologie, Chemie, Wirtschaftskunde, Französisch und Physik – und ist um jeweils ein Textbuch ergänzt. Damit verzichtet man darauf, große Informationsblöcke ins Programm zu integrieren und nutzt die Software lediglich zum interaktiven Lernen. Diese Kombination von Buch und Programm wird solange angeboten werden, wie Heimcomputer-Benutzer vorwiegend mit Acht-Bit-Maschinen arbeiten. Sobald andere Prozessoren mit größerer Speicherkapazität zur Verfügung stehen, kann auf Begleitmaterial verzichtet werden.

Buch und Programm

In der Pan Course Tutor-Serie stellen die Programme Fragen zu den einzelnen Themenbereichen. Der Vorteil der Computerbenutzung ist psychologisch begründet: Man kann nicht einfach zur „Antwortseite" umblättern. Erfolgt eine falsche Antwort, erhält man Hinweise für die richtige Lösung. Ist die zweite Antwort ebenfalls falsch, wird ein weiterer Tip gegeben – gewöhnlich in Form eines Verweises auf eine Seite oder ein Kapitel im Lehrbuch, wo der Schüler nachschlagen kann. Die Antworten werden überprüft und festgehalten. Damit kann das Programm die Ergebnisse analysieren und sogar Vorschläge für den künftigen Lernschritt machen. Durch den Wechsel zwischen Buch und Computer ist es Schülern möglich, indivi-

duell zu arbeiten.

Für Kinder, die die Funktionen eines Computers kennenlernen wollen, ist es leichter, einer Lehrstunde auf dem Monitor zu folgen – die entsprechenden Befehle einzugeben und ein Feedback zu bekommen –, als ein Handbuch durchzulesen.

Monitor-Unterricht

Die Expansion dieses potentiellen Marktes für Ausbildungs-Software hat leider zur Folge, daß auch sehr schlechte Programme angeboten werden, die die Entwicklung geeigneterer Software eher verhindern. Ein bekannter Hersteller produzierte ein aufwendiges, aus vier Programmen bestehendes Paket. Eines davon enthielt das Spiel „Rounders". Der Anmerkung zufolge sollte das Spiel dazu dienen, Kindern an Regentagen die Regeln des gleichnamigen Spiels zu vermitteln. Da den Programmierern aber bei der Entwicklung beträchtliche Fehler unterliefen – und wesentliche Regeldetails unerwähnt blieben –, dürften Kinder, die versuchten „Rounders" zu erlernen, einige Probleme haben. Desgleichen wurden Versprechungen bezüglich des pädagogischen Wertes gemacht, so: „Das Programm bietet viele Möglichkeiten für den Umgang mit der Tastatur . . . beginnend damit, daß ein oder zwei Spieler ihre Namen einzutippen haben." Das Spiel ist in BASIC geschrieben und so langsam, daß nicht einmal die Spannung eines „pädagogisch wertlosen" Videospiels aufkommt.

Mit der Weiterentwicklung von Computern wird die Software von Lehrern immer kritischer betrachtet werden. Es ist damit zu rechnen, daß es zukünftig weniger CAL-Programme geben wird, die dafür aber einen wesentlich höheren Standard aufweisen und für spezielle Lehrstoffe besonders geeignet sind.

Die Trennlinie zwischen dem, was als computergestütztes Lernen und anderer Ausbildungs-Software bezeichnet wird, bei dem das Kind größere Kontrollmöglichkeiten hat – existiert eigentlich nicht. Die Lehrstoffdarstellung kann mit Simulationen und Experimenten gemischt sein, die normalerweise in den Übungsräumen der Schulen nicht durchführbar wären. Und selbst die hartnäckigsten Kritiker müssen zugeben, daß Programme, beispielsweise für Studenten, eine entscheidende Hilfe sind.

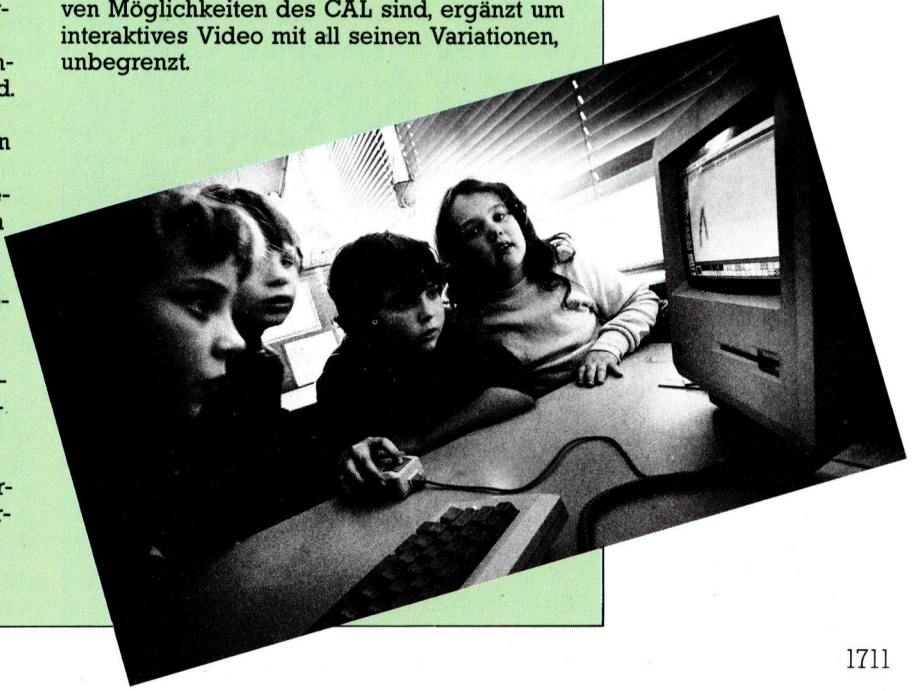
Wiederholung

Derzeit vorhandene CAL-Software ist sinnvoll für Lehrstoffe, die mit standardisierten Tests meßbar sind. Der Phantasie des Lernenden indes bleibt wenig belassen. Damit das CAL von morgen eine wirklich interessante und wertvolle Ausbildungshilfe sein wird, müssen drei Faktoren geändert werden. Antiquierte Lerntechniken, wie das „Pauken", das in vielen CAL-Programmen stattfindet, sind zu eliminieren. Leistungsfähigere Computer werden benötigt. Die zur Zeit vorhandenen Microcomputer mit ihrer geringen Speicherkapazität sind nicht imstande, die Software, die für wirklich interaktives, computergestütztes Lernen erforderlich wäre, zu verarbeiten. Der wichtigste Faktor aber ist, daß die Programmgestaltung von Pädagogen durchgeführt wird.

Ein früher Schritt in diese Richtung war PILOT, eine Computersprache, die es Lehrern ermöglichte, Lernprogramme für Kinder selbst zu schreiben. Die Arbeit in diesem Bereich wurde im Palo Alto-Forschungszentrum von Xerox fortgesetzt. Dort hat man eine „Wiederholungs-Welt" geschaffen, in der man sich einer visuellen Programmier-Umgebung bedient. In diesen Räumen entstand auch der Apple Macintosh. Mit „Rehearsal World" – so die Originalbezeichnung – können Programmierunkundige Lernsoftware erstellen. Basis dafür ist SMALLTALK. Der Programmierer bewegt „Darsteller" auf einer „Bühne" und lehrt sie Interaktion durch Übermittlung von „Tips". All dies erfolgt auf interaktiv-grafische Art.

Schon heute bietet der Computer Auszubildenden viele Lernmöglichkeiten. Eine Kombination von Autorensprachen wie MICROTEX oder Apple SUPER PILOT mit interaktivem Video würde dem Lehrer unbegrenzte Kontrollmöglichkeiten über den Lernfortschritt bzw. -prozeß geben.

In der Schule ließen sich interaktive Videos für den Klassenunterricht benutzen. Individuelles Lernen unter Einbezug der vielen interaktiven Möglichkeiten ist sicherlich eine Perspektive. Eine Quelle für Datenzugriff könnte die Schulbibliothek sein. Die kreativen Möglichkeiten des CAL sind, ergänzt um interaktives Video mit all seinen Variationen, unbegrenzt.



Die Bandbreite der neuen, auf Grafik basierenden Betriebssysteme und GEM von Digital Research geben auch Leuten, die nichts über Computer wissen, beachtliche Möglichkeiten. Alle vorhandenen CAL-Programme basierten darauf, daß der Anwender auf den Rechner keine Zugriffsmöglichkeit hat und nichts anderes tun kann, als Fragen zu beantworten.

Austausch von Variablennamen

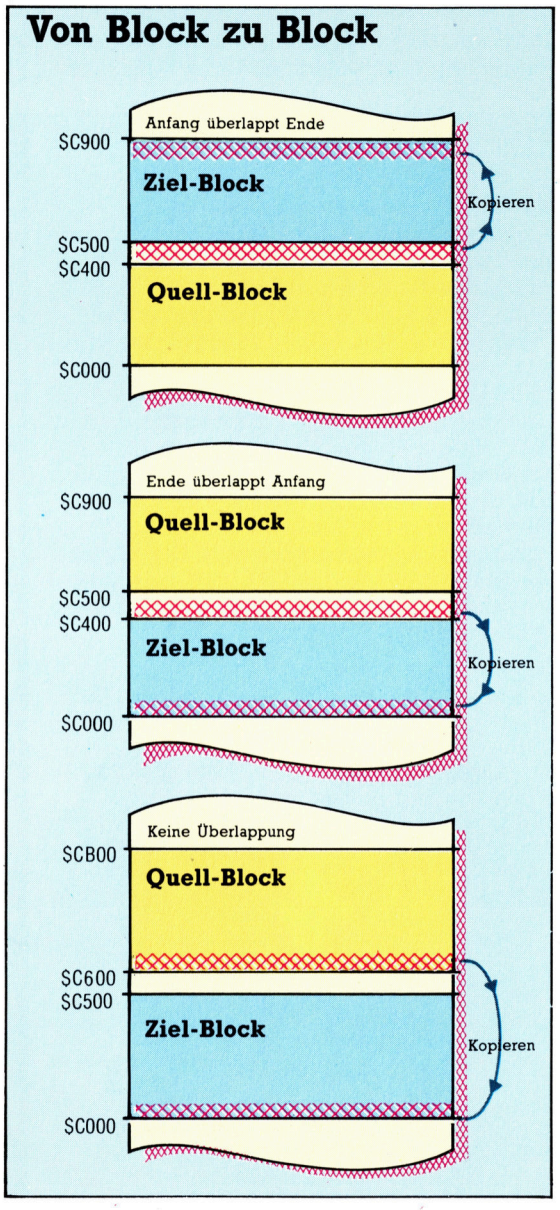
Nachdem wir uns damit näher befaßt haben, wie ein BASIC-Programm gespeichert wird, können wir nun das Variablen-Suchprogramm erweitern, so daß es einen Namen auch austauschen kann.

Das Programm zum Ersetzen von Variablennamen ist eine sehr komplexe Utility. Aus diesem Grund müssen wir ein Maschinensprache-Programm hinzufügen. Die 6502 CPU des Acorn B und 6510 CPU des Commodore verfügen über denselben Befehlssatz, so daß wir sie parallel behandeln können.

Unsere erste Aufgabe ist, eine Methode herauszufinden, mit der gleichzeitig zwei verschiedene Programme im Speicher abgelegt werden können. Wie bereits erklärt, ist dies beim Acorn B durch Änderung der Variablen PAGE und HIMEM möglich. Beim Commodore benötigen wir ein Maschinensprache-Programm, um die verschiedenen Zeiger im Zero-Page-Speicher zu ändern. Der erste Teil des Assembler-Listings (ab Label SWITCH), führt dies aus.

Die Routine SWITCH gestattet uns, zwei BASIC-Programme zu verbinden: Eins beginnt bei Adresse 800 hex (dem normalen Platz) und das andere bei Adresse 9000 hex. SWITCH untersucht zuerst den Zeiger TXTTAB, um festzustellen, welcher Programmbereich gerade aktiviert ist, und ändert entsprechend die Zeiger.

Wird TXTTAB so geändert, daß es auf den Beginn des neuen Speicherbereichs zeigt, müssen FRETOP und MEMSIZ auf das Byte nach dem letzten Byte des neuen Bereichs zeigen, wogegen FRESPC auf das Ende des Bereichs zeigt. Anschließend durchsucht das Programm unter Verwendung von VARTAB die Link-Adreßzeiger, um das Ende des BASIC-Programms zu finden. Findet es die beiden Null-Link-Adreßbytes (=Programmende), erhöht es den vorherigen Zeiger zweimal und kopiert das Ergebnis in ARYTAB und STREND. Auf diese Art zeigen VARTAB, ARYTAB und STREND auf das Byte direkt nach dem BASIC-Programm.



Die Austauschroutine muß große Blöcke des BASIC-Programms im Speicher nach oben oder unten verschieben, wenn neue Variablennamen unterschiedlicher Länge eingefügt werden. Hierbei identifiziert sie die vier möglichen Konditionen der Quell- und Zieladresse. Wenn sie immer vom Start des Quell-Blockes aus kopiert, würde, wenn der Kopf des Ziel-Blockes das Ende des Quell-Blockes überlappt, durch das Kopieren ein Teil der Quell-Daten überschrieben. Eine „intelligente" Kopierroutine vermeidet eine Überschneidung, indem sie den Quell-Block vom Ende aus kopiert.

Zeilenlängen und Zeiger

Die Hauptänderungen am BASIC-Programm sind die Unterroutinen bei den Zeilen 30500 und 30600. Die erste ermittelt das Ende des BASIC-Programms. Dabei verwendet sie beim Acorn B die Zeilenlängen-Bytes und beim Commodore die Zeiger zur nächsten Zeile.

Die Unterroutine bei Zeile 30600 ist für den Austausch der Namen verantwortlich. Haben der alte und neue Name dieselbe Länge, kann der neue einfach über den alten geschrieben werden. Unterscheiden sich die Längen, wird der Vorgang komplizierter. Entweder muß zusätzlicher Platz geschaffen oder unnötige Bytes müssen beseitigt werden, außerdem muß die Änderung an den Variablen, die die Position des Programms verzeichnen, durchgeführt werden. Ebenso ist beim Acorn B das Zeilenlängen-

Byte und beim Commodore der Zeiger zur nächsten Zeile zu modifizieren.

Für diese Aufgaben verwenden wir zwei Maschinensprache-Routinen: UP schafft zusätzlichen Platz, und DOWN beseitigt unnötige Bytes. Details über den Programmblock, der verschoben werden soll, werden über die Speicherstellen CURR, LAST und DIFF an die Routinen weitergegeben.

Für die Unterroutine UP müssen die folgenden Speicherstellen initialisiert werden:

CURR: Adresse der unteren Blockgrenze.

LAST: Wert um 1 kleiner als obere Grenze des zu verschiebenden Blockes.

DIFF: Anzahl der freizumachenden Bytes.

Für DOWN müssen sie wie folgt initialisiert werden:

CURR: Adresse der oberen Grenze des zu verschiebenden Blockes.

LAST: Wert um 1 kleiner als untere Grenze des zu verschiebenden Blockes.

DIFF: Anzahl der zu beseitigenden Bytes.

Beachten Sie, daß die beiden Unterroutinen an den entgegengesetzten Enden des zu verschiebenden Blockes beginnen und die Verschiebung in entgegengesetzte Richtung vornehmen. Auf diese Art wird verhindert, daß die Daten überschrieben werden.

Um die Acorn-B-Version des Programms zu verwenden, müssen Sie zuerst das Programm, das den Maschinencode im Speicher ablegt, eingeben und starten. Laden Sie dann das zu ändernde Programm und geben Sie ein:

P%=PAGE

Hiermit wird die Startadresse des Programms an das Variablen-Programm übergeben. Setzen Sie dann PAGE auf einen Wert über LOMEM. Anschließend laden und starten Sie das Programm. Um zum bearbeiteten Programm zurückzuführen, geben Sie ein:

HIMEM = PAGE — 1
PAGE = P%

Um die Commodore-Version des Programms zu verwenden, müssen Sie ebenfalls das Maschinenprogramm im Speicher ablegen, was mit dem BASIC-Ladeprogramm oder durch Assemblierung des Source-Codes möglich ist. Wenn Sie den Source-Code assemblieren oder den Maschinencode direkt als Maschinencode einladen, müssen Sie Nullen in die Adressen 36864, 36865 und 36866 POKEn.

Nach Einladen des Maschinen-Codes kann man mit SYS 49152 von einem Bereich auf den anderen umschalten. Haben Sie einmal vergessen, in welchem Bereich Sie sind, können Sie dies mit PRINT PEEK(44) herausfinden. Das Ergebnis ist entweder 8 (normaler Bereich) oder 144 (alternativer Bereich).

Ist der Maschinencode im Computer abgelegt, können Sie das zu ändernde Programm in den normalen Programmspeicherbereich und das Programm zum Ersetzen der Variablen in den alternativen Bereich einladen und dann das Variablenprogramm starten.

C64 Schalten und Kopieren

```
10 ADDR=49152
20 FOR LINE=1000 TO 1180 STEP 10
30 S=0
40 FOR ADDR = ADDR TO ADDR+7
50 READ BYTE
60 POKE ADDR,BYTE
70 S=S+BYTE
80 NEXT ADDR
90 READ CHECKSUM
100 IF S<>CHECKSUM THEN PRINT"DATA ERROR IN
    LINE";LINE:END
110 NEXT LINE
120 POKE36864,0:POKE36865,0:POKE36866,0
1000 DATA162,0,164,44,192,8,208,9,787
1010 DATA160,160,32,72,192,169,144,208,1137
1020 DATA7,160,144,32,72,192,169,8,784
1030 DATA162,1,134,43,133,44,134,45,696
1040 DATA133,46,160,0,177,45,170,200,931
1050 DATA177,45,224,0,208,240,201,0,1095
1060 DATA208,236,230,45,208,2,230,46,1205
1070 DATA136,16,247,165,45,133,47,133,922
1080 DATA49,165,46,133,48,133,50,96,720
1090 DATA134,51,132,52,134,55,132,56,746
1100 DATA202,136,134,53,132,54,96,160,967
1110 DATA0,177,251,164,255,145,251,160,1403
1120 DATA0,196,251,208,2,198,252,198,1305
1130 DATA251,165,253,197,251,208,234,165,1724
1140 DATA254,197,252,208,228,96,164,255,1654
1150 DATA177,251,160,0,145,251,230,251,1465
1160 DATA208,2,230,252,165,253,197,251,1558
1170 DATA208,236,165,254,197,252,208,230,1750
1180 DATA96,0,0,0,0,0,0,0,96
```

Acorn-Kopie

```
 10 MODE 7
 20 HIMEM=HIMEM-&100
 30 CURR = &70
 40 LAST = &74
 50 DIFF = &78
 60 FOR PASS = 1  TO 2
 70 P%=HIMEM
 80[ OPT 1
 90 .UP      LDY #0
100 .UP1     LDA (CURR),Y
110          LDY DIFF
120          STA (CURR),Y
130          LDY #0
140          CPY CURR
150          BNE UP2
160          DEC CURR+1
170 .UP2     DEC CURR
180          LDA LAST
190          CMP CURR
200          BNE UP1
210          LDA LAST+1
220          CMP CURR+1
230          BNE UP1
240          RTS
250 .DOWN    LDY DIFF
260          LDA (CURR),Y
270          LDY #0
280          STA (CURR),Y
290          INC CURR
300          BNE DOWN1
310          INC CURR+1
320 .DOWN1   LDA LAST
330          CMP CURR
340          BNE DOWN
350          LDA LAST+1
360          CMP CURR+1
370          BNE DOWN
380          RTS
390]
400 NEXT PASS
410PRINT~UP,~DOWN
```

Variablen-Austausch

Commodore 64
Führen Sie folgende Änderungen am Hauptprogramm aus:

```
30005 INPUT "REPLACE BY"; RS
30006 CURR = 251
30007 LAST = 253
30008 DI = 255
30035 GOSUB 30500
30036 IF ERR THEN PRINT "CAN'T FIND END OF
   PROGRAM": END
30060 IF NXTLINE = 0 THEN END
30065 CURRLINE = TEXTPTR
delete line 30085
30460 IF NAME$ = T$ THEN GOSUB 30600
30465 IF ERR THEN PRINT "NO ROOM AT LINE";
LINENO: END
```

Acorn B
Ändern Sie das Hauptprogramm wie folgt ab:

```
30005 INPUT "Replace by"; REPLACEMENT$
30006 CURR = &70
30007 LAST = &74
30008 DIFF = &78
30055 GOSUB 30500
30056 IF Outofroom THEN PRINT "Can't find end
   of program": PRINT "PAGE in wrong place?":
   END
30060 Textpointer = P%
30070 IF Lineno >32767 THEN END
30105 Lengthbyte = Textpointer
30460 IF Name$ = Target$ THEN GOSUB 30600
30465 IF Outofroom THEN PRINT "No room at
   line"; Lineno: END
```

Dateizugriff

Wir stellen das Dateisystem des Acorn B vor, das mit einer Reihe von Betriebssystemroutinen Daten an das Speichermedium sendet oder von dort empfängt. Hier untersuchen wir den Aufruf von OSFILE.

