CHIP WISSEN

Rüdeger Baumann

Computerspiele und Knobeleien programmiert in BASIC

Ein Buch von CHIP, der Zeitschrift
für Mikrocomputer-Technik

Rüdeger Baumann

Computerspiele und Knobeleien, programmiert in BASIC

CHIP WISSEN

Rüdeger Baumann

Computerspiele und Knobeleien programmiert in BASIC

3. Auflage

VOGEL-BUCHVERLAG
WÜRZBURG

Rüdeger B a u m a n n

1936 geboren, von 1956 bis 1966 Studium der Soziologie und der Mathematik an den Universitäten Münster, Berlin und Frankfurt. Seit 1968 im Schuldienst. Derzeitige Tätigkeit: Mathematik- und Informatik-Unterricht am Johanneum und an der Volkshochschule Lüneburg sowie Lehrerfortbildung.

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Baumann, Rüdeger:
Computerspiele und Knobeleien programmiert in BASIC / Rüdeger Baumann. - 3. Aufl. - Würzburg: Vogel, 1983.
(Chip-Wissen: Applikation)
ISBN 3-8023-0703-8

ISBN 3-8023-0703-8
3. Auflage. 1983

Printed in Germany

Herstellung: Vogel-Druck Würzburg

Inhaltsverzeichnis

Vorwort VII

Überblick IX

1 Zur Einführung

1.1 Welche Hand? 1
1.2 Additionstrainer 4
1.3 Zu groß - zu klein 7
1.4 Ziel 100 10
1.5 Wer paßt zu wem? 13

2 Lernspiele

2.1 Alphabet 19
2.2 Zahlen-Senso 21
2.3 Begriffe merken 23
2.4 Karten wenden 26
2.5 Rechentrainer 30
2.6 Termumformung 32
2.7 Fahrlehrer 35

3 Graphikspiele

3.1 Elementare Bildschirmaktionen 42
3.2 Fledermaus 53
3.3 Sterne jagen 56
3.4 Eidechse 60
3.5 Ballons 63
3.6 Labyrinth 67
3.7 Zahlenlabyrinth 74
3.8 Badinerie 82
3.9 Lebensspiel 85

4 Such- und Ratespiele

4.1 Nicomachus 100
4.2 Intervallsuche 102
4.3 Begriffe raten 104
4.4 Verwürfelt 106
4.5 Knack den Code 108
4.6 Spionsuche 111
4.7 Lapland 115

5 Glücksspiele

5.1 Die böse Sechs 122
5.2 Craps 126
5.3 Ein Münzwurfspiel 130
5.4 Siebzehn und vier 136
5.5 Roulette 142
5.6 Einarmiger Bandit 148

6 Strategiespiele

6.1 Nim mit mehreren Schalen 160
6.2 Kegelspiel 171
6.3 Treib die Dame in die Ecke 176
6.4 Kringel und Kreuze 183
6.5 Brückenspiel 195
6.6 Mikroschach 202
6.7 Vier gewinnt 215

7 Gemischte Strategien

7.1 Einfaches Pfennigknobeln 221
7.2 Rate die Karte 230

8 Geduldspiele

8.1 Magische Quadrate 241
8.2 n-Damen-Problem 253
8.3 Springertour 260
8.4 Fünfzehnerspiel 266
8.5 Umfüllaufgabe 278
8.6 Zauberwürfel 281

9 Vermischte Knobeleien 291

10 Anhang

10.1 Anmerkungen 299
10.2 Literaturverzeichnis 301
10.3 Die Programmiersprache BASIC 303

Vorwort

Der Computer ist ein Spielzeug,
welches die Natur uns zuwarf -
zur Unterhaltung und zum Troste
in der Finsternis.
d'ALEMBERT

Übrigens ist mir alles verhaßt,
was mich bloß belehrt,
ohne meine Tätigkeit zu vermehren
oder zu beleben.
GOETHE

Um Mißverständnisse gar nicht erst aufkommen zu lassen: dies Buch ist k e i n e Sammlung von Spielkonserven im Sinne fertiger Programme, die nur in den Computer eingetippt oder eingelesen und anschließend konsumiert zu werden brauchen. Vielmehr wird der Leser zu einem aktiven und schöpferischen Umgang mit Computerspielen aufgerufen und angeleitet: aus der Spielidee entwickelt sich die Spielstrategie und hieraus das Programm. Das Programmieren des Computers selbst ist das Spiel; so lernt der Leser spielend das Programmieren.
Programmierkenntnisse werden keine vorausgesetzt - wohl aber die Bereitschaft, Ideen aufzunehmen und selbsttätig weiterzuführen. Die vorgelegten Programmbeispiele sind keineswegs perfekt; sie harren vielmehr der Ausgestaltung und Vervollkommnung durch den Leser. Die zahlreichen Aufgaben sollen der Phantasie und Eigentätigkeit des Lesers zur Anregung dienen.

Ich habe aus vielerlei Quellen geschöpft; allen voran ist hier Martin Gardner zu nennen, dessen Bücher (siehe das Literaturverzeichnis) und Spalten im "Scientific American" bzw. im "Spektrum der Wissenschaft" einen Riesenschatz an Spielideen darstellen. Ferner die Zeitschriften "Creative Computing" und "Jeux et Stratégies", in deren Beiträge wiederum viel von Gardner eingeflossen ist, der seinerseits die Ideen zahlreicher Autoren verwertet. Der erste und wahre Erfinder vieler Spiele ist selten noch auszumachen.

Spielen ist so alt wie die Menschheit selbst: alle Völker und Kulturen kennen und betreiben es. Neuen Auftrieb erhält es durch die sogenannte d r i t t e i n d u s t r i e l l e R e v o l u t i o n: einmal, indem diese die Freizeit der Menschen erweitert, und zum anderen, indem sie im Computer einen Spielmeister und Spielpartner von großer Gewandtheit und Flexibilität bereitstellt. Das Faszinierende daran ist, wie man einer Maschine Verhaltensweisen beibringen kann, daß sie als Spielgegner ernstzunehmen ist. Der Computer ist so universell, daß jedes Spiel - das ums Überleben und das rein zum Spaß - auf ihm nachgeahmt oder gegen ihn gespielt werden kann.

Oft wird gesagt, ein Spiel habe seinen Reiz verloren, wenn seine Theorie bekannt sei. Für den Computer-Liebhaber wird es jedoch jetzt erst interessant. Spielen ist reizvoll, aber das Programmieren eines Spiels ist interessanter, denn letzteres kann nur, wer das Spiel vollständig verstanden hat.

Das Spiel ist, im Unterschied zur Arbeit, eine Tätigkeit, die um ihrer selbst willen, d.h. um des aus ihr fließenden Vergnügens willen, ausgeübt wird; es ermöglicht eine Form von Freiheit - auch und gerade in der technisierten und verwalteten Welt. So vermag der Computer, dem man gerne menschenfeindliche Tendenzen nachsagt, zur ästhetischen Erziehung der Menschen beizutragen.

Lüneburg R . B a u m a n n

Überblick

Das Gebiet der Spiele ist unübersehbar groß und es wird ständig größer, denn die Phantasie der Menschen ist unerschöpflich. Um nicht in der Flut des Materials unterzugehen, müssen wir uns auf weniges beschränken; glücklicherweise hat die Beschränkung auf Computerspiele eine heilsame Auswahl zur Folge.

Wir wollen uns jetzt durch Einteilung der Spiele nach gewissen Gesichtspunkten einen Überblick verschaffen und danach die Themen dieses Buches einordnen.

Zunächst kann man die Spiele hinsichtlich der Rolle des Computers einteilen:

- Der Computer kann Spielmeister sein, d.h. er regelt und überwacht den Spielverlauf, wertet die Aktionen der Spieler aus und entscheidet über Gewinn oder Verlust.
- Der Computer kann aber auch Mitspieler, Spielpartner oder Spielgegner sein. In diesem Fall müssen wir ihn mit einer gewissen Intelligenz ausstatten (wenn es sich nicht um reine Glücksspiele handelt): dies ergibt die interessantesten, aber auch die schwierigsten Programmieraufgaben, denn wer dem Computer eine Strategie vermitteln will, muß das betrachtete Spiel vollständig analysiert haben.

Ein weiterer Einteilungsgesichtspunkt ist die Anzahl der Spieler.

- Bei Einpersonenspielen handelt es sich um die bekannten Geduldspiele und Knobeleien; der Computer ist Spielmeister und -medium zugleich. Dank seiner kombinatorischen Fähigkeiten ist er eine wertvolle Hilfe beim schnellen Durchmustern von Lösungsmöglichkeiten (Kapitel 8 und 9).
- Den Zweipersonenspielen sind die Kapitel 4 bis 7 gewidmet: hier kann der Computer die Rolle mit dem Spieler tauschen.
- Mehrpersonenspiele werden seltener vorkommen; jedoch wird der Leser häufig aufgefordert werden, ein Spiel zum Mehrpersonenspiel (mit dem Computer als Spielmeister oder als einem der Mitspieler) auszubauen.

Die dritte Einteilung betrifft den Einfluß, den wir dem Zufall auf das Spielgeschehen einräumen.

- Ist der Zufall ausgeschlossen, der Spielverlauf also nur von den Entscheidungen der Spieler abhängig, spricht man von reinen Strategiespielen; ihnen ist Kapitel 6 gewidmet.
- Bei reinen Glücksspielen besitzt der Spieler keine Einwirkungsmöglichkeit auf den Spielverlauf. An ihnen läßt sich das Wirken des Zufalls studieren. Ihren spezifischen Reiz (etwa die Casinoatmosphäre) kann der Computer nur unvollkommen wiedergeben (Kapitel 5).
- Bei Spielen mit gemischter Strategie setzt der Spieler den Zufall als Bestandteil seiner Strategie: Spiele dieser Art sind besonders interessant, aber auch schwer zu verstehen (Kapitel 7).-

Wir haben bisher nicht nach dem Spielmaterial unterschieden: Brett, Würfel, Karten, Schreibzeug. Es ist nicht leicht darzustellen; um dies zu tun, muß man die Graphikmöglichkeiten des Computers kennen und benutzen: dem ist Kapitel 3 gewidmet. Grundsätzlich können alle Spiele (auch) Graphikspiele sein, wenn man den Bildschirm zur Darstellung des Spielverlaufs benutzt; insofern ist unsere Einteilung der Spiele logisch nicht ganz einwandfrei.

Nimmt man die vom Spieler erwartete Tätigkeit als Gliederungsmittel, so

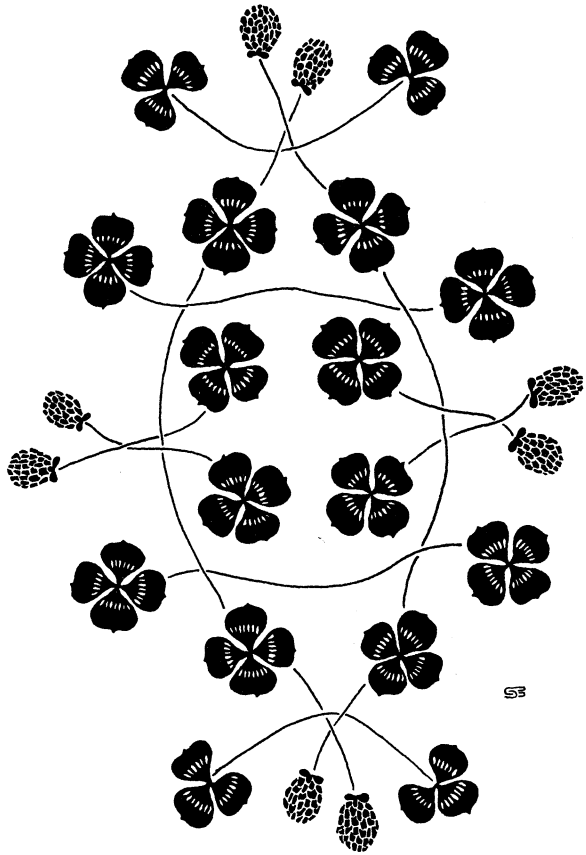
bekommt man die Einteilung in

- Such- und Ratespiele (Kapitel 4),
- Konzentrations- und Gedächtnisspiele (Kapitel 2),
- Reaktions- und Geschicklichkeitsspiele (hauptsächlich in Kapitel 3),
- Denkspiele.

Auch diese Einteilung ist logisch nicht konsistent, denn beim Suchen und Raten kommt man ums Denken nicht herum, ja man wird eine Such- oder Ratestrategie (Kapitel 6) anwenden.-

Noch eine Bemerkung zur verwendeten Programmiersprache (BASIC). Mit Recht wird ihr vorgeworfen, daß sie zum 'Hacken', dem wilden, unsystematischen Programmieren verführe, dessen Ergebnis der gefürchtete 'Spaghetticode' ist. Sprechende Beispiele dafür sind die in Zeitschriften und Sammelwerken abgedruckten Programme (z.B. in [1], [2], [6], [21] und [23] des Literaturverzeichnisses): sie sind vollkommen unlesbar und damit unverständlich.
Ein Hauptanliegen des vorliegenden Buches ist der Nachweis, daß man auch in BASIC gut strukturierte und damit verständliche Programme schreiben kann. Der Geschwindigkeitsfanatiker wird natürlich die Nase rümpfen: "zu langsam!". Wem es an Speicherplatz mangelt, wird den zuweilen großen Umfang der Programme als Verschwendung empfinden. Hat man ein Programm verstanden, ist es jedoch leicht,eine schnelle bzw. knappe Fassung anzufertigen.

Zum Schluß möchte ich die Leser um Kritik, Verbesserungsvorschläge und Anregungen bitten.
Meine Anschrift: R . B a u m a n n , In den Stuken 16 , 2120 Lüneburg .



1
Zur Einführung

Die Spiele dieses Kapitels sind - bis auf das vierte - e i n f a c h in doppeltem Sinne: einfach zu spielen und einfach zu programmieren. Denn Sie - der Leser - sollen ja zum Umgang mit dem Computer angeregt werden. Das erste Spielchen wird dem Computer scheinbar telepathische Fähigkeiten verleihen; es benutzt einen alten Trick, der auf dem Zahlenwert von Münzen beruht.

Beispiel 1: Welche Hand?

Der Computer fordert den Spieler auf, einen Groschen in die eine und einen Pfennig in die andere Faust zu nehmen. Nun bittet er darum, den Wert der Münze in der rechten Hand mit 8 (oder irgendeiner anderen geraden Zahl) und den der Münze in der linken Hand mit 5 (oder irgendeiner anderen ungeraden Zahl) zu multiplizieren, die Ergebnisse zu addieren und dem Computer mitzuteilen, ob die Summe gerade oder ungerade ist. Daraufhin teilt der Computer dem Spieler mit, in welcher Hand sich der Groschen und in welcher sich der Pfennig befindet.

Ein D i a l o g zwischen Computer und Spieler könnte folgendermaßen ablaufen:

```
Computer:  Ist die Summe der von Ihnen errechneten Zahlen
           eine gerade oder eine ungerade Zahl?

Spieler:   gerade

Computer:  Der Groschen befindet sich links, der Pfennig rechts.
```

Bevor wir uns klarmachen, worauf der Trick beruht, wollen wir ihn programmieren. Das Verfahren, welches dem Programm zugrunde liegt, läßt sich so beschreiben:

WELCHE HAND

```
Ausgabe: Spielregeln und Frage  (* des Computers *)
Eingabe: Antwort                (* des Spielers *)
WENN  Antwort = "gerade"  DANN
  Groschenhand := links
  Pfennighand  := rechts
SONST
  Groschenhand := rechts
  Pfennighand  := links
ENDE-WENN
Ausgabe: "Der Groschen befindet sich "; Groschenhand;
         "der Pfennig  befindet sich "; Pfennighand .
```

Die Übersetzung dieses Verfahrens in die Programmiersprache befindet sich auf der folgenden Seite. -

Worauf beruht nun der Trick? Zwei Fälle können auftreten:

1. Fall: Groschen rechts, Pfennig links.

Wert rechts:	$8 \cdot 10$,	allgemein: $2m \cdot 10$	(gerade Zahl)
Wert links:	$5 \cdot 1$,	allgemein: $(2n+1) \cdot 1$	(ungerade Zahl)
Summe:	85,	allgemein: $2(10m + n) + 1$	(ungerade Zahl).

2. Fall: Groschen links, Pfennig rechts.

Wert rechts:	$8 \cdot 1$,	allgemein: $2m \cdot 1$	(gerade Zahl)
Wert links:	$5 \cdot 10$,	allgemein: $(2n+1) \cdot 10$	(gerade Zahl)
Summe:	58,	allgemein: $2(m + 10n + 5)$	(gerade Zahl).

Der Trick beruht also letztlich auf der Regel "gerade + gerade = gerade" und "gerade + ungerade = ungerade".

```
100 : PRINT "□"
110 : PRINT "                    WELCHE HAND?"
111 : PRINT "                    ------------"
112 :
120 REM DER COMPUTER BETAETIGT SICH ALS ZAUBERER, D.H. ER SCHEINT
121 REM GEDANKEN LESEN ZU KOENNEN AUF GRUND EINER EINFACHEN
122 REM ARITHMETISCHEN BEZIEHUNG FUER GERADE ZAHLEN
123 :
200 REM === BEKANNTGABE DER SPIELREGELN ==============================
201 :
210 : PRINT : PRINT
220 : PRINT "NEHMEN SIE BITTE IN DIE EINE HAND EINEN GROSCHEN, IN DIE
230 : PRINT "ANDERE HAND EINEN PFENNIG. ICH WERDE ERRATEN, IN WELCHER
240 : PRINT "HAND SIE WELCHE MUENZE HABEN, WENN SIE MIR FOLGENDE FRAGE
245 : PRINT "BEANTWORTEN:
250 : PRINT "MULTIPLIZIEREN SIE DEN WERT DER MUENZE IN IHRER RECHTEN
255 : PRINT "MIT 8 (ODER MIT IRGEND EINER ANDEREN GERADEN ZAHL) UND
260 : PRINT "MULTIPLIZIEREN SIE DEN WERT DER MUENZE IN IHRER LINKEN
265 : PRINT "MIT 5 (ODER IRGEND EINER ANDEREN UNGERADEN ZAHL).
270 : PRINT
290 :
300 REM === FRAGE UND ANTWORT ======================================
301 :
310 : PRINT
320 : PRINT "IST DIE SUMME DER VON IHNEN ERRECHNETEN ZAHLEN
330 : PRINT "EINE GERADE ODER EINE UNGERADE ZAHL?
335 : PRINT "ANTWORTEN SIE BITTE MIT 'GERADE' ODER 'UNGERADE'!
340 : PRINT
350 : INPUT "IHRE ANTWORT "; A$
360 :
370 : IF A$ = "GERADE"   THEN 400
380 : IF A$ = "UNGERADE" THEN 450
390 : PRINT "FALSCHE EINGABE!": GOTO 335
395 :
400 : REM SUMME IST GERADE
405 :
410 :   LET G$ = "LINKS"  : REM DER GROSCHEN BEFINDET SICH LINKS
420 :   LET P$ = "RECHTS" : REM DER PFENNIG BEFINDET SICH RECHTS
430 :   GOTO 500
440 :
450 : REM SUMME IST UNGERADE
455 :
460 :   LET G$ = "RECHTS" : REM DER GROSCHEN BEFINDET SICH RECHTS
470 :   LET P$ = "LINKS"  : REM DER PFENNIG BEFINDET SICH LINKS
490 :
500 REM === ANTWORT DES COMPUTERS ==================================
501 :
510 : PRINT : PRINT
520 :
530 : PRINT "DER GROSCHEN BEFINDET SICH ";G$;", DER PFENNIG ";P$;"."
540 :
590 : END
```

E r l ä u t e r u n g e n zum Programm "Welche Hand":

1. Zeile 100 dient zum Löschen des Bildschirms. Man schreibt PRINT , setzt ein Anführungszeichen und drückt dann (zusammen mit der Umschalttaste) die Taste CLEAR.