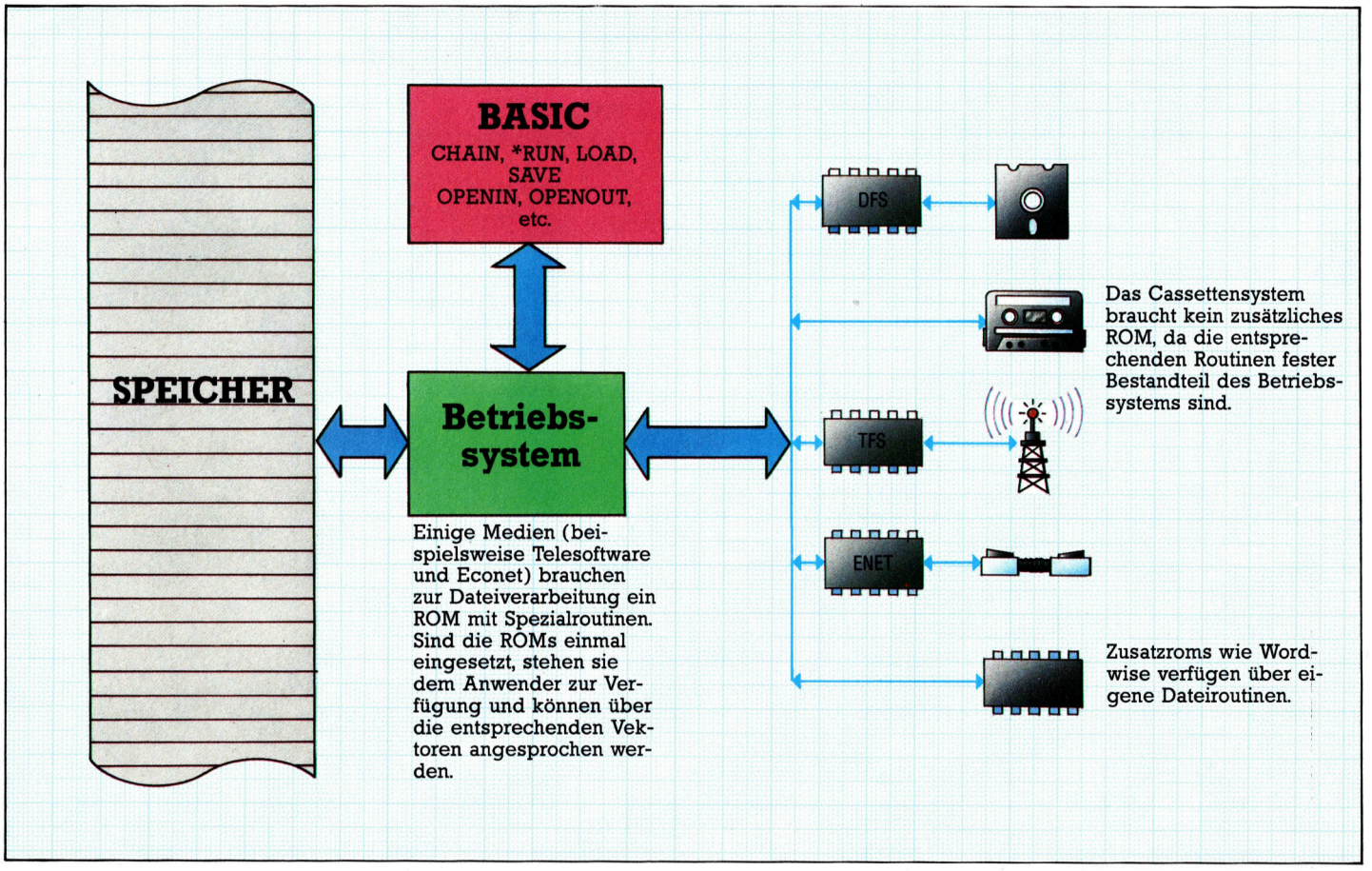
Dateisysteme enthalten eine Gruppe von Routinen, mit denen sich Daten zwischen dem Arbeitsspeicher, permanenten Speichern und anderen Medien (darunter auch Telesoftware und andere Netzwerke) übertragen lassen.

Das Dateisystem eines Computers steuert die Datenübertragung zwischen dem Arbeitsspeicher und anderen Speichermedien, beispielsweise einer Cassette oder Diskette. Der Acorn B kann neben diesen Speichermedien auch andere Dateisysteme – beispielsweise das Econet-Netzwerk und ROM-Cartridges – ansprechen. Obwohl eigentlich zu erwarten wäre, daß der Acorn B für jedes Gerät eine eigene Routine braucht, verwendet das Dateisystem die gleichen OS-Module für alle Geräte. Das heißt jedoch nicht, daß alle Geräte auf die gleiche Weise funktionieren. Viele Speichermedien brauchen ein eigenes ROM, um die physische Dateistruktur (die Anordnung der Datenbytes auf dem Medium) bearbeiten zu können. Diese Struktur wird von OS-Routinen aufbereitet, die die logische Struktur (Datensätze, Felder etc.) festlegen und für alle Speichermedien identisch sind.

Die Routinen der Dateisystem-ROMs werden über Vektoren angesprochen. Wenn Sie ein bestimmtes Dateisystem einsetzen, veranlassen Sie das OS zunächst, sich dem neuen Dateisystem anzupassen. Dabei kann es natürlich passieren, daß einige Abläufe – beispielsweise der Befehl SAVE an ein ROM – in einem bestimmten System nicht möglich sind.

Es gibt eine Vielzahl von Möglichkeiten, ein Dateisystem anzusprechen. Wir haben hier nur die direkten Methoden aufgeführt:

- Wenn Sie das Dateisystem-ROM an den dafür vorgesehenen Steckplatz setzen, wird es bei RESET automatisch aktiviert. Befindet sich mehr als ein Dateisystem-ROM in der Maschine, wird zuerst der Chip angesprochen, der unmittelbar links von BASIC-ROM liegt – vorausgesetzt, das BASIC-ROM wurde an den rechtsaußen liegenden Steckplatz für paged ROMs gesetzt.
- BREAK mit Zusatztaste: Wenn sich das entsprechende ROM in der Maschine befindet,



aktivieren BREAK und die Taste N das Econet-Dateisystem.

● Sternbefehle: Der Befehl *TAPE12 aktiviert das Cassettensystem mit 1200 BAUD, *DISC das Diskettensystem, *TELESOFT das Telesoftsystem und *NET das Econetsystem.

Das Dateisystem wird bei jedem Einschalten der Maschine aktiviert, selbst wenn es nur für das Laden oder Speichern eines Programms benutzt wird. Die beiden BASIC-Befehle SAVE und LOAD verwenden ebenso wie CHAIN, OPENUP, OPENOUT, OPENIN, INPUT, PRINT etc. die Dateiroutine des OS. Auch einige *-Befehle arbeiten mit Dateisystemroutinen, die meisten greifen jedoch auf die speziellen ROMs ihres eigenen Dateisystems zu.

Wir können nicht alle Einsatzmöglichkeiten der Routinen aufführen, die für Dateivorgänge des Acorn B gebraucht werden. Wir werden daher nur auf die wichtigsten eingehen.

Sieben OS-Routinen (OSFILE, OSARGS, OSBGET, OSBPUT, OSFSC, OSFIND und OSGBPB) und einige Aufrufe von OSWORD und OSBYTE bearbeiten Dateivorgänge.

Attribute

Da OSFILE mit vollständigen Dateien arbeitet, bietet es sich an, zuerst auf diese Routine einzugehen. OSFILE wird von Befehlen eingesetzt, die wie SAVE, LOAD, *SAVE etc. große Datenblöcke verarbeiten. Die Dateien enthalten außer ihren Daten auch sogenannte „Attribute", die Auskunft über die Ladeadresse (LOAD ADDRESS), Ausführadresse (EXECUTION ADDRESS) und die Dateilänge geben. Mit OSFILE kann der Programmierer diese Attribute steuern. Wie OSWORD braucht auch OSFILE einen Parameterblock, dessen Adresse mit den Registern X und Y an OSFILE übergeben wird. OSFILE arbeitet mit Vektoren und wird über die Adresse &FFDD aufgerufen. Der Ablauf bei Routine OSFILE hängt davon ab, welcher Wert sich beim Aufruf im A-Register befindet. Die erste der beiden folgenden Tabellen zeigt die Struktur des Parameterblocks von OSFILE und die zweite die möglichen Abläufe.

OSFILE benötigt für die Angabe der Speicherstellen nur eine 16-Bit-Adresse. In diesem Fall sind nur das LSB und das nächsthöhere Byte der Adreßangabe von Bedeutung. In der zweiten Tabelle wird deutlich, daß OSFILE für Cassetten längst nicht soviele Möglichkeiten bietet wie für ein Diskettensystem. Die Dateiattribute hängen von dem eingesetzten Dateisystem ab. Bei einem Diskettensystem ist es sehr praktisch, beispielsweise die Ladeadresse einer Datei durch eine Änderung des entsprechenden Eintrags im Parameterblock versetzen zu können. Dafür braucht nur der entsprechende Wert in A gespeichert und OSFILE aufgerufen werden.

Das folgende Programm speichert sich über einen OSFILE-Aufruf selbst:

Parameterblock für den Aufruf von OSFILE

Offset von der Adresse in den X- und Y- Registern	Beschreibung
0	LSB der Adresse des Dateinamens
1	MSB der Adresse des Dateinamens
2	LSB der Ladeadresse der Datei
3	keine Angabe
4	keine Angabe
5	MSB der Ladeadresse der Datei
6	LSB der Ausführadresse der Datei
7	keine Angabe
8	keine Angabe
9	MSB der Ausführadresse der Datei
10	LSB der Adresse der Daten, die gespeichert werden, oder Bestimmungsadresse für gelesene Daten
11	keine Angabe
12	keine Angabe
13	MSB der Datenadresse
14	LSB der Endadresse Daten + 1 für Schreibvorgänge oder Informationen über Dateiattribute bei Lesevorgängen
15	keine Angabe
16	keine Angabe
17	MSB der Datenadresse

Abläufe von OSFILE

Inhalt von A	Beschreibung der Abläufe	
	Diskette	Cassette
0	Speicherbereich sichern	Speicherbereich sichern
1	In den Katalog schreiben	Datei laden
2	Ladeadresse schreiben	Datei laden
3	Ausführadresse schreiben	Datei laden
4	Dateiattribute schreiben	Datei laden
5	Katalog lesen	Datei laden
6	Datei löschen	Datei laden
255	Datei laden	Datei laden

```
10 DIM block 20
20 DIM name 10
30 BASIC=&8023:REM für BASIC II. &801F
   für BASIC I
40 X%=block MOD 256:REM Zeiger auf den
   Parameterblock
50 Y%=block DIV 256
```

```
60 A%=0:REM Schreibvorgang angeben
70 block!0=name:REM Addresse des
   Dateinamens
80 block!2=PAGE:REM Ladeadresse des
   Programms
90                 :REM wird PAGE
100 block!6=BASIC:REM Ausführadresse
   wird der
110                :REM Einsprungspunkt für
   BASIC
120 block!10=PAGE:REM Erstes
   geschriebenes Byte
130                :REM ist PAGE
140 block!14=(TOP+1):REM Ende der
   Daten +1
150 $name="FRED":REM setzt einen legalen
   Dateinamen ein
160 CALL & FFDD:REM ausführen
170 END
```

Auch das Löschen eines Dateinamens ist einfach – man muß den Namen in den Speicher laden, mit den ersten zwei Einträgen des Parameterblocks darauf zeigen, das A-Register auf 6 setzen und OSFILE aufrufen. Nach einem Aufruf von OSFILE mit A=5 enthalten Byte 2 und 5 des Parameterblocks die Ladeadresse der Datei und Bytes 6 und 9 die Ausführadresse. Wenn für den Namen, auf den die Adresse des Parameterblocks zeigt, kein Eintrag auf der Diskette gefunden wird, dann enthält A bei der Rückkehr von der Routine den Wert 200. Wird eine Datei gefunden, steht A auf 1.

*-Befehle der OS-Dateisystemroutinen

*Befehl	Beschreibung
*CAT oder *	Legt einen Katalog der Dateien an, die im Augenblick auf dem Dateisystem verfügbar sind.
*SAVE	Schreibt einen Byteblock vom Arbeitsspeicher auf das permanente Speichermedium. Beispiel: *SAVE "Dateiname" Anfang Ende Ausführung "Anfang" ist die Hexadezimaladresse des ersten Bytes, das gespeichert werden soll, und "Ende" die Hexadezimaladresse des letzten Byes +1. "Ausführung" ist die Ausführadresse der so entstandenen Bytedatei. „Ausführung" ist ein Parameter, der wahlweise eingesetzt werden kann. Wenn er fehlt, wird die Ausführadresse auf die Startadresse gesetzt. "Ende" kann durch "+Länge" ersetzt werden, das die Anzahl der Bytes angibt, die gespeichert werden sollen.
*LOAD	Lädt eine Bytedatei an die angegebene Adresse im Arbeitsspeicher. Der Befehl hat folgendes Format: *LOAD "Dateiname" Adresse "Adresse" ist die Hexadezimaladresse der Position, an die die Bytes geladen werden sollen. Fehlt der Parameter, dann werden die Bytes an die Adresse geladen, von der aus sie gesichert wurden.
*RUN	Entspricht einer Maschinencodeversion des BASIC-Befehls CHAIN. *RUN "Dateiname" lädt eine Bytedatei an die Adresse, von der aus sie gespeichert wurde und führt sie als Maschinencodeprogramm von der beim SAVE angegebenen Adresse an aus.
*SPOOL und *EXEC	Diese beiden Befehle wurden schon früher erklärt.
*OPT1,x	Dieser Befehl teilt dem OS mit, ob Meldungen dargestellt werden sollen. Bei x=0 werden keine Meldungen angezeigt; x=1 liefert kurze, x=2 längere Meldungen.
*OPT2,x	Dieser Befehl steuert die Abläufe des OS, wenn während eines Dateivorgangs ein Fehler auftritt. Bei x=0 wird der Fehler ignoriert, x=1 teilt dem Anwender mit, es nochmals zu versuchen (der Fehler wurde gefunden), und x=3 veranlaßt bei einem Fehler den Abbruch des Dateivorgangs.
*OPT3,x	Legt Pausen fest, die beim Schreiben auf Cassette eingesetzt werden, x gibt Einheiten von 0,1 Sekunden an.
*OPT4,x	Dieser Befehl wird nur bei Diskettensystemen eingesetzt. Er setzt auf der Diskette die SHIFT-BREAK-Optionen (siehe betreffendes Handbuch).

Frühe Formeln

Obwohl FORTRAN schon Mitte der fünfziger Jahre entwickelt wurde, hat sich die Sprache viele ihrer ursprünglichen Merkmale erhalten. Im ersten Artikel dieser neuen Serie sehen wir uns den allgemeinen Programmaufbau an und untersuchen, wie Ein- und Ausgabe und die mathematischen Abläufe funktionieren.

FORTRAN wird im allgemeinen als die erste Hochsprache angesehen. Die Sprache wurde in einer Zeit entwickelt, als Computer fast ausschließlich für die Zahlenverarbeitung eingesetzt wurden und kaum Möglichkeiten der Stringverarbeitung hatten. FORTRAN ist noch immer eine der wenigen Sprachen, die komplexe Zahlen direkt verarbeiten. Mit ihrer mathematischen Struktur lassen sich schnelle und leistungsfähige Routinen schreiben, die sich auch für Grafik eignen. Es gibt eine Reihe Maschinen, deren Betriebssysteme zumindest teilweise in FORTRAN geschrieben wurden.

Über Jahrzehnte entstand eine große Zahl Subroutinen für ein breites Spektrum numerischer Anwendungen. Dies ist einer der Hauptgründe für die Beliebtheit der inzwischen schon recht betagten Sprache. Da FORTRAN die einfachste Methode ist, diese Software einzusetzen, scheint sie noch für einige Zeit eines der wichtigsten Programmierinstrumente für Wissenschaftler und Ingenieure zu bleiben.

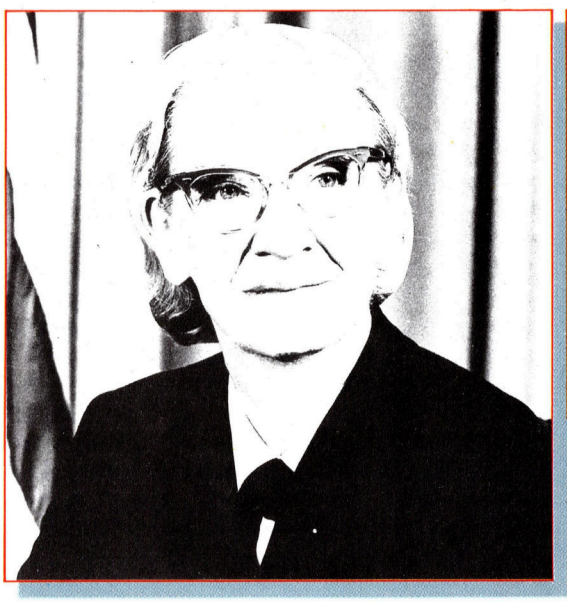
Auch für Microcomputer gibt es FORTRAN-Versionen, die jedoch nur selten den vollen Standard des aktuellen FORTRAN 77 erreichen. Die meisten sind Abwandlungen von FORTRAN IV mit entsprechenden Erweiterungen. Wir werden mit FORTRAN IV beginnen, da diese Version den besten Einblick in die Sprache bietet und später die Erweiterungen und Abweichungen vom Standard erläutern.

Der Aufbau von FORTRAN wird wesentlich von der Tatsache geprägt, daß früher für die meisten Ein- und Ausgaben 80-spaltige Lochkarten verwandt wurden. Die letzten acht Spalten einer Karte enthielten normalerweise die Kartennummer, so daß für jeden Befehl 72 Spalten zur Verfügung standen. FORTRAN kann daher pro Zeile nur einen Befehl mit einer Länge von maximal 72 Zeichen verstehen. Wenn ein Befehl sich über mehrere Zeilen erstrecken soll, müssen die folgenden Zeilen als Fortsetzung markiert sein. Die 72 Spalten werden noch weiter eingeschränkt: Die ersten fünf Spalten sind für die Nummer des Befehls reserviert, während die sechste anzeigt, ob eine Fortsetzungszeile vorliegt. Der eigentliche Befehl beginnt erst ab Spalte 7.

FORTRAN-Zeilen brauchen nicht wie in BASIC numeriert zu werden. Jede Zeile, die von einem anderen Befehl angesprochen wird, muß jedoch mit einer Zahl als Label versehen sein. Diese Nummern brauchen nicht in einer bestimmten Reihenfolge stehen, müssen aber eindeutig sein. Ein C in Spalte Eins markiert die Zeile als Kommentar. Weil Datentypen und Steuerstrukturen fehlen, sind FORTRAN-Programme schwer zu strukturieren. Da die Sprache schwer zu lesen ist, spielen Kommentare eine wichtige Rolle.

Moderne Zeiten

1954	Herausgabe der ersten FORTRAN-Beschreibung
1957	Erster Compiler für FORTRAN I
	Herausgabe der Beschreibung für FORTRAN II
1958	Erster Compiler für FORTRAN II
1962	Herausgabe der Beschreibung für FORTRAN IV
1963	Compiler für FORTRAN IV
1977	Herausgabe der Beschreibung für das neue FORTRAN 77
1978	Die ersten Compiler für FORTRAN 77



Grace Hopper
Grace Hopper war die treibende Kraft bei der Entwicklung der ersten Programmiersprachen. In den frühen fünfziger Jahren arbeitete sie bei Remington Rand und half bei der Entwicklung eines der ersten Compiler – A-2 – und den Sprachen ARITHMATIC und FLOW-MATIC. Die Firma IBM verfeinerte in ihrer ersten FORTRAN-Version das Konzept der compilierten Sprachen.

Wie in BASIC müssen Variablen nicht deklariert werden. Sie lassen sich an jedem Punkt des Programms einfach durch Angabe eines neuen Namens einführen. Damit tritt die vom System vorgegebene Vereinbarung in Kraft, daß Variablennamen, die mit I,J,K,L,M oder N anfangen, Ganzzahlen bezeichnen und alle anderen Namen reelle Zahlen.

Die Komplexität von Ein- und Ausgabe bereitet FORTRAN-Anfängern die meisten Schwierigkeiten. Auch dieses Problem entstand aus der ursprünglichen Verwendung des starren Lochkartenformats. Jeder E/A-Vorgang braucht zwei Befehle: READ oder WRITE bestimmen das E/A-Gerät und die eingesetzten Variablen, während FORMAT Typ und Anordnung der Daten auf der Lochkarte, im Datensatz oder der Eingabezeile des Terminals angibt:

```
READ(1,100) X, I, ICHAR
100 FORMAT(F7.2,I4,A1)
```

READ veranlaßt die Ausführung eines Eingabevorgangs. Die erste Zahl hinter der Klammer bestimmt das Eingabegerät. Die Gerätenummern werden von der Soft- und Hardwareimplementation festgelegt. Die Nummern bezeichnen Terminals, Drucker (nur Ausgabe), Lochkartenleser oder Stanzer und die Dateien der Diskette oder Cassette.

Die zweite Zahl der Klammer bezieht sich auf die Nummer des entsprechenden FORMAT-Befehls. (Beachten Sie, daß FORMAT in unserem Beispiel mit der gleichen Zahl versehen wurde.) Da FORMAT nicht ausgeführt wird, sondern nur Informationen liefert, kann es an einer beliebigen Stelle des Programms stehen. Die Befehlsnummer ist der einzige Bezug, und so können mehrere READs oder WRITEs das gleiche FORMAT einsetzen. Der READ-Klammer folgt die Variablenliste, denen die Eingabewerte zugeordnet werden sollen.

FORMAT enthält eine Beschreibung jeder angegebenen Variable. Hier die wichtigsten:

- F: F7.2 zeigt beispielsweise an, daß die Eingabe für X eine reelle Zahl ist, die ersten sieben Spalten der Zeile, Karte oder des Datensatzes belegt und hinter dem Punkt zwei Ziffern hat.
- I: I4 bestimmt die Eingabe in I als Ganzzahl, die vier Spalten belegt.
- A: A1 legt den Wert der nächsten Spalte als Zeichen fest.