2. Die Anweisung REM (von engl. remark = Bemerkung) leitet einen Kommentar ein und sagt dem Computer, was hinter ihr auf der gleichen Zeile steht, nicht zu beachten.
   Wollen Sie das Programm von Hand eingeben und nicht allzuviel schreiben, so lassen Sie die Zeilen 120-122, 200, 300, 400 einfach weg; sie müssen dann aber Sprunganweisungen (GOTO), die diese Zeilen als Ziel haben, ändern (Zeile 430 muß dann beispielsweise 'GOTO 510' heißen).

3. Zeile 350: Die Variable A$ (gelesen "A string" von engl. string = Zeichenkette) dient zur Aufnahme Ihrer Antwort; das Dollarzeichen $ kennzeichnet Variablen für Zeichenketten. Die Anweisung INPUT dient zur Eingabe der Antwort.

4. In Zeile 370 tritt eine b e d i n g t e A n w e i s u n g (Entscheidungsanweisung) auf.

```
IF  A$ = "GERADE"  THEN 400
```

   bedeutet:

```
WENN  A$ = "GERADE"  DANN
  fahre mit Programmzeile 400 fort
SONST
  fahre mit der folgenden Programmzeile fort
ENDE-WENN
```

5. Wenn der Spieler weder "gerade" noch "ungerade" als Antwort eingegeben hat, so bewirkt Zeile 390 einen Rücksprung zu Zeile 335, wo er erneut zur Eingabe aufgefordert wird.

*

Nun etwas eher Nützliches, bei dem das Programmieren Spaß macht und auch (eine Weile) das Spielen.

Beispiel 2 : Additionstrainer

Der Computer stellt eine Additionsaufgabe und der Benutzer tippt seine Lösung ein. Hierauf antwortet der Computer mit 'richtig' oder 'falsch' und geht im Falle einer zutreffenden Antwort zu einer neuen Aufgabe über. Die Anzahl der Aufgaben soll vorher wählbar sein.

Einen D i a l o g zwischen Spieler und Computer finden Sie auf der folgenden Seite.

```
                Additionstrainer

Wieviel Aufgaben wünschen Sie? 2

Wieviel ist  28 + 13 ?  41
Richtig!

Wieviel ist  39 + 15 ?  64
Leider daneben. Versuchen Sie es noch einmal!
Wieviel ist  39 + 15 ?  54
Richtig!


Noch eine Aufgabenserie (j/n) ?  n
Es hat mich gefreut, auf Wiedersehen.
```

Das Verfahren, nach dem der Computer vorgeht, lautet so:

ADDITIONSTRAINER

```
Eingabe: anzahl  (* der gewünschten Aufgaben *)
FÜR i VON 1 BIS anzahl WIEDERHOLE
  Wähle zwei Zufallszahlen  z1 , z2
  zwischen  1  und  100
  Frage: "Wieviel ist " z1 + z2 ?
  Eingabe: summe
  WENN  z1 + z2 = summe  DANN
    Ausgabe: "Richtig!"
  SONST
    Ausgabe: "Falsch!"
  ENDE-WENN
ENDE-WIEDERHOLE
```

Hier ist allerdings noch nicht berücksichtigt, daß der Computer bei falscher Antwort die Frage wiederholen soll.

Das BASIC-Programm dazu lautet:

```
100 : PRINT "□"
110 : PRINT "          ADDITIONSTRAINER
111 : PRINT "          ----------------
112 :
120 REM DER BENUTZER KANN SEINE FAEHIGKEITEN IM ADDIEREN
121 REM NATUERLICHER ZAHLEN VERBESSERN
129 :
130 : PRINT
140 : INPUT "WIEVIEL AUFGABEN WUENSCHEN SIE "; N
190 :
200 : FOR I = 1 TO N  : REM HAUPTSCHLEIFE ============================
201 :
210 :   REM --- WAHL ZWEIER ZUFALLSZAHLEN ---
220 :
230 :     LET Z1 = INT(99*RND(1))+1
240 :     LET Z2 = INT(99*RND(1))+1
250 :
260 :   REM --- FRAGE UND ANTWORT -----
270 :
280 :     PRINT
290 :     PRINT "WIEVIEL IST ";Z1;"+";Z2;
300 :     INPUT S
310 :     PRINT
320 :     IF S = Z1+Z2 THEN PRINT "RICHTIG!": GOTO 360
330 :     PRINT "LEIDER DANEBEN.";
340 :     PRINT "VERSUCHEN SIE ES NOCH EINMAL!": GOTO 280
350 :
360 : NEXT I  : REM ENDE DER HAUPTSCHLEIFE ==========================
390 :
400 REM --- VERABSCHIEDUNG ODER WIEDERHOLUNG ----------------------
401 :
410 : PRINT : PRINT : PRINT
420 : INPUT "NOCH EINE AUFGABENSERIE(J/N) "; A$
430 : IF A$ = "J" THEN RUN
440 : PRINT
450 : PRINT "ES HAT MICH GEFREUT, AUF WIEDERSEHEN.
460 : PRINT : PRINT
490 : END
```

Erläuterungen:

1. Neu an diesem Programm ist das Auftreten von Schleifen (Wiederholungsanweisungen).

```
200  FOR I = 1 TO N
     <Anweisungsfolge>
360  NEXT I
```

ist eine sogenannte Zählschleife: die Variable I zählt von 1 bis N, und bei jedem Zählschritt wird die Anweisungsfolge ausgeführt; diese wird also genau N-mal wiederholt.

2. Um den Rechner dazu zu veranlassen, uns immer neue Aufgaben zu stellen, rufen wir in den Zeilen 230 und 240 die Funktion RND(1) auf, welche 'zufällige' Zahlen zwischen 0 und 1 liefert (engl. random = zufällig). Sollen die Zufallszahlen g a n z e Zahlen zwischen den Grenzen A und B sein, so müssen wir

```
LET Z = INT((B - A + 1)*RND(1)) + A
```

setzen. Für A = 1 und B = 99 ergibt dies

```
LET Z = INT(99*RND(1)) + 1 .
```

3. In Zeile 320 prüft der Rechner, ob die Antwort des Benutzers korrekt ist: in diesem Falle fährt er mit Zeile 360 (NEXT I) fort, d.h. er erhöht die Zählvariable I um eins und liefert - falls N noch nicht überschritten ist - die nächste Aufgabe.
4. In Zeile 420 wird nach einer möglichen Wiederholung gefragt. Lautet die Antwort "J", so beginnt das Programm von vorn (Kommando RUN), andernfalls verabschiedet es sich.

*

Als drittes Beispiel dieser Einführung nehmen wir uns ein einfaches Ratespiel vor.

Beispiel 3: Zu groß - zu klein (Zahlenraten)

Eine zwischen 1 und n vom Computer zufällig gewählte Zahl soll erraten werden; nach jedem Rateversuch gibt der Computer einen Hinweis "zu groß" oder "zu klein". Wurde richtig geraten, nennt der Computer die Anzahl der Versuche.

Ein D i a l o g kann sich so abspielen:

```
                Zahlenraten

Ich habe mir eine natürliche Zahl zwischen 1 und 14 gedacht.

Welche Zahl ist es?  7
7 ist zu groß!

Welche Zahl ist es?  4
4 ist zu klein!

Welche Zahl ist es?  5
Getroffen!

Sie haben 3 Versuche benötigt.
```

Damit die Raterei nicht zu simpel wird, lassen wir den Computer zunächst die obere Grenze des Ratebereichs zufällig bestimmen; im Dialog auf der vorigen Seite hat der Computer als obere Grenze die Zahl 14 ausgewählt. Das Vorgehen läßt sich so beschreiben:

ZU GROSS - ZU KLEIN

```
obere_grenze  := Zufallszahl zwischen 1 und 1000
gemerkte_zahl := Zufallszahl zwischen 1 und obere_grenze
Ausgabe: "Ich habe mir eine Zahl zwischen 1 und " obere_grenze
         "gemerkt"
versuchsanzahl := 0
WIEDERHOLE
  WIEDERHOLE
    Ausgabe: "Welche Zahl ist es?"
    Eingabe: ratezahl
    versuchsanzahl := versuchsanzahl + 1
    WENN  ratezahl > gemerkte_zahl  DANN
      Ausgabe: ratezahl "ist zu groß"
    WENNABER  ratezahl < gemerkte_zahl  DANN
      Ausgabe: ratezahl "ist zu klein"
    SONST
      Ausgabe: "Getroffen!"
    ENDE-WENN
  BIS  ratezahl = gemerkte_zahl
  ENDE-WIEDERHOLE
  Ausgabe: "Noch einmal?
  Eingabe: antwort
BIS antwort ≠ "ja"
ENDE-WIEDERHOLE
Ausgabe: "Auf Wiedersehen."
```

Das zugehörige BASIC-Programm findet sich auf der folgenden Seite.

```
100 : PRINT "□"
110 : PRINT "        ZAHLENRATEN"
111 : PRINT "        -----------"
112 :
120 REM DER BENUTZER SOLL EINE VOM COMPUTER GEMERKTE ZAHL RATEN
190 :
200 REM === FESTLEGUNG DER ANFANGSWERTE ============================
201 :
210 : LET OG = INT(1000*RND(1)) : REM OBERE GRENZE DER ZAHLEN
220 : LET Z = INT(OG*RND(1))+1  : REM GEMERKTE ZAHL
230 : PRINT
240 : PRINT "ICH HABE MIR EINE ZAHL ZWISCHEN 1 UND ";OG;" GEMERKT."
250 :
260 : LET V = 0    : REM ANZAHL DER VERSUCHE
290 :
300 REM === RATEVORGANG ============================================
301 :
310 : PRINT
320 : INPUT "WELCHE ZAHL IST ES "; R
330 : LET V = V+1          : REM EIN WEITERER VERSUCH
340 : IF R > Z THEN PRINT R;" IST ZU GROSS": GOTO 310
350 : IF R < Z THEN PRINT R;" IST ZU KLEIN": GOTO 310
360 : PRINT : PRINT : PRINT "GETROFFEN! ";
370 : PRINT " SIE HABEN ";V;" VERSUCHE BENOETIGT."
390 :
400 REM === WIEDERHOLUNG ODER VERABSCHIEDUNG =======================
401 :
410 : PRINT : PRINT : PRINT
420 : PRINT "NOCH EINMAL (J/N)?";
430 : GET A$ : IF A$ = "" THEN 430
440 : IF A$ = "J" THEN PRINT "□": GOTO 200
450 : PRINT : PRINT : PRINT "AUF WIEDERSEHEN."
490 : END
```

Erläuterungen:

1. In Zeile 210 wird mit OG die obere Grenze des Ratebereichs als Zufallszahl zwischen 1 und 1000 ermittelt. Die übrigen Anweisungen sind uns bereits vertraut.

2. Neu ist jedoch die Eingabe in Zeile 430: die Anweisung GET A$ bewirkt das Einlesen genau eines Zeichens vom Tastenfeld in die Variable A$, jedoch ohne (wie bei INPUT) das Programm anzuhalten. Mittels `IF A$ = "" THEN 430` wird abgefragt, ob überhaupt ein Zeichen in A$ eingelesen wurde. Ist dies n i c h t der Fall, d.h. gilt A$ = "" (leere Zeichenkette), so bleibt der Rechner in der Schleife von Zeile 430 hängen; er verläßt sie erst und fährt mit der Programmausführung fort, wenn in A$ ein Zeichen eingelesen worden ist.
   Verwechseln Sie bitte die leere Zeichenkette "" nicht mit dem Leerzeichen " " (Zwischenraum)!

Beim folgenden Beispiel benutzen wir zum ersten Mal den für Spiele grundlegenden Begriff der S p i e l s t r a t e g i e .

Beispiel 4 : Ziel 100

Mit einer zufällig gewählten Zahl zwischen 1 und 30 beginnend addieren die Spieler - einer davon ist der Computer - abwechselnd eine natürliche Zahl zwischen 1 und 10 einschließlich. Wer zuerst das Ziel 100 erreicht, ist Sieger.

Ein typischer D i a l o g für dies Spiel sieht so aus:

```
Startzahl: 14

Wieviel addieren Sie? 6
Neuer Stand: 20

Ich addiere 3
Neuer Stand: 23

Wieviel addieren Sie? 5
Neuer Stand: 28

.....

Ich addiere 4
Neuer Stand: 89

Wieviel addieren Sie? 8
Neuer Stand: 97

Ich addiere 3
Neuer Stand: 100

Ich habe gewonnen.
Noch eine Partie?(j/n) n
Vielen Dank, es hat mich sehr gefreut.
```

Unsere Aufgabe besteht nun darin, den Rechner so mit Intelligenz auszustatten, daß er einen derartigen Dialog führen; mehr noch: daß er - sofern möglich - gewinnen kann.
Jede einzelne Partie des Spiels besteht aus einer Folge von S p i e l s t e l l u n g e n , nämlich den jeweils erreichten Zwischensummen (bei der oben gespielten Partie ist dies die Folge 14, 20, 23, 28, ..., 97, 100).

Diese Zahlen geben den Spielverlauf genau wieder. Warum hat nun der (menschliche) Spieler gegen den Computer verloren?
Schauen wir uns die letzte Spielstellung vor Erreichen des Ziels an: es ist die 97, und sie kennzeichnet eine G e w i n n s t e l l u n g , weil der nächste Zug (Addition von 3 durch den Computer) zum Gewinn der Partie führte. Jede der Zahlen 90, 91, ..., 99 ist für den, der am Zuge ist, eine Gewinnstellung; dagegen ist die 89 eine V e r l u s t s t e l l u n g , weil jeder Zug zu einer Gewinnstellung führt - und dann ist ja der Gegenspieler dran. Man erkennt mit dem gleichen Schluß die 78, 67, 56, 45, 34, 23, 12, 1 als Verluststellungen und alle übrigen Zahlen als Gewinnstellungen.
Wir werden den Computer so programmieren, daß er die genannten Verluststellungen immer anspielt, d.h. dem Gegner übergibt. Wenn der Benutzer dagegen eine Verluststellung an den Computer übergibt, so macht dieser einen Verlegenheitszug, indem er 1 addiert. Dieser Verhaltensplan des Computers heißt S p i e l s t r a t e g i e: zu jeder Spielstellung hat der den richtigen Zug parat.
Das BASIC-Programm findet sich auf der nächsten Seite. Es ist in die Teile

- Wahl der Startzahl
- Spielerzug
- Ziel erreicht?
- Computerzug
- Ziel erreicht?
- weitere Partie?

gegliedert.
In den Zeilen 520 - 540 durchläuft der Computer eine kleine Schleife, um die nächste 'Gewinnzahl' (so nennt er die Verluststellungen des Gegners) zu finden, nämlich die kleinste der Zahlen G, die größer oder gleich der Spielstellung S sind. RESTORE veranlaßt den Computer, die Datenzeile 810 mittels READ von Beginn an zu lesen.
Ist G = S, so macht er seinen Verlegenheitszug (Zeile 550), andernfalls addiert er mit Z eine Zahl derart, daß der neue Spielstand eine Verluststellung für den Gegner ist.

```
100 : PRINT "␣
110 : PRINT "            ZIEL HUNDERT
111 : PRINT "            ------------
112 :
120 REM EINFACHES NIM-AEHNLICHES SPIEL
121 :
130 : PRINT "BEI JEDEM ZUG IST EINE ZAHL ZWISCHEN 1 UND 10
140 : PRINT "ZUR BISHERIGEN GESAMTZAHL ZU ADDIEREN.
150 : PRINT "WER ZUERST 100 ERREICHT, HAT GEWONNEN.
190 :
200 REM === WAHL DER STARTZAHL ========================================
201 :
210 : LET S = INT(29*RND(1)+1)
230 : PRINT: PRINT "STARTZAHL:  "; S
290 :
300 REM === SPIELERZUG ================================================
301 :
320 : PRINT: INPUT "WIEVIEL ADDIEREN SIE "; Z
330 : IF Z < 1 OR Z > 10 THEN PRINT "FALSCHE EINGABE!": GOTO 320
340 : LET S = S+Z
350 : PRINT "NEUER STAND: "; S
390 :
400 REM === ZIEL ERREICHT? ============================================
401 :
410 : IF S >= 100 THEN PRINT: PRINT "SIE HABEN GEWONNEN!": GOTO 700
490 :
500 REM === COMPUTERZUG ===============================================
501 :
510 : REM --- ERMITTLUNG DER NAECHSTEN GEWINNZAHL
515 :
520 :   RESTORE
525 :   LET I = 0
530 :   LET I = I+1
535 :   READ G : REM EINLESEN DER GEWINNZAHL
540 :   IF G < S THEN 530
544 :
545 : REM --- WAHL DES COMPUTERS
546 :
550 :   IF G = S THEN LET Z = 1 : GOTO 570 : REM VERLEGENHEITSZUG
560 :   LET Z = G-S       : REM DIFFERENZ ZUR NAECHSTEN GEWINNZAHL
570 :   PRINT: PRINT "ICH ADDIERE "; Z
580 :   LET S = S+Z
590 :   PRINT "NEUER STAND: "; S
599 :
600 REM === ZIEL ERREICHT? ============================================
601 :
610 : IF S < 100 THEN 300
620 : PRINT: PRINT "SIE HABEN LEIDER VERLOREN.
690 :
700 REM === WEITERE PARTIE? ===========================================
701 :
710 : PRINT: PRINT "NOCH EINE PARTIE? (J/N)"
720 : GET A$: IF A$ = "" THEN 720
730 : IF A$ = "J" THEN PRINT "␣": GOTO 200
740 : PRINT
750 : PRINT "VIELEN DANK, ES HAT MICH SEHR GEFREUT."
790 :
800 REM === GEWINNZAHLEN ==============================================
801 :
810 DATA 1, 12, 23, 34, 45, 56, 67, 78, 89, 100
880 :
890 END
```

Alle bisher betrachteten Spiele liefen auf einen Dialog zwischen Mensch und Computer hinaus. Zum Abschluß dieses einführenden Kapitels wollen wir ein Spiel kennenlernen, das für mehrere Personen gedacht ist; Sie können es z.B. auf einer privaten Geselligkeit Ihren Gästen vorführen.

Beispiel 5 : Wer paßt zu wem?

Der Computer stellt - nicht immer ganz ernst zu nehmende - Fragen an die Gäste beiderlei Geschlechts. Anschließend gibt er für je zwei Personen (eine männlich, eine weiblich) Zahlen aus, welche die Übereinstimmung (auf Grund der Antworten) zwischen diesen Personen kennzeichnen.

Um dies schon recht umfangreiche Programm übersichtlich zu gestalten, wurde es in H a u p t p r o g r a m m und U n t e r p r o g r a m m e gegliedert, die vom Hauptprogramm aufgerufen werden, und zwar

1. Initialisierung
2. Namenseingabe
3. Frage und Antwort
4. Auswertung
5. Verabschiedung.

Der 1. Teil dimensioniert die Tabellen für Namen und Punkte (d.h. er veranlaßt den Rechner, Speicherplatz bereitzustellen) und liest die Fragen ein. Wenn Ihnen die in den Zeilen 3400 - 3490 stehenden Fragen nicht zusagen, können Sie leicht andere einfügen.
Im 2. Teil werden die Namen der Mitspieler und ihr Geschlecht eingegeben; dabei achtet der Computer darauf, daß gleich viele weibliche wie männliche Gäste teilnehmen (4500 und 4520).
Im 3. Teil werden die Fragen gestellt und beantwortet; dabei erscheinen die Antworten nicht auf dem Bildschirm (Eingabe über GET , 5600 - 5900).
Der 4. Teil dient der Auswertung der Antworten und der Bekanntgabe der Ergebnisse. Jeder Teilnehmer beantwortet die Fragen durch Eingabe der Zahlen 1 ('entschieden nein'), ... , 5 ('entschieden ja'); aus den Punktzahlen PM(i,k) = Antwortzahl des i-ten Herrn auf die k-te Frage
und PW(j,k) = Antwortzahl der j-ten Dame auf die k-te Frage wird in Zeile 6200 eine Differenz berechnet, und diese Differenzen werden addiert. Ideale Übereinstimmung zwischen Herr i und Dame j ist durch

die Punktsumme S = 0 gekennzeichnet, totale Nicht-Übereinstimmung durch die Punktsumme S = 40 (bei 10 Fragen). Dieser Programmteil sowie die Ergebnisausgabe sind noch verbesserungsfähig.

```
1000 : PRINT "□"
1010 : PRINT "         WER PASST ZU WEM?
1011 : PRINT "         ----------------
1012 :
1020 : PRINT
1030 : PRINT "   ICH, DER COMPUTER, STELLE DURCH
1040 : PRINT "   FRAGEN, DIE SIE BEANTWORTEN MOEGEN,
1050 : PRINT "   GEMEINSAMKEITEN ZWISCHEN DEN GAESTEN
1060 : PRINT "   FEST UND NENNE EINE BEWERTUNGSZAHL.
1070 : PRINT
1090 :
1095 :
1100 REM   VARIABLEN:
1110 :
1120 REM   M$(I) .....I-TER MAENNLICHER NAME
1130 REM   W$(J) .....J-TER WEIBLICHER  NAME
1140 REM   F$(K) .... K-TE FRAGE
1150 REM   PM(I,K) .. ANTWORTZAHL DES I-TEN HERRN AUF DIE K-TE FRAGE
1160 REM   PW(J,K) .. ANTWORTZAHL DER J-TEN DAME  AUF DIE K-TE FRAGE
1170 REM   N ........ MAXIMALZAHL DER MITSPIELENDEN PAARE
1900 :
1999 :
2000 REM *** HAUPTPROGRAMM *****************************************
2001 :
2300 : GOSUB 3000 : REM INITIALISIERUNG
2400 : GOSUB 4000 : REM NAMENSEINGABE
2500 : GOSUB 5000 : REM FRAGE UND ANTWORT
2600 : GOSUB 6000 : REM AUSWERTUNG
2700 : GOSUB 7000 : REM VERABSCHIEDUNG
2900 : END
2970 :
2980 REM *** ENDE DES HAUPTPROGRAMMS ********************************
2995 :
2999 :
3000 REM +++ UNTERPROGRAMM INITIALISIERUNG +++++++++++++++++++++++++
3001 :
3100 : LET N = 10 : REM MAXIMAL N PAARE KOENNEN MITSPIELEN
3150 :
3200 : DIM M$(N), W$(N), F$(10), PM(N,10), PW(N,10)
3300 :
3400 DATA BEEINFLUSSEN DIE GESTIRNE DEN CHARAKTER?
3410 DATA SOLLTEN FRAUEN DEN BERUF EINES PILOTEN AUSUEBEN?
3420 DATA ERZAEHLEN SIE GUTE WITZE WEITER?
3430 DATA SCHNARCHEN SIE?
3440 DATA SOLL DIE STARTBAHN WEST GEBAUT WERDEN?
3450 DATA IST RAUCHEN SCHAEDLICH?
3460 DATA LIEBEN SIE BRAHMS?
3470 DATA VERDIENEN AERZTE ZUVIEL?
3480 DATA SOLLTEN VERHEIRATETE GETRENNTEN URLAUB MACHEN?
3490 DATA GLAUBEN SIE AN EIN HOEHERES WESEN?
3495 :
3500 : FOR K = 1 TO 10 : READ F$(K) : NEXT K
3900 :
```

```
4000 REM +++ UNTERPROGRAMM NAMENSEINGABE ++++++++++++++++++++++++++++
4001 :
4010 : PRINT
4100 : PRINT "           WER SPIELT MIT?
4115 : PRINT
4200 : PRINT "   GEBEN SIE VORNAME UND GESCHLECHT EIN!
4210 : PRINT "   NACH DEM LETZTEN NAMEN 'ENDE'.
4215 :
4300 : LET S = 1 : LET T = 1
4305 : LET M = 0 : LET W = 0
4310 : PRINT
4320 : INPUT "VORNAME "; N$
4325 : IF N$ = "ENDE" THEN 4500
4330 : INPUT "GESCHLECHT (M/W) "; G$
4340 : IF G$ = "M" THEN M$(S) = N$ : M = M+1 : S = S+1 : GOTO 4310
4350 : IF G$ = "W" THEN W$(T) = N$ : W = W+1 : T = T+1 : GOTO 4310
4360 : GOTO 4330
4400 :
4500 : IF M < W THEN PRINT "ES FEHLEN NOCH ";W-M;" HERREN.": GOTO 4310
4520 : IF W < M THEN PRINT "ES FEHLEN NOCH ";M-W;" DAMEN." : GOTO 4310
4530 :
4540 : GOTO 2500
4999 :
5000 REM +++ UNTERPROGRAMM 'FRAGE UND ANTWORT' ++++++++++++++++++++
5001 :
5005 : PRINT "□"
5100 : FOR I = 1 TO M
5110 :   PRINT
5120 :   PRINT "■";M$(I)
5130 :   GOSUB 5500   : REM ANLEITUNG
5150 :   FOR K = 1 TO 10
5200 :     GOSUB 5600 : REM ANTWORT
5250 :     LET PM(I,K) = VAL(A$)
5300 :   NEXT K
5320 : NEXT I
5340 :
5350 : PRINT "□"
5360 : FOR J = 1 TO W
5365 :   PRINT
5370 :   PRINT "■";W$(J)
5375 :   GOSUB 5500   : REM ANLEITUNG
5380 :   FOR K = 1 TO 10
5400 :     GOSUB 5600 : REM ANTWORT
5450 :     LET PW(J,K) = VAL(A$)
5460 :   NEXT K
5470 : NEXT J
5480 :
5490 : RETURN : REM ENDE UNTERPROGRAMM 'FRAGE UND ANTWORT'
5499 :
5500 : REM +++ UNTERPROGRAMM ANLEITUNG +++
5501 :
5510 :   PRINT
5520 :   PRINT "SO ANTWORTEN SIE BITTE:
5525 :   PRINT "(IHRE ANTWORT ERSCHEINT NICHT "
5526 :   PRINT "AUF DEM BILDSCHIRM.)
5530 :   PRINT
5540 :   PRINT "ENTSCHIEDEN NEIN .....1
5545 :   PRINT "BEDINGT NEIN .....2
5550 :   PRINT "UNENTSCHIEDEN .....3
5555 :   PRINT "BEDINGT JA .....4
5560 :   PRINT "ENTSCHIEDEN JA .....5
5570 :   PRINT
5590 :   RETURN
```

```
5592 :
5595 : REM +++ UNTERPROGRAMM ANTWORT +++
5596 :
5600 :   PRINT F$(K) : REM FRAGE DES COMPUTERS
5610 :   GET A$ : IF A$ = "" THEN 5610
5620 :   IF A$="1" OR A$="2" OR A$="3" OR A$="4" OR A$="5" THEN RETURN
5670 :   PRINT
5680 :   PRINT "FALSCHE EINGABE!" : GOTO 5510
5900 :
6000 REM +++ UNTERPROGRAMM AUSWERTUNG ++++++++++++++++++++++++++++++++
6001 :
6002 : PRINT "□"
6003 : PRINT "          ERGEBNISSE
6004 : PRINT "          ----------
6005 :
6006 : PRINT
6007 :
6100 : PRINT
6120 : FOR I = 1 TO M
6150 :   FOR J = 1 TO W
6160 :     LET S = 0
6180 :     FOR K = 1 TO 10
6200 :       LET D = ABS(PM(I,K)-PW(J,K))
6250 :       LET S = S+D
6300 :     NEXT K
6310 :     PRINT "DER ABSTAND VON ";M$(I);" ZU ";
6320 :     PRINT W$(J);" BETRAEGT: "; S;" PUNKTE."
6340 :   NEXT J
6360 : NEXT I
6490 : PRINT : PRINT
6500 : PRINT "WENN SIE GENUG HABEN, DRUECKEN SIE EINE TASTE.
6550 : GET T$ : IF T$ = "" THEN 6550
6560 : RETURN
6900 :
7000 REM +++ UNTERPROGRAMM VERABSCHIEDUNG ++++++++++++++++++++++++++
7001 :
7100 : PRINT "□"
7200 : PRINT : PRINT
7300 : PRINT "   VIELEN DANK FUER DIE
7400 : PRINT " INTERESSANTEN EINBLICKE!
7450 : PRINT
7500 : PRINT "      AUF WIEDERSEHEN.
7600 : PRINT : PRINT
7700 : FOR I = 1 TO 1000 : NEXT I
7800 : PRINT "□"
7900 : RETURN
```

Aufgaben

1. Das Programm 'Additionstrainer'soll so erweitert werden, daß nach drei falschen Antworten das richtige Ergebnis genannt und mit der nächsten Aufgabe fortgefahren wird.

2. Eine andere Variante des Programms 'Additionstrainer': am Schluß der Aufgabenserie wird die Anzahl der richtigen Antworten genannt und ein Kommentar (bzw. eine Zensur) gegeben.

3. Bauen Sie in den Additionstrainer eine Zeitschranke ein: wenn sie überschritten ist, gilt die Aufgabe als nicht gelöst.

4. Die Zeitschranke kann auch für die gesamte Aufgabenserie gelten: es zählen am Schluß nur die innerhalb der gesetzten Frist richtig gelösten Aufgaben.

5. Erweitern Sie das Programm 'Additionstrainer', indem Sie zu Beginn die Wahl unterschiedlicher Schwierigkeitsgrade ermöglichen, und zwar einmal durch die Größe der Summanden, zum anderen durch Zeitschranken.

6. Entwerfen Sie einen Subtraktionstrainer, einen Multiplikations- und einen Divisionstrainer!

7. Versehen Sie das Programm 'Zu groß - zu klein' mit einer Eingabekontrolle, d.h. bei Eingabe einer Zahl, die kleiner als 1 oder grösser als die obere Grenze ist, soll der Computer auf diesen Tatbestand hinweisen und die Eingabe zurückweisen.

8. Programmieren Sie ein Zahlenrate-Spiel 'Wärmer-kälter', welches dem Ratenden die Entfernung zur unbekannten Zahl entweder unmittelbar oder verschlüsselt, z.B. durch eine entsprechende Anzahl Zeichen (etwa *) mitteilt.

9. Der Benutzer soll in Beispiel 3 nicht beliebig dumm raten dürfen: nach einer gewissen Anzahl von Rateversuchen bricht der Rechner das Spiel ab. Welche Anzahl ist angemessen?

10. Die Gewinnzahlen (Verluststellungen für den Spielgegner) im Programm 'Ziel 100' lassen sich auch durch eine mathematische Formel erzeugen, nämlich $g = 11 \cdot k + 1$ für $k = 0,\ldots,9$. Benutzen Sie diese Formel statt der DATA-Zeile 810.

11. Schreiben Sie ein Programm, das den Computer nicht zum Gegenspieler, sondern zum Spielmeister für 'Ziel 100' macht: er läßt sich die Namen der Spieler nennen, ruft denjenigen auf, der an der Reihe ist, prüft die Eingabe (die eingegebene Zahl darf nicht kleiner als 1 und nicht größer als 10 sein) und ruft den Sieger aus.

12. Ändern Sie das Programm 'Ziel 100' so ab, daß mehrere Partien gespielt werden können und der Computer am Ende einer Serie die Anzahl der gewonnenen bzw. verlorenen Partien angibt ("Es steht 5:3 für ... Sie").

13. Schreiben Sie das Programm 'Ziel 100' für andere Bereiche der Summanden (d.h. nicht von 1 bis 10) und andere Ziele ("Ziel 152"). Lassen Sie Ziel und Bereich zu Beginn wählen oder zufällig festlegen.

14. Die Ergebnisanzeige im Programm 'Wer paßt zu wem?' vermag noch nicht zu befriedigen. Ändern Sie sie so ab, daß eine Tabelle etwa folgender Art auf dem Bildschirm erscheint:

	A	B	C	D
U	0	12	18	24
X	*	0	21	8
Y	*	*	0	38
Z	*	*	*	0

15. Eine weitere Verbesserung von Programms 'Wer paßt zu wem?': Der Computer soll selbständig diejenigen Personen nennen, die - gemäß den gegebenen Antworten - am besten zueinander passen, d.h. die niedrigsten Entfernungszahlen aufweisen. In der obigen Tabelle paßt Herr D am besten zu Dame X, Herr B paßt am besten zu Dame U u.s.w.

16. Auch das Bewertungsschema können Sie ändern - etwa indem Sie die Fragen gewichten. Lassen Sie Ihre Phantasie spielen!

2
Lernspiele

Ludendo discimus
LEIBNIZ

Die Spiele dieses Kapitels mag mancher als eher ernsthaft und nützlich, also als 'unspielerisch' ansehen. Wer nach den Programmen lernen muß, wird sich vielleicht wenig freuen - dafür erfreut das Schreiben der Programme umso mehr. Vielleicht können Sie als Programmautor durch aufmunternde Zwischenbemerkungen die saure Arbeit des Lernens etwas erleichtern.

Beispiel 1: Alphabet

Ein Programm soll entworfen werden, das Kindern das Aufsagen des Alphabets beibringt.

Dialog:

```
Gib bitte der Reihe nach die Buchstaben
des Alphabets ein!

a b c d e f g h i k

Dieser Buchstabe war leider falsch. Noch einmal von vorn!
Gib bitte der Reihe nach die Buchstaben
des Alphabets ein!

a b c d e f g h i j k l m n o p q r s t u v w x y z

Das hast du gut gemacht. Auf Wiedersehen.
```

```
100 PRINT "□"
110 PRINT "                ALPHABET"
111 PRINT "                --------"
112 :
120 REM SOLL KINDERN AB DREI JAHREN
121 REM DAS AUFSAGEN DES ALPHABETS BEIBRINGEN
129 :
130 PRINT
140 PRINT "GIB BITTE DER REIHE NACH DIE BUCHSTABEN DES ALPHABETS EIN!
200 PRINT
205 :
210 FOR N = 65 TO 90 : REM ASCII- NUMMERN DER BUCHSTABEN
220 :
230 : GET B$ : IF B$ = "" THEN 230 : REM BUCHSTABENEINGABE
250 : PRINT B$;"  ";
260 :
270 : IF B$ = CHR$(N) THEN 340 : REM RICHTIG, ZUM NAECHSTEN BUCHSTABEN
280 : PRINT : PRINT
290 : PRINT "DER LETZTE BUCHSTABE WAR LEIDER FALSCH.";
300 : PRINT " NOCH EINMAL VON VORN!
310 : PRINT : PRINT
320 : GOTO 140 : REM SPRUNG ZUM ANFANG DER SCHLEIFE
330 :
340 NEXT N
350 :
360 PRINT : PRINT
370 PRINT"DAS HAST DU GUT GEMACHT. AUF WIEDERSEHEN.
380 :
390 END
```

Das obige Programm ist sehr einfach. Die Zählvariable N in Zeile 210 durchläuft die Zahlen 65 bis 90: das sind die Nummern der Buchstaben von A bis Z im sogenannten ASCII (American Standard Code for Information Interchange), der im Handbuch Ihres Rechners aufgeführt ist.
In Zeile 270 wird abgefragt, ob der eingegebene Buchstabe B$ gleich dem vom Computer vorgegebenen Buchstaben ist. CHR$(N) ist das zur Zahl N gehörige Zeichen; z.B. gilt CHR$(65) = "A" und CHR$(90) = "Z".
Wurde ein falscher Buchstabe eingegeben, so beginnt das Programm von vorne; es entläßt den Benutzer nur, wenn dieser das Alphabet fehlerlos eingetippt hat. -

*

Nun folgt ein Spiel zum Gedächtnis- und Konzentrationstraining.

Beispiel 2 : Zahlen-Senso

Der Computer bietet eine Folge zufällig gewählter Ziffern an, der Spieler soll sie wiederholen. Nach jeder erfolgreichen Wiederholung wird die Folge um eine Ziffer länger ...

D i a l o g :

```
Ziffernfolge jetzt: 4
Geben Sie die Folge wieder! 4

Ziffernfolge jetzt: 4 7
Geben Sie die Folge wieder! 4 7

Ziffernfolge jetzt: 4 7 1
Geben Sie die Folge wieder! 4 7 2

Leider falsch. Die Ziffernfolge lautet richtig:
4 7 1
```

Das Verfahren lautet:

ZAHLEN-SENSO

```
WIEDERHOLE
  erzeuge und speichere zufällige ziffer
  gib die bisherige ziffernfolge aus
  lies die ziffernfolge des spielers ein
BIS  falsche ziffer eingegeben wird
ENDE-WIEDERHOLE
```

Im 1. Teil des Programms auf der folgenden Seite wird eine Dezimalziffer zufällig erzeugt (Zeile 240) und in einer Tabelle gespeichert.
Im 2. Teil werden alle bis dahin gespeicherten Ziffern kurz auf dem Bildschirm gezeigt (die Zeitspanne ist durch die Warteschleife in Zeile 380 festgelegt). Jetzt hat der Benutzer die Ziffernfolge im 3. Teil wiederzugeben. Tippt er eine falsche Ziffer ein, geht das Programm in den

4. Teil (Abbruch des Versuchs) über, andernfalls springt es zum ersten Teil zurück. Es lautet:

```
100 : PRINT "■"
110 : PRINT "                    ZAHLEN-SENSO
111 : PRINT "                    ------------
112 :
120 REM SPIEL ZUM GEDAECHTNIS- UND KONZENTRATIONSTRAINING
130 :
140 : DIM A(100) : REM TABELLE ZUR AUFNAHME DER ZIFFERN
190 :
200 REM === ERZEUGEN DER ZIFFERN ====================================
201 :
210 : PRINT
220 : LET N = 1                  : REM ZIFFERNZAEHLER
225 : PRINT : PRINT
230 : PRINT N;". VERSUCH: "
240 : LET Z = INT(9*RND(TI)+1) : REM ZUFALLSZIFFER
250 : LET A(N) = Z               : REM ABSPEICHERN DER ZIFFER
290 :
300 REM === PRAESENTATION DER ZIFFERNFOLGE ===========================
301 :
310 : PRINT
320 : PRINT "ZIFFERNFOLGE BIS JETZT: ";
340 : FOR I = 1 TO N : PRINT "■"; A(I);: NEXT I
370 :
380 : FOR J = 1 TO 500 : NEXT J : REM WARTESCHLEIFE
390 : PRINT "■"
399 :
400 REM === WIEDERHOLUNG AUS DEM GEDAECHTNIS =========================
401 :
410 : PRINT
420 : PRINT "GEBEN SIE JETZT DIE ZIFFERNFOLGE EIN: "
430 : FOR I = 1 TO N
440 :   GET R$ : IF R$ = "" THEN 440        : REM EINGABE
460 :   IF R$ < "1" OR R$ > "9" THEN 440    : REM EINGABEPRUEFUNG
470 :   LET R = VAL(R$)                     : REM UMWANDLUNG IN ZAHL
480 :   PRINT R;
490 :   IF R <> A(I) THEN 600               : REM FALSCH ERINNERT
500 : NEXT I
510 :
520 : PRINT : PRINT "RICHTIG GERATEN!"
530 :
540 : LET N = N+1 : GOTO 225               : REM RUECKSPRUNG
590 :
600 REM === ABBRUCH ==================================================
601 :
610 : PRINT : PRINT
620 : PRINT "LEIDER FALSCH. DIE ZIFFERNFOLGE LAUTET RICHTIG:
630 : PRINT
640 : FOR I = 1 TO N : PRINT A(I);: NEXT I
650 :
660 : PRINT : PRINT
690 :
699 : END
```

Beispiel 3 : Begriffe merken

Auf dem Bildschirm erscheinen kurze Zeit an zufällig ausgewählten Stellen Worte (Namen, Begriffe). Der Spieler soll möglichst viele davon wiedergeben (eintippen).