FORTRAN kann Zeichen nur verarbeiten, wenn sie als Ganzzahlvariablen gespeichert werden. Die maximale Zahl pro Variable hängt von der Länge ab, die das System einer Ganzzahlvariablen zuweist. Bei Ganzzahlen im 16-Bit-Format können nur A1 und A2 verwandt werden. In einigen Sprachversionen ist es jedoch auch möglich, dafür Variablen für reelle Zahlen einzusetzen. Da sich Variablen sowohl numerisch als auch alphanumerisch verarbeiten lassen und die Speicherung nicht immer dem ASCII-Code entspricht, kann hier leicht Verwirrung entstehen. So müssen längere Strings z. B. über Ganzzahlenarrays bearbeitet werden.

Die eingegebenen Werte müssen genau den Variablenbeschreibungen entsprechen, da sonst Ablauffehler auftreten. Die meisten modernen Systeme verfügen jedoch über ein „freies Eingabeformat", beispielsweise READ (1,*)A,B,C oder auch READ A,B,C, das sich speziell für die Tastatureingabe eignet, bei der exakte Abstände zwischen den einzelnen Werten nur schwer einzuhalten sind. Dieses freie Eingabeformat arbeitet wie der BASIC-Befehl INPUT. Ausgaben (mit dem WRITE-Befehl) funktionieren ähnlich:

```
WRITE(1,200) A,B,C
200 FORMAT(3F7.2)
```

Beachten Sie, daß Beschreibungen sich mehrfach einsetzen lassen und nicht für jede Variable neu angelegt werden müssen.

Zeichenketten lassen sich mit der speziellen H-Beschreibung ausgeben. Sie ist jedoch recht umständlich, da darin die exakte Zeichenzahl aufgeführt sein muß:

Vergleichende Logik

Logische Operatoren und Verbindungselemente:

FORTRAN	BASIC
.EQ.	=
.NE.	<>
.GT.	>
.GE.	>=
.LT.	<
.LE.	<=
.NOT.	NOT
.AND.	AND
.OR.	OR

Schnelle Zahlen

FORTRAN IV kennt eine Reihe numerischer Datentypen:

- **INTEGER und REAL sind Ganzzahlen und reelle Zahlen**
- **DOUBLE PRECISION sind Fließkommazahlen, die über die doppelte Menge an Speicherwörtern verfügen wie normale reelle Zahlen.**
- **COMPLEX steht für komplexe Zahlen, die einen reellen und einen imaginären Teil besitzen.**
- **LOGICAL entspricht dem PASCAL-Typ BOOLEAN und kann die Werte .TRUE. und .FALSE. annehmen (die Punkte sind notwendig).**
- **Variablennamen können bis zu sechs Zeichen lang sein. Sie dürfen nur alphanumerische Zeichen enthalten. Das erste Zeichen muß ein Buchstabe sein.**
- **Variablen lassen sich nur am Anfang des Programms mit einem Deklarationsbefehl anlegen, etwa:**

```
INTEGER NUM1,NUM2
LOGICAL FNSH
```

```
      WRITE(1,300)ANS
  300 FORMAT(18HDIE ANTWORT
     LAUTET,F7.2)
```

Bei neueren FORTRAN-Versionen lassen sich auch in Anführungsstrichen eingeschlosse Strings einsetzen:

```
      WRITE(1,300)ANS
  300 FORMAT(,DIE ANTWORT
     LAUTET',F7.2)
```

Bei der Ausgabe an einen Drucker setzt FORTRAN das erste gedruckte Zeichen zur Steuerung des Papiers ein. Ein typisches System kann beispielsweise das Leerzeichen für normale Zeichenabstände und die Zeichen 1,2,3 etc. für die gewünschte Anzahl Leerzeilen verwenden. Da diese Eigenheit oft vergessen wird, geht leicht das erste Zeichen der Ausgabe verloren, während der Drucker unvorhersehbare Dinge mit dem Papier anstellt. Hier ein Beispiel für die Druckerausgabe:

```
      WRITE(2,400)I,J,K,L,M,N
  400 FORMAT(1H,2(I6,5X))
```

Beachten Sie das zusätzliche 1H, für die Papiersteuerung. X erzeugt die angegebene Zahl Leerzeichen. Eine Beschreibungsfolge kann geklammert werden – zum Beispiel 2(I6,5X).

Wir verwenden in allen Beispielen die Gerätenummer 1 für die Ein- und Ausgabe per Terminal. Der korrekte Wert hängt allerdings von der eingesetzten Implementation ab.

Sterne statt Pfeil

Das mathematische Format und die Zuordnungen von FORTRAN sind BASIC sehr ähnlich, mit dem einzigen Unterschied, daß FORTRAN für Potenzen statt des Aufwärtspfeils (↑) doppelte Sterne (**) einsetzt. In den mathematischen Ausdrücken lassen sich reelle Zahlen, Ganzzahlen und sogar Werte mit doppelter Präzision vermischen. Dabei ist jedoch Vorsicht geboten, da leicht ein Teil der Berechnung mit der Ganzzahlenarithmetik ausgeführt wird, auch wenn das Ergebnis ein reeller Wert sein soll.

```
I=2
J=3
A=I/J
```

Hier hat A den Wert 0, da die Teilung in der Ganzzahlenarithmetik erfolgte. Die Funktion FLOAT kann die Ganzzahlenwerte jedoch in reelle Zahlen verwandeln:

```
A=FLOAT(I)/FLOAT(J)
```

liefert das korrekte Ergebnis.

Da FORTRAN als mathematische Sprache konzipiert wurde, besitzt sie selbstverständlich viele Standardfunktionen. Die Liste ist jedoch zu lang, um sie hier abdrucken zu können. Sie enthält außer den üblichen triogonometrischen und logarithmischen Funktionen auch viele, die nur für Mathematiker und Ingenieure von Bedeutung sind. Die meisten Funktionen haben eine Version für reelle Zahlen, eine für Zahlen mit doppelter Präzision und, falls nötig, auch noch das Ganzzahlenformat.

Einer der Hauptnachteile von FORTRAN war das Fehlen von Steuerstrukturen. FORTRAN 77 beseitigt dieses Problem jedoch teilweise. So wird der vertraute GOTO-Befehl sehr oft eingesetzt. Er kann jeden mit Nummer versehenen ausführbaren Befehl (nicht aber FORMAT) ansprechen.

Es gibt zwei Variationen des IF-Befehls: Das logische IF entspricht dem IF in BASIC. Es hat das Format:

```
IF (logischer Ausdruck) ausführbarer Befehl
```

Das arithmetische IF wird oft als das schlechteste Beispiel einer Steuerstruktur zitiert. Es hat folgendes Format:

```
IF (arithmetischer Ausdruck) S1, S2, S3
```

wobei S1, S2 und S3 Befehlsnummern sind. Die Steuerung wird an die Befehlsnummer S1 übergeben, wenn der Wert des arithmetischen Ausdrucks kleiner als Null ist, an S2, wenn er gleich Null ist und an S3, wenn er darüber liegt. Für die Programmierung einer dreifachen Verzweigung ist dies sehr praktisch, führt aber zu einem noch erheblich komplizierteren „Spaghetticode" als GOTO.

Durchschnitt

```
C     DAS PROGRAMM LIEST ZAHLEN
C     UND DRUCKT DEN DURCHSCHNITT AUS
C     DER INPUT WIRD DURCH DIE EINGABE
C     EINER
C     NEGATIVEN ZAHL BEENDET
C
C     ZAEHLER UND SUMME INITIALISIEREN
C
      SUMME=0
      IZAEHLER=0
C
C     NAECHSTE ZAHL LESEN
C
100   READ(1,10)X
C
C     TESTEN, OB SIE NEGATIV IST
C
      IF (X) 300,200,200
C
C     POSITIVE ZAHL
C
200   IZAEHLER=IZAEHLER+1
      SUMME=SUMME+X
      GOTO 100
C
C     ENDE DER EINGABE
C
300   AVGE=SUMME/FLOAT(IZAEHLER)
      WRITE(I,20)AVGE
      STOP
C
C     FORMATIEREN
C
10    FORMAT(F9.2)
20    FORMAT (20HDER DURCHSCHNITT
      IST,F9.2)
      END
```

Namens-erweiterung

In unserer CP/M-Serie haben wir uns bisher auf die Befehlsstruktur des Betriebssystems konzentriert. Wir wenden uns nun dem Dateiaufbau zu und beginnen mit Namenserweiterungen.

In diesem Artikel gehen wir auf die Namenserweiterungen von CP/M-Dateien genauer ein. Es wurde bereits erwähnt, daß CP/M-Dateien durch einen Hauptnamen (bis zu acht Zeichen), einen Punkt und eine Erweiterung (bis zu drei Zeichen) gekennzeichnet sind, wobei die Erweiterung auch weggelassen werden darf. Theoretisch läßt sich jede beliebige Zeichenkombination an den Hauptnamen anhängen. Einige Bezeichnungen sind jedoch für Spezialaufgaben reserviert – beispielsweise .COM, die CP/M mitteilt, daß diese Datei zu den Diskettenbefehlen gehört.

Unter CP/M entwickelte BASIC-Programme sollten mit der Erweiterung .BAS gespeichert werden. Viele BASIC-Versionen mit CP/M fügen die Erweiterung automatisch an. Obwohl BASIC-Programme normalerweise als Quell-(text)-Datei gespeichert werden, besteht ein großer Unterschied zwischen ihnen und einer üblichen Textdatei: CP/M legt die Programme im Tokenformat ab und nicht als sequentielle ASCII-Daten. Oft werden BASIC-Programme vor ihrem Einsatz compiliert. Diese Dateien enthalten Objectcode und haben die Erweiterung .INT (INTermediary – Zwischendatei).

Maschinencodeprogrammen können je nach Status mehrere Erweiterungen zugeordnet werden. Ein Programm in Assemblersprache sollte die Erweiterung .ASM haben, da der Assembler sonst eine Fehlermeldung ausgibt.

Für die Assemblierung stehen zwei Erweiterungen zur Verfügung: .HEX und .PRN. Der Ausdruck .HEX bedeutet, daß das Ergebnis der Assemblierung eine Objektdatei im Hexadezimalformat ist, während der Assembler mit .PRN ein Protokoll des Assembliervorgangs, den Objektcode, die Variablenzuordnungen und eine Fehlerliste druckt. PRN ist daher eine große Hilfe bei der Fehlersuche in Assemblerprogrammen.

Die letzte wichtige Gruppe von Erweiterungen wird für Textdateien eingesetzt. Viele CP/M-Versionen enthalten ein kleines Hilfsmodul für das Editieren von Texten (die Befehlsdatei ED). Andere Versionen gehen davon aus, daß ein umfangreicheres Textsystem vorhanden ist. Unabhängig davon werden jedoch in beiden Fällen die gleichen Erweiterungen verwandt.

Außer den reservierten Erweiterungen läßt sich fast jede Kombination von drei Zeichen als Erweiterung verwenden.

Wenn Sie eine Textdatei korrigiert haben und sich nach dem Speichern das Directory ansehen, werden Sie feststellen, daß zwei Dateien entstanden sind: Eine enthält die von Ihnen angegebene Erweiterung und eine .BAK. CP/M speichert immer eine Sicherungsdatei mit der Erweiterung .BAK (BAK-up).

Wird beim Editieren eine Datei unbeabsichtigt gelöscht, können Sie unter CP/M immer auf die Sicherungskopie zurückgreifen, die die Version vor dem jeweils letzten Sichern enthält. Jedes Speichern aktualisiert daher auch die .BAK-Version. Wenn Sie Ihre Texte oft sichern, haben Sie immer eine aktuelle Sicherheitskopie zur Verfügung.

Obwohl es nicht notwendig ist, Erweiterungen anzugeben, sind sie eine große Hilfe bei der Dateiorganisation – nicht nur aus Gründen der Übersichtlichkeit, sondern für die gesamte Diskettenverwaltung. Dabei können sogenannte „Wildcard"-Zeichen eingesetzt werden, über die sich ganze Dateigruppen mit einem Befehl ansprechen lassen.

Nehmen Sie an, der Geschäftsführer einer Firma hat im Juli eine Reihe Memos geschrieben, die zwei Monate später nicht mehr aktuell sind und aus dem Directory herausgenommen werden sollen. Er könnte sich nun das Directory ansehen und mit dem Befehl ERA jede Datei

CP/M-Steuerzeichen

In CP/M gibt es eine Reihe Steuerzeichen, die Funktionen des Systems ausführen. Obwohl einige von ihnen inzwischen überholt sind, werden sie in neuere Versionen mit übernommen, damit die Kompatibilität zwischen den verschiedenen Computern gewährleistet ist.

CTRL C	Löst einen Warmstart von CP/M aus
CTRL C	Initialisiert eine neu eingelegte Diskette
CTRL M	Beendet den PIP-Befehl und gibt die Steuerung an CP/M zurück
CTRL Z	Übergibt nach Einfügen an ED zurück
CTRL P	Leitet alle Bildschirminhalte zum Drucker Ein 2. CTRL-P schaltet Funktion wieder ab
CTRL U	Löscht die Zeile, auf der der Cursor steht
CTRL X	Löscht die Zeile, auf der der Cursor steht und stellt den Cursor an den Zeilenanfang
CTRL M	Führt die aktuelle Befehlszeile aus (statt RETURN)
CTRL H	Rückwärts- und Löschtaste
CTRL E	Ermöglicht die Eingabe einer langen Befehlszeile (ohne sie auszuführen)
CTRL R	Wiederholt die aktuelle Befehlszeile

einzeln löschen. Bei einer großen Zahl von Dateien ist dies jedoch ein langwieriger Vorgang.

Wenn er in weiser Voraussicht jedoch alle Memos des Monats Juli mit der Erweiterung .JUL versehen hat, kann er das Löschen der Dateien mit einem Befehl erledigen: ERA D:*.JUL (D gibt das Diskettenlaufwerk an, kann aber weggelassen werden). Der Stern vor dem Punkt veranlaßt CP/M, den Hauptnamen zu ignorieren und alle Dateien mit der Erweiterung .JUL zu löschen. Auch rechts vom Punkt läßt sich der Stern einsetzen. Wenn auf beiden Seiten des Punktes Sterne stehen, werden alle Dateien der Diskette gelöscht (ERA *.*).

Nehmen Sie an, der Geschäftsführer unserer Beispielfirma hat alle Juli-Dateien mit .JUL versehen, möchte aber nur einen Teil davon löschen. In diesem Fall läßt sich der Stern nicht verwenden, da damit sämtliche Dateien mit .JUL verschwinden würden. Wenn die Memo-Dateien jedoch als MEM1.JUL, MEM2.JUL, etc. gekennzeichnet sind, kann wieder ein einziger Löschbefehl eingesetzt werden, ohne daß andere Dateien gefährdet sind: ERA D:MEM?.JUL. CP/M löscht nun alle Dateien, deren Namen mit MEM anfangen und mit .JUL enden. Das ? an der vierten Stelle teilt CP/M mit, daß dieses Zeichen ignoriert werden kann.

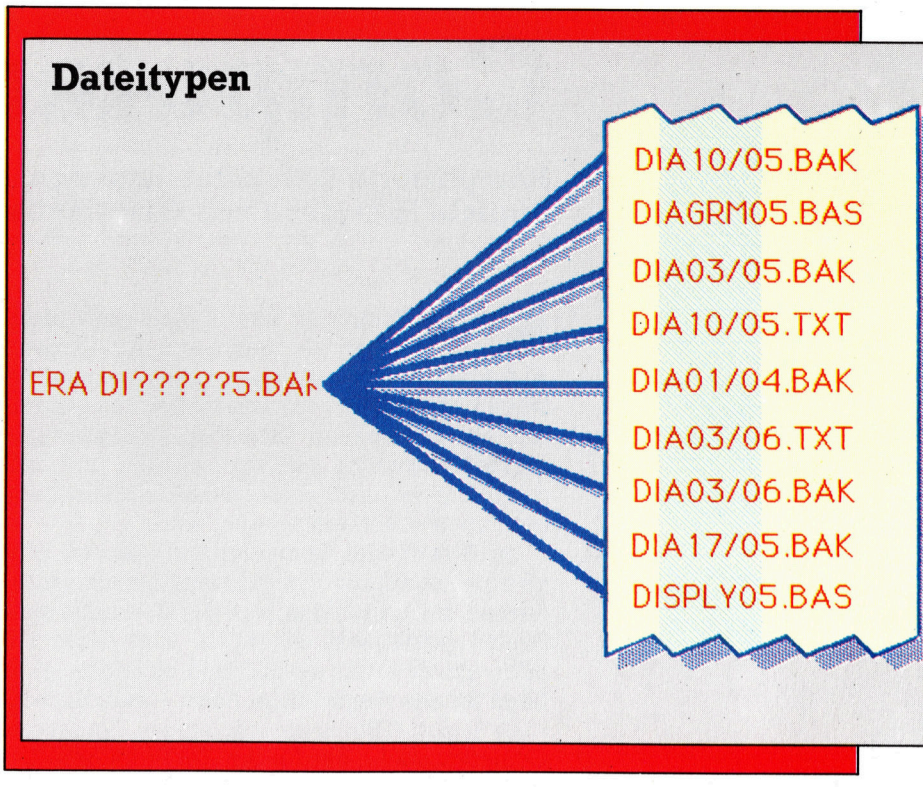
Dieser Vorgang ist für alle elf Positionen eines Dateinamens möglich. Das ? ist besonders praktisch, wenn beispielsweise nicht nur die Memos für JUL, sondern auch für JUN gelöscht werden sollen. Das Format ist hier MEM?.JU?. Auch Kombinationen zwischen ? und * sind möglich. ERA D:????Q???.* löscht beispielsweise alle Dateien, in deren Namen an fünfter Stelle ein Q steht.

Wildcard-Zeichen

Die Wildcard-Zeichen * und ? sind nicht auf ERA beschränkt, sondern lassen sich auch mit PIP, STAT und REN einsetzen. Der Befehl, alle Dateien einer Diskette auf eine andere zu kopieren, lautet: PIP B:=A:*.*.

In dieser Serie haben wir uns bisher nur mit Befehlen zur Steuerung von CP/M beschäftigt. Das Betriebssystem setzt jedoch auch Steuerzeichen (Control-Characters) ein. Das bekannteste Steuerzeichen ist vermutlich CTRL-C (das heißt das gleichzeitige Drücken von CTRL und C), das einen „Warmstart" des Computers auslöst und das Betriebssystem wieder in den Computer lädt. CTRL-C wird aber nicht nur für das Laden von CP/M eingesetzt, sondern auch beim Einlegen einer neuen Diskette.

Denken Sie daran, daß CP/M sogenannte „Log-On"-Informationen der Diskette in den Arbeitsspeicher kopiert. Wird eine Diskette gewechselt, muß das Betriebssystem darüber informiert werden, da sonst Fehler entstehen können. Ein Warmstart von CP/M legt diese Informationen für alle eingesetzten Laufwerke neu an, und so kann nach einem CTRL-C problemlos weitergearbeitet werden. Mit CTRL-C läßt sich auch die Ausführung einiger Programme anhalten, da ein Warmstart das gesamte System unterbricht.



Die sogenannten „Wildcard"-Zeichen sind eine wichtige Eigenschaft von CP/M. Durch die Eingabe der Wildcards an unterschiedlichen Positionen ignoriert der Computer beim Durchsuchen des Directories die Zeichen dieser Position. Bei einer vorausschauenden Planung der Dateinamen lassen sich daher ganze Dateilisten mit einem Befehl bearbeiten.

Viele Steuerzeichen beziehen sich auf das Editieren des Textes und werden auch von vielen Textsystemen eingesetzt, die mit CP/M arbeiten. So löscht CTRL-H beispielsweise das letzte Zeichen, CTRL-U und CTRL-X die ganze Zeile, und CTRL-P sendet den Text statt zum Bildschirm zum Drucker. Ein weiteres CTRL-P schaltet diese Funktion wieder ab.

CP/M enthält viele Steuerzeichen, die durch den Fortschritt von Hard- und Software inzwischen überflüssig geworden sind.

Viele der frühen Microcomputer besaßen weder Löschtaste noch Cursortasten und mußten ihre Bildschirmeditierung mit Hilfe des Betriebssystems ausführen. Während sich nun die ASCII-Codes der alphanumerischen und CTRL-Tasten über die Jahre nicht verändert haben, wechselt die Belegung der neueren Tasten jedoch des öfteren. Mit den ursprünglichen CTRL-Tasten sind die Systeme jedoch in jedem Fall miteinander kompatibel. Bei Veränderungen oder Tastenneubelegungen sollten auch die CP/M-Anwender berücksichtigt werden. Sie haben sich fast ein Jahrzehnt an diese Steuerzeichen gewöhnt und müssen bei einer Veränderung des Systems die gesamte Steuerung von neuem lernen.

Nach diesem Überblick über Befehle, Dateiformate und den normalen Einsatz von CP/M werden wir in der nächsten Folge tiefer in den Aufbau des Systems einsteigen und untersuchen, wie das Programm den Computer und die Peripheriegeräte anspricht.