Ein möglicher D i a l o g :

```
Meise

                          Amsel

          Elster
              Drossel
```

Kurze Zeit später:

```
Geben Sie nun die Begriffe wieder!

Elster    Richtig!
Vogel     Leider falsch!
Meise     Richtig

Dies waren die angebotenen Begriffe:
Amsel  Meise  Drossel  Elster

Und diese haben Sie behalten:
Meise  Elster
```

Das Programm gliedert sich in die fünf Teile Initialisierung, Spielregeln, Anbieten der Begriffe, Wiedergabe der Begriffe und Auswertung. Im ersten Teil wird eine Tabelle B$ für 100 Begriffe bereitgestellt und mit den in den DATA-Zeilen ab 700 gespeicherten Worten gefüllt. Dann gibt der Rechner (auf Wunsch) die Spielregeln wieder. Im dritten Teil

sucht er sich zunächst eine Zufallszahl K zwischen 1 und N (der Anzahl eingelesener Begriffe) und bestimmt den zugehörigen Begriff (Zeile 435). Dann legt er mit X und Y eine zufällige Stelle auf dem Bildschirm fest und steuert den Cursor dorthin. (Die in Zeile 460, 465 vorkommenden Cursorbewegungen werden wir im folgenden Kapitel ausführlich studieren.) Schließlich druckt der das vorher bestimmte Wort an diese Stelle. Das ganze wird D-mal wiederholt (im Programm ist die Zahl D = 8 , d.h. es werden 8 Begriffe präsentiert). Im vierten Programmteil gibt der Spieler die Worte ein, die er sich gemerkt hat. Zum Schluß nennt der Computer alle von ihm zuvor ausgegebenen Worte und die, an die sich der Spieler erfolgreich erinnert hat.

Das Programm lautet:

```
100 : PRINT "■"
110 : PRINT "          BEGRIFFE MERKEN
111 : PRINT "          ---------------
112 :
120 REM SPIEL ZUM AUFFASSUNGS- UND GEDAECHTNISTRAINING
129 :
200 REM === INITIALISIERUNG ================================================
201 :
210 : DIM B$(100)              : REM TABELLE DER BEGRIFFE
220 : LET D = 8                : REM ANZAHL DER ANGEBOTENEN BEGRIFFE
230 : LET T = 200              : REM ZEITKONSTANTE
240 :
250 : FOR I = 1 TO 100
255 :   READ B$
260 :   IF B$ = "@" THEN 280 : REM ALLE BEGRIFFE SIND EINGELESEN
265 :   LET B$(I) = B$
270 : NEXT I
280 : LET N = I-1              : REM ANZAHL ALLER BEGRIFFE
290 :
300 REM === SPIELREGELN ====================================================
301 :
310 : PRINT
320 : PRINT "MOECHTEN SIE DIE SPIELREGELN KENNENLERNEN?(J/N)
330 : GET A$ : IF A$ = "" THEN 330
340 : IF A$ <> "J" THEN 400 : REM UEBERSPRINGEN DER SPIELREGELN
345 :
350 : PRINT : PRINT
360 : PRINT "AUF DEM BILDSCHIRM ERSCHEINEN KURZE ZEIT EINIGE WOERTER
361 : PRINT "ODER BEGRIFFE. SIE SOLLEN VON DIESEN ANSCHLIESSEND
362 : PRINT "SO VIELE WIEDER EINGEBEN, WIE SIE KOENNEN.
365 :
370 : PRINT : PRINT
380 : PRINT "TASTE DRUECKEN!
385 : GET T$ : IF T$ = "" THEN 385
390 :
```

```
400 REM === ANBIETEN DER BEGRIFFE =====================================
401  :
410  : PRINT "▣"
420  : FOR J = 1 TO D
430  :   LET K = INT(N*RND(TI)+1)              : REM NUMMER DES BEGRIFFS
435  :   LET M$(J) = B$(K)                    : REM BEGRIFF MERKEN
440  :   LET X = INT(68*RND(TI)+1)            : REM SPALTE
445  :   LET Y = INT(24*RND(TI)+1)            : REM ZEILE
450  :   PRINT "▣"                            : REM CURSOR NACH HAUSE
455  :   PRINT "□□";                          : REM CURSOR NACH OBEN
460  :   FOR R = 1 TO X : PRINT "■";: NEXT R  : REM CURSOR NACH RECHTS
465  :   FOR R = 1 TO Y : PRINT "▣";: NEXT R  : REM CURSOR NACH UNTEN
470  :   PRINT M$(J)                          : REM BEGRIFF DRUCKEN
480  : NEXT J
485  :
490  : FOR R = 1 TO T : NEXT R : PRINT "▣"    : REM WARTESCHLEIFE
495  :
500 REM === WIEDERGABE DER BEGRIFFE ===================================
501  :
510  : PRINT
520  : PRINT "GEBEN SIE NUN DIE BEGRIFFE WIEDER!
525  : PRINT "WENN SIE ALLE EINGEGEBEN HABEN, ";
526  : PRINT "DRUECKEN SIE '@'!
529  : PRINT
530  :
540  : FOR J = 1 TO D
550  :   INPUT B$
555  :   IF B$ = "@" THEN 600       : REM ALLE BEGRIFFE EINGEGEBEN
559  :   PRINT "■■■■■■■■■■■■■■■□"; : REM CURSOR NACH RECHTS UND OBEN
560  :   FOR L = 1 TO D
570  :     IF B$ = M$(L) THEN PRINT "RICHTIG!" : N$(L) = B$ : GOTO 590
580  :   NEXT L
585  :   PRINT "LEIDER FALSCH!
590  : NEXT J
599  :
600 REM === AUSWERTUNG ================================================
601  :
610  : PRINT : PRINT : PRINT
620  : PRINT "DIES WAREN DIE ANGEBOTENEN BEGRIFFE:
625  : PRINT
630  : FOR J = 1 TO D : PRINT M$(J);" ";: NEXT J
640  : PRINT : PRINT : PRINT
650  : PRINT "UND DIESE HABEN SIE BEHALTEN:
660  : PRINT
670  : FOR J = 1 TO D : PRINT N$(J);" ";: NEXT J
690  :
700 DATA HUND, KATZE, MAUS, ESEL, PFERD, KUH, SCHWALBE, RATTE, WURM
701 DATA ZEBRA, SCHWEIN, ENTE, SCHWAN, KUECKEN, BAUM, STRAUCH, ULME
702 DATA BUCHE, EICHE, TANNE, FICHTE, REH, HIRSCH, ELCH, SCHAF, BOCK
703 DATA FINK, STAR, MEISE, AMSEL, DROSSEL, MUECKE, SCHABE, ASSEL
704 DATA HIER, KOENNEN, SIE, NOCH, WEITERE, 65, UNTERBRINGEN,
705 DATA ALS, LETZTES, ZEICHEN, IMMER, DAS, AFFEN-A, @
```

*

Auch das folgende Spiel dient der Schulung der Konzentrationsfähigkeit und des Gedächtnisses.

Beispiel 4 : Karten wenden

Ein Kartenspiel wird gemischt, und die Karten werden verdeckt auf dem Tisch ausgebreitet. Der Spieler, welcher am Zug ist, deckt zwei Karten auf; stimmen diese im Wert (Zahl oder Bild) überein, darf er die Karten wegnehmen und zwei weitere Karten aufdecken. Andernfalls werden die Karten verdeckt zurückgelegt, und der nächste Spieler ist am Zug. Wer am Schluß die meisten Karten beisammen hat, ist Sieger. Die Schwierigkeit und der Reiz des Spiels bestehen darin, sich die Lage und den Wert der aufge- und wieder verdeckten Karten zu merken (weshalb das Spiel im englischen Sprachbereich auch 'Memory' genannt wird). Es ist auf dem Computer zu simulieren.

Eine P a r t i e kann sich auf dem Bildschirm wie folgt abspielen:

```
Spieler HUGO ist dran! Welche Karten decken Sie auf?

    1  2  3  4  5  6  7  8  9  10 11 12
1   X  X  X  X  X  X  X  X  X  X  X  X
2   X  X  X  X  X  X  X  X  X  X  X  X
3   X  X  X  X  X  X  X  X  X  X  X  X
4   X  X  X  X  X  X  X  X  X  X  X  X

Erste Karte? 1,2
Zweite Karte? 4,7

    1  2  3  4  5  6  7  8  9  10 11 12
1   X  6♥ X  X  X  X  X  X  X  X  X  X
2   X  X  X  X  X  X  X  X  X  X  X  X
3   X  X  X  X  X  X  X  X  X  X  X  X
4   X  X  X  X  X  X  6♦ X  X  X  X  X

Treffer!
```

```
100 : PRINT "□"
110 : PRINT "            KARTEN WENDEN
111 : PRINT "            ~~~~~~~~~~~~
119 :
120 REM KONZENTRATIONS- UND GEDAECHTNISSPIEL
129 :
199 :
200 REM *** HAUPTPROGRAMM ********************************************
201 :
210 : GOSUB 400 : REM SPIELREGELN
220 : GOSUB 500 : REM INITIALISIERUNG
230 : LET S = 0 : REM SPIELERNUMMER
235 :
240 : FOR K = 1 TO 24
241 :
245 :   PRINT "□"
250 :   PRINT : PRINT "SPIELER "; "▣"; N$(S);"■";" IST JETZT DRAN!";
260 :   PRINT " WELCHE KARTEN DECKEN SIE AUF?
265 :   LET X=0 : LET Y=0 : LET I=0 : LET V=0
269 :
270 :   GOSUB 800 : REM SPIELFELDAUSGABE
280 :   GOSUB 700 : REM EINGABE DER KARTEN
290 :   GOSUB 800 : REM SPIELFELDAUSGABE
300 :   GOSUB 900 : REM TREFFERPRUEFUNG
310 :
320 :   FOR W = 1 TO 1500 : NEXT W : REM WARTESCHLEIFE
330 :
340 :   IF T$ <> "TREFFER" THEN LET S = 1-S : GOTO 245
350 :
360 : NEXT K
370 :
380 : END
390 :
397 REM *** ENDE DES HAUPTPROGRAMMS ***********************************
398 :
399 :
400 REM +++ UNTERPROGRAMM SPIELREGELN +++++++++++++++++++++++++++++++++
401 :
410 : PRINT
415 : PRINT "SOLL ICH IHNEN DIE SPIELREGELN NENNEN?(J/N)
420 : GET A$ : IF A$ = "" THEN 420
425 : IF A$ <> "J" THEN RETURN
430 :
440 : REM --- HIER SOLLTEN DIE SPIELREGELN STEHEN ---
450 :
490 : RETURN
499 :
500 REM +++ UNTERPROGRAMM INITIALISIERUNG ++++++++++++++++++++++++++++
501 :
510 :   PRINT : PRINT "WER SPIELT MIT?
515 :   PRINT
520 :   INPUT "ERSTER  SPIELER "; N$(0)
530 :   INPUT "ZWEITER SPIELER "; N$(1)
531 :
532 :   LET T(0) = 0 : LET T(1) = 0 : REM TREFFERZAEHLER
535 :
540 :   PRINT : PRINT "BITTE ETWAS GEDULD! ";
541 :   PRINT "ICH MISCHE DIE KARTEN.
545 :
```

```
550 : REM --- SPIELKARTEN BEREITSTELLEN ---
551 :
560 :   DIM F$(4,12)
565 :
570 :   FOR I = 1 TO 4
575 :     FOR J = 1 TO 12 : READ F$(I,J) : NEXT J  : REM EINLESEN
580 :   NEXT I
590 :
600 : REM --- KARTEN MISCHEN ---
601 :
610 :   FOR I = 1 TO 4
620 :     FOR J = 1 TO 12
630 :       LET X = INT(4*RND(TI)+1)
635 :       LET Y = INT(12*RND(TI)+1)
640 :       LET H$ = F$(I,J)
645 :       LET F$(I,J) = F$(X,Y)
650 :       LET F$(X,Y) = H$
660 :     NEXT J
670 :   NEXT I
675 :
680 DATA "2♠","3♠","4♠","5♠","6♠","7♠","8♠","9♠","B♠","D♠","K♠","A♠"
681 DATA "2♥","3♥","4♥","5♥","6♥","7♥","8♥","9♥","B♥","D♥","K♥","A♥"
682 DATA "2♣","3♣","4♣","5♣","6♣","7♣","8♣","9♣","B♣","D♣","K♣","A♣"
683 DATA "2♦","3♦","4♦","5♦","6♦","7♦","8♦","9♦","B♦","D♦","K♦","A♦"
685 :
690 : RETURN
699 :
700 REM +++ UNTERPROGRAMM EINGABE ++++++++++++++++++++++++++++++++++++
701 :
705 : PRINT
710 : INPUT "ERSTE  KARTE "; X,Y
715 : IF X < 0 OR X > 4 OR Y < 0 OR Y > 12 THEN 710
720 : IF F$(X,Y) = " " THEN PRINT "KARTE IST SCHON WEG!": GOTO 710
725 :
730 : INPUT "ZWEITE KARTE "; U,V
735 : IF U < 0 OR U > 4 OR V < 0 OR V > 12 THEN 730
740 : IF U = X AND V = Y THEN PRINT "DIE GLEICHE?": GOTO 730
745 : IF F$(U,V) = " " THEN PRINT "KARTE IST SCHON WEG!": GOTO 730
760 :
790 : RETURN
799 :
800 REM +++ UNTERPROGRAMM SPIELFELD AUSGEBEN ++++++++++++++++++++++++++
801 :
803 : PRINT : PRINT
805 : FOR J = 1 TO 12 : PRINT TAB(4*J+6) J;    : NEXT J : PRINT
806 : PRINT
809 :
810 : FOR I = 1 TO 4
815 :   PRINT TAB(4) I;
820 :   FOR J = 1 TO 12
830 :     IF I = X AND J = Y      THEN 855
835 :     IF I = U AND J = V      THEN 855
840 :     IF F$(I,J) = " "        THEN 855
850 :     PRINT TAB(4*J+7) "▒";: GOTO 860
855 :     PRINT TAB(4*J+7) F$(I,J);
860 :   NEXT J
865 :   PRINT
870 : NEXT I
880 :
890 : RETURN
899 :
```

```
900 REM +++ UNTERPROGRAMM TREFFERPRUEFUNG ++++++++++++++++++++++++++++
901 :
910 : IF LEFT$(F$(X,Y),1) = LEFT$(F$(U,V),1) THEN 930
911 :
915 : PRINT
920 : PRINT "LEIDER DANEBEN!" : LET T$ = "N" : GOTO 960
925 :
930 : PRINT
935 : PRINT "TREFFER! " : LET T$ = "TREFFER"
940 : LET F$(X,Y) = " " : LET F$(U,V) = " " : REM KARTEN ENTFERNEN
945 : LET T(S) = T(S)+1                     : REM TREFFER SUMMIEREN
950 :
960 : PRINT : PRINT "SPIELSTAND: ";
970 : PRINT "■"; N$(0);":"; T(0);
975 : PRINT "■"; N$(1);":"; T(1)
990 :
999 : RETURN
```

Erläuterungen:

Das Programm ist für zwei Spieler geschrieben. Das Abwechseln geschieht mit Hilfe der Variablen S, welche die Werte 0 und 1 annehmen kann. Zu Beginn ist S = 0 (Zeile 230) und damit ist der Spieler mit dem Namen N$(0) am Zug (Zeile 250). Wenn er zwei nicht übereinstimmende Karten erwischt hat, setzt das Programm S := 1 - S, d.h. jetzt gilt S = 1 (Zeile 340) und N$(1) ist am Zug (Zeile 250). Wurde ein Treffer erzielt, so geht das Programm zum nächsten Wert von K über (Zeile 360); da 24 Kartenpaare vorhanden sind, zählt die Variable K von 1 bis 24 (Zeile 240). Die zueinander passenden Karten werden nach dem Aufdekken aus dem Spiel genommen (Zeile 940), d.h. nicht mehr angezeigt (840).

*

Nun kommt ein Programm, mit dessen Hilfe man ernsthaft lernen kann - und zwar das Rechnen mit ganzen Zahlen. Es handelt sich um eine erweiterte Fassung von Beispiel 2 (Kapitel 1).

Beispiel 5 : Rechentrainer

Der Computer legt - in buntem Wechsel - Aufgaben zu den vier Grundrechenarten mit ganzen Zahlen zwischen 10 und 100 vor. Der Benutzer soll sie lösen.

```
1000 : PRINT "□"
1010 : PRINT "        RECHENTRAINER
1011 : PRINT "        -------------
1012 :
1020 REM TRAINIERT DIE VIER GRUNDRECHENARTEN
1021 :
1100 REM *** HAUPTPROGRAMM ******************************************
1101 :
1110 : PRINT
1120 : INPUT "WIE VIELE AUFGABEN "; N
1130 : PRINT
1140 : PRINT "SCHWIERIGKEITSGRAD L(EICHT, M(ITTEL, S(CHWER ?"
1150 : GET A$ : IF A$ = "" THEN 1150
1160 :
1170 : IF A$ = "L" THEN LET W = 10 : GOTO 1220  : REM SEKUNDEN
1180 : IF A$ = "M" THEN LET W =  6 : GOTO 1220  : REM PRO
1190 : IF A$ = "S" THEN LET W =  3 : GOTO 1220  : REM AUFGABE
1200 : GOTO 1150                                : REM EINGABEFEHLER
1210 :
1220 : LET L = N*W*60                           : REM ZEITLIMIT
1230 :
1240 : LET T = TI                               : REM STELLEN DER UHR
1250 :
1260 : LET R = 0                                : REM ANZAHL RICHTIGER
1290 :
1300 : DEF FNR(X) = INT(8*RND(X)+2)             : REM ZUFALLSFUNKTION
1310 : DEF FNS(X) = INT(89*RND(X)+11)           : REM ZUFALLSFUNKTION
1390 :
1400 : REM --- SCHLEIFENBEGINN ------------------------------------
1401 :
1410 :   FOR I = 1 TO N
1420 :
1430 :   REM --- WAHL DER RECHENART ---
1431 :
1440 :     LET K = INT(4*RND(1)+1)
1450 :     ON K GOSUB 2000, 3000, 4000, 5000
1490 :
1500 :   REM --- FRAGE UND ANTWORT -----
1501 :
1510 :     PRINT : PRINT "WIEVIEL IST "; Z1;OP$;Z2;" ";
1520 :     INPUT A
1530 :
1540 :     IF TI-T >= L THEN PRINT:PRINT "ZEIT ABGELAUFEN!": GOTO 1600
1550 :
1560 :     IF A = E THEN PRINT"RICHTIG!": LET R = R+1 : GOTO 1590
1570 :       PRINT "LEIDER FALSCH"
1580 :
1590 :   NEXT I
1595 :
1598 : REM --- SCHLEIFENENDE -------------------------------------
1599 :
```

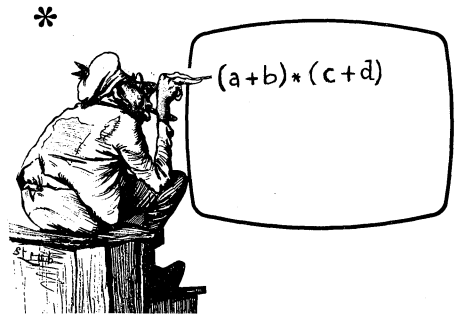
```
1600 : REM --- AUSWERTUNG ------------------------------------------------
1601 :
1610 :    LET Q = INT(1000*R/N+0.5)/1000
1620 :    PRINT : PRINT
1630 :    PRINT "SIE ERREICHTEN UNTER ";N;" AUFGABEN ";R;" RICHTIGE, ";
1640 :    PRINT "DAS SIND ";100*Q;" PROZENT."
1690 :
1700 : REM --- WIEDERHOLUNG ODER VERABSCHIEDUNG ----------------------------
1701 :
1710 :    PRINT : PRINT : PRINT
1720 :    PRINT "NOCH EINE SERIE?(J/N)"
1730 :    GET A$ : IF A$ = "" THEN 1730
1740 :    IF A$ = "J" THEN RUN    : REM SPRUNG ZUM ANFANG
1750 :    PRINT : PRINT : PRINT "AUF WIEDERSEHEN."
1760 :
1790 : END
1799 :
1900 REM *** ENDE HAUPTPROGRAMM ************************************
1999 :
2000 REM +++ UNTERPROGRAMM ADDITION ++++++++++++++++++++++++++++++++
2001 :
2010 : LET OP$ = "+"
2020 : LET Z1 = FNS(1)
2030 : LET Z2 = FNS(1)      : REM WAHL DER SUMMANDEN
2040 : LET E = Z1+Z2        : REM ERGEBNIS
2080 :
2090 : RETURN
2900 :
3000 REM +++ UNTERPROGRAMM SUBTRAKTION +++++++++++++++++++++++++++++++
3001 :
3010 : LET OP$ = "-"
3020 : LET Z1 = FNS(1)
3030 : LET Z2 = FNS(2)      : REM WAHL DER OPERANDEN
3040 : LET E = Z1-Z2        : REM ERGEBNIS
3080 :
3090 : RETURN
3900 :
4000 REM +++ UNTERPROGRAMM MULTIPLIKATION ++++++++++++++++++++++++++++
4001 :
4010 : LET OP$ = "*"
4020 : LET Z1 = FNR(1)      : REM ERSTER  FAKTOR EINSTELLIG
4030 : LET Z2 = FNS(1)      : REM ZWEITER FAKTOR ZWEISTELLIG
4040 : LET E = Z1*Z2        : REM ERGEBNIS
4080 :
4090 : RETURN
4900 :
5000 REM +++ UNTERPROGRAMM DIVISION ++++++++++++++++++++++++++++++++++
5001 :
5010 : LET OP$ = "/"
5020 : LET Z2 = FNR(1)      : REM EINSTELLIGER  DIVISOR
5030 : LET  E = FNS(2)      : REM ZWEISTELLIGER QUOTIENT
5040 : LET Z1 = E*Z2        : REM DIVIDEND
5050 :
5090 : RETURN
5095 :
5900 : END
```

Erläuterungen zum Programm 'Rechentrainer':

Die Rechenart wird in den Zeilen 1440 und 1450 gewählt: zuerst erzeugt der Computer eine der Zahlen 1,2,3,4 zufällig; in Abhängigkeit von dieser Zahl springt er in das Unterprogramm für Addition, Subtraktion, Multiplikation oder Division.
Die Wahl des Schwierigkeitsgrades geschieht in den Zeilen 1170 bis 1190: L = N*W*60 ist die zur Verfügung stehende Zeitspanne (in Sekunden) zum Lösen der N Aufgaben. Die Uhr wird in Zeile 1240 gestellt:
TI (von engl. time = Zeit) ist eine Variable, welche die seit Einschalten des Rechners verflossene Zeit in 60stel Sekunden enthält. In Zeile 1540 wird geprüft, ob das Zeitlimit L überschritten ist.

Beispiel 6: Termumformung

Ein algebraischer Term wird eingegeben und dann umgeformt; der Computer sagt, ob die Umformung korrekt, d.h. ob der neue Term zum gegebenen äquivalent ist.



Dialog:

```
Geben Sie den Term ein!
(a+b)↑2

Äquivalenter Term ?
a↑2 + 2*a*b + b↑2

Richtig!

Äquivalenter Term ?
@

Auf Wiedersehen.
```

```
100 : PRINT "■"
110 : PRINT "       TERMUMFORMUNG
111 : PRINT "       -------------
112 :
120 REM LERNPROGRAMM FUER DEN ALGEBRA-UNTERRICHT (KLASSE 7/8)
129 :
130 : PRINT
135 : PRINT "SIE KOENNEN EINEN TERM, GEBILDET AUS ZAHLEN,
136 : PRINT "BUCHSTABEN, DEN RECHENZEICHEN +,-,*,/,↑ UND
137 : PRINT "KLAMMERN EINGEBEN UND UMFORMEN.
138 : PRINT "ICH SAGE IHNEN DANN, OB DER UMGEFORMTE TERM
139 : PRINT "ZUM GEGEBENEN AEQUIVALENT IST.
140 : PRINT "WENN SIE GENUG HABEN, DRUECKEN SIE @.
190 :
200 REM *** HAUPTPROGRAMM *********************************************
201 :
210 : GOSUB 400 : REM BELEGUNG DER VARIABLEN MIT ZUFALLSZAHLEN
220 :
230 : PRINT
240 : INPUT "WELCHEN TERM MOECHTEN SIE UMFORMEN "; A$
250 : PRINT
255 : PRINT "UMZUFORMENDER TERM: "; A$
260 : PRINT
265 : INPUT "AEQUIVALENTER TERM: "; B$
270 : IF B$ = "@" THEN END
275 :
280 : GOSUB 600 : REM BERECHNUNG DES TERMWERTS
290 :
300 : PRINT
310 : PRINT "DIE UMFORMUNG IST ";
320 :
330 : IF ABS(DF) > 0.001 THEN PRINT "LEIDER FALSCH." : GOTO 250
340 : PRINT "RICHTIG !! " : LET A$ = B$ : GOTO 250
350 :
390 REM *** ENDE DES HAUPTPROGRAMMS ***********************************
399 :
400 REM +++ UNTERPROGRAMM 'BELEGUNG MIT ZUFALLSZAHLEN' +++++++++++++++
401 :
404 : PRINT : PRINT
405 : PRINT "SIE ERLEBEN JETZT DIE BELEGUNG DER VARIABLEN ";
406 : PRINT "MIT ZUFALLSZAHLEN.
407 : PRINT "TASTE DRUECKEN!
408 : GET A$ : IF A$ = "" THEN 408
409 :
410 : FOR I1 = ASC("A") TO ASC("Z")
415 :   PRINT "■■";CHR$(I1);" = ";RND(0);" : GOTO 440"
420 :   GOSUB 500 : REM BELEGUNG DES TASTATURPUFFERS
430 :   END
440 : NEXT I1
450 : POKE 158, 0 : REM KEIN ZEICHEN IM TASTATURPUFFER
460 :
490 : RETURN
499 :
500 REM +++ UNTERPROGRAMM 'BELEGUNG DES TASTATURPUFFERS' ++++++++++++++
501 :
510 : POKE 623, ASC("■") : REM CURSOR HEIM
520 : POKE 624, 13       : REM RETURN
530 : POKE 158, 2        : REM ZWEI ZEICHEN IM TASTATURPUFFER
540 :
590 : RETURN
599 :
```

```
600 REM +++ UNTERPROGRAMM 'BERECHNUNG DES TERMWERTS' ++++++++++++++++++
601 :
610 : PRINT "DF = (";A$;") - (";B$;") : GOTO 650 "
630 : GOSUB 500 : REM BELEGUNG DES TASTATURPUFFERS
640 : END
650 : FOR I = 32768 TO 32768+239 : POKE I,32 : NEXT I
655 : PRINT
660 :
690 : RETURN
699 :
```

Erläuterungen: Zunächst werden die Variablen a bis z mit Zufallszahlen zwischen 0 und 1 belegt. Nach jeder Umformung wertet der Rechner den ursprünglichen und den umgeformten Term aus und vergleicht beide Werte. Sind sie gleich, so sind die Terme mit großer Wahrscheinlichkeit äquivalent, andernfalls sicher nicht. Der Rechner gibt entsprechende Meldungen ab und fordert erneut zur Umformung auf.-

Die folgenden Überlegungen sind etwas knifflig; sie setzen das Verständnis der Inhalte des dritten Kapitels voraus. Wir wollen nämlich erreichen, daß der jeweils umgeformte Term unmittelbar, d.h. ohne Programmunterbrechung, ausgewertet wird. Normalerweise sind Funktionsterme Teile des Programms; will man sie ändern, so hat man das Programm geändert und muß es nach der Änderung neu starten. Wir wollen aber das Programm ohne Änderung durchlaufen lassen.
Um dies zu erreichen, benutzen wir die Tatsache, daß die END - Anweisung den Rechner in den Direktmodus übergehen läßt und ihn dann veranlaßt, so viele Zeichen aus dem Tastaturpuffer (Speicherzellen 623 - 632) zu holen, wie der Anzahlspeicher (Zelle 158) angibt, um sie anschließend zu verarbeiten. Schreibt man vor der Ende-Anweisung mittels PRINT eine Anweisung auf den Bildschirm und bringt den Cursor an den Anfang dieser Zeile, so kann man erreichen, daß der Computer nach Beendigung des Programms mit der Abarbeitung dieser Zeile fortfährt. Dazu muß man in den Tastaturpuffer die Stelle geben, wo der Cursor hinspringen soll,sowie das Kommando RETURN.
Genau dies macht unser Programm. Für die Buchstaben a bis z wird

(*) LET buchstabe = zufallszahl : GOTO 440

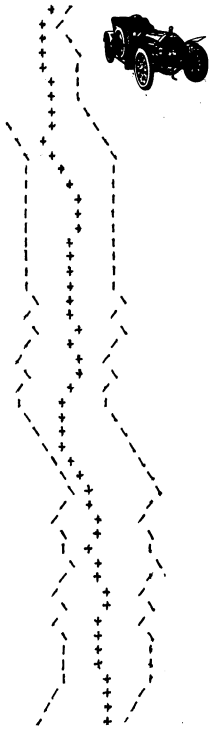
auf den Bildschirm geschrieben (Zeile 415). Das Unterprogramm 'Belegung des Tastaturpuffers' veranlaßt, daß der Rechner nach der Ende-Anweisung in Zeile 430 den Cursor in die linke obere Ecke führt, wo er die Anweisung (*) vorfindet; anschließend erhält er aus dem Tastatur-Puffer das Kommando RETURN (ASCII-Nummer 13). Dies bewirkt die Ausführung der Anweisung (*), d.h. Belegung des Buchstabens mit einer Zufallszahl und Fortfahren mit Zeile 440 (NEXT I1).
Nach genau dem gleichen Muster wird in Zeile 610 die Differenz DF der beiden Terme A$ (ursprünglicher Term) und B$ (umgeformter Term) ausgewertet.

*

Als letztes Beispiel dieses Kapitels entwickeln wir ein Programm, das schon zum Thema 'Graphikspiele' überleitet.

Beispiel 7 : Fahrlehrer

Eine Straße wird simuliert, die sich in zufälligen Windungen über den Bildschirm schlängelt, und in deren Mitte ein Punkt ('Wagen') zu halten ist.



Wir wollen die Aufgabe mit ganz bescheidenen Mitteln, also noch ohne die graphischen Möglichkeiten des folgenden Kapitels lösen. Die Straße kommt am unteren Bildfeldrand in den Blick (wir haben also die Vorstellung, rückwärts zu fahren). Die Steuerung des Wagens (durch ein Kreuz + markiert) erfolgt mittels einer Variablen T$. Wenn der Spieler keine der Tasten 1 oder 2 drückt, wartet das Programm eine kleine Weile (Zeile 320) und läßt dann den Wagen senkrecht nach unten fahren (Beibehalten der Richtung). Zufallszahlen ändern die Straßenrichtung (Zeile 420), und diese bestimmt, welches der Zeichen /, |, oder \ in der Variablen D$ zur Markierung des Straßenrandes benutzt wird. Gerät der Wagen in den Straßengraben (Zeile 360), heißt es 'Krach!', und das Programm ist beendet.

```
100 : PRINT "□"
110 : PRINT "          FAHRLEHRER"
111 : PRINT "          ----------
112 :
120 REM EINFACHES GESCHICKLICHKEITSSPIEL
121 :
130 : PRINT "SIE STEUERN IHREN WAGEN '+' MIT DEN TASTEN
135 : PRINT "1 (NACH LINKS) UND 2 (NACH RECHTS)
140 :
150 : PRINT : PRINT "TASTE DRUECKEN!
155 : GET A$ : IF A$ = "" THEN 155
190 :
```

```
200 REM --- INITIALISIERUNG ------------------------------------------------
201 :
210 : LET B = 10          : REM STRASSENBREITE
215 : LET W = 100         : REM ZEITKONSTANTE FUER WARTESCHLEIFE
220 :
230 : LET LR = 20         : REM LINKER STRASSENRAND
235 : LET RR = LR + B     : REM RECHTER STRASSENRAND
240 :
250 : LET P = 24          : REM ANFANGSPOSITION DES AUTOS
260 : LET M = 0           : REM PUNKTZAHL
290 :
300 REM --- STEUERUNG DES WAGENS -------------------------------------------
301 :
310 : GET T$
315 : IF T$ <> "" THEN 325          : REM STEUERTASTE WURDE GEDRUECKT
320 : FOR I = 1 TO W : NEXT I : GOTO 360       : REM VERZOEGERUNG
325 : IF NOT(T$ = "1" OR T$ = "2") THEN 310    : REM EINGABEKONTROLLE
330 :
340 : IF T$ = "1" THEN LET P = P-1 : GOTO 360 : REM WAGEN NACH LINKS
345 :                  LET P = P+1             : REM WAGEN NACH RECHTS
350 :
360 : IF P = LR OR P = RR THEN 600        : REM IN DEN GRABEN GEFAHREN
365 :
370 : LET M = M+1                         : REM PUNKTEZAEHLER ERHOEHEN
390 :
400 REM --- ZUFALLSBEWEGUNG DER STRASSE ------------------------------------
401 :
420 : LET X = INT(3*RND(TI))-1
425 :
430 : IF X = -1 THEN LET D$ = "/" : GOSUB 500 : GOTO 310
435 : IF X = +1 THEN LET D$ = "\" : GOSUB 500 : GOTO 310
440 :                LET D$ = "I" : GOSUB 500 : GOTO 310
490 :
500 REM +++ UNTERPROGRAMM 'AENDERUNG DER STRASSENKOORDINATEN' +++++++++
501 :
510 : LET LR = LR + X                          : REM LINKER RAND
520 : IF LR < 1 THEN LET LR = 1                : REM KONTROLLE LINKS
530 : LET RR = LR + B                          : REM RECHTER RAND
540 : IF RR > 39 THEN LET LR = 29 : GOTO 530   : REM KONTROLLE RECHTS
550 :
560 : PRINT TAB(LR) D$; TAB(P) "+"; TAB(RR) D$ : REM STRASSE ZEIGEN
570 :
590 : RETURN
599 :
600 REM --- AUSWERTUNG -----------------------------------------------------
601 :
610 : PRINT TAB(P) "▒     KRACH!!"
620 : PRINT : PRINT
630 : PRINT "SIE ERREICHTEN "M" PUNKTE.
640 :
650 : FOR I = 1 TO 1500 : NEXT I
660 : PRINT "□" : GOTO 200     : REM DAS SPIEL BEGINNT VON VORNE
```

A u f g a b e n

1. Erweitern Sie das Programm 'Zahlen-Senso' dergestalt, daß auch für das Erinnern eine Zeitschranke vorgegeben wird. Sie soll variabel und als Schwierigkeitsgrad vorher wählbar sein.

2. Beim 'Zahlen-Senso' soll die Rateleistung (Länge der wiedergegebenen Zahlenfolge beim Abbruch) bewertet und kommentiert werden.

3. Ein Programm ist zu entwerfen, das überprüft, ob der Benutzer z.B. Wochentage oder Monatsnamen richtig aufsagt.

4. Schreiben Sie ein Programm, welches prüft, ob der Benutzer die auf eine gegebene natürliche Zahl folgende Zahl richtig angibt. Beispiel: "Welche Zahl kommt nach 9909?"

5. Im Programm 'Begriffe merken' kann es vorkommen, daß ein Begriff mehrfach auf dem Bildschirm erscheint. Er kann auch einen anderen (teilweise) überdecken. Stellen Sie diesen Mißstand ab!

6. Im Programm 'Begriffe merken' soll die Zeitspanne zum Memorieren der Begriffe und die Anzahl der erscheinenden Begriffe zu Beginn wählbar sein.

7. Beim Programm 'Begriffe merken' erscheinen die Begriffe nach und nach auf dem Bildschirm. Ändern Sie das Programm so ab, daß die Wahl der Begriffe und die der Stellen auf dem Bildschirm vorher geschieht, und die Begriffe dann 'schlagartig' erscheinen.

8. Vervollständigen Sie das Programm 'Karten wenden' durch die Spielregeln und die Angabe eines Endstandes.

9. Im Programm 'Karten wenden' wandert das Spielfeld über den Bildschirm. Erzeugen Sie ein 'stehendes' Spielfeld!

10. Erweitern Sie das Programm 'Karten wenden' so, daß mehrere Personen damit spielen können.

11. Ändern Sie das Zufallsgesetz, nach dem die Straße im Programm 'Fahrlehrer' erscheint, so ab, daß längere Geradenstücke und Kurven wahrscheinlicher werden.

12. Eine vorher wählbare Anzahl n von Buchstaben soll vom Computer zufällig erzeugt und dem Benutzer zum Erinnern (Wiedergeben) präsentiert werden. Dabei ist eine gewisse Zeit einzuhalten. Der Computer stellt die Anzahl der richtig wiedergegebenen Buchstaben fest und bewertet das Ergebnis.

13. Schreiben Sie ein Programm, welches das Lernen von Vokabeln unterstützt.

14. Lesegeschwindigkeit
Auf dem Bildschirm erscheint kurze Zeit ein Satz, der gelesen und anschließend wiedergegeben werden soll.

15. Bruchrechentrainer
Schreiben Sie ein Programm, das - analog zum Rechentrainer - dem Benutzer Aufgaben zur Bruchrechnung vorlegt und die Lösungen beurteilt.

16. Zahlumwandlung
Ein Programm ist zu entwerfen, mit dem man die Fähigkeiten des Benutzers zur Umwandlung von Dezimalzahlen ins Dualsystem und umgekehrt prüfen und verbessern kann. Dialog-Beispiel:

```
Dualzahl:  11000
Dezimal?   24

Richtig!

Dezimalzahl: 7
Dual?        1111

Leider falsch.
```

17. Überschlagsrechnung
Ein Programm ist gesucht, welches das Überschlagsrechnen (vier Grundrechenarten und Wurzelziehen) trainiert. Beispiel:

```
1.85 * 2132 = ?  4000

Gut! Ihre Schätzung ist auf 1.5% genau.

Quadratwurzel von 230 ?   15

Fehlerschranke: 1.2%
```

Das Programm soll darauf achten, daß höchstens zwei geltende Ziffern eingegeben werden; z.B. wäre die Eingabe 3995 zurückzuweisen (ein solches Ergebnis kann nicht durch Überschlag zustandegekommen sein).

18. Quadratische Gleichung
Ein Programm ist zu entwerfen, welches eine quadratische Gleichung zur Lösung vorlegt; z.B.:

```
x*x + x - 6  = 0

erste Lösung? 2
zweite Lösung? -3

Richtig!
```

19. Konzentration
Auf dem Bildschirm erscheinen in regelmäßigen Zeitabständen ganze Zahlen. Sobald eine Zahl zum zweiten Mal erscheint, ist die Leertaste zu drücken.

20. Im Programm 'Termumformung' müssen Terme eingegeben werden, die im Sinne von BASIC korrekt gebildet sind; andernfalls wird eine Fehlermeldung ausgegeben (SYNTAX ERROR). Ändern Sie das Programm so ab, daß der Rechner nicht aus dem Programm aussteigt, sondern den Fehler diagnostiziert.

21. Die nächste Zahl, bitte.
Der Computer gibt den Anfang einer Zahlenfolge und fragt nach der nächsten Zahl. Beispiel:

```
1   6   11   16
Die nächste, bitte? 21
Richtig!
```

Das Bildungsgesetz dieser Folge lautet $x_{n+1} = x_n + 5$ (rekursiv) oder $x_n = 5 \cdot n + 1$ (explizit).
Als Folgen können Sie einprogrammieren:

a) $x_n = a \cdot n + b$

b) $x_n = a \cdot n + (-1)^n b$

c) $x_n = a \cdot n^2 + b$

d) $x_n = a \cdot x_{n-1} + b$

e) $x_n = a \cdot x_{n-1} + b \cdot x_{n-2} + c$

Achtung! Eine endliche Zahlenfolge läßt sich auf unendlich viele Weisen fortsetzen. Es gibt hier also nur ein Richtig oder Falsch, wenn der Typ der Folge vorher festgelegt ist.