Computerbilder

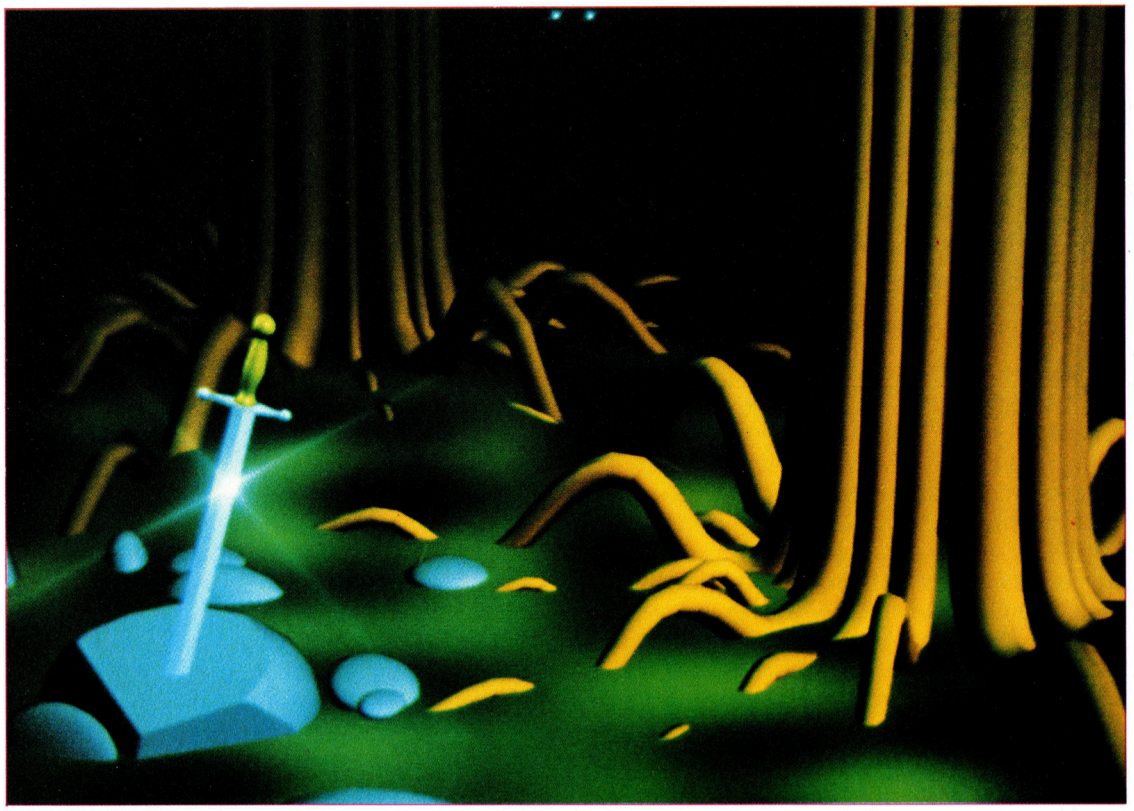
Komplizierte Darstellungsformen, die eigentlich eine Lüge sind – digitale Illusionen von Oberflächen und Spiegelungen

Soll ein Computer lügen? Sicherlich nicht, wenn man an die vielen lebenswichtigen Aufgaben denkt, mit denen Computer mittlerweile betraut sind; ganz abgesehen von den immensen finanziellen Verlusten, die durch „lügende" Computer eintreten können und aufgrund krimineller Manipulationen von Programmen auch schon eingetreten sind.

Im Bereich der Computergrafik jedoch gibt es eine Anzahl von komplizierten Darstellungsformen, bei denen das, was der Computer zum Schluß präsentiert, eigentlich eine Lüge ist, oder, um es etwas abmildernd zu sagen, eine Lüge für das menschliche Auge – eine Illusion.

Dr. James F. Blinn, ein vielseitiger Pionier der Computergrafik, der für die „Jet Propulsion Laboratories" in Pasadena, Kalifornien, arbeitet und hier unter anderem für die NASA Raumflüge wie etwa den von Voyager II zum Neptun simuliert, hat das Problem einmal so ausgedrückt: „Es gibt zwei Arten, realistische Computergrafik zu machen. Man kann mit einem riesigen Kraftakt jedes einzelne Detail erzeugen, oder man findet heraus, wie es für den Computer am einfachsten geht. Bei der ersten Methode bekommt man einfach einen Schreck vor der vielen Arbeit, und dann beginnt man zu überlegen, wie der Computer das Auge betrügen kann. Und in diesem Geschäft wird das Auge oft betrogen." Eine dieser Techniken ist das sogenannte „texturemapping". Bei dieser Methode werden modellierte Oberflächen mit materialcharakteristischen Texturen erzeugt, ohne daß jede individuelle Unregelmäßigkeit definiert werden muß. Oberflächenstrukturen können, etwa per Videokamera, von einem Objekt, oder, per Scanner, von einer Fotografie übernommen und dann um den computergenerierten Körper „geformt" werden. So kann etwa ein „Ziegelstein-Muster" auf das Modell eines Hauses gebracht werden oder auch wahlweise Holzmaserung oder die Struktur geschliffenen Kristalls auf ein noch „undefiniertes" Gefäßmodell. Fachleute sind jedoch der Meinung, daß „texture-mapping" nur eine Übergangslösung für das Problem sein kann, grundsätzlich geometrische Bilder in realitätsnahe Darstellungen zu verwandeln. Unterstützt wird „texture-mapping" jetzt schon durch die verschiedenen Schattierungstechniken, wie etwa die von Henri Gouraud, bei der zuerst die Intensitätswerte der Polygone an den Spitzen und dann entlang der Kanten gemittelt werden, um gleichmäßige Übergänge zu erreichen. Das weiterentwickelte Verfahren von Bui-Tuong Phong reagiert durch eine detailliertere Berechnung noch empfindlicher auf die Richtungseffekte spiegelnder Schlaglichter.

Nur auf den ersten Blick kompliziert wirkt diese rein geometrische Konstruktion. Hochleistung wird vom Computer jedoch für die Darstellung der Glaskugeln an den Eckpunkten gefordert. Hier sind für Spiegelung und gleichzeitiges Durchscheinen „Illusions-Techniken" wie das „ray-tracing" gefragt.



Noch kein „texture-mapping“ erfahren hat diese Darstellung eines Motivs aus der Sage um König Artus. Strukturen entstehen hier nur durch die Rasterung des Bildschirms.

So hochrealistisch diese Gefäße mit ihrer matt reflektierenden Oberfläche auch erscheinen – mangelnde Schattenführung auf der gekachelten Standfläche sorgt für eine Irritation der Wahrnehmung: Die Gefäße scheinen nicht zu stehen, sondern über den Kacheln zu schweben.

Auf Armeslänge . . .

Im letzten Abschnitt wurde unser Robot-Arm mechanisch und elektrisch vervollständigt. Jetzt geht es an den Entwurf der Steuer-Software für den Acorn B, Commodore 64 und den Sinclair Spectrum. Im ersten Teil gehen wir speziell auf den Acorn B ein.

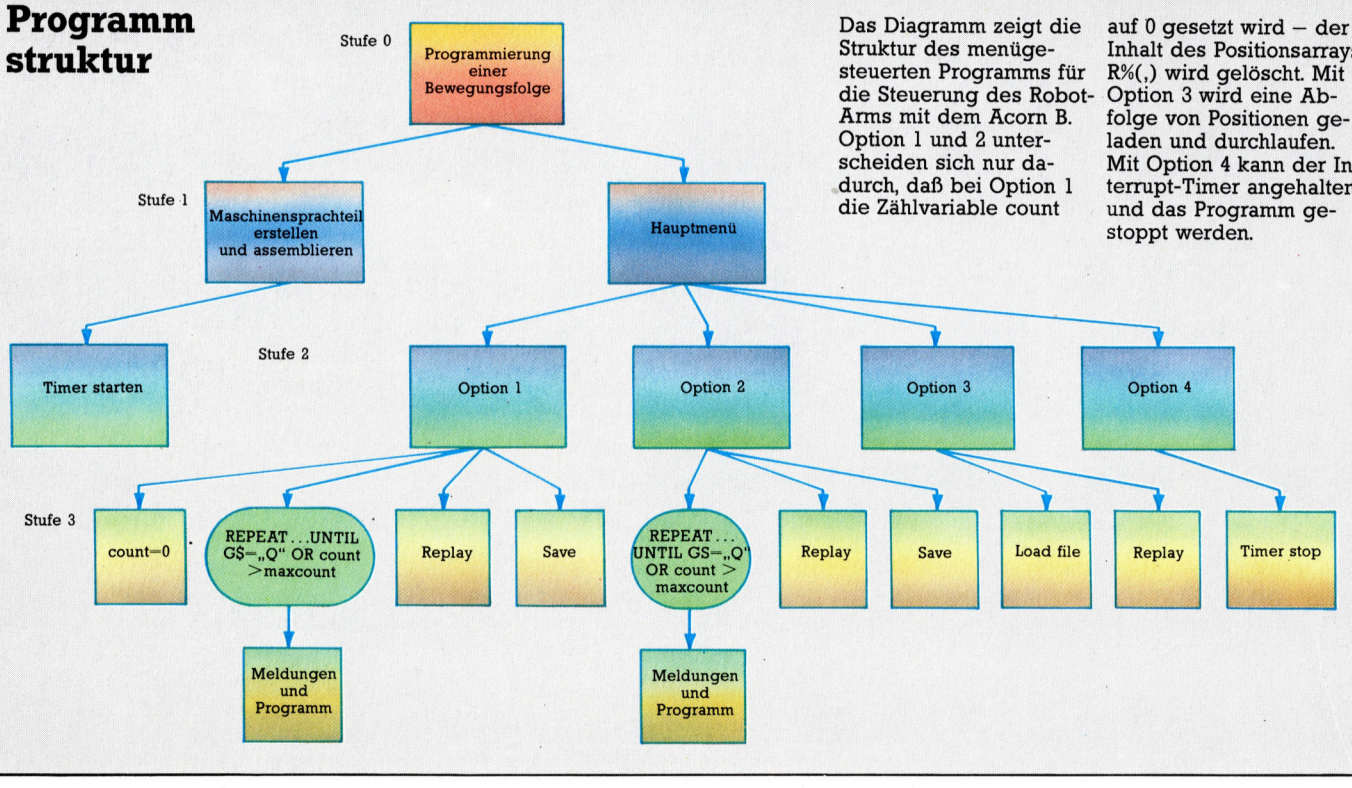
Von den Motoren des Robot-Arms führen wir Datenleitungen zum steuernden Computer, beim Acorn B sind sie mit den User-Port-Anschlüssen D0 bis D3 verbunden. Zum Ansteuern und Halten einer Position müssen die Servomotoren ständig Steuerimpulse erhalten. Mit BASIC allein geht das nicht – die Impulse werden von einem ins BASIC eingebundenen Maschinenprogramm erzeugt, das interruptgesteuert jede Sechzigstelsekunde neue Signale zu den Motoren schickt. Diese Steuertechnik wurde in einem der vorangehenden Artikel genau beschrieben. Mit dem Testprogramm für mehrere Servomotoren kann der Robot-Arm auf einwandfreie Funktion geprüft werden. Das Programm ist zwar nicht besonders komfortabel, reicht aber zum Prüfen der einzelnen Motoren aus.

Wenn das Maschinenprogramm im Hintergrund läuft, kann die BASIC-Tastaturabfrage dazu genutzt werden, die Werte direkt an die Speicherplätze für Steuerung der Motorpositionen zu bringen. Die Adresse des ersten Speicherplatzes ist „angle" – hier liegt der Wert für die Stellung von Motor 0. Speicherplatz angle+1 steuert Motor 1, usw. Motor 0 ist der „Hüft"-Motor des Roboters, Motor 1 sein Schultergelenk, Motor 2 steuert den Ellenbogen, und Motor 4 schließlich dient zum Öffnen und Schließen des Greifers. Falls einer der Werte erhöht oder erniedrigt wird, reagieren die Motoren mit einer Änderung ihres Stellwinkels. Bei großen Stellwinkel-Änderungen, etwa von 0 auf 128, werden die Bewegungen des Arms mit Verzögerungen ausgeführt.

Komplizierte Armkontrolle

Bei der Prüfung des Arms mit dem einfachen Testprogramm merkt man schnell, daß die Steuer-Software komfortabler sein könnte. Die Kontrolle des Arms ist nicht ganz einfach. Einige Industrieroboter werden durch manuelles Führen des Arms programmiert. Der Roboter „merkt" sich dabei automatisch die beschriebenen Wege, die einzelnen Stellungen und ihre Abfolge. Dazu ist jedoch eine hochentwickelte Technik nötig, mit der unser Robot-Arm nicht

Programm struktur



Das Diagramm zeigt die Struktur des menügesteuerten Programms für die Steuerung des Robot-Arms mit dem Acorn B. Option 1 und 2 unterscheiden sich nur dadurch, daß bei Option 1 die Zählvariable count auf 0 gesetzt wird – der Inhalt des Positionsarrays R%(,) wird gelöscht. Mit Option 3 wird eine Abfolge von Positionen geladen und durchlaufen. Mit Option 4 kann der Interrupt-Timer angehalten und das Programm gestoppt werden.

aufwarten kann – so haben Industrieroboter etwa Winkel-Sensoren, mit denen sie der Steuerelektronik die Stellung ihrer einzelnen Gelenke melden können.

Im industriellen Einsatz kommt es aber auch häufig vor, daß das Arbeitsfeld eines Roboters viel zu gefährlich für das Einüben der Bewegungen durch die Hand eines Menschen ist. Als Alternative dient die Steuerung des Roboters über eine Fernbedienung, bei der die Bewegungen auf Tastendruck ausgeführt werden. Die einzelnen Schritte und Stellungen werden dabei in einem Array gespeichert und können später beliebig wiederholt werden. Der in Form des Arrays definierte Bewegungsablauf kann auch auf Diskette, Lochstreifen oder Magnetband gespeichert oder neu geladen werden – eine Methode, die wir für unseren Arm übernehmen wollen.

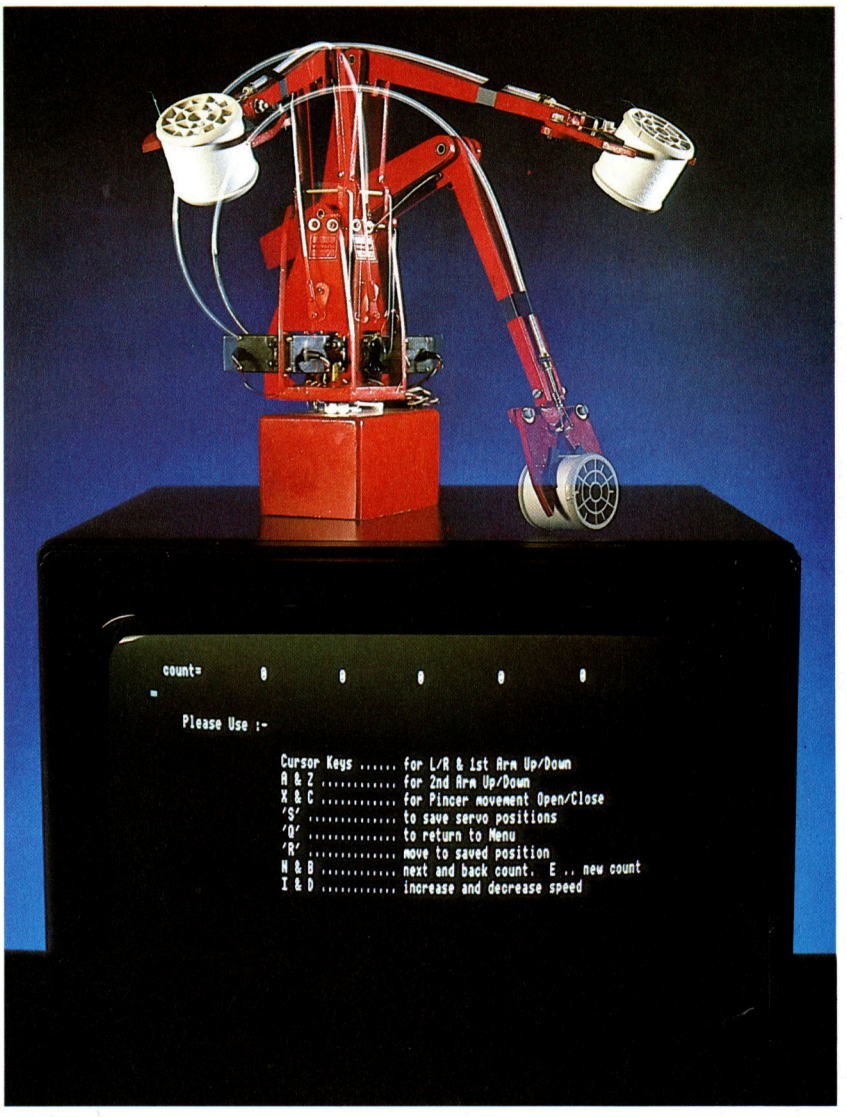
Der erste Schritt zur Entwicklung unserer Software ist es, den verschiedenen Bewegungen des Arms Tasten zuzuordnen, so daß er über die Tastatur in die Stellungen gebracht werden kann, die er sich „merken" soll. Für die Rechts/Linksdrehung des Tragkörpers und die Auf/Ab-Bewegungen des Oberarms nutzen wir die Cursortasten des Acorn B, die Tasten A und Z sollen den Unterarm bedienen. Mit X und C wird das Öffnen und Schließen des Greifers gesteuert. I und D dienen zur Regulierung der Positionierungsgeschwindigkeit, damit sich schwierige Manöver langsamer als weit ausgreifende Bewegungen ausführen lassen. Mit S wird die aktuelle Stellung des Arms in einem Array gespeichert, R stellt den Arm auf die letzte gespeicherte Position. Die einprogrammierte Bewegungsfolge kann zum Ändern mit der Taste N vorwärts und der Taste B rückwärts durchlaufen werden. E soll die festgelegten Positionen innerhalb der Sequenz ansteuern. Mit den drei letzten Tasten kann jeder programmierte Ablauf verändert werden. Die Unterprogramme „inform" und „program" im Listing zeigen die Tastenbelegung an und sorgen für fortlaufende Tastaturabfrage.

„replay" und „save"

Die Unterprogramme „replay" und „save" verwalten das Positionsarray R%(,), so daß der Ablauf wiederholt und gespeichert werden kann. Der Programmteil „loadfile" holt die in einem sequentiellen File gespeicherten Werte von R%(,) wieder zurück – damit haben wir bereits ein recht komfortables Programm, das neben der Eingabe auch das Verändern und Speichern der Bewegungsabläufe ermöglicht.

Bei der Adresse angle beginnend, steuert die Interrupt-Routine die Einstellung der Servomotoren durch Abfrage von vier Speicherplätzen, die den Stellwinkel für jeden Motor angeben. Ein Grund für die abrupten Armbewegungen im Testprogramm war, daß vom BASIC aus neue Werte direkt auf diese Speicherplätze gePOKEt wurden. Die Motoren steuern dann die neuen Positionen mit Höchstgeschwindigkeit an. Damit sich der Arm etwas gemächlicher bewegt, sollten die Werte allmählich verändert werden. Für diesen Zweck gibt es einen zweiten Speicherbereich, der bei newpos% anfängt. Hier werden die über BASIC eingegebenen neuen Werte erst einmal zwischengespeichert. Dann kann dem Hauptprogramm eine kurze Maschinenroutine hinzugefügt werden, die den Wert der vier angle-Speicherplätze in Schritten von je einer Einheit bis auf den gewünschten Wert in newpos% erhöht bzw. erniedrigt. Die entsprechende Routine wird durch das Unterprogramm „moveservo" aufgerufen.

Diese Zusatz-Routine hat noch einen weiteren Nutzen: Durch Veränderung des Endwertes der Verzögerungsschleife im Maschinenprogramm kann die Bewegungsgeschwindigkeit des Arms reguliert werden. Dieser Verzögerungsfaktor befindet sich am Speicherplatz &81 und kann zu Beginn des Unterprogramms replay verändert werden. Dann läuft eine vorher programmierte Bewegungsfolge wahlweise beschleunigt oder verlangsamt ab.



Die zu Beginn des Robot-Arm-Projekts gestellte Aufgabe kann nun ausgeführt werden: Der Arm ist fähig, Gegenstände vom Gewicht und der Größe einer Garnrolle aufzuheben und zu bewegen. Mit dem Arm-Steuerprogramm können einprogrammierte Bewegungsabläufe auch gespeichert und neu geladen werden – sie lassen sich also jederzeit wiederholen.