3
Graphikspiele

Die käuflichen Computerspiele machen von graphischen Möglichkeiten regen Gebrauch: da schweben merkwürdig geformte Wesen vor imaginären Landschaften, Gebirge türmen sich auf und gebären fiktive Invasionsheere, Spielfelder entstehen , Straßen rollen unter dem Betrachter weg, und überall wird geschossen ...; getroffene Objekte zerplatzen in tausend Trümmer und formen sich alsbald zu neuen Gestalten.
Wie läßt sich diese bunte und bewegte Welt auf den Bildschirm zaubern? Schaut man sich eines der Spielprogramme näher an, um hinter sein Geheimnis zu kommen, so entdeckt man neben Texten im wesentlichen Graphiksymbole, Steuerzeichen und Zahlen - und das ganze ist vollkommen unverständlich.
Es bedarf sorgfältiger Anleitung, um die Möglichkeiten der graphischen und akustischen Programmierung zu nutzen. In diesem Kapitel sollen einige grundlegende Techniken des direkten Speicherzugriffs gezeigt werden, wie man sie bei der graphischen und akustischen Programmierung benutzt, und sie sollen bei Entwurf und Programmierung graphischer Spiele angewandt werden. Die hier vorgestellten Spiele sind bewußt ganz e i n f a c h , um das Wesentliche herauszustellen; sie stellen nur einen Rahmen dar, den der Leser, wenn er die Techniken beherrscht, mit Inhalt füllen und ausgestalten soll.

Beispiel 1: Elementare Bildschirmaktionen

Es sollen einige für Graphikspiele grundlegende Bildschirmoperationen wie z.B. das Zeichnen eines Rahmens und das Steuern eines beweglichen Punktes programmiert werden.

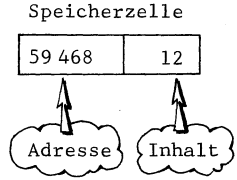
Für die genannten Aufgaben ist folgendes nützlich zu wissen. Ein Computer besitzt zahlreiche S p e i c h e r z e l l e n , von denen jede einer natürlichen Zahl, ihrer A d r e s s e , zugeordnet ist. Arbeitet man im 'normalen BASIC', so hat man auf diese Speicherzellen keinen direkten Zugriff. Es gibt jedoch zwei Anweisungen, mit denen unmittelbare Speicherplatzoperationen möglich sind: POKE und PEEK.

Die Anweisung

```
POKE <adresse>,<codezahl>
```

ermöglicht, in die Speicherzelle mit der gegebenen adresse direkt eine Zahl einzuschreiben (zwischen 0 und 255), die der Code einer Anweisung, eines Zahlenwerts oder eines Zeichens ist.

Die Anweisung `POKE 59 468 , 12` bewirkt beispielsweise, daß in die Speicherzelle Nr. 59 468 die Zahl 12 geschrieben wird:



Dieser Inhalt der genannten Speicherzelle (nämlich die 12) bezeichnet ein Kommando an das Betriebssystem des Rechners, nämlich die Umschaltung auf Großschreibung und Graphiksymbole.

Die Funktion

```
PEEK (<adresse>)
```

gibt den Inhalt der Speicherzelle mit der genannten adresse (in dezimaler Schreibweise) an. `PRINT PEEK(59 468)` druckt auf den Bildschirm eine 12 (wenn vorher obige POKE-Anweisung erteilt wurde).

Das Wort 'peek' kommt von engl. to peek = spähen, gucken, lugen (süddeutsch: spicken): man späht in die Speicherzelle hinein.

Das Wort 'poke' kommt von engl. to poke = stoßen, stochern: man stößt etwas in die Speicherzelle hinein.
Beachten Sie bitte: während POKE ... , ... eine selbständige Anweisung an den Rechner ist, ist PEEK(...) der Wert einer Funktion, also eine Zahl. Er muß also immer erst in eine Anweisung eingebaut werden, z.B.: PRINT PEEK(...) oder LET X = PEEK(...).

*

Jedem der 40 x 25 = 1000 Punkte des Bildschirms ist genau eine Speicherzelle zugeordnet, die die Information enthält, welches Zeichen auf dem Bildschirm an der genannten Stelle erscheint. Die linke obere Ecke des Bildschirms entspricht der Speicherzelle Nr. 32 768 , die rechte untere Ecke entspricht der Speicherzelle mit der Adresse 33 767. Es ist nützlich, sich ein Übersichtsblatt der folgenden Art anzulegen:

		Spalte						
		0	1	2	3		38	39
Zeile	0	32768	32769	32770	32771	...	32806	32807
	1	32808	32809	32810	32811	...	32846	32847
	2	32848	32849	32850	32851	...	32886	32887
	3	32888	32889	32890	32891	...	32926	32927
		⋮						
	23	33688	33689	33690	33691	...	33726	33727
	24	33728	33729	33730	33731	...	33766	33767

Adresse der Speicherzelle, welche für die Bildschirmstelle in Zeile 3 und Spalte 2 verantwortlich ist

Wollen wir nun ein bestimmtes Zeichen an eine vorgegebene Stelle des Bildschirms setzen, so brauchen wir nur die POKE-Anweisung mit Adresse und Codezahl des gewünschten Zeichens zu erteilen. Beispielsweise

wird durch die Anweisung `POKE 33 267, 42` das Zeichen * in die Mitte des Bildschirms plaziert (33 267 = 32 768 + 499); die Zahl 42 ist die Codezahl des Sterns im sogenannten Bildschirmcode.
Wollen wir uns den gesamten Zeichenvorrat auf dem Bildschirm ansehen, so geben wir einfach die Anweisung

```
FOR I=0 TO 255: POKE 32 768+I, I : NEXT I
```

*

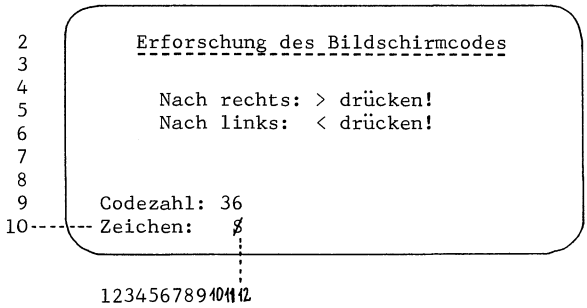
Den Bildschirmcode können wir entweder aus dem Benutzerhandbuch oder direkt vom Rechner erfahren. Wollen wir beispielsweise die Codezahl des Zeichens # kennenlernen, so schreiben wir es in die linke obere Ecke des Bildschirms, gehen mit dem Cursor in die nächste Zeile und schreiben PRINT PEEK(32 768) . Dann erscheint die Zahl 35; sie ist die gesuchte Codezahl. Möchten wir umgekehrt das Zeichen zu einer gegebenen Codezahl kennenlernen, so beschreiben wir (beispielsweise) den Bildschirm wie folgt:

```
POKE 32848,35

#
READY
■
```

Etwas komfortabler ist folgendes kleine Programm:

```
100 PRINT "■"
110 PRINT "   ERFORSCHUNG DES BILDSCHIRMCODES
115 PRINT "   ------------------------------
119 PRINT
120 PRINT "      NACH RECHTS: > DRUECKEN!
130 PRINT "      NACH LINKS:  < DRUECKEN!
140 PRINT
145 :
150 INPUT "CODEZAHL DES ERSTEN ZEICHENS "; C
160 IF C < 0 OR C > 255 THEN 150
165 :
170 POKE 32768 + 10*40 + 12, C
180 PRINT "■■■■■■■■■■CODEZAHL: "; C
190 PRINT "■■■■■■■■■■ZEICHEN: ";
195 :
200 GET T$ : IF T$ = ">" THEN LET C = C+1 : GOTO 170
210 :      : IF T$ = "<" THEN LET C = C-1 : GOTO 170
220 :                                     : GOTO 200
230 END
```

Es erzeugt folgenden Bildschirminhalt:



Zeile 170 gibt die Anweisung, das Zeichen mit der Codezahl C an die Stelle 32 768 + 10*40 + 12, d.h. in die 10. Zeile und die 12. Spalte zu drucken.-
Bitte verwechseln Sie den Bildschirmcode nicht mit dem ASCII-Code. An manchen Stellen stimmen beide überein, z.B. zwischen 20 und 63; der ASCII-Code enthält aber neben den druckbaren Zeichen auch die wichtigen Steuerzeichen, dafür aber keine Graphiksymbole. Um den ASCII-Code kennenzulernen, braucht man nur die Anweisungen (beispielsweise)

`PRINT ASC("$")` bzw. `PRINT CHR$(36)`

zu erteilen.

*

Die Zeilen 180 und 190 des obigen Programms zur Erforschung des Bildschirmcodes geben uns Gelegenheit, über die Steuerung des Cursors unter Programmkontrolle zu sprechen. Es gibt Zeichen für

(1) Bildschirm löschen,
(2) Cursor heim (d.h. Cursor in die linke obere Bildschirmecke),
(3) Cursor nach rechts
(4) Cursor nach links
(5) Cursor nach unten
(6) Cursor nach oben

(7) Negative Zeichendarstellung (dunkle Schrift auf hellem Grund),
(8) Positive Zeichendarstellung (helle Schrift auf dunklem Grund).

Obwohl diese Steuerzeichen nicht druckbar sind, erscheint bei ihrer Eingabe ein Zeichen, und zwar der Art, wie es die folgende Tabelle zeigt (leider nicht das Zeichen, das auf den entsprechenden Tasten aufgebracht ist):

	Funktion	Steuertaste	Symbol auf dem Bildschirm (beim Druck)	Symbol auf der Schreibmaschine
1	Bildschirm löschen	CLR	◻	C
2	Cursor heim	HOME	◻	H
3	Cursor nach rechts	⇒	◻	R
4	Cursor nach links	⇐	◻	L
5	Cursor nach unten	⇓	◻	U
6	Cursor nach oben	⇑	◻	O
7	Negativdarstellung	RVS	◻	N
8	Positivdarstellung	OFF	■	P

Die Spalte ganz rechts hat folgende Bedeutung: da auf der Schreibmaschine, mit der dies Buch geschrieben wird, die Steuerzeichen nicht verfügbar sind, werden sie im Text so wiedergegeben, wie es die rechte Spalte angibt. Die Zeile 180 des Programms auf der vorvorigen Seite lautet im Schreibmaschinentext:

180 PRINT "HUUUUUUUUUCODEZAHL: "; C

Achtung! Versuchen Sie bitte nicht, das entsprechende Graphiksymbol auf den Bildschirm zu setzen, wenn Sie ein Steuerzeichen anbringen wollen.

Nunmehr sind wir in der Lage, die im Beispiel 1 angesprochenen Aufgaben zu lösen:

a) Der Bildschirm soll mit einem Rahmen von beliebig wählbaren Zeichen umgeben werden.

Lösung: Beginnen wir mit dem oberen Querbalken! Um ihn zu zeichnen, brauchen wir nur in die Speicherzellen von A = 32768 bis B = 32807 die Codezahl Z des gegebenen Zeichens zu schreiben:

```
FOR I=A TO B : POKE I , Z : NEXT I
```

Beim rechten senkrechten Balken müssen wir eine Schrittweite von H = 40 wählen, um die Zeilen zu überspringen; es ist A = 32847 und B = 33767:

```
FOR I=A TO B STEP 40 : POKE I , Z : NEXT I
```

Für den unteren Querbalken ist eine Schrittweite von H = -1 angebracht, und für den senkrechten Balken links eine Schrittweite von H = -40.
Das eigentliche Zeichnen legen wir in ein kleines Unterprogramm, dem wir vor jedem Aufruf die Parameter A , B und H übergeben. Die Auswahl des Zeichens erfolgt durch eine Mehrfachverzweigung (Zeile 330 des Programms auf der nächsten Seite).
Dies Programm zum Zeichnen eines Rahmens ist sehr ausführlich und aufwendig; soll es innerhalb eines anderen Programms auftreten, fassen wir uns natürlich knapper. Man kann die Aufgabe auch ganz anders lösen, wie schon das folgende Beispiel zeigt:

b) Es soll ein Punkt über den Bildschirm wandern; seine Richtung soll durch Wahl einer passenden Schrittweite festgelegt werden können. Stößt er auf den Rahmen, soll er in die Mitte des Bildschirms rückversetzt werden.

Lösung: Während beim Rahmenzeichnen das jeweils gesetzte Zeichen stehenbleibt, muß es beim 'laufenden Punkt' nach dem Weiterrücken gelöscht werden, um die Illusion einer Bewegung zu erzeugen. Durch POKE I , 81 wird das Bällchen ● an die Stelle I gebracht; durch POKE I-1 , 32 wird das Leerzeichen (Codezahl 32) an die Stelle I-1 gesetzt (an der sich eben noch das Bällchen befand), d.h. das Zeichen wird an der eben

```
100 : PRINT "□"
110 : PRINT "        RAHMEN ZEICHNEN
111 : PRINT "        ---------------
112 :
120 REM ZEICHNET EINEN RAHMEN UM DEN BILDSCHIRM
121 :
200 REM === WAHL DES ZEICHENS ========================================
201 :
210 : PRINT
220 : PRINT "AUS WELCHEM ZEICHEN SOLL DER RAHMEN BESTEHEN?
230 : PRINT
240 : PRINT "KREUZ ................. 1 DRUECKEN
250 : PRINT "STERN ................. 2 DRUECKEN
260 : PRINT "PFEIL ................. 3 DRUECKEN
270 : PRINT "BALL .................. 4 DRUECKEN
280 : PRINT
290 : GET A$ : IF A$ = "" THEN 290
300 : IF A$ < "1" OR A$ > "4" THEN 290
320 :
330 : ON VAL(A$) GOTO 350, 360, 370, 380
340 :
350 : LET Z = 43 : GOTO 390 : REM STERN *
360 : LET Z = 42 : GOTO 390 : REM KREUZ +
370 : LET Z = 30 : GOTO 390 : REM PFEIL ↑
380 : LET Z = 81 :           : REM BALL  ■
385 :
390 : PRINT "□"  : REM BILDSCHIRM LOESCHEN
395 :
400 REM === ZEICHNUNG DES OBEREN QUERBALKENS ==========================
401 :
410 : LET A = 32768 : LET B = 32806
420 : LET H = 1     : REM SCHRITTWEITE 1 (KOENNTE ENTFALLEN)
430 : GOSUB 800     : REM ZUM UNTERPROGRAMM 'BALKENZEICHNEN'
490 :
500 REM === ZEICHNUNG DES SENKRECHTEN BALKENS RECHTS ==================
501 :
510 : LET A = 32807 : LET B = 33727
520 : LET H = 40    : REM SCHRITTWEITE 40
530 : GOSUB 800     : REM ZUM UNTERPROGRAMM 'BALKENZEICHNEN'
590 :
600 REM === ZEICHNUNG DES UNTEREN QUERBALKENS =========================
601 :
610 : LET A = 33767 : LET B = 33729
620 : LET H = -1    : REM SCHRITTWEITE -1
630 : GOSUB 800     : REM ZUM UNTERPROGRAMM 'BALKENZEICHNEN'
690 :
700 REM === ZEICHNUNG DES SENKRECHTEN BALKENS LINKS ==================
701 :
710 : LET A = 33728 : LET B = 32808
720 : LET H = -40   : REM SCHRITTWEITE -40
730 : GOSUB 800     : REM ZUM UNTERPROGRAMM 'BALKENZEICHNEN'
790 : END
799 :
800 REM +++ UNTERPROGRAMM 'BALKENZEICHNEN' ++++++++++++++++++++++++++
801 :
810 : FOR I = A TO B STEP H : POKE I,Z : NEXT I : RETURN
```

verlassenen Stelle gelöscht. Das Programm

```
10 LET A = 32768 + 3*40
20 LET B = A + 39
30 FOR I = A TO B
40 : POKE I,81
50 : POKE I-1,32
60 NEXT I
```

läßt in der dritten Zeile ein Bällchen von links nach rechts über den Bildschirm laufen.

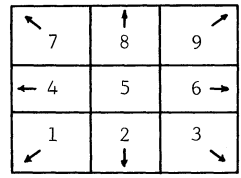
Wichtig ist das Kennenlernen der verschiedenen Möglichkeiten, durch Wahl einer geeigneten Schrittweite H die Richtung des laufenden Punktes festzulegen. Diesem Zweck dient das folgende Programm:

```
100 : PRINT "□"
110 : PRINT "          LAUFENDER PUNKT
111 : PRINT "          ---------------
112 :
120 REM DIENT ZUM STUDIEREN DER BAHN EINES BEWEGLICHEN PUNKTS
121 REM IN ABHAENGIGKEIT VON DER SCHRITTWEITE
122 :
130 : LET P = 42           : REM CODEZAHL DES PUNKTES
140 : LET M = 32768 + 499  : REM BILDSCHIRMMITTE
150 :
160 : PRINT
170 : INPUT "SCHRITTWEITE "; H
190 :
200 REM === RAHMEN ZEICHNEN ==========================================
201 :
210 : LET A$ = "########################################"
220 : LET B$ = "#                                      #"
230 : PRINT "□"A$
240 : FOR I = 1 TO 22 : PRINT "□"B$ : NEXT I
250 : PRINT "□"A$"■"
290 :
300 REM === PUNKTBEWEGUNG ============================================
301 :
310 : LET X = M                    : REM BILDSCHIRMMITTE
320 : POKE X,P                     : REM PUNKT PLAZIEREN
330 :
340 : LET Y = X+H : REM NEUE POSITION, SCHLEIFENANFANG ---------------
350 :
360 :   IF NOT(Y < 32768 OR Y > 33727 OR PEEK(Y) = 35) THEN 390
370 :   LET Y = 32768 + INT(960*RND(1)+40) : REM ZUFALLSPOSITION
380 :
390 :   FOR I = 1 TO 100 : NEXT I : REM WARTESCHLEIFE
400 :
410 :   POKE Y,P                   : REM PUNKT PLAZIEREN
420 :   POKE X,32                  : REM ALTE POSITION LOESCHEN
430 :   LET X = Y                  : REM NEUE POSITION WIRD ZUR ALTEN
440 :
450 : GOTO 340 : REM SCHLEIFENENDE ----------------------------------
460 :
490 : END
```

Nun zur dritten Aufgabe:

c) Ein beweglicher Punkt soll mittels Tasten über den Bildschirm gesteuert werden.

Lösung: Zuerst zeichnen wir wieder einen Rahmen; er dient u.a. dazu, daß der Punkt nicht aus dem Bildschirm herauslaufen kann. Zur Steuerung des Punktes benutzen wir die Zahltasten in der nebenstehend abgebildeten Weise. Jeder Richtung entspricht eine Schrittweite; wir bilden also eine Tabelle, die jeder Zahltaste die vereinbarte Schrittweite und damit Richtung zuordnet.