Robot-Arm Steuerprogramm für den Acorn B

```
   10 REM ************************
   11 REM ************************
   12 REM **                      **
   13 REM ** BBC ARM SEQUENCE    **
   14 REM **     PROGRAMMER       **
   15 REM **                      **
   16 REM ************************
   17 REM ************************
   20 :
   25 MODE 3
   30 PROCsetup
   40 CLS:PRINT TAB(5,6)"Would you like to : "
   50 PRINT TAB(25,10)"1.........Program new arm sequence"
   60 PRINT TAB(25,12)"2.........Add moves to program"
   70 PRINT TAB(25,14)"3.........Replay a file"
   80 PRINT TAB(25,16)"4.........Leave Program"
   90 G$=GET$:IF G$="1" THEN count=0
  100 IF G$="1" OR G$="2" THEN PROCinform:PROCprogram:PROCreplay:PROCsave
  110 IF G$="3" THEN PROCloadfile:PROCreplay
  120 IF G$="4" THEN CLS:PROCendtimer:END
  130 GOTO 40
  140
  150
  160 DEF PROCinform:CLS
  170 PRINTTAB(5,5)"Please Use :- "
  180 PRINTTAB(20,7)"Cursor Keys ..... for L/R & 1st Arm Up/Down"
  190 PRINTTAB(20)"A & Z ............ for 2nd Arm Up/Down"
  200 PRINTTAB(20)"X & C ............ for Pincer movement Open/Close"
  210 PRINTTAB(20)"'S' .............. to save servo positions"
  220 PRINTTAB(20)"'Q' .............. to return to Menu"
  230 PRINTTAB(20)"'R' .............. move to saved position"
  240 PRINTTAB(20)"N & B ............ next and back count.  E .. new count"
  250 PRINTTAB(20)"I & D ............ increase and decrease speed"
  260  PRINTTAB(2,2)"count="count"  ";:FOR I%=0TO3:PRINTR%(count,I%);"  ";:NEXT:
PRINT
  270 ENDPROC
  280
  290
  300 DEF PROCprogram:?&81=1
  310 REM re-assign cursor move keys
  320 *FX4,1
  330 DX=2
  340 REPEAT  :REM scan keys
  350  IF INKEY(-38)THEN DX=DX+1 :REM "I"
  360  IF INKEY(-51)THEN DX=DX-1 :IF DX<1 THEN DX=1 :REM "D"
  380  IF INKEY(-26)THEN newpos%?0=(newpos%?0)+DX: REM "LEFT ARROW" servo 0  rig
ht
  390  IF INKEY(-122)THEN newpos%?0=(newpos%?0)-DX:REM "RIGHT ARROW" servo 0 lef
t
  400  IF INKEY(-58)THEN newpos%?1=(newpos%?1)+DX:REM "UP ARROW" servo 1 up
  410  IF INKEY(-42)THEN newpos%?1=(newpos%?1)-DX:REM "DOWN ARROW" servo 1 down
  420  IF INKEY(-66)THEN newpos%?2=(newpos%?2)-DX:REM "A" servo 2 up
  430  IF INKEY(-98)THEN newpos%?2=(newpos%?2)+DX:REM "Z" servo 2 down
  440  IF INKEY(-67)THEN newpos%?3=(newpos%?3)+DX*3:REM "X"  servo 3 open jaw
  450  IF INKEY(-83)THEN newpos%?3=(newpos%?3)-DX*3:REM "C" servo 3 close jaw
  460  IF INKEY(-52) THEN FOR I%=0TO3:newpos%?I%=R%(count,I%):NEXT :REM "R" repl
ay position
  470  IF INKEY(-86)THEN count=count+1:IF count>maxcount THEN count=0
  480  IF INKEY(-101)THEN count=count-1:IF count<0 THEN count=maxcount
  490  IF INKEY(-35) THEN PRINTTAB(10,20)"GIVE VALUE FOR count ";:INPUT count :R
EM "E" keyin new count
  500  IF INKEY(-82)THEN FOR I%=0 TO Noservoes%:R%(count,I%)=newpos%?I%:NEXT:cou
nt=count+1 :REM "S" save position at current count
  510 IF count>lastmaxc THEN lastmaxc=count
  520  IF oldc<>count THEN PRINTTAB(2,2)"count="count"  ";:FOR I%=0TO3:PRINTR%(c
ount,I%);"  ";:NEXT:PRINT
  530  oldc=count
  540 FOR I%=0 TO 3:IF newpos%?I%>200 THEN newpos%?I%=200
  550 IF newpos%?I%<10THEN newpos%?I%=10
  560 NEXT
  570 PROCmoveservo
  580 G$=INKEY$(1):UNTIL G$="Q" OR count>maxcount
  590 *FX4,0
  600 ENDPROC
  610
  620
  630 DEF PROCreplay
  640 REPEAT
  650 PRINTTAB(11,23)"Would you like a replay of movements ? (Y/N) R to repeat"
  660 G$=GET$
  670 UNTIL G$="Y" OR G$="N" OR G$="R"
  680 IF G$="N" THEN ENDPROC
  690 IF G$<>"R" THEN INPUTTAB(11,22)"What delay period between response  (1 to
255) then (return)";dly%:IF dly%>255 OR dly%<1 GOTO 690
  700 ?&81=dly%
  710 FOR PosNo=0 TO lastmaxc:PRINTTAB(19,2)"No. of sequential position    =
";PosNo;"  ";:FOR servo%=0 TO Noservoes%:newpos%?servo%=R%(PosNo,servo%):NEXT:PRO
Cmoveservo:NEXT
  720 GOTO640
  730
  740
  750 DEF PROCmoveservo
  760 CALLmoveservo  :REM the machine code version
  820 ENDPROC
  830
  831
  840 DEF PROCsave:CLS
  850 PRINT TAB(15,10)"Would you like to save the sequence on file ?  Y/N ":IF G
ET$<>"Y" THEN ENDPROC
  860 INPUT TAB(15,12)"Please name the File to save : ";file$ ':file$="D."+file
$
  870 PROCendtimer
  880 X=OPENOUT file$:PRINT#X,lastmaxc:FOR A=0 TO lastmaxc:FOR B=0 TO Noservoes%
:PRINT#X ,R%(A,B):NEXT:NEXT
  890 CLOSE#X
  900 PROCstarttimer
  910 ENDPROC
  920
  930
  940 DEF PROCloadfile:CLS
  950 INPUT TAB(15,10)"Please name the File to load : ";file$:file$="D."+file$
  960 PROCendtimer
  970 X=OPENIN file$ :INPUT#X,count:lastmaxc=count:FOR A=0 TO count:FOR B=0 TO N
oservoes%:INPUT#X ,R%(A,B):NEXT:NEXT
  980 oldc=0
  990 CLOSE#X
 1000 PROCstarttimer
 1010 ENDPROC
 1020
 1030
 1040 DEF PROCsetup
 1050 count=0:Noservoes%=3:REM NO OF MOTORS-1
 1060 lastmaxc=0 :oldc=0 :dly%=10
 1070 maxcount=100
 1080 DIM R%(maxcount,Noservoes%),newpos% 8 ,timer% 12 ,read% 12
 1090 PROCassembletime
 1100 ENDPROC
 1110
 1120
 1130 DEF PROCassembletime
 1140 REM ****************************************
 1150 REM       Set up the timer etc
 1160 REM ****************************************
 1170 osbyte=&FFF4
 1180 A%=&97 :X%=&62 :Y%=&FF
 1190 CALL osbyte:REM set up port B for output
 1200 DIM p%(8)
 1210 DIM timer% 12 ,read% 12
 1220 xtimer=timer% MOD 256
 1230 ytimer=timer% DIV 256
 1240 xread=read% MOD 256
 1250 yread=read% DIV 256
 1260 PROCinitial
 1270 FOR I%=angle TO angle+8:angle?I%=128:NEXT
 1280 t=.02  :REM sec between pulses
 1290 time%=&FFFFFFFF -(t*100) +1
 1300 timer%?4=&FF    :REM load highest byte
 1310 !timer%=time%   :REM set up timer, enable events
 1320 PROCstarttimer
 1330 ENDPROC
 1340
 1350 DEF PROCstarttimer
 1360 *FX14,5
 1370 A%=4 :X%=xtimer :Y%=ytimer :CALL &FFF1
 1380 ENDPROC
 1390
 1400 DEF PROCendtimer
 1410 *FX13,5
 1420 ENDPROC
 1430
 1440
 1450 DEF PROCinitial
 1460  REM ****************************************
 1470  REM     Assemble the machine code
 1480  REM ****************************************
 1490 DIM space% 500
 1500 FOR C=0 TO 3 STEP 3
 1510 zeropage=&70    :REM free for users
 1520 portb=&FE60 :osword=&FFF1
 1530 P%=space%
 1540 angle=P% :P%=P%+8  :REM potentially 8 motors
 1550 table=P% :P%=P%+256  :REM 256 possible pulse lengths
 1560 FORI%=table TO table+&100:?I%=&FF:NEXT
 1570 lowtable%=table MOD 256
 1580 hightable%=table DIV 256
 1590 ?zeropage=lowtable% :zeropage?1=hightable%
 1600 [OPT C
 1610 .eventhandler
 1620 PHP:PHA:TYA:PHA:TXA:PHA
 1630 LDA #&04
 1640 LDX #xtimer
 1650 LDY #ytimer
 1660 JSR osword
 1670 LDY #&0
 1680  \Start pulse, for some motors it may be possible to start
 1690  \before filling table and so reduce the wait loop below
 1700 LDA #&FF :STA portb
 1710  \fill table with exceptions
 1720 LDX #&7 :LDA #&FF :CLC \set up bit pattern
 1730 .exceptions
 1740       ROR A :PHA  \bit pattern
 1750       LDY angle,X \get offset corresponding to angle of motor X
 1760       AND (zeropage),Y \keep existing bit pattern
 1770       STA (zeropage),Y \but modified for motor X
 1780       PLA :DEX
 1790       BPL exceptions
 1800 \table is now loaded, fill in some time
 1810 LDY #&60
 1820 .wait DEY
 1830       BNE wait
 1840 LDA #&FF  \all pulses on
 1850 LDY #&0
 1860 .loop AND (zeropage),Y  \but mask off with each table
 1870       STA portb  \element in turn
 1880       INY
 1890       BNE loop
 1900 LDX #&7 :LDA #&FF
 1910 .clear
 1920       LDY angle,X  \clear all the exceptions again
 1930       STA (zeropage),Y
 1940 DEX
 1950 BPL clear
 1960  \all pulses should now be finished
 1970 PLA:TAX:PLA:TAY:PLA:PLP
 1980 RTS
 1990
 2000
 2010 .moveservo \ .......  mch.code smooth move  .......
 2020 .Loop2
 2030 LDX #0
 2040 LDY #0
 2050 .Loop1
   LDA newpos%,X  \  new position for servo x
   CMP angle,X    \  is old position the same?
   BEQ moved      \ if = then do nothing
     BCS add      \ if > then add
        DEC angle,X   \ else take one away
        JMP incrementx
     .add
        INC angle,X
        JMP incrementx
     .moved
        INY
   .incrementx
   INX :CPX #4
   BNE Loop1      \ have i checked all 4 servos?
 2200 JSR waitmotors
 2210 CPY #4
 2220 BNE Loop2        \ continue if not all finished
 2230 RTS              \ got this far then all servos are in correct pos
 2240
 2250
 2260 .waitmotors
 2270 TXA:PHA:TYA:PHA
 2280 LDY #&FF :LDX &81
 2290 .waiting
   DEY :NOP :NOP
   BNE waiting   \ 8 clock cycles loop, ie 1024 cycles all in
 2320 LDY #&FF
 2330 DEX
 2340 BNE waiting
 2350 PLA:TAY:PLA:TAX
 2360 RTS
 2370
 2380 ]
 2390 NEXT
 2400 !&220=eventhandler OR (!&220 AND &FFFF0000)
 2410 ENDPROC
```

Schatzsuche

Die Aussicht auf wertvolle Gewinne, die für die Lösung kniffliger Probleme in Abenteuerspielen ausgesetzt sind, hat schon manchen in geheimnisvolle Welten gelockt. „Quo Vadis" ist mit 1024 Bildschirmszenen ausgestattet, und als Preis winkt dem Spieler ein Zepter aus Gold und Silber.

„Quo Vadis" von Softek ist ein Abenteuerspiel mit einer Fülle von grafischen Kulissen, bei dem es wirklich etwas zu gewinnen gibt. In einem gefährlichen Höhlenlabyrinth müssen Sie nach Hinweisen suchen und auch noch drei Rätsel lösen, um herauszufinden, wo das Zepter aus Gold und Silber vergraben ist.

Die Abenteuer-Arcadespiele verbinden Elemente des klassischen Arcarde- und des Abenteuerspiels miteinander und sind dank einer neuen Methode noch attraktiver geworden, die durch das englische Buch „Masquerade" von Kit Williams bekannt geworden ist. Die „Masquerade" (unter gleichem Titel in dt. Übersetzung) enthält in Wort und Bild verschlüsselte Fingerzeige, wie der Leser einen echten Goldhasen finden kann, der nächtlich an geheimnisvoller Stätte in der Gegend von Bedford vergraben wurde. Als erster Softwarehersteller griff die Firma Automata mit dem Spiel „Pimania" auf diese Idee zurück und setzte als „Finderlohn" eine goldene Sonnenuhr aus. Danach kam der „Hare Raiser", ähnlich wie das Buch Masquerade aufgebaut und sogar mit demselben Gewinn. Weitere Beispiele sind unter anderem das Abenteuerspiel „Eureka" mit einer Prämie von umgerechnet über 100 000 Mark für die erste richtige Lösung oder ein Spiel von Firebird mit einem Porsche als Preis.

Das Abenteuerspiel „Quo Vadis" der Firma Softek liegt auf der gleichen Ebene: Wer als erster die Geheimnisse lüftet und das Spiel erfolgreich zu Ende führt, bekommt ein Zepter aus Gold und Silber. Das Spiel überrascht vor allem durch seine ungewöhnliche Kulissenvielfalt – es umfaßt 1024 Bildschirmszenen.

Bei „Quo Vadis" schlüpft der Spieler in die Rolle eines Ritters, der ein Zepter sucht, das in einem unterirdischen Labyrinth verborgen ist. Das Gangsystem enthält zudem 118 Höhlen.

Der Weg durch die Kavernen erfordert kühne Sprünge von einem Felsensims zum andern, teils auch waghalsige Seilkletterei. Natürlich lauern überall tödliche Gefahren wie Lavalöcher im Höhlenboden und ähnlich Einladendes. In dem Labyrinth hausen 38 verschiedene Monster, die den Ritter beim Betreten der Höhlen attackieren, und ihm hilft dann nur die Rundumverteidigung durch gezielte Joystick-„Schüsse".

Wird der Ritter von den Monstern erwischt, verliert er immer mehr von seinen anfangs 100 Energiepunkten; wenn alle verbraucht sind, hat er verloren. Er kann seinen Energievorrat aber aus Schatztruhen auffrischen, die in manchen Höhlen zu finden sind. Außerdem sind Rätselsprüche an der Wand zu entdecken, die Hinweise auf das Versteck des Zepters enthalten.

Angesichts der Weitläufigkeit der Unterwelt waren anderwärts Zugeständnisse unvermeidlich, und so ist die Grafik- und Tonqualität zugegebenermaßen nicht umwerfend. Die Bilder gewinnen aber an Atmosphäre, je länger man sich mit dem Spiel beschäftigt, und bei flackernden Fackeln, tropfenden Kerzen, düsteren Treppen und glitschigen Wänden stellt sich bald eine fabelhafte Gruselstimmung ein.

Daß dieses umfangreiche Spiel überhaupt in den Speicher des Commodore 64 hineinpaßt, zeugt von einer äußerst platzsparenden Programmiertechnik unter Ausnutzung aller Möglichkeiten zur Datenkompression.

Einige Monate nach Veröffentlichung des Spiels brachte Softek den „Quo Vadis Generator" auf Diskette heraus – damit sind, falls Ihnen die 1024 noch nicht reichen, nach Aussage des Herstellers dann eine Million verschiedener Bildschirmszenen darstellbar!

Quo Vadis: Commodore 64
Produzent: Softek International, 12/13 Henrietta St., Covent Garden, London WC2E 8LH
Datenträger: Cassette oder Diskette
Joystick: erforderlich

Allein auf hoher See

Während unser Schiff Kurs auf die Neue Welt nimmt, beeinflussen diverse Ereignisse die Fahrtdauer und die Stärke der Mannschaft. Wir setzen die Programmierung dieser Ereignisse mit einem Modul fort, das sich mit dem Fangen frischer Fische und dem Auffrischen der Trinkwasservorräte befaßt.

Der Programmcode für die Zwischenfälle besteht aus kleineren Programmabschnitten, die per Zufall durch die Unterroutine ab Zeile 5500 gewählt werden. Im vorangegangenen Artikel haben wir die ersten fünf Ereignisse behandelt, so daß wir nach Hinzufügen der neuen insgesamt dreizehn Zwischenfälle haben (somit muß auch RM in Zeile 46 auf 13 gesetzt werden). Die Liste der Zeilennummern hinter ON X GOTO in Zeile 5525 muß den neuen Routinen ebenfalls angepaßt werden. Das Hauptprogramm wählt aus dieser erweiterten Liste „geringfügigerer Ereignisse" jeweils einen Zwischenfall pro Woche per Zufall aus.

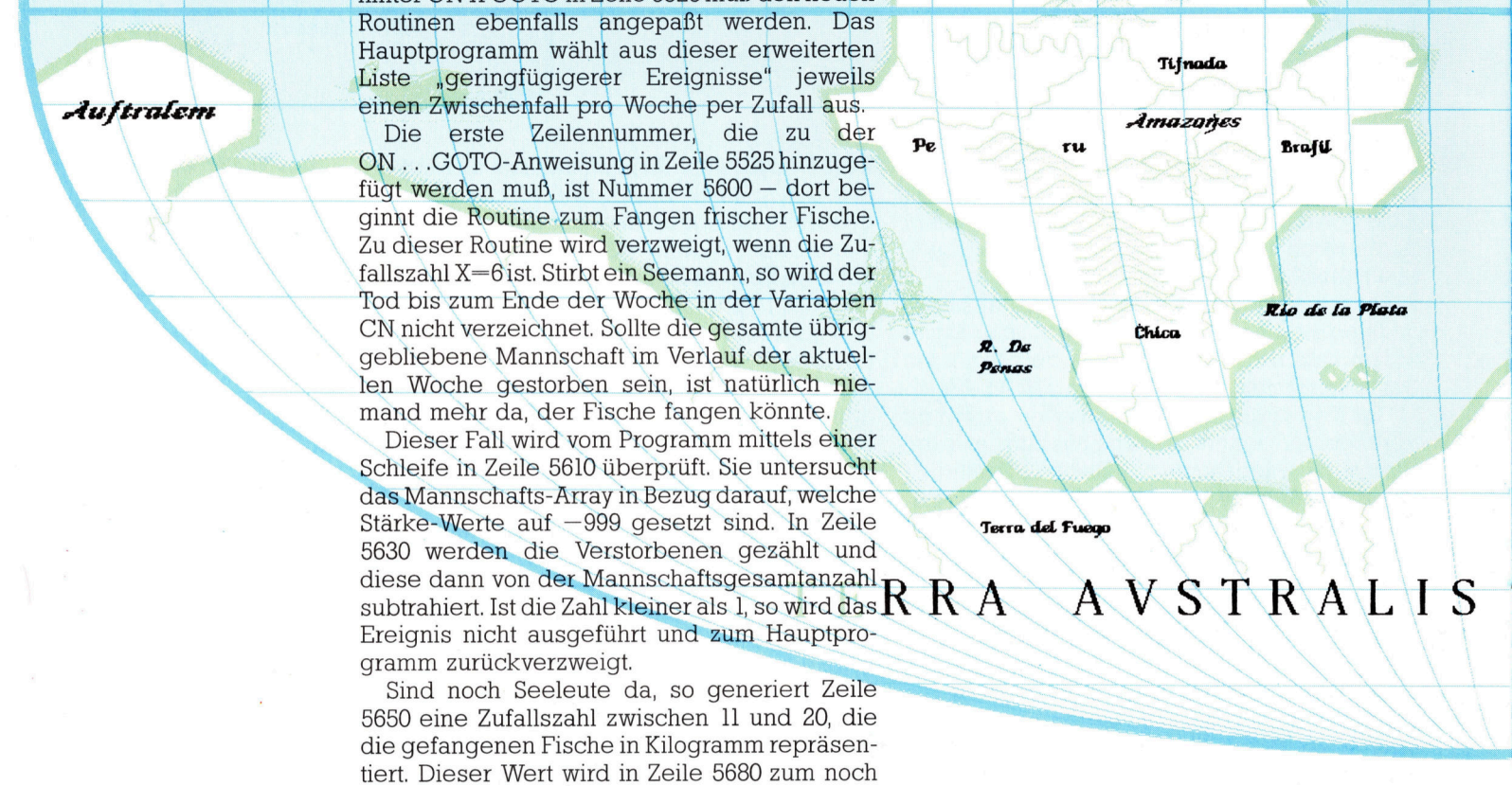
Die erste Zeilennummer, die zu der ON . . .GOTO-Anweisung in Zeile 5525 hinzugefügt werden muß, ist Nummer 5600 – dort beginnt die Routine zum Fangen frischer Fische. Zu dieser Routine wird verzweigt, wenn die Zufallszahl X=6 ist. Stirbt ein Seemann, so wird der Tod bis zum Ende der Woche in der Variablen CN nicht verzeichnet. Sollte die gesamte übriggebliebene Mannschaft im Verlauf der aktuellen Woche gestorben sein, ist natürlich niemand mehr da, der Fische fangen könnte.

Dieser Fall wird vom Programm mittels einer Schleife in Zeile 5610 überprüft. Sie untersucht das Mannschafts-Array in Bezug darauf, welche Stärke-Werte auf −999 gesetzt sind. In Zeile 5630 werden die Verstorbenen gezählt und diese dann von der Mannschaftsgesamtanzahl subtrahiert. Ist die Zahl kleiner als 1, so wird das Ereignis nicht ausgeführt und zum Hauptprogramm zurückverzweigt.

Sind noch Seeleute da, so generiert Zeile 5650 eine Zufallszahl zwischen 11 und 20, die die gefangenen Fische in Kilogramm repräsentiert. Dieser Wert wird in Zeile 5680 zum noch vorhandenen Fleisch addiert, und das Programm informiert Sie, wie lange der Vorrat noch ausreicht. Abschließend wird in Zeile 5685 die Gesamtfleischmenge durch die Zahl der Mannschaftsmitglieder und deren Wochenbedarf dividiert und eine neue Schätzung ausgegeben, wie viele Wochen der Vorrat noch reicht.

Das nächste Ereignis ist ein Gewitter. Die entsprechende Routine beginnt bei Zeile 5700. Durch den Regen ist die Mannschaft in der Lage, Regenwasser aufzufangen und so die Wasservorräte aufzufrischen. Das Programm generiert in Zeile 5735 eine Zufallszahl zwischen 11 und 20, die die Menge aufgefangenen Wassers in Barrel angibt. Dieser Wert wird in Zeile 5750 zum noch vorhandenen Wasservorrat addiert. Abschließend wird in Zeile 5755 berechnet und ausgegeben, wie lange der Wasservorrat noch reicht.

Das achte mögliche Ereignis sind günstige Winde, die die Geschwindigkeit des Schiffes erhöhen und somit die Fahrzeit um eine halbe Woche verkürzen (Zeile 5835). Die „Good Weather"-Routine (Ereignis Nummer 9) beginnt in Zeile 5850. Das gute Wetter ermöglicht der Mannschaft, sich zu regenerieren, was dadurch simuliert wird, daß mittels einer Schleife in Zeile 5882 die Stärkewerte in TS(,) erhöht werden. Damit ein toter Seemann durch eine solche Erhöhung nicht wieder zum Leben erweckt wird, überprüft Zeile 5884, ob irgendwelche Werte auf 0 oder −999 gesetzt sind. In Zeile 5886 wird dann ein Zufallswert zwischen 5 und 15 Einheiten zum Stärkewert jedes lebenden Seemannes addiert.



Ereignis Nummer 10 beginnt bei Zeile 5900 – die Unterroutine zum „Verlieren von Medizin". Durch das schlechte Wetter sind die Hälfte der Medizinflaschen zerbrochen, und das kann bei einem Krankheitsfall erhebliche Auswirkungen haben. Zuerst überprüft das Programm, ob überhaupt Medizinflaschen an Bord sind, indem das erste Element des Vorrats-Arrays OA() untersucht wird. Ist der Wert auf 0 oder −999 gesetzt, ist keine Medizin vorhanden und es erfolgt Rücksprung zum Hauptprogramm. Ansonsten wird durch Zeile 5925 der Vorrat halbiert und das Resultat ausgegeben.

Das elfte Ereignis simuliert, daß durch hohen Seegang einige Waffen naß und somit unbrauchbar geworden sind. Dies ist sehr ungünstig, da Waffen zur Verteidigung, Nahrungsgewinnung und zum Handel gebraucht werden. Die Unterroutine überprüft in Zeile 5955, ob Waffen an Bord sind, halbiert den Betrag in Zeile 5975 und gibt den Integerwert des Ergebnisses aus.

Das zwölfte Ereignis beginnt bei Zeile 6000 und besteht darin, daß Mäuse die Hälfte der Kleidung angefressen haben. Kleidung ist das vierte Element im Vorrats-Array OA(), und das Programm überprüft in Zeile 6005, ob Kleidung an Bord ist. Wurde Kleidung eingekauft, beginnen die Mäuse, sie zu fressen (Zeile 6025), und der Spieler wird nach der „Mahlzeit" informiert, wie viele Ballen noch übrig geblieben sind.

Das letzte Ereignis mit der Nummer 13 ist das Sichten eines Albatros. Dieses Ereignis hat eine Besonderheit, da es zweierlei Dinge auslösen kann: Wird es bei ON X GOTO gewählt, passiert noch nichts Ungewöhnliches. Wird der Albatros jedoch gesichtet, wenn die Mannschaft auf halbe Fleischrationen gesetzt ist, erhält sie die Möglichkeit, den Vogel abzuschießen. Die verschiedenen Ereignisse werden im Hauptprogramm mittels GOSUB in Zeile 860 aufgerufen.

Ein Himmelsbote

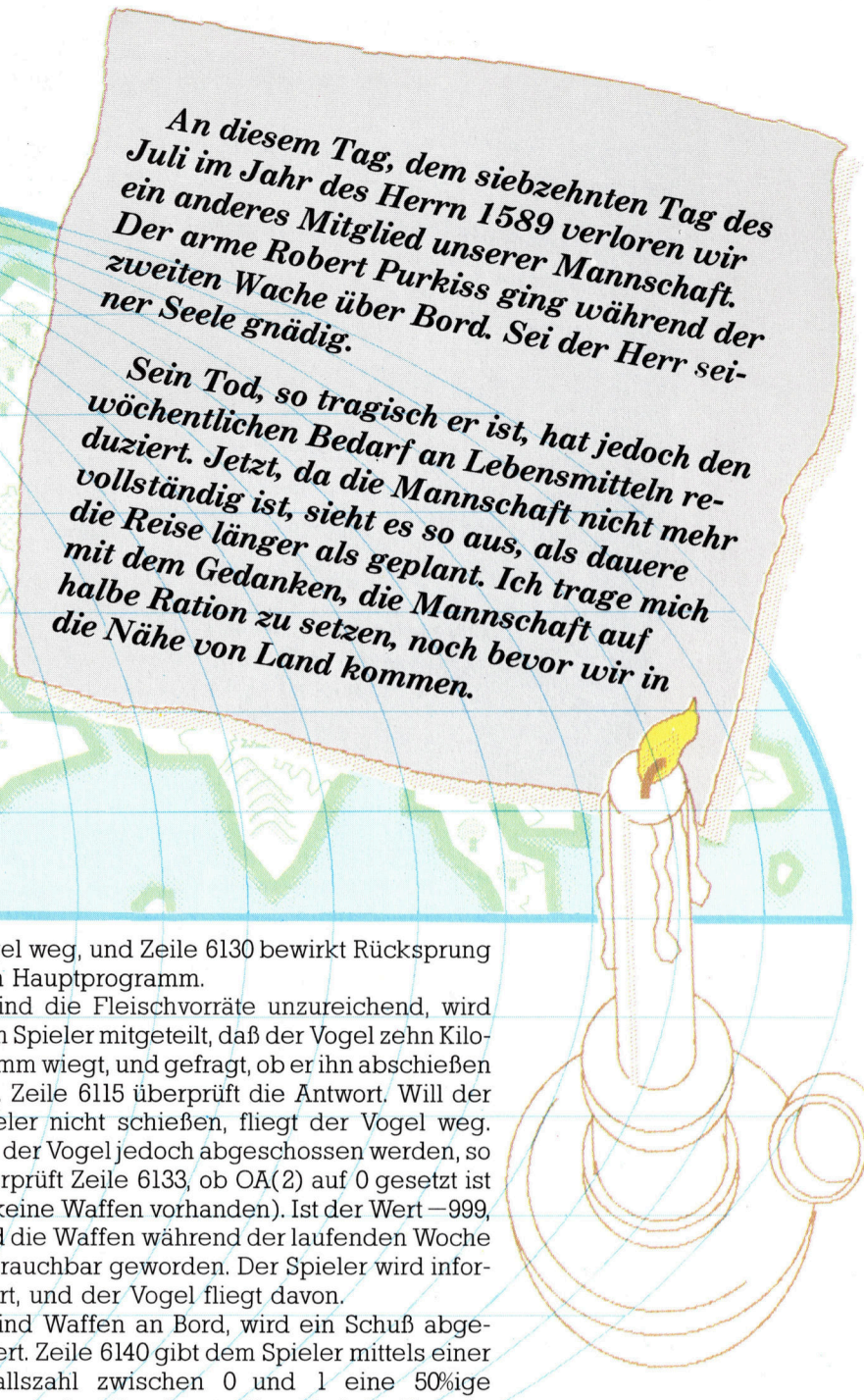
Die schwerwiegenderen Zwischenfälle folgen ab Zeile 870. Zeile 875 überprüft, ob die Mannschaft auf halbe Fleischrationen gesetzt ist (HR(3)=0.5). Ist dies der Fall, gibt ein Zufallsfaktor – AND RND(1)<.5 – eine 50%ige Chance, den Albatros zu sichten. Wird der Vogel entdeckt, verzweigt das Programm zur „Albatros"-Unterroutine bei Zeile 6050.

Das Sichten eines Albatros bringt Glück. Daher wird eine Variable A$ in Zeile 6055 auf „Y" gesetzt. Sie wird zu einem späteren Zeitpunkt vom Programm verwendet, um das Risiko einer Meuterei zu reduzieren. Danach überprüft das Programm in Zeile 6075, ob die Mannschaft nur noch geringe Fleischvorräte hat. Hierzu wird bestimmt, ob der vorhandene Fleischvorrat größer oder gleich dem wöchentlichen Bedarf der Mannschaft bis zum geschätzten Ende der Reise ist. Ist der Vorrat ausreichend, fliegt der Vogel weg, und Zeile 6130 bewirkt Rücksprung zum Hauptprogramm.

Sind die Fleischvorräte unzureichend, wird dem Spieler mitgeteilt, daß der Vogel zehn Kilogramm wiegt, und gefragt, ob er ihn abschießen will. Zeile 6115 überprüft die Antwort. Will der Spieler nicht schießen, fliegt der Vogel weg. Soll der Vogel jedoch abgeschossen werden, so überprüft Zeile 6133, ob OA(2) auf 0 gesetzt ist (= keine Waffen vorhanden). Ist der Wert −999, sind die Waffen während der laufenden Woche unbrauchbar geworden. Der Spieler wird informiert, und der Vogel fliegt davon.

Sind Waffen an Bord, wird ein Schuß abgefeuert. Zeile 6140 gibt dem Spieler mittels einer Zufallszahl zwischen 0 und 1 eine 50%ige Chance, den Vogel zu treffen. Sind die Fleischvorräte während der laufenden Woche ausgegangen, wird der entsprechende Wert in Zeile 6160 von −999 auf 0 gesetzt, damit der zusätzliche Vorrat verzeichnet werden kann. Die Menge an Fleisch in PA(3) wird um 10 Kilo erhöht, und der Spieler erfährt die neue Menge.

Eine Warnung wird dem Spieler in unserem Spiel gegeben. In Zeile 6162 wird B$ auf „Y" gesetzt. Dies ist der Pech-Faktor, der erheblich schwerer wiegt als der Glück-Faktor in Zeile 6055. Die genauen Auswirkungen besprechen wir später. Der Spieler wird über den Grund nicht informiert, doch werden Faktoren geändert, die die Fahrt behindern sollen.



Modul sieben: Mehr Random-Ereignisse

Für die Haupt-Reise-Schleife

```
875 IF HR(3)=.5ANDRND(1)<.5THENPRINTCHR$
(147):GOSUB6050
```

Unterroutine für mehr Random-Ereignisse

```
5600 REM EVENT 6 - CATCH SOME FISH
5605 X=0
5610 FORT=1TO16
5615 IFTS(T,2)=-999THENX=X+1
5620 REM COUNT DEAD THIS WEEK
5625 NEXT
5630 IFCN-X<1THENRETURN
5635 REM NO ACTION IF ALL CREW DEAD
5640 PRINT
5645 S$="  DURING THE WEEK*":GOSUB9100
5646 PRINT:GOSUB9200
5650 S$="ONE OF THE CREW CAUGHT*":GOSUB9100
5655 X=INT(RND(1)*10)+11
5660 REM 10 TO 20 KILOS
5662 PRINTX;"KILOS OF FISH"
5665 PRINT:GOSUB9200
5670 S$="YOUR MEAT SUPPLY IS NOW*":GOSUB9100
5675 S$="ENOUGH FOR APPROXIMATELY*":GOSUB9100
5678 IFPA(3)=-999THENPA(3)=0
5680 PA(3)=PA(3)+X
5685 PRINTINT(PA(3)/(CN*PN(3)));"WEEKS"
5690 GOTO5530
5700 REM EVENT 7 - CATCH SOME WATER
5705 PRINT
5710 S$="  DURING THE WEEK*":GOSUB9100
5715 PRINT:GOSUB9200
5720 S$="A RAINSTORM REFILLED YOUR*":GOSUB9100
5725 S$="WATER BARRELS*":GOSUB9100
5730 PRINT:GOSUB9200
5735 X=INT(RND(1)*10)+11
5736 REM 10 TO 20 BARRELS
5740 S$="YOUR WATER SUPPLY IS NOW*":GOSUB9100
5745 S$="ENOUGH FOR APPROXIMATELY*":GOSUB9100
5748 IFPA(4)=-999THENPA(4)=0
5750 PA(4)=PA(4)+X
5755 PRINTINT(PA(4)/(CN*PN(4)));"WEEKS"
5760 GOTO5530
5800 REM EVENT 8 - GOOD WINDS
5805 PRINT
5810 S$="STRONG FOLLOWING WINDS ALL WEEK*":GOSUB9100
5815 PRINT:GOSUB9200
5820 S$="YOU HAVE MADE GOOD SPEED*":GOSUB9100
5825 S$="AND YOUR JOURNEY TIME IS*":GOSUB9100
5830 S$="REDUCED BY 1/2 A WEEK*":GOSUB9100
5835 EW=EW-.5
5839 GOTO5530
5850 REM EVENT 9 - GOOD WEATHER
5855 PRINT
5860 S$="GOOD WEATHER ALL WEEK*":GOSUB9100
5865 PRINT:GOSUB9200
5870 S$="THE CREW IS FEELING HAPPIER*":GOSUB9100
5875 GOSUB9200
5880 S$="                AND HEALTHIER!*":GOSUB9100
5882 FORT=1TO16
5884 IFTS(T,2)=0 OR TS(T,2)=-999THEN5888
5886 TS(T,2)=TS(T,2)+INT(RND(1)*11)+5
5888 NEXT
5889 GOTO5530
5900 REM EVENT 10 - LOSE MEDICINE
5905 IF OA(1)=0 OR OA(1)=-999THENRETURN
5910 PRINT
5915 S$="YOU DISCOVER THAT HALF YOUR*":GOSUB9100
5920 S$="MEDICINE BOTTLES HAVE BROKEN*":GOSUB9100
5925 OA(1)=INT(OA(1)/2)
5930 PRINT:GOSUB9200
5935 S$="YOU NOW HAVE ONLY*":GOSUB9100
5940 PRINTOA(1);"BOTTLES LEFT"
5945 GOTO5530
5950 REM EVENT 11 - RUSTY GUNS
5955 IF OA(2)=0 OR OA(2)=-999THENRETURN
5960 PRINT
5965 S$="YOU DISCOVER THAT HALF YOUR*":GOSUB9100
5970 S$="GUNS HAVE GONE RUSTY*":GOSUB9100
5972 S$="AND NO LONGER WORK*":GOSUB9100
5975 OA(2)=INT(OA(2)/2)
5980 PRINT:GOSUB9200
5985 S$="YOU NOW HAVE ONLY*":GOSUB9100
5990 PRINTOA(2);"GUNS LEFT"
5995 GOTO5530
6000 REM EVENT 12 - LOSE CLOTH
6005 IF OA(4)=0 OR OA(4)=-999THENRETURN
6010 PRINT
6015 S$="YOU DISCOVER THAT HALF YOUR*":GOSUB9100
6020 S$="BALES OF CLOTH HAVE BEEN*":GOSUB9100
6022 S$="GNAWED BY MICE*":GOSUB9100
6024 S$="AND ARE NOW WORTHLESS*":GOSUB9100
6025 OA(4)=INT(OA(4)/2)
6030 PRINT:GOSUB9200
6035 S$="YOU NOW HAVE ONLY*":GOSUB9100
6040 PRINTOA(4);"BALES LEFT"
6045 GOTO5530
6050 REM EVENT 13 - ALBATROSS
6055 PRINT:A$="Y"
6060 S$="AN ALBATROSS FLIES OVER THE SHIP*":GOSUB9100
6062 GOSUB9200
6065 S$="THIS IS A GOOD OMEN*":GOSUB9100
6068 S$="AND THE CREW IS HAPPY*":GOSUB9100
6070 PRINT:GOSUB9200
6075 IFPA(3)<(CN*PN(3)*(JL-WK+1))THEN6090
6080 REM NOT SHORT OF MEAT
6085 GOTO6122
6090 S$="YOU'RE RUNNING SHORT OF MEAT*":GOSUB9100
6095 S$="AND THE BIRD WEIGHS 10 KILOS!*":GOSUB9100
6100 PRINT:GOSUB9200
6105 S$="WOULD YOU LIKE TO CATCH IT?*":GOSUB9100
6110 INPUTI$
6112 PRINT:GOSUB9200
6115 IFLEFT$(I$,1)="Y"THEN6133
6120 S$="PROBABLY JUST AS WELL!*":GOSUB9100
6122 PRINT:GOSUB9200
6125 S$="THE ALBATROSS FLIES AWAY....*":GOSUB9100
6130 GOTO5530
6133 IFOA(2)=00ROA(2)=-999THEN6180
6135 S$="A SHOT IS FIRED.............*":GOSUB9100
6138 GOSUB9200:GOSUB9200
6140 IFRND(1)<.5THEN6150
6145 S$=".........BUT MISSES!*":GOSUB9100
6148 GOTO6122
6150 S$="AND THE BIRD FALLS TO THE DECK!*":GOSUB9100
6155 PRINT:GOSUB9200
6160 IFPA(3)=-999THENPA(3)=0
6162 PA(3)=PA(3)+10:B$="Y"
6165 S$="YOU NOW HAVE 10 MORE KILOS*":GOSUB9100
6167 S$="OF MEAT.....*":GOSUB9100
6170 S$="BUT YOU MAY NOT HAVE MUCH*":GOSUB9100
6172 S$="GOOD LUCK FROM NOW ON!!*":GOSUB9100
6174 GOTO5530
6180 S$="YOU CAN'T - YOU HAVE NO GUNS*":GOSUB9100
6190 GOTO6122
```

BASIC-Dialekte

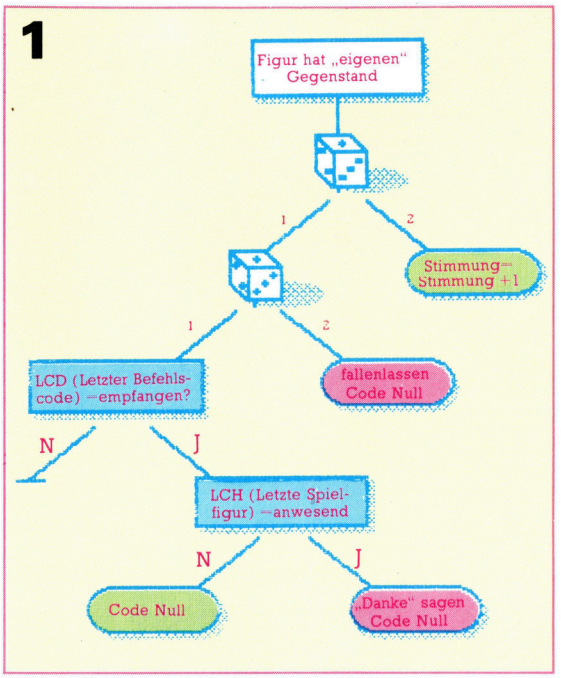
Spectrum:
Führen Sie die folgenden Änderungen aus:

```
875  IF HR(3)=.5ANDRND(1)<.5THENCLS
     :GO SUB 6050
5527 IF X=6 THEN GO TO 5600
5528 IF X=7 THEN GO TO 5700
5529 IF X=8 THEN GO TO 5800
5530 IF X=9 THEN GO TO 5850
5531 IF X=10 THEN GO TO 5900
5532 IF X=11 THEN GO TO 5950
5533 IF X=12 THEN GO TO 6000
5534 IF X=13 THEN GO TO 6050
5535 PRINT:S$=K$:GO SUB 9100
5536 LET I$=INKEY$:IF INKEY$="" THEN GO
     TO 5536
6115 IF I$(1 TO 1)="Y" THEN GO TO 6133
```

Acorn B
Ändern Sie das Programm wie folgt:

```
875  IF HR(3)=.5AND RND(1)<.5
     THEN CLS: GOSUB6050
```

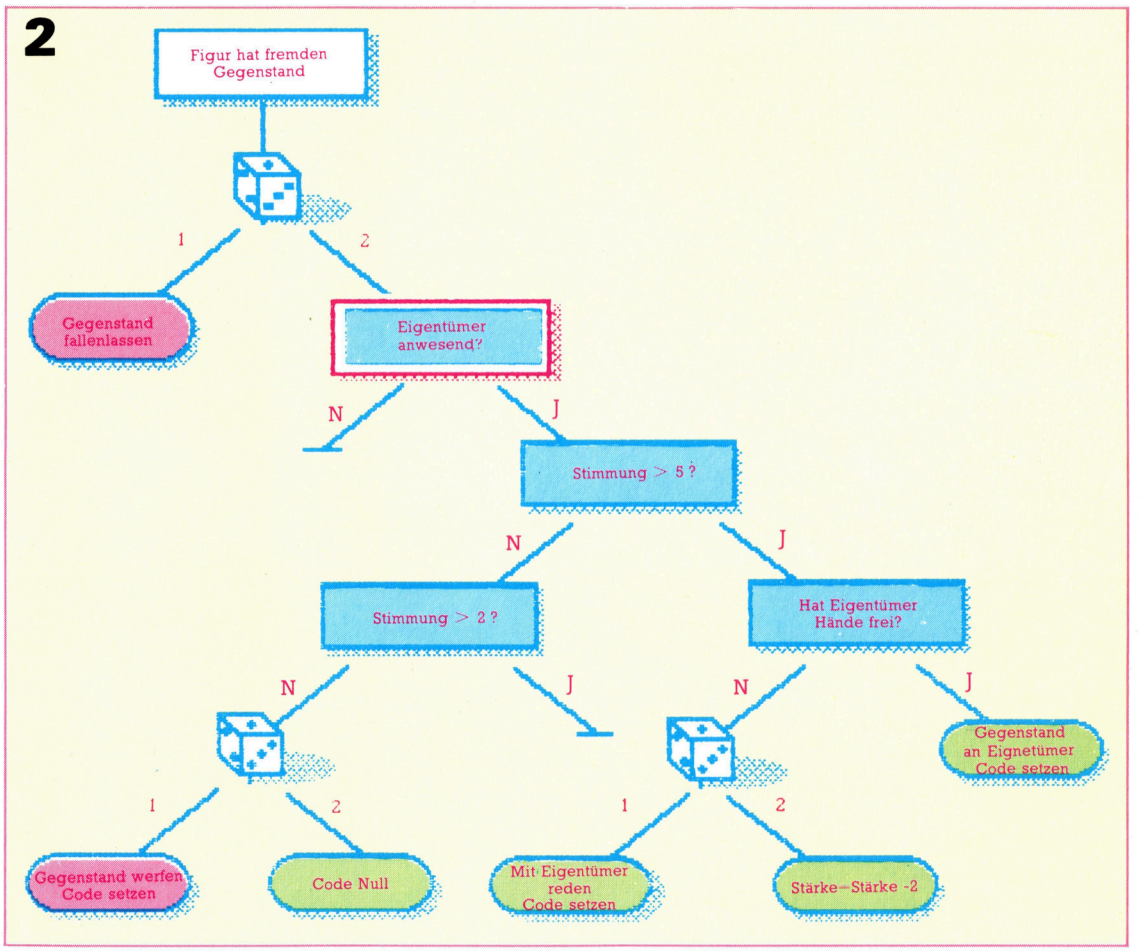
Was tun mit Essen und Trinken?