Die Steuerung des Punktes geschieht nach folgendem Algorithmus:

PUNKTSTEUERUNG

```
x := Anfangsposition
Plaziere das Zeichen auf Position x
WIEDERHOLE
  Lies Taste
  Bestimme die neue Position y
  WENN y auf dem Rahmen DANN
    y := x  (* alte Position wird beibehalten *)
  ENDE-WENN
  Plaziere das Zeichen auf Position y
  Lösche das Zeichen auf der alten Position x
  x := y  (* die neue Position wird zur alten Position *)
ENDE-WIEDERHOLE
```

Die Anweisung 'Lies Taste' geschieht mittels der Anweisung GET wie folgt: in die Variable T$ wird die gedrückte Ziffer eingelesen. Ist keine Taste gedrückt (Bedingung T$ = ""), so wird die bisherige Richtung beibehalten, andernfalls wird durch `LET Y = H(T)` die neue Richtung übernommen. Die Abfrage `IF PEEK(Y) = R THEN LET Y = X` verhindert, daß der Punkt über den Bildschirm hinausschießt (er bleibt auf seiner vorigen Position).

```
100 : PRINT "□"
110 : PRINT "     PUNKT STEUERN"
111 : PRINT "     ---------------"
112 :
120 REM EIN AN EINE ZUFAELLIGE STELLE DES BILDSCHIRMS GESETZTES ZEICHEN
121 REM WIRD MITTELS DER ZAHLTASTEN IN EINE GEWUENSCHTE RICHTUNG
122 REM GESTEUERT; STOESST ES AUF DEN RAND, BLEIBT ES STEHEN
129 :
130 : LET R = 90   : REM ZEICHEN FUER DEN RAHMEN ("♠")
140 : LET P = 81   : REM ZEICHREN FUER DEN BEWEGLICHEN PUNKT ("●")
190 :
200 REM === RAHMEN ZEICHNEN ============================================
201 :
210 : PRINT "□"
220 : FOR I = 32768 TO 32807
230 :   POKE I,R : POKE 960+I,R
240 : NEXT I
250 : FOR I = 32768 TO 33727 STEP 40
260 :   POKE I,R : POKE 39+I,R
270 : NEXT I
290 :
300 REM === RICHTUNGSTABELLE FESTLEGEN ================================
301 :
310 : DIM H(9)
311 : LET H(1) =  39 : REM NACH LINKS UNTEN
312 : LET H(2) =  40 : REM SENKRECHT NACH UNTEN
313 : LET H(3) =  41 : REM NACH RECHTS UNTEN
314 : LET H(4) =  -1 : REM NACH LINKS
315 : LET H(5) =   0 : REM STEHENBLEIBEN
316 : LET H(6) =   1 : REM NACH RECHTS
317 : LET H(7) = -41 : REM NACH LINKS OBEN
318 : LET H(8) = -40 : REM SENKRECHT NACH OBEN
319 : LET H(9) = -39 : REM NACH RECHTS OBEN
390 :
400 REM === ZUFALLSSTELLE SUCHEN ======================================
401 :
410 : LET X = INT(1000*RND(1)) + 32768
420 : IF PEEK(X) = R THEN 410          : REM NICHT AUF DEN RAND
490 :
500 REM === PUNKT SETZEN UND STEUERN ==================================
501 :
510 : POKE X,P   : REM SCHLEIFENANFANG, PUNKT SETZEN ------------------
520 :
530 :   GET T$                          : REM TASTE EINLESEN
540 :   IF T$ <> "" THEN LET T = VAL(T$) : REM TASTE WURDE GEDRUECKT
550 :   LET Y = X + H(T)                 : REM NEUE POSITION
560 :
570 :   IF PEEK(Y) = R THEN LET Y = X     : REM NICHT AUF DEN RAND
580 :
590 :   POKE Y,P                         : REM PUNKT SETZEN
600 :   POKE X,32                        : REM ALTE POSITION LOESCHEN
610 :   LET X = Y                        : REM AUS NEU WIRD ALT
620 :
630 : GOTO 510   : REM SCHLEIFENENDE ----------------------------------
690 :
699 END
```

An dieser Stelle sind noch einige Bemerkungen zum sogenannten Tastaturpuffer angebracht. "Puffer" heißt "Zwischenspeicher"; der Tastaturpuffer ist ein Zwischenspeicher, der jedes Zeichen, das über die Tastatur eingegeben wird, zunächst aufnimmt. Ein Vorteil des Tastaturpuffers besteht darin, daß man - noch während der Rechner mit etwas anderem beschäftigt ist - Zeichen eingeben kann; sie werden dann nach Abschluß der Tätigkeit aus dem Tastaturpuffer geholt und bearbeitet. Das folgende kleine Programm demonstriert, wie man die Anzahl der Zeichen im Tastaturpuffer beschränken kann. Denn manchmal ist es lästig - z.B. nach dem Abbruch eines Spiels - , daß der Rechner noch Zeichen auf den Bildschirm bringt, die von Tastendrücken während des Spiels stammen.

```
100 PRINT "      DEMONSTRATION DES TASTATURPUFFERS
101 PRINT "      ---------------------------------
110 PRINT
120 INPUT "WIEVIELE ZEICHEN SOLL ICH MIR MERKEN? (0 BIS 9) "; N
140 IF N < 0 OR N > 9 THEN 110
150 :
200 PRINT
210 PRINT "ICH BIN NUNMEHR EINE WEILE BESCHAEFTIGT.
220 PRINT "GEBEN SIE EINE BELIEBIGE ANZAHL VON ZEICHEN EIN!
230 :
250 FOR I = 1 TO 5000 : NEXT I : REM SIMULIERTE BESCHAEFTIGUNG
260 :
270 POKE 158,N : REM FESTLEGUNG DER ANZAHL GUELTIGER ZEICHEN
280 :
290 PRINT
300 PRINT "DIE UNTER DEM 'READY' STEHENDEN ";N;"ZEICHEN HATTE
310 PRINT "ICH MIR GEMERKT; JETZT BIN ICH IM EINGABEMODUS.
320 :
330 END
```

Manchmal ist es äußerst nützlich, wenn man in den Tastaturpuffer Botschaften hinterlegen kann, die der Rechner nach Beendigung eines Programms noch aufnimmt (siehe z.B. des Programm 'Termumformung' in Kap.2). Besteht eine solche Botschaft aus einem Steuerzeichen, so führt er die zugehörige Anweisung aus. Z.B. kann er den Cursor auf eine vorher

auf den Bildschirm geschriebene Zeile setzen. Diese wird dann - wie im Direktmodus - als Anweisung oder - wie im Eingabemodus - als Eingabe interpretiert. Kommt nun vom Tastenpuffer ein RETURN (Codezahl 13), so wirkt dies wie ein RETURN im Direktmodus bzw. im Eingabemodus.
Auf diese Weise kann ein Programm noch "aus dem Grabe", d.h. nach seiner Beendigung, wirken. Man nennt dergleichen 'programmierter Direktmodus' (eigentlich ein Widerspruch).

*

Als Anregung zu eigenen Programmiertätigkeiten empfehle ich Ihnen die Aufgaben 1 bis 8 .

*

Nachdem wir nun einen Punkt beliebig über den Bildschirm steuern können, wollen wir uns an unser erstes kleines Graphikspiel wagen. Es liegt nahe, dem Punkt Hindernisse in den Weg zu legen und an den Ausweichversuchen die Geschicklichkeit des Spielers zu erproben.

Beispiel 2 : Fledermaus

Wie eine Fledermaus, die mit nachtwandlerischer Sicherheit Hindernisse umfliegt, soll der vom Spieler gesteuerte Punkt Gegenstände, die ihm im Weg liegen, umgehen. Das Hinterhältige daran ist nur, daß die Anzahl der Gegenstände, vom Zufall beeinflußt, ständig wächst ...

Unser Programm soll aus folgenden Teilen bestehen:

1. Ausgabe der Spielregeln (falls gewünscht)
2. Initialisierung, d.h. Festlegung der Anfangswerte der Variablen
3. Zeichnen des Spielfelds
4. Spiel
5. Auswertung
6. Verabschiedung oder Wiederholung.

Das eigentliche Spiel (Teil 4) folgt nachstehendem Algorithmus:

SPIEL

```
Setze Punkt auf Anfangsposition  x
WIEDERHOLE
  Lies Taste
  WENN  Taste nicht gedrückt  DANN  Schrittweite := 1
  Bestimme neue Position  y
  ABBRUCH FALLS  Zusammenprall mit Hindernis
  Setze Punkt auf Position  y
  Lösche Punkt auf Position x
  x := y  (* neue Position wird zur alten *)
  Plaziere neues Hindernis  (* zufällig *)
ENDE-WIEDERHOLE
```

Und hier ist das Programm:

```
100 : PRINT "□"
110 : PRINT "              FLEDERMAUS"
111 : PRINT "              ----------"
112 :
115 : PRINT : PRINT
116 : PRINT "       EIN SPIEL ZUM UEBERLEBEN"
117 :
120 REM EINFACHES REAKTIONS- UND GESCHICKLICHKEITSSPIEL
121 :
130 REM  VARIABLEN:
131 :
140 REM    P ..... CODE DES VOM SPIELER GESTEUERTEN PUNKTES
145 REM    R ..... CODE DES RAHMENS
150 REM    L ..... CODE DES LEERZEICHENS
155 REM    N ..... ZEITKONSTANTE
160 REM    H(..)... RICHTUNGSTABELLE
170 REM    X, Y ... POSITIONEN DES VOM SPIELER GESTEUERTEN PUNKTES
175 REM    T$, T .. EINGABEZEICHEN (ZIFFERNTASTEN)
180 REM    H ..... SCHRITTWEITE FUER DIE PUNKTSTEUERUNG
190 :
199 :
200 REM === SPIELREGELN ================================================
201 :
210 : PRINT : PRINT : PRINT : PRINT : PRINT
220 : PRINT "SOLL ICH DIE SPIELREGELN AUSGEBEN (J/N)"
225 : GET A$
230 : IF A$ <> "J" AND A$ <> "N" THEN 225
240 : IF A$ = "N" THEN 300
245 :
250 : PRINT "□"
255 : PRINT "                    SPIELREGELN"
260 : PRINT "                    -----------"
265 : PRINT : PRINT
270 : PRINT "SIE KOENNEN MIT DEN VIER ZIFFERNTASTEN 2,4,6,8 DEN STERN
275 : PRINT "IN DER UEBLICHEN WEISE BEWEGEN. WENN SIE AUF EIN HINDERNIS
280 : PRINT "PRALLEN, IST DAS SPIEL ZU ENDE.
```

```
290 : PRINT "DRUECKEN SIE EINE TASTE!"
295 : GET Z$ : IF Z$ = "" THEN 295
299 :
300 REM === INITIALISIERUNG ============================================
301 :
310 : LET H(2) = 40 : LET H(4) = -1 : LET H(6) = 1 : LET H(8) = -40
320 :
330 : LET P = 42     : REM CODEZAHL DES STERNS
340 : LET R = 160    : REM CODENUMMER VON ■
350 : LET L = 32     : REM CODEZAHL DES LEERZEICHENS
360 : LET X = 33270 : REM ANFANGSPOSITION
370 : LET N = 150    : REM BESTIMMT GESCHWINDIGKEIT DES PUNKTES
390 :
400 REM === SPIELFELD ZEICHNEN ========================================
401 :
410 : PRINT "□"
420 :
430 : FOR I = 32768 TO 32807
440 :   POKE I, R : POKE I+960, R
445 : NEXT I
450 :
460 : FOR I = 32768 TO 33727 STEP 40
470 :   POKE I, R : POKE I+39, R
475 : NEXT I
480 :
500 REM === SPIEL ======================================================
501 :
510 : POKE X, P
515 :
520 : GET T$
525 : IF T$ = "" THEN LET H = 1 : GOTO 540 : REM PUNKT NACH RECHTS
530 :   LET T = VAL(T$)
535 :   LET H = H(T)
540 :   LET Y = X+H
545 :
550 : IF PEEK(Y) <> L THEN 600              : REM ZUSAMMENPRALL
555 :
560 :   POKE Y, P
565 :   POKE X, L
570 :   LET X = Y
575 :
580 : POKE INT(960*RND(TI))+32768, R        : REM HINDERNIS
585 :
590 : FOR I = 1 TO N : NEXT I : REM WARTESCHLEIFE
592 : LET N = 0.99*N           : REM ZEITSPANNE WIRD VERKLEINERT
595 :
597 : GOTO 510
598 :
599 :
600 REM === AUSWERTUNG ================================================
601 :
610 : FOR I = 1 TO 150 : POKE X,R : POKE X,P : NEXT I : REM ZAPPELN
620 :
630 : PRINT "□"
640 : PRINT : PRINT
650 : PRINT "SIE HABEN LEIDER VERLOREN."
660 :
690 :
```

```
700 REM === VERABSCHIEDUNG ODER WIEDERHOLUNG ========================
701 :
710 : PRINT : PRINT
720 : PRINT "NOCH EIN VERSUCH?(J/N)"
730 : GET A$
740 : IF A$ <> "J" AND A$ <> "N" THEN 730
750 : IF A$ = "J" THEN RUN
760 : PRINT
770 : PRINT "AUF WIEDERSEHEN."
780 : PRINT : PRINT
790 :
799 : END
```

Erläuterungen:

Damit der Spieler den Punkt nicht an einer Stelle stehen läßt, bekommt dieser eine Eigenbewegung: in Zeile 525 wird die Schrittweite H = 1 gesetzt, d.h. der Punkt läuft - wird keine Taste gedrückt - nach rechts weg. Die Zeitkonstante N (Zeile 370) bestimmt über die Warteschleife (Zeile 590) die Geschwindigkeit des Punktes; diese wird allmählich immer größer, da in 592 LET N = 0.99 * N gesetzt wird. Zeile 610 bewirkt ein Vibrieren (Zappeln) des Punktes, wenn er auf ein Hindernis gestoßen ist: in schneller Folge werden die Zeichen * und ▒ abwechselnd auf die gleiche Stelle gesetzt.-

Leider kann man bei diesem Spiel nur verlieren; versuchen Sie, es zu einem Spiel mit Gewinnchancen abzuwandeln (siehe die Aufgaben)!

*

Während der Spieler als 'Fledermaus' zur Passivität verdammt ist, soll er jetzt die Möglichkeit bekommen, Hindernisse durch einen gezielten Schuß aus dem Wege zu räumen.

Beispiel 3: Sterne jagen

Es soll ein Graphikspiel entworfen werden, das wie folgt funktioniert: Auf dem Bildschirm (mit Rahmen) erscheinen zufällig gesetzte Punkte (Sterne) in einer vorher wählbaren Anzahl. Mittels eines beweglichen Punktes ('Raumschiff') sollen sie gejagt werden. Bewegt sich das Raumschiff auf einen Stern zu, kann es in Bewegungsrichtung einen Schuß abfeuern, der - wenn er trifft - den Stern zum Verlöschen bringt. Stößt das Raumschiff auf einen Stern, muß es verglühen und das Spiel ist verloren.

Wir gliedern das Spielprogramm in fünf Unterprogramme:

```
1000 : PRINT "■"
1010 : PRINT "              STERNE JAGEN
1020 : PRINT "              ------------
1030 :
1040 REM REAKTIONS- UND GESCHICKLICHKEITSSPIEL
1050 :
1090 :
2000 REM *** HAUPTPROGRAMM *********************************************
2100 :
2300 : GOSUB 3000 : REM AUSGABE DER SPIELREGELN
2350 :
2400 : GOSUB 4000 : REM SPIELFELD UND STERNE
2450 :
2500 : GOSUB 5000 : REM SPIEL
2550 :
2600 : GOSUB 6000 : REM AUSWERTUNG
2650 :
2700 : GOSUB 7000 : REM  VERABSCHIEDUNG ODER WIEDERHOLUNG
2750 :
2900 : END
2950 :
2960 : REM *** ENDE HAUPTPROGRAMM ***************************************
2980 :
2999 :
```

Interessant ist nur das Unterprogramm 'Spiel'. Beim Bewegen des Raumschiffs muß abgefragt werden, ob es auf einen Stern geprallt ist:

```
5300  IF PEEK(Y) = 42 THEN E$ = "BUMS" : RETURN
```

In diesem Fall wird die Variable E$, welche sich Erfolg oder Mißerfolg merkt, mit dem Wort 'Bums' belegt und das Unterprogramm beendet.
Nicht ganz einfach ist das Abfeuern eines Schusses: die Bewegungsrichtung des Raumschiffs ist in der Variablen H(T) aufbewahrt; sie wird in Zeile 5630 dem Schuß mitgeteilt: LET U = Z+H(T) . Ist das Zeichen an der Stelle U ein Karo (Codezahl 90), so liegt ein Fehlschuß vor: man hat - ohne zu treffen - auf den Rand geschossen. In diesem Fall wird zu 5200 GET T$ zurückgesprungen und auf einen neuen Tastendruck gewartet - bzw. das Raumschiff bewegt sich in der bisherigen Richtung weiter.
Ist ein Stern getroffen, so erfolgt eine kleine Detonation: es erscheint kurzfristig das Zeichen X , dann ist Stille. Den ganzen Schuß betten wir in eine Zählschleife ein, um die Angelegenheit übersichtlicher zu gestalten: 5620 FOR I = 1 TO 50 ... . Diese Schleife wird in jedem Fall vorzeitig verlassen: entweder trifft der Schuß auf die Wand oder auf einen Stern.