Obwohl die Anzahl der verfügbaren Gegenstände bei „Dog and Bucket" nicht besonders groß ist, spielt der richtige Umgang mit ihnen doch eine große Rolle.

Wie unsere Spielfiguren mit Gegenständen umgehen, ist für den Unterhaltungswert des Programms von entscheidender Bedeutung – dabei ist eine große Zahl von Einzelfaktoren zu berücksichtigen. Die Gegenstände lassen sich in zwei Kategorien einteilen: Eßbar (Sandwich und Pastete) und trinkbar. Es gibt noch mehr Objekte, sie haben aber keine Funktion im Spiel und dienen eher der Dekoration.

Sehr wichtig ist, wem ein Gegenstand gehört: Bestimmte Getränke gehören zu bestimmten Spielfiguren – der „Besitz" der Spielfiguren muß also beachtet werden. Ebenso wichtig ist es, die Wünsche der Figuren zu kennen. In einer komplexen Baumstruktur lassen sich alle



diese Eigenheiten zusammenfassen. Zur Vereinfachung gehen wir dabei in vier Stufen vor.

Zuerst wollen wir betrachten, was eine Spielfigur mit einem ihr zugeteilten Objekt anstellen kann. Das erste Diagramm zeigt eine einfache Baumstruktur der Möglichkeiten. Wichtig sind dabei die Würfelsymbole, die Zufallsentscheidungen darstellen. Wohin an diesen Stellen verzweigt wird, hängt von der „erwürfelten" Zufallsvariablen ab.

Neben den Zufallsverzweigungen gibt es drei weitere Verzweigungs-Typen im Diagramm. Endpunkte, bei denen eine Meldung am Bildschirm erscheint und die Variablen aktualisiert werden, sind rot dargestellt. Auch die grünen Felder sind Endpunkte, allerdings werden hier nur die Variablen auf den neuesten Stand gebracht, ohne daß eine Bildschirmmeldung erfolgt. Diese Punkte werden separat behandelt. Schließlich sind auch noch blaue Felder vorhanden, die einfache Entscheidungen symbolisieren.

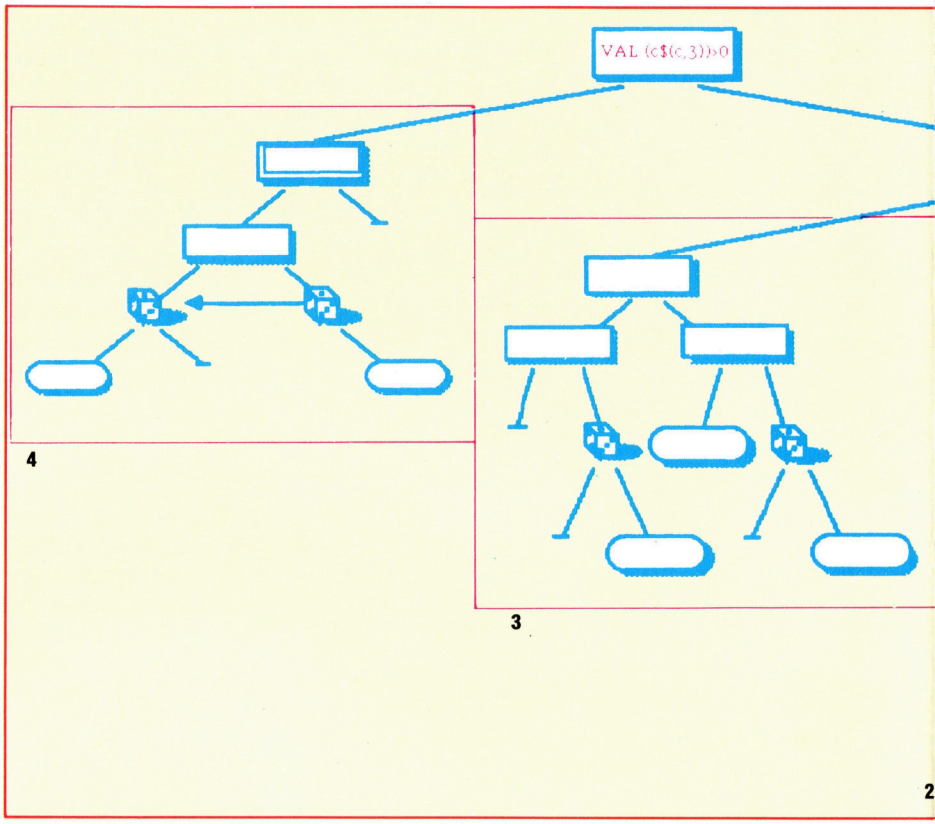
Aus dem Diagramm wird deutlich, daß eine Spielfigur mit einem „eigenen" Gegenstand entsprechend den vier Endpunkten vier Verhaltensmöglichkeiten hat. So könnte sich bei günstiger Zufallsentscheidung die Stimmung heben, weil die Figur sich an ihrem Lieblingsgetränk laben kann. Weniger wahrscheinlich ist, daß die Spielfigur den Gegenstand fallen läßt oder, wenn sie ihn von einem noch anwesenden Spender bekommen hat, daß sie sich bedankt. Es ist zu beachten, daß drei der Endpunkte die Werte LCH und LCD auf Null setzen, wodurch Interaktionen mit einer anderen Spielfigur beendet werden.

Diese Strukturen sind rein willkürlich – wir können also ganz unterschiedliche Bäume für den gleichen Zweck aufbauen. Also weiter zum nächsten Baum. Er ist für Situationen zuständig, in denen Spielfiguren fremde Gegenstände in Besitz haben.

Ist der Eigentümer da?

Die Verzweigungen sind hier farbig ähnlich unterschieden wie im vorigen Beispiel, bis auf das Entscheidungsfeld mit dem roten Rand. Es arbeitet ähnlich wie ein Endpunkt: Wird das Feld erreicht, verzweigt das Programm zu einer anderen Routine. Je nachdem, wie diese Situation ausgeht, kehrt das Programm manchmal auch wieder an den alten Punkt zurück.

Um herauszufinden, ob der Eigentümer des zu einer anderen Spielfigur gehörigen Objektes gerade anwesend ist, müssen die Umgebungsbedingungen aller Figuren geprüft werden (c$(character,2)). Der erwähnte Verzweigungspunkt führt dazu aus dem laufenden Programm heraus zu einem Unterprogramm. Ist die betreffende Person wirklich da, wird zum nächsten Entscheidungspunkt des Baumes zurückgesprungen (Stimmung>5?), ist sie aber nicht da, geht es an der vorherigen Stelle weiter.



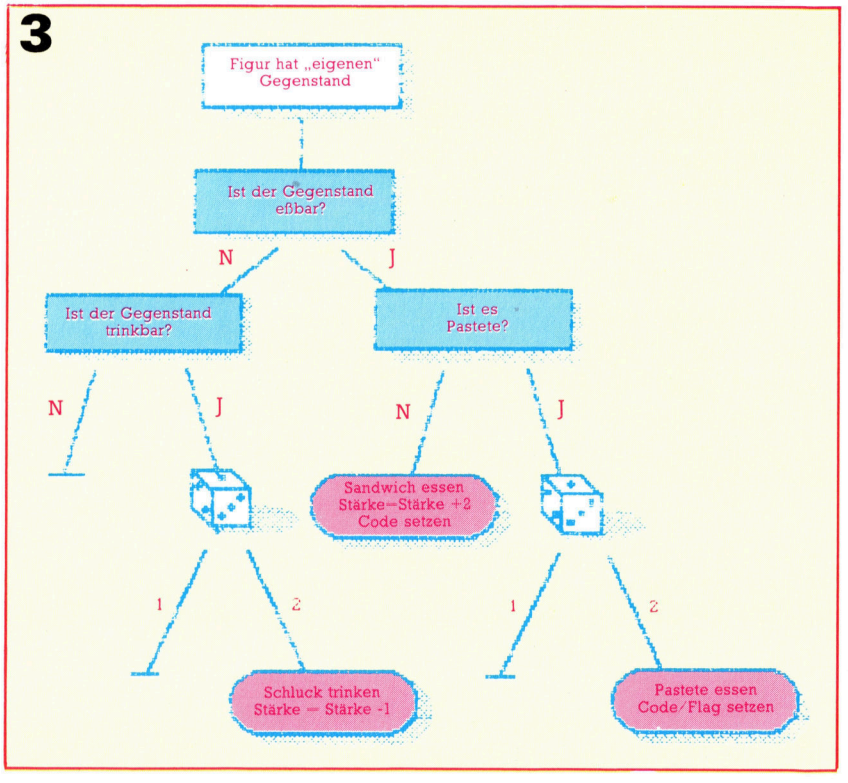
4

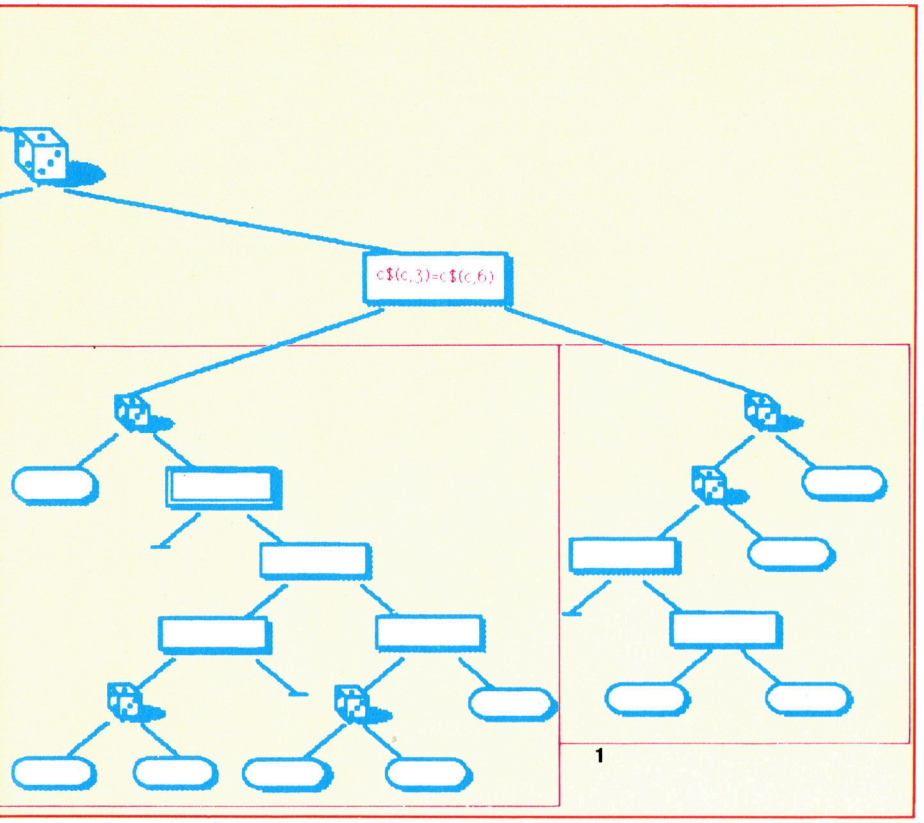
3

2

Die behandelten vier Baumstrukturen können nach der obigen Abbildung zu einem einzigen, größeren Baum zusammengefaßt werden. Das Programm prüft zuerst den Wert der Bedingung (VAL(c$(c,3))>0). Das Ergebnis ist WAHR, wenn die betreffende Spielfigur einen Gegenstand trägt, sonst als FALSCH. Bei FALSCH wird die Programmsteuerung vom Hauptbaum an den in Bild 4 dargestellten Baum abgegeben. Bei WAHR erfolgt eine Zufallsentscheidung, die entweder zum Baum von Bild 3 (Essen/Trinken) führt oder mit dem Ausdruck (c$(c,3)=c$(c,6)) auf „eigene" Gegenstände hin prüft. Dann geht die Steuerung auf eine der Strukturen über.

3

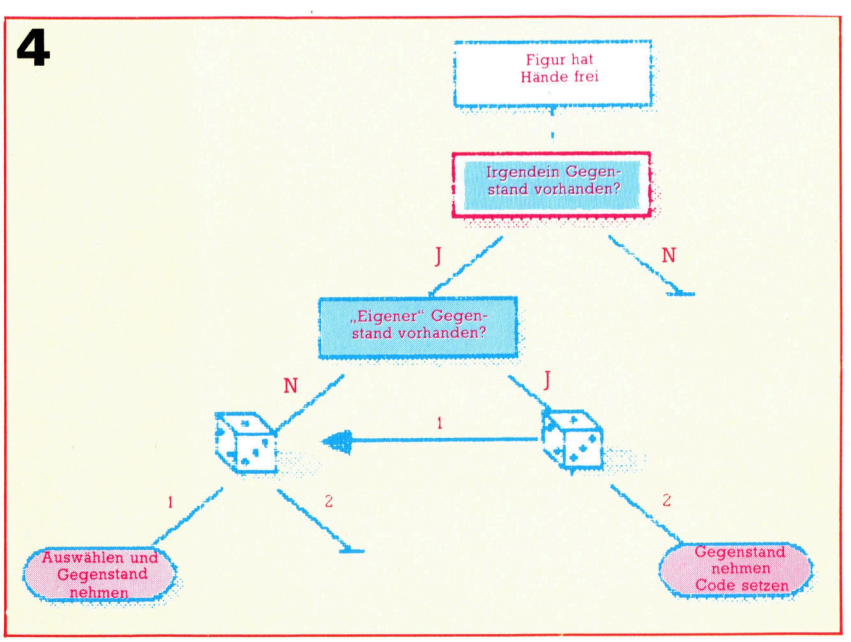




1

Meist läßt die Figur den Gegenstand fallen, wenn nicht, prüft das Programm, ob der Eigentümer anwesend ist. Ist er in der Nähe, entscheidet das Programm, ob er den Gegenstand überreicht bekommt – leere Hände vorausgesetzt – oder damit beworfen wird, falls die Figur in schlechter Stimmung ist. Hat der Eigentümer die Hände voll, gibt es eine Zufallsentscheidung, ob die Spielfigur sich womöglich für ihr Fehlverhalten entschuldigt.

Weiter ist zu beachten, daß eines der Endfelder die Stärke der Figur um zwei Punkte reduziert, falls sie etwa aus dem fremden Glas trinkt



– schließlich weiß jedes Kind, daß man nicht mehrere Getränke durcheinander zu sich nehmen sollte. Das Essen und Trinken ist bei der Entscheidungsfindung in den Bäumen bisher noch nicht berücksichtigt worden. Dafür gibt es wiederum eine eigene Struktur, die in Bild 3 gezeigt wird.

Die Farbkennungen im dritten Baum entsprechen den Farben der beiden anderen. Die Stärke der Figuren kann auch hier durch Trinken vermindert werden. Das Essen des Sandwichs steigert die Spielstärke um zwei Punkte und setzt den LCD entsprechend. Essen der Pastete erzeugt ein Flag, das für den weiteren Spielablauf benötigt wird.

Wenn wir schon einmal dabei sind, wollen wir auch schnell die Handlung des Spiels verraten. Die Katzenfutter-Büchsen in der Küche sind Ihnen sicher aufgefallen – Sie werden bereits vermutet haben, daß die berühmte Dog-and-Bucket-Pastete in Wirklichkeit aus billigem Katzenfutter gemacht wird.

Mysterium oder Mord

Das Vertilgen einer Pastete führt zum vorzeitigen Verschwinden einer oder mehrerer Spielfiguren, wobei in jedem Spiel andere Figuren betroffen sind. Wir brauchen daher ein Flag, das die möglichen Opfer einer Fleischvergiftung kennzeichnet. Ob der Tod einer Figur als Mysterium oder als Mord durch die anderen Kneipengänger aufgefaßt wird, hängt von den Faktoren ab, die wir später noch behandeln. Vorher soll aber die Erläuterung der Steuerstrukturen vervollständigt werden.

Hat eine Figur leere Hände, kommt der Baum im vierten Diagramm zum Einsatz. Er enthält ebenfalls einen Punkt, bei dem zu einem Unterprogramm verzweigt wird. Dieses prüft, ob noch andere Gegenstände im Raum vorhanden sind, und springt zurück bzw. prüft weiter, ob das „eigene" Objekt der Spielfigur dabei ist. Durch einen Zufallsprozeß wird entschieden, ob sie es an sich nimmt. Falls nicht, kann auch ein anderer Gegenstand genommen werden. Falls die Person nicht ihren „eigenen" Gegenstand nimmt, hat sie immer noch die Möglichkeit, etwas anderes zu wählen. Das ist möglich, weil der Entscheidungspunkt quer durch den Baum verzweigt. Die entsprechenden Gegenstände werden also nicht lange herrenlos herumstehen, die Personen werden aber mit größerer Wahrscheinlichkeit ihre eigenen Dinge bekommen.

Die vier Bäume bilden gemeinsam eine größere Struktur, wobei jede zu prüfende Bedingung als einfacher logischer Ausdruck betrachtet werden kann. Da einige Entscheidungspunkte dieselben Bedingungen zu verschiedener Zeit prüfen wollen, fassen wir die Werte der möglichen Eigenschaften zu einem neuen Array zusammen, dessen Elemente wir beim Aufbau der Baumstrukturen nutzen.

Die Vielzahl der verschiedenen Faktoren, die das Ergebnis eines Rennens beeinflussen, machen dieses Programm zu einem preisgünstigen Übungswerkzeug, um „Wetten zu lernen".
Für den Wettenden wird der Computer so zum nützlichen Hilfsmittel, mit dem sich Gewinne maximieren und Fehleinsätze vermeiden lassen.

Auf der Rennbahn

Wir beginnen hier mit einer vierteiligen Serie, die sich mit dem Einsatz von Microcomputern bei Glücksspielen befaßt. Zum Auftakt bringen wir vier Programme, die für Anhänger des Pferderennsports geschrieben wurden.

Betrachtet man die ungeheure Popularität von Glücksspielen, überrascht es kaum, daß mit der Einführung des Computers viele Programmierer angeregt wurden, Algorithmen zu entwickeln, die dazu dienen, Gewinne zu maximieren, Verluste zu minimieren und Vorhersagen zu treffen. In dieser vierteiligen Serie beschäftigen wir uns mit den Techniken und der Theorie, die hinter Computer-„Spiel"-Anwendungen stehen, und wagen eine bescheidene Vorhersage darüber, ob Ihr Micro Sie zum Millionär machen wird oder nicht. Welcher Computerbesitzer ist noch nicht auf die Idee gekommen, ein Programm zur Lottozahlermittlung zu erstellen!?

In diesem Teil stellen wir vier Programme vor, die dem Wettenden angeblich Vorteile bringen sollen, und schätzen die möglichen Gewinne (oder Verluste!) für die einzelnen Programme ein. In den nächsten Folgen befassen wir uns mit dem Einsatz von Computern an der Rennbahn, der Statistik und Wahrscheinlichkeitsrechnung.

COURSEWINNER

Eines der interessantesten Programme dieser Kategorie ist „Coursewinner" von Selec Software. Vorausgesetzt wird dabei, daß dem Spieler ein Exemplar der englischen Wettzeitschrift „Sporting Life" oder einer vergleichbaren Rennzeitschrift zur Verfügung steht und somit die entsprechenden Daten vorhanden sind. Der Spieler hat dabei die Möglichkeit, drei verschiedene „Tempi" einzugeben, wie sie in der Sport- und Allgemein-Presse veröffentlicht werden oder von Fachleuten ermittelt wurden.

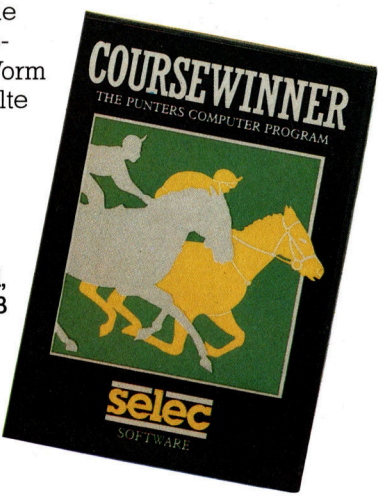
Das menügesteuerte Programm ist einfach in der Anwendung, wenn auch gewisse Fehlermöglichkeiten gegeben sind. Voraussetzung für einen guten Start ist die richtige Eingabe. Allerdings gibt es auch dann keine Katastrophe, wenn man z. B. in der Acorn-B-Version versehentlich die „Escape"-Taste drückt.

Sind die wichtigen Faktoren eingegeben, wird erwartet, daß ein Gebot erfolgt. Das Programm versucht nämlich – neben der Berechnung der „wirklichen Vorteile" der Pferde –, dahingehend Rat zu geben, ob die Chancen beim Buchmacher gut sind. Ist dies der Fall, wird eine „Dreierwette" empfohlen. Da Wettschluß erst zehn Minuten vor dem Start ist, läßt sich dies nicht immer durchführen.

Überdies gibt es die Möglichkeit, das Gebot entsprechend der „Verfassung" des Pferdes zu korrigieren, und somit die Gewinnchancen neu zu berechnen.

Das Programm wurde in 18 Rennen mit zehn oder weniger Pferden getestet. Das Ergebnis der Dreierwette mit drei Siegern bei zwölf Chancen war nicht ungewöhnlich: Sie brachte bei einem Einsatz von 4 Mark einen Gewinn von 6,50 Mark. Pferde aber, auf die man mehr setzte oder denen man höhere Chancen einräumte, brachten vergleichsweise mehr. So bei der Wette „Drei aus Fünf" umgerechnet einen Gewinn von 16 Mark. Schlechter bewertete Pferde, 6–4, (irreführenderweise im Programm als 3–2 bezeichnet) brachten bei „Zwei aus Sechs" immerhin einen Gewinn von 2,80 Mark.

„Coursewinner" ist ein interessantes, gut gestaltetes Programm mit Soundeffekten. Natürlich gilt hierfür – wie auch für andere Programme – eines: Vorm „echten" Einsatz sollte man trocken üben! Das spart Geld.



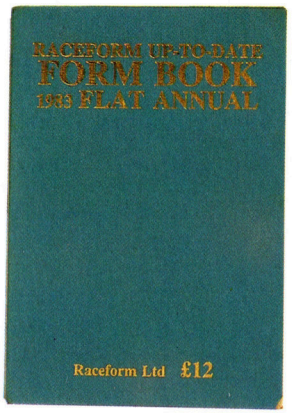
Coursewinner:
Für Spectrum, C 64, Schneider CPC, Atari, Apple II und Acorn B

Vertrieb:
Selec Software, 37 Councillor Lane, Cheadle, Cheshire

SPRINT FORMULA

„Brimardon Computer Racing Service" bietet mehrere Produkte an, die dem anspruchsvollen Wetter, der mehr mit seinem Computer machen möchte, einige Anregungen bringen. So die Möglichkeit, sein eigenes Programm zu schreiben, fertig konfektionierte Programme speziellen Gegebenheiten anzupassen, maßgeschneiderte Programme zu erstellen sowie „Sprint Formula". Letzteres ist ein Programm für den erfahrenen Wetter. Es konzentriert sich auf Vorgaberennen bis zu sieben Achtelmeilen, da diese schweren Rennen ein offenes Wetten erlauben. Die Autoren behaupten, daß das Programm eine „goldene" Gelegenheit für den Wetter ist, der zu großen Gewinnen kommen will.

Die Laufgeschwindigkeit des Pferdes kann durch die Bodenbeschaffenheit erhöht oder gesenkt werden, ferner die Streckenlänge und die Art des Rennens, in der die Bewertung erfolgte. Ergänzend lassen sich Faktoren wie die derzeitige Verfassung des Pferdes usw. hinzufügen. Ziel ist es natürlich, den größten Gewinn zu erzielen.



Dazu verhilft das System auch den Leuten, die bereit sind, auf die beiden höchstbewerteten Pferde zu setzen. Bisher sind aus 123 Wetten die zwei besten 28mal als Sieger hervorgegangen, was einem Einsatz von 79 Punkten entspricht – bei einem Ertrag von 62 Prozent. Der durchschnittliche Gewinn lag bei „13–2". Wer bereit war, die gelegentlichen Niederlagen hinzunehmen, wurde mit Gewinnen von bis zu „20–1" belohnt.

Die Autoren geben zu, daß die Bildschirmdarstellung von „Sprint Formula" aufgrund der geringen Speicherkapazität nicht optimal ist. Die Input-Routinen wirken teilweise etwas unbeholfen. Fehlersuche gibt es nicht, und es liegt am Spieler, Falscheingaben zu vermeiden. Um das auszugleichen, wird das Programm mit umfangreicher, hilfreicher Dokumentation geliefert. Insgesamt erweist es sich als hilfreich für den erfahrenen Wetter.

Sprint Formular: Für Acorn B, Acorn Electron, Spectrum, C 64, Schneider CPC, Tandy, Dragon, Oric und Atari
Vertrieb: Brimardon Computer Racing Service, 48 Pierremont Road, Darlington

Z5 HORSE RACE FORECAST

Das Programm „Z5 Horse Race Forecast" von Professor Frank George basiert ebenfalls auf Daten aus „The Sporting Life" sowie Laufgeschwindigkeiten aus „Raceform Handicap Book". Es arbeitet mit erheblich weniger Daten als „Coursewinner 3" und akzeptiert zwei Werte – die Geschwindigkeits-Bestleistung sowie die Form-Daten.

Auf verschiedene Anfragen werden Informationen eingegeben, anschließend offensichtliche Fehler beseitigt und die eingegebenen Daten nach Richtigkeit abgefragt. Daraufhin wird jedes Pferd nach folgendem System bewertet: optimale Wette, gute Wette, mögliche Wette, schlechte Wette, Außenseiter-Wette und löschen. Leider wird kein Versuch gemacht, diese Bewertungen zu quantifizieren. Der Anwender muß das Programm also ausführlich testen, um den echten statistischen Wert nutzen zu können.

Enttäuschend bei diesem Programm ist die Regelmäßigkeit, mit der bis zu fünf Pferde in ein und demselben Rennen bei zehn oder weniger Konkurrenten Spitzenwerte zugeteilt bekommen. Hier zwei Empfehlungen: Entweder nicht setzen oder auf alle Spitzen-„Reiter".

Z5 Horse Race Forecast: Für Spectrum, Acorn B, C 64
Hersteller: Bureau Of Information Science, Chalfont, St. Giles

Die Benutzung von „Coursewinner"

Nach dem Laden des Programms erscheint ein Menü. Danach ist „C" einzugeben, worauf mehrere Rennbahnen auf dem Bildschirm zu sehen sind. Nun wird die der Bahn entsprechende Nummer gewählt. Anschließend muß die Entfernung zur nächsten Achtelmeile eingegeben werden. Das Programm kehrt zum Hauptmenü zurück, und mit „I" erfolgt die Eingabe der Renndaten. Das Programm greift auf ein Submenü mit vier Routinen zurück, womit die Eingabe der verschiedenen Daten ermöglicht wird. In dem mit „Schnelligkeit" überschriebenen Sektor können drei der folgenden Bewertungen benutzt werden: Stopwatch (Sporting Life), Form Ratings (Sporting Life), Spotform (Daily Mirror), Formcast (Daily Mail) und Timeform.

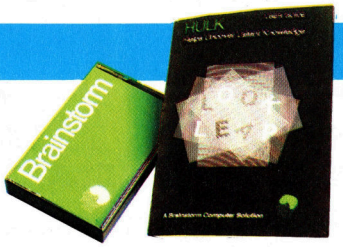
Ist man wieder im Hauptmenü, wird „P" eingegeben. Darauf erfolgt die Frage nach dem Startwert. Steht ein solcher nicht zur Verfügung, wird die Wett-Vorhersage benutzt. Dabei sollte man berücksichtigen, daß dies die empfohlenen Dreierwetten abwertet. Die Programmberechnung der „echten Bedingungen" wird allerdings nicht unbedingt beeinflußt.

Im Hauptmenü ist nun die Abfrage der Renn-Analyse wie dargestellt vorzunehmen. Je niedriger die berechneten Minuswerte, desto besser die Chancen des betreffenden Pferdes. Anfangs ist es vielleicht empfehlenswert, nur auf die Pferde zu setzen, die „1–1" gewettet werden. Bieten die Buchmacher Wetten von „6–4" und mehr, um so besser.

Hulk I und II

Die Autoren von „Hulk" beschreiben das Programm als „Wissenschaftler des kleinen Mannes", da es bei der Entwicklung eigener Expertensysteme hilft. Akzeptiert man, daß ein erfolgreicher „Wetter" Experte in seinem Gebiet ist – warum sollte man dann nicht versuchen, dieses Wissen auf den Computer zu übertragen? „Hulk" scheint deshalb besonders für den erfahrenen Wetter geeignet, der mit einem „System" arbeiten möchte – einer Reihe von Regeln auf der Basis vergangener Ereignisse (=Daten), die – eine gewisse Konsequenz vorausgesetzt – Gewinne erzeugen und – Profit bringen.

Die zeitaufwendigste Aufgabe ist die Erstellung der Datenbank. Speicherkapazitätsprobleme beim Acorn B stellen hohe Anforderungen an den Anwender, der genau entscheiden muß, welche Faktoren wichtig sind, und dazu dann noch zwecks Analyse ein bestimmtes Rennen einzugeben hat. Typisch für eine Strategie mag das Studium der Vorgaberennen über eine gegebene Distanz in Zehner-Rennen sein, wobei man sich bei der Vorhersage auf die ersten drei oder vier konzentrieren sollte. Dabei ist entscheidend, daß ein statistisch repräsentatives Muster gewählt wurde.



Die Daten werden von einem mit „Look" bezeichneten Programm analysiert, das die Entwicklung einer Reihe von Regeln (durchschnittlich sieben oder acht) erlaubt, um eine Vorhersage zu treffen (in diesem Fall, ob ein Pferd siegt oder nicht). Steht eine genügende Menge von Regeln zur Verfügung, können diese für weitere Rennen durch ein anderes Programm namens „Leap" aufgerufen werden. Damit sind Vorhersagen in Form von Gewinnprozenten möglich.

„Hulk" ist ein faszinierendes Programm für den „System"-Wetter, der sich der statistischen Analyse zur Siegerermittlung bedient. Mit diesem Programm ist es möglich, Regeln für eine Gewinnvorhersage auf selektiver Basis zu entwickeln. Die Trefferquote liegt bei 60 Prozent.

Hulk I und II: Für Acorn B Hulk I, Hulk II für dBase II, MS-DOS und PC-DOS-Rechner
Hersteller: Warm Boot Ltd., Finsbury Business Centre, 40 Bowling Green Lane, London EC1 ONE, und Brainstorm Computer Solutions, 103a Seven Sisters Road, London, N7 7QN

1 — 18.30 Uhr

Hans-im-Glück-Rennen

4.000 DM (2200, 1000, 500, 300) **Autostart**
Für 3- bis 6jährige inl. Pferde
deren Startsumme nicht mehr als 7.000 DM beträgt.

Rennstrecke 2140 m — **blaue Decke, gelbe Nummer**

Sieg: _____
Platz: _____
Zweierwette: _____
Dreierwette: _____

2140 Meter	**Cassina**	6j. dbr. Stute	12.10.	5.26,0	E1700	/W.Hehemann	260	Snowgirl/Fenne/Dambo
1	**R.Petersen**	Gristow-Casale	26.10.	4.25,8	E2100	/W.Hehemann	184	Kiwit/Hallo Kaiser/Cäsario
			16.11.	7.25,0	E1700	/W.Hehemann	264	Chartreus/Skidia/Speed Master
	24,2	Frau E. Gatermann	7.12.	5.29,1	E2100A	/R.Marcus	136	Mecki+Pjotre/Tiffy
	4.050,–	R. Petersen	21.12.	4.26,0	E1700	/R.Petersen	132	Dancing Qu./Keyst.Adolf/Dio Kan
	Frivol Han.	5j. dbr. Hengst	31.10.	1.26,5	H2140	/H.Rathjen	52	Frivol H./Imp.Dragon/Bel.Lee
2	**H.Rathjen**	Hans Han.-Frivola	20.11.	1.24,0	H2140	/H.Rathjen	36	Frivol Han./Leddybo/Oqueta
			5.12.	3.25,7	H2140	/H.Rathjen	16	Jilly Starl./Right Scot/Frivol H.
	24,0	Dr. H. Pohle	12.12.	dis.r.	H2140	/H.Rathjen	19	Anbros/Zomisto Fort./Tanagra
	4.600,–	H. Rathjen	26.12.	5.25,7	H2140	/H.Rathjen	44	Markos/Corola/Zomisto Fortuna
	Keyst. Adolf	5j. dbr. Hengst	3.11.	dis.r.	H2140	/F.König	80	Maxim/Glücksklee/Hänschen
3	**F.König**	Keyst. Stew.-Ena	24.11.	4.24,2	H2140	/F.König	92	La Guardia/Glücksklee/Tölle
			29.11.	dis.r.	R2000	/F.König	148	Pontedera/True Sally/Uta Pride
	22,8	W. Seubert	14.12.	4.27,5	E1700	/F.König	36	La Singla/Ellator/Valenza
	4.650,–	F. König	21.12.	2.25,4	E1700	/F.König	88	Dancing Qu./Keyst.Adolf/Dio Kan
	Alcan Senator	5j. br. Wallach	20.10.	5.gdZ.	H2140	/H.Christiansen	84	Wuttke/Muscadet/Annett Han.
4	**H.Christiansen**	Paw Senator-Alcantara	7.11.	dis.r.	H2140	/H.Christiansen	112	Favoritin/Annett Han./Laslola
			20.11.	9.23,2	H2140	/H.Christiansen	324	Fortalessa/Immertreuer/Wasja
	23,2	Frau B.+K. Fleischer	1.12.	dis.r.	H2140	/O.J.Schröder	356	Sel.Quiche/Elikon/Zom.Fortuna
	4.900,–	H. Rathjen	29.12.	dis.r.	M2000	/H.Wewering	100	Donum/Erril Nev./Arkansas Nev.
	Hallo Kaiser	5j. hbr. Stute	9.11.	3.25,6	E2100	/Chr.Ohmer	44	Corturally/Dornaxel/HalloKaiser
5	**E.Ohmer**	My Nevele-Torry P.	14.11.	4.25,2	H2140	/E.Ohmer	68	Chilly/Itala/Olycier
			17.11.	8.25,0	H2140	/E.Ohmer	108	Fania Lobell/Ulzera/Herbstgold

Wettdaten

Der obige Ausschnitt stammt aus einer Wettzeitschrift und enthält wichtige Daten über das „Hans-im-Glück-Rennen" und die in diesem Rennen startenden Pferde und Fahrer. Es ist das erste Rennen über eine Distanz von 2140 Meter. Insgesamt ist das Rennen mit 4000 Mark dotiert. Davon erhält der Sieger 2200 Mark, der zweite 1000 Mark. Weiter wie angegeben. Gestartet wird das Rennen hinter einem Auto. Die Numerierung in den einzelnen Spalten von oben nach unten gibt die Startposition an. 1 gleich Innenkante der Rennbahn. Es folgen der Pferdename, Fahrername, Bestzeit über die Rennstrecke, hier fehlt allerdings eine 1 vor der Sekundenangabe und darunter die bislang gewonnenen Preisgelder. In der nächsten Spalte das Alter des Pferdes und das Geschlecht, sowie Besitzer und Trainer. Dann folgen die letzten fünf Starttermine mit dem jeweiligen Fahrer, die Quote und die Reihenfolge des Einlaufs. Im Rennen am 12. 10. gab es für den Sieg von Snowgirl 260 für 10 Mark Wetteinsatz am Totalisator.

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.

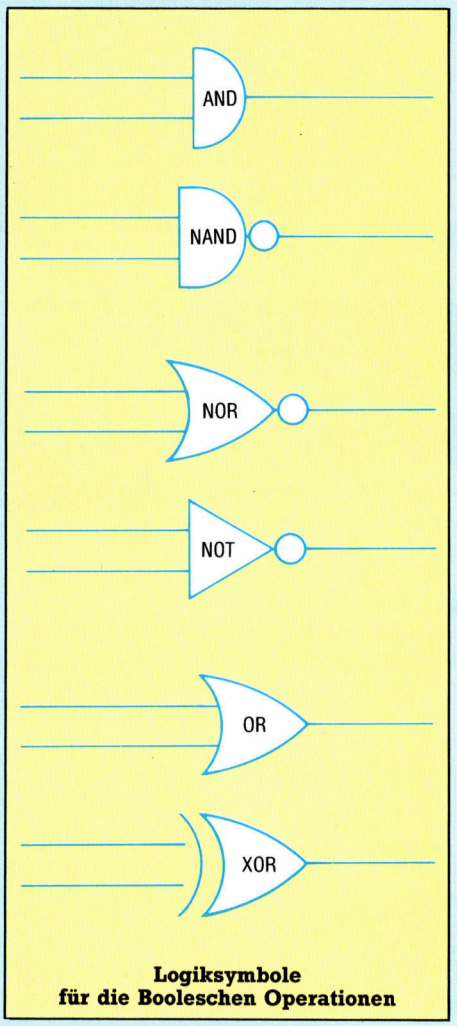
Logic Gate = Logikgatter
Schaltkreise zur Realisierung logischer Verknüpfungen heißen Logik-Gatter. Ein Gatter hat ein oder zwei Ausgänge und bis zu acht Eingänge; je nach den Eingangsspannungspegeln und der logischen Funktion des Gatters werden die Ausgänge auf „High" (Betriebsspannung) oder „Low" (Nullpegel) gesetzt.

Die Arbeitsweise eines Gatters läßt sich mit einer Anzahl Schalter vergleichen, die in einer bestimmten Anordnung betätigt werden; ein ODER-Gatter entspricht beispielsweise zwei parallel liegenden Schaltern. Als elektronische Schalter werden dabei bipolare sowie Feldeffekt-Transistoren verwendet. Ein Transistor hat drei Anschlüsse oder Elektroden, sie heißen beim bipolaren Transistor Emitter, Kollektor und Basis. Die Basis ist die Steuerelektrode; das dort anliegende Signal bestimmt, ob zwischen Emitter und Kollektor eine leitende Verbindung besteht oder ob der Transistor gesperrt ist. Die Gattereingänge sind jeweils mit Basis-Elektroden verbunden, und die Emitter-Kollektor-Strecken schalten definierte Spannungspegel auf die Ausgänge durch. Die Gattertypen – AND/OR/NAND/NOR/NOT/EXOR – unterscheiden sich in der Anordnung der Schalttransistoren.

Logic State = Logischer Zustand
Den Null- und Eins-Binärwerten lassen sich die logischen Zustände „wahr" und „falsch" zuordnen.

Der zugehörige Spannungspegel hängt dagegen von der verwendeten „Logikfamilie" ab. Bei einem Gatter, das beispielsweise mit +5V Betriebsspannung arbeitet, entsprechen diese 5V bei positiver Logik dem Zustand „wahr", 0 V bedeutet dementsprechend „falsch".

Logic Symbols = Logiksymbole
Jeglicher Systementwurf – ob nun in Form eines Flußdiagramms oder einer elektronischen Schaltung – sollte sich möglichst auf einen Satz von grafischen Symbolen stützen, mit dem die verschiedenen Komponenten in eindeutiger und allgemeinverständlicher Weise zu kennzeichnen sind.



Logiksymbole für die Booleschen Operationen

Für den Entwurf logischer Schaltkreise gibt es international genormte Logiksymbole. Die Zeichen für die gebräuchlichsten Verknüpfungen der Booleschen Algebra sind in der Abbildung wiedergegeben; neben diesen Symbolen werden noch genormte Sinnbilder für die verschiedenen Flipflop-Typen verwendet.

Logging On = Einloggen
Beim Rechnerbetrieb ist es oft unerläßlich, den Zugriff unbefugter Personen auf die Anlage insgesamt oder auf bestimmte Dateien zuverlässig zu verhindern. Daher muß sich der Benutzer bei größeren Systemen meist erst einmal „einloggen".

Zum Einloggen gehört normalerweise ein Paßwort. Wenn viele Leute das gleiche Terminal benutzen, wird häufig ein gemeinsames Paßwort als generelle Zugangsberechtigung verwendet und dazu noch ein persönliches, das die eigenen Dateien oder andere Systembereiche freigibt.

Nach Beendigung seiner Tätigkeit muß sich der Benutzer über eine „Log off"-Routine abmelden (ausloggen). Damit ist die Verbindung zwischen Terminal und Rechner unterbrochen. Bei Großanlagen wird der Benutzer im allgemeinen automatisch abgehängt, wenn länger keine Eingabe erfolgt.

Vor allem mit der wachsenden Verbreitung von Computernetzen in den letzten Jahren hat auch die Zahl der Hacker zugenommen, die Paßworte knacken und unberechtigt in fremde Systeme eindringen. Zumindest für Telefon-Datennetze gibt es jedoch neuerdings eine wirkungsvolle Methode, die den Hackern kaum noch Chancen läßt: Der Rechner unterbricht die Verbindung, nachdem das Paßwort eingegeben ist, und ruft selbst den rechtmäßigen Besitzer an.

Bildnachweise
1709: Nick Harris
1710: Liz Heany
1711: Tony Sleep
1712, 1721, U3: Liz Dixon
1714, 1724: Kevin Jones
1717: Sperry Ltd.
1722: Raster Technologies
1723: G. Entis, R. Chuang
1725, 1728, 1729: Ian McKinnell
1731–1733: Caroline Clayton
1734: Sporting Pictures
1734–1736: Marcus Wilson-Smith

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

computer kurs

Heft 63

Große Probleme
In unserem Adventure-Spiel treten schwerwiegende Ereignisse ein, darunter Pest-Epidemie und Rettung Schiffbrüchiger.

Robotsteuerung
Für den Robotarm ist eine unterschiedliche Steuer-Software nötig.
Diesmal: Commodore-Basic.

Brot und Spiele
Es gibt eine Cassette, deren Verkaufserlös Bedürftigen in Afrika zugute kommt.

Zahlenakrobatik
FORTRAN bietet sehr gute Möglichkeiten der Zahlenverarbeitung.

Top-Modell
Commodores Amiga beeindruckt durch große Leistungsfähigkeit und zeigt sich im modernen Gewand.