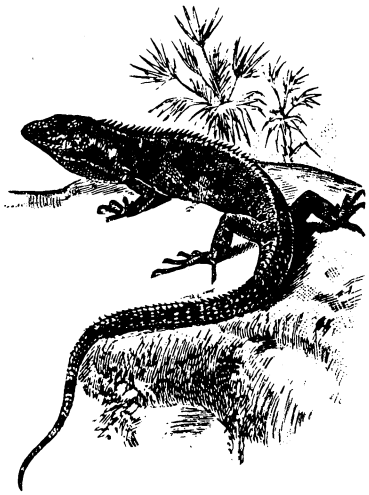
```
3000 REM +++ UNTERPROGRAMM 'AUSGABE DER SPIELREGELN' ++++++++++++++++++
3010 :
3100 : PRINT : PRINT
3200 : PRINT "WOLLEN SIE DIE SPIELREGELN KENNENLERNEN?(J/N)"
3300 : GET A$
3350 : IF A$ <> "J" THEN IF A$ <> "N" THEN 3300
3400 : IF A$ = "N" THEN RETURN
3490 :
3500 : PRINT "■"
3600 : PRINT "                              SPIELREGELN"
3610 : PRINT "                              ----------"
3620 : PRINT : PRINT
3630 : PRINT "SO SIEHT IHR RAUMSCHIFF AUS: o,"
3640 : PRINT "UND DIES SIND DIE STERNE, DIE SIE JAGEN SOLLEN: *"
3650 : PRINT "SIE STEUERN DAS RAUMSCHIFF MIT DEN UEBLICHEN TASTEN"
3660 : PRINT "UND FEUERN MIT TASTE 5 IN BEWEGUNGSRICHTUNG."
3690 :
3700 : PRINT "VERSTANDEN? DANN DRUECKEN SIE TASTE '='!"
3750 :
3800 : GET A$ : IF A$ <> "=" THEN 3800
3850 :
3900 : RETURN
3999 :
4000 REM +++ UNTERPROGRAMM 'SPIELFELD, STERNE UND RAUMSCHIFF' ++++++++
4010 :
4100 : PRINT : PRINT
4200 : INPUT "■WIEVIEL STERNE SOLLEN ES SEIN "; N
4250 : IF N < 1 OR N > 50 OR INT(N) <> N THEN 4200
4290 :
4300 : REM --- RAHMEN ZEICHNEN
4310 :
4320 :   PRINT "■"
4330 :   FOR I = 32768 TO 32807
4340 :     POKE I,90 : POKE I+960,90
4350 :   NEXT I
4390 :
4400 :   FOR I = 32768 TO 33727 STEP 40
4420 :     POKE I,90 : POKE I+39,90
4430 :   NEXT I
4490 :
4500 : REM --- STERNE PLAZIEREN
4510 :
4520 :   FOR I = 1 TO N
4530 :     LET S = INT(1000*RND(TI)) + 32768
4540 :     IF PEEK(S) = 42 OR PEEK(S) = 90 THEN LET I = I-1 : GOTO 4560
4550 :     POKE S,42
4560 :   NEXT I
4590 :
4600 : REM --- RAUMSCHIFF PLAZIEREN
4610 :
4620 :   LET X = INT(1000*RND(TI)) + 32768
4630 :   IF PEEK(X) <> 32 THEN 4620
4690 :
4700 : REM --- RICHTUNGSTABELLE
4710 :
4720 :   DIM H(9)
4730 :   LET H(1)=39 : LET H(2)=40 : LET H(3)=41 : LET H(4)=-1
4740 :   LET H(6)= 1 : LET H(7)=-41: LET H(8)=-40: LET H(9)=-39
4790 :
4800 : LET E = 0                    : REM ZAEHLER DER ERFOLGE
4850 :
4900 : RETURN
```

```
4999 :
5000 REM +++ UNTERPROGRAMM 'SPIEL' +++++++++++++++++++++++++++++++++++
5010 :
5100 : POKE X,87                              : REM RAUMSCHIFF ERSCHEINT
5150 : FOR I = 1 TO 50 : NEXT I               : REM WARTESCHLEIFE
5180 :
5200 : GET T$
5250 : IF T$ = "5" THEN 5600                  : REM FEUERN
5260 :   IF T$ <> "" THEN T = VAL(T$)         : REM TASTE WURDE GEDRUECKT
5270 :     LET Y = X + H(T)                   : REM NEUE POSITION
5280 :
5300 :   IF PEEK(Y) = 42 THEN E$ = "BUMS": RETURN : REM ZUSAMMENSTOSS
5350 :   IF PEEK(Y) = 90 THEN LET Y = X       : REM RAHMEN BERUEHRT
5390 :
5400 : POKE X,32                              : REM ALTE POSITION LOESCHEN
5450 : LET X = Y : GOTO 5100                  : REM RUECKSPRUNG
5490 :
5600 : REM +++ FEUERN +++
5610 :
5615 :   LET Z = X                            : REM KUGELPOSITION
5620 :   FOR I = 1 TO 50                      : REM SCHUSSCHLEIFE
5630 :     LET U = Z+H(T)                     : REM NEUE KUGELPOSITION
5640 :     LET P = PEEK(U)
5650 :     IF P = 90 THEN 5200                : REM FEHLSCHUSS, NEUE TASTE
5655 :     POKE U,46                          : REM PUNKT ALS KUGEL
5656 :     IF P = 42 THEN 5670                : REM TREFFER, ZUR DETONATION
5657 :     POKE U,32 : LET Z = U              : REM LOESCHEN UND POSITION
5660 :   NEXT I                                             MERKEN
5665 :
5670 :     POKE U,86                          : REM DETONATION
5675 :     FOR K = 1 TO 50 : NEXT K
5680 :     POKE U,32
5700 :     LET E = E+1                        : REM TREFFER ADDIERT
5800 :     IF E < N THEN 5200                 : REM RUECKSPRUNG
5850 :     LET E$ = "ERFOLG"                  : REM ALLE STERNE WEG
5890 :
5900 : RETURN
5999 :
6000 REM +++ UNTERPROGRAMM 'AUSWERTUNG' +++++++++++++++++++++++++++++
6010 :
6050 : PRINT"□"
6100 : IF E$ = "ERFOLG" THEN 6500
6200 : PRINT"■■■■■■■■■■■▯▯▯▯▯▯▯▯ZUSAMMENSTOSS!"
6300 : GOTO 6800
6350 : RETURN
6390 :
6500 : PRINT "SIE HABEN ES GESCHAFFT!"
6800 : FOR I = 1 TO 1000 : NEXT I
6900 : RETURN
6999 :
7000 REM +++ UNTERPROGRAMM 'VERABSCHIEDUNG ODER WIEDERHOLUNG' ++++++++
7010 :
7100 : PRINT "□"
7200 : PRINT : PRINT : PRINT
7300 : PRINT "MOECHTEN SIE ES NOCH EINMAL VERSUCHEN?(J/N)"
7400 : GET A$ : IF A$ <> "J" THEN IF A$ <> "N" THEN 7400
7500 : IF A$ = "J" THEN RUN
7600 : PRINT "□WAS MAN NICHT KANN, SPIELT MAN EBEN NICHT."
7700 : PRINT "ICH EMPFEHLE MICH."
7800 :
7900 : RETURN
```

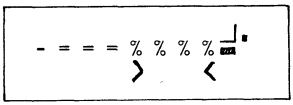
Nach den vielen Schießereien wollen wir nunmehr ein ganz friedliches Tier betrachten.

Beispiel 4: Eidechse

Eine Eidechse kriecht über den Bildschirm, über ihr summt eine Fliege. Plötzlich schnellt die Zunge der Eidechse vor ...



Bisher haben wir nur Objekte über den Bildschirm bewegt, die aus jeweils e i n e m Zeichen bestanden. Unsere Eidechse jedoch bauen wir aus m e h r e r e n Zeichen auf, und zwar so:



Erkennen Sie die Ähnlichkeit?

Diese Zeichen haben folgende Codezahlen: % (37), = (61), - (64), . (46), ⌟ (122), < (60), > (62) und ▭ (121).

Geben wir der Mundpartie ▭ die Position K, so hat z.B. das Vorderbein die Position K+39; wir geben also die Anweisung POKE K+39,60 . Die gesamte Eidechse wird in den Zeilen 515 bis 550 des Programms auf der folgenden Seite gezeichnet.

Ihre Bewegung geht in einer FOR-NEXT-Schleife vor sich (Zeile 510 bzw. Zeile 710). Wird K um 1 erhöht, d.h. macht die Eidechse einen Schritt nach rechts, so müssen nur diejenigen Positionen des Eidechsenleibs mit einem Leerzeichen belegt werden, die nicht von nachrückenden Körperteileilen überschrieben werden.

Während ihrer Vorwärtsbewegung kann die Eidechse ihre Zunge senkrecht nach oben schnellen; dies geschieht ganz ähnlich wie beim Schuß im vorigen Programm (wir haben es in ein kleines Unterprogramm gelegt). Die Eidechse hat 60 Sekunden Zeit, auf Jagd zu gehen, dann ist das Spiel zuende; die Zeit ist jeweils oben ins Bild eingeblendet. Die Uhr wird in Zeile 360 auf Null gestellt und jeweils in Zeile 620 abgelesen.

```
100 : PRINT "■"
110 : PRINT "                  EIDECHSE
111 : PRINT "                  ---------
112 :
120 REM REAKTIONS- UND GESCHICKLICHKEITSSPIEL
121 :
200 REM === SPIELREGELN ================================================
201 :
210 : PRINT
220 : PRINT "DIE EIDECHSE VERSUCHT, DIE FLIEGE
225 : PRINT "ZU ERHASCHEN.
230 : PRINT "ZUM BETAETIGEN DER ZUNGE DRUECKEN SIE
235 : PRINT "IRGENDEINE TASTE.
240 : PRINT
250 : PRINT "DRUECKEN SIE JETZT EINE TASTE!
260 :
270 : GET T$ : IF T$ ="" THEN 260
290 :
300 REM === VORBEREITUNG ================================================
301 :
310 : PRINT "■"
320 : PRINT TAB(14) "■■■■■■■■■■■FERTIGMACHEN!"
330 :
340 : FOR I = 1 TO 1000 : NEXT I    : REM VORWARNZEIT
350 :
360 : LET TI$ = "000000"             : REM UHR WIRD AUF NULL GESTELLT
370 :
380 : LET Z = 0                      : REM FLIEGENZAEHLER
390 :
400 REM === SPIEL ======================================================
401 :
410 : PRINT"■"
420 :
430 : REM --- FLIEGE PLAZIEREN ---
440 :
450 :   LET R = INT(10*RND(TI))
460 :   LET X  = 32768+11+R*40+INT(29*RND(TI))
470 :   POKE X,42
499 :
500 : REM --- EIDECHSE LAUFEN LASSEN ---
501 :
510 :   FOR K = 33696 TO 33726    : REM K IST DER KOPF DER EIDECHSE
512 :
515 :     POKE K-1,32   : POKE K,32      : POKE K-41,32 : POKE K-40,32
520 :     POKE K,121    : POKE K-1,37    : POKE K-2,37  : POKE K-3,37
530 :     POKE K-4,37   : POKE K-5,61    : POKE K-6,61  : POKE K-7,61
540 :     POKE K-8,64   : POKE K-40,122  : POKE K-39,46 : POKE K-9,32
550 :     POKE K+39,60  : POKE K+38,32   : POKE K+36,62 : POKE K+35,32
610 :
620 :     PRINT "■";" ZEIT: "; TI$      : REM ZEITANSAGE
630 :
640 :     GET A$
650 :     IF A$ = "" THEN 710 : REM NAECHSTER SCHRITT DER EIDECHSE
655 :
660 :     GOSUB 800 : REM FANGVERSUCH
670 :
680 :     IF (TI-T0)/60 < 60 THEN 400 : REM AUF EIN NEUES
690 :     GOTO 900                    : REM ZEIT IST ABGELAUFEN
700 :
710 :   NEXT K
720 :
750 :   GOTO 400                      : REM NEUER VERSUCH
799 :
```

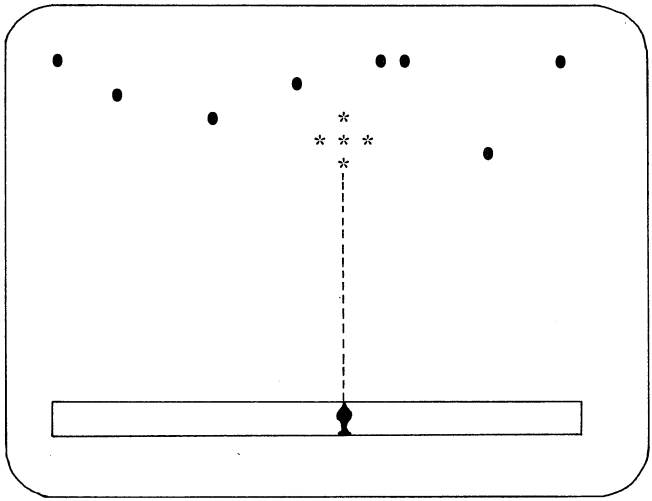
```
800 REM +++ UNTERPROGRAMM 'FANGVERSUCH' ++++++++++++++++++++++++++++++
801 :
805 : REM --- ZUNGE SCHNELLT NACH OBEN ---
809 :
810 :    FOR T = K-80 TO K-960 STEP -40
815 :        POKE T,103
819 :
820 :        IF PEEK(T-40) <> 42 THEN 870   : REM NOCH NICHTS ERWISCHT
825 :        FOR H = T TO K STEP 40         : REM BEUTE EINROLLEN
830 :          POKE H-40,32 : POKE H,81
832 :          FOR I = 1 TO 5 : NEXT I
834 :        NEXT H
835 :
840 :      REM --- BEUTE VERSCHLINGEN ---
841 :
845 :        POKE K-39,32 : POKE K-40,32 : POKE K,98
847 :        POKE K+1,98
849 :        FOR I = 1 TO 50 : NEXT I
850 :        POKE K,98 : POKE K+1,111
855 :
860 :        GOSUB 880 : RETURN : REM SCHLUCKEN
864 :
865 :        POKE T,103 : POKE T,32
869 :
870 :    NEXT T
874 :
875 :    RETURN
879 :
880 : REM +++ UNTERPROGRAMM 'SCHLUCKEN' +++
881 :
885 :    PRINT : PRINT " SCHMATZ!!
888 :    LET Z = Z+1
890 :    PRINT : PRINT Z;"FLIEGEN GESCHNAPPT"
892 :    FOR I = 1 TO 1000 : NEXT I : REM VERDAUUNGSPAUSE
895 :
896 :    RETURN
899 :
900 REM === SPIELENDE ================================================
901 :
910 : PRINT : PRINT : PRINT
920 : PRINT "DIE MINUTE IST ABGELAUFEN.
930 :
990 : END
```

Das folgende Beispiel weist schon eine recht komplexe Struktur auf. Es soll zeigen, wie schwierig es ist, einem etwas umfangreicheren Spielprogramm einen klaren und verständlichen Aufbau zu geben. (Leider wird bei diesem Spiel wieder geschossen.)

Beispiel 5 : Ballons

Vom Himmel rieseln Ballons hernieder. Sie dürfen mittels einer beweglichen Kanone zum Platzen gebracht werden.

Hier der Bildschirm:



Das Programm weist die übliche Grobstruktur "Anleitung - Anfangswerte - Spielfeld - Spiel - Bewertung" auf. Schwierig ist der Programmteil 'Spiel'; hier müssen drei Bewegungsvorgänge koordiniert werden:

a) die Hin- und Herbewegung der Kanone,

b) das Herabrieseln der Ballons (vom Zufall gesteuert),

c) der Schuß.

Diese Teile sind durch Kommentare zwar gekennzeichnet, doch ist der Programmaufbau alles andere als durchsichtig. Sorgen Sie für Abhilfe!

```
1000 : PRINT "■"
1010 : PRINT "              BALLONS
1011 : PRINT "              --------
1012 :
1020 REM GRAPHISCHES REAKTIONS- UND GESCHICKLICHKEITSSPIEL
1090 :
2000 REM *** HAUPTPROGRAMM *****************************************
2001 :
2300 : GOSUB 3000 : REM ANLEITUNG
2310 :
2400 : GOSUB 4000 : REM ANFANGSWERTE
2410 :
2500 : GOSUB 5000 : REM SPIELFELD
2510 :
2600 : GOSUB 6000 : REM SPIEL
2610 :
2700 : GOSUB 7000 : REM BEWERTUNG
2710 :
2800 : END
2810 :
2900 REM *** ENDE HAUPTPROGRAMM *************************************
2980 :
2990 :
3000 REM +++ UNTERPROGRAMM 'ANLEITUNG' +++++++++++++++++++++++++++++++
3001 :
3010 : PRINT"■■IHRE AUFGABE BESTEHT DARIN, VOM HIMMEL
3020 : PRINT"■HERABRIESELNDE BALLONS ABZUSCHIESSEN.
3030 : PRINT"■SIE HABEN 25 VERSUCHE FUER 10 BALLONS.
3040 : PRINT"■IHRE KANONE BEWEGT SICH UNUNTERBROCHEN
3050 : PRINT"■AM BODEN HIN UND HER.
3060 : PRINT"■SIE SCHIESSEN MIT DER ■LEERTASTE■.
3070 : PRINT"■■TASTE DRUECKEN UND LOS GEHT'S!
3080 :
3200 : GET Q$ : IF Q$ = "" THEN 3200
3290 :
3300 : PRINT"■"
3390 :
3900 : RETURN
3950 :
4000 REM +++ UNTERPROGRAMM 'ANFANGSWERTE' ++++++++++++++++++++++++++++
4001 :
4010 : ZK =    193   :  REM KANONE
4020 : ZB =     81   :  REM BALLON
4030 : ZS =     93   :  REM KUGEL
4040 : ZR =    102   :  REM RAND
4050 : LE = 33528    :  REM LINKE  ECKE
4060 : RE = 33567    :  REM RECHTE ECKE
4070 : BP = 33548    :  REM BASIS-PUNKT (KANONE)
4080 : BR =      1   :  REM RICHTUNG
4090 : AB =     10   :  REM BALLONANZAHL
4110 : SZ =      0   :  REM ANZAHL DER SCHUSSZAHL
4120 : VZ =     25   :  REM ANZAHL DER VERSUCHE
4130 : ZL =     32   :  REM LEERZEICHEN
4140 :
4900 : RETURN
4995 :
```

```
5000 REM +++ UNTERPROGRAMM 'SPIELFELD' ++++++++++++++++++++++++++++++++
5001 :
5010 : FOR I = 32768 TO 32807
5020 :   POKE I,ZR
5030 :   POKE 760+I,ZR
5040 :   POKE 960+I,ZR
5060 : NEXT I
5090 :
5100 : FOR I = 33568 TO 33688 STEP 40
5110 :   POKE I,ZR
5120 :   POKE 39+I,ZR
5130 : NEXT I
5190 :
5200 : DIM B(40)
5290 :
5300 : FOR I = 1 TO 10
5310 :   LET K = INT(40*RND(TI)) + 32807
5320 :   IF PEEK(K) <> 32 THEN 5310
5330 :   LET B(I) = K
5340 :   POKE K,ZB
5350 : NEXT I
5390 :
5400 : POKE BP,ZK
5410 :
5500 PRINT"SQQQQQQQQQQQQQQQQQQQQQQ█VERSUCHE      SCHUESSE      BALLONS"
5510 :
5900 : RETURN
5999 :
6000 REM +++ UNTERPROGRAMM 'SPIEL' +++++++++++++++++++++++++++++++++++
6001 :
6100 :   PRINT"SQQQQQQQQQQQQQQQQQQQQQQQ"TAB(4)VZ"█ ";
6105 :   PRINT TAB(18)"█ " SZ TAB(31) AB "█ "
6140 :
6150 :   GET A$ : IF A$ = " " THEN 6300  : REM SCHUSS
6190 :
6200 :     REM --- HIN-UND-HERBEWEGUNG DES BASISPUNKTS (KANONE)
6201 :
6210 :       IF NOT(LE < BP AND BP < RE) THEN LET BR = -BR
6220 :       POKE BP,ZR : LET BP = BP+BR : POKE BP,ZK
6230 :
6290 :   GOTO 6150 : REM NEUER TASTENDRUCK
6295 :
6300 : REM --- SCHUSS UND FALLEN DER BALLONS
6310 :
6330 :   LET VZ = VZ-1 : LET SZ = SZ+1 : LET P = BP
6340 :
6350 :   LET P = P-40
6360 :   IF PEEK(P) <> ZB THEN 6410 : REM BALLON NOCH NICHT GETROFFEN
6370 :   LET AB = AB-1              : REM GETROFFEN, EIN BALLON WENIGER
6380 :   GOSUB 6700                 : REM DETONATION
6390 :   GOSUB 6900                 : REM LOESCHEN EINES TIEFFLIEGERS
6395 :   GOTO 6460                  : REM SCHUSS LOESCHEN
6400 :
6410 :   IF PEEK(P) <> ZR THEN 6430 : REM NOCH NICHT AUF DEM RAND
6415 :   POKE P+40, ZL              : REM SCHUSS AUF DEM RAND
6420 :   GOTO 6460                  : REM ZUM LOESCHEN
6425 :
6430 :   POKE P,ZS                  : REM KUGEL
6440 :   GOTO 6350                  : REM KUGEL RUECKT VOR
6450 :
```

```
6460 :    FOR I = BP-680 TO BP-40 STEP 40
6470 :      POKE I,ZL                    : REM SCHUSS LOESCHEN
6480 :    NEXT I
6490 :
6500 :    IF AB <= 0 OR VZ <= 0 OR VZ< AB THEN RETURN : REM SPIEL AUS
6510 :
6520 :    FOR I = 1 TO 10              : REM FALLEN DER BALLONS
6530 :      LET K = INT(10*RND(TI))+1
6540 :      IF B(K) = 0 THEN 6590
6550 :      LET B(K) = B(K)+40
6560 :      IF PEEK(B(K)) = ZR THEN LET VZ = 0 : REM BALLON UNTEN
6570 :      POKE B(K)-40,ZL
6580 :      POKE B(K),ZB
6590 :    NEXT I
6595 :
6610 :
6690 : GOTO 6100
6695 :
6700 : REM +++ UNTERPROGRAMM 'DETONATION' +++
6701 :
6710 :    FOR I = P-2 TO P+2 : POKE I,42 : NEXT I
6720 :
6730 :    FOR I = P-80 TO P+80 STEP 40
6740 :      IF PEEK(I) = ZR THEN 6760
6750 :      POKE I,42
6760 :    NEXT I
6770 :
6780 :   FOR I = 1 TO 14
6785 :     IF B(I) = P THEN LET B(I) = 0 : GOTO 6800
6790 :   NEXT I
6795 :
6800 :   FOR I = P-2 TO P+2 : POKE I,32 : NEXT I
6810 :
6820 :   FOR I = P-80 TO P+80 STEP 40
6830 :     IF PEEK(I) = ZR THEN 6850
6840 :     POKE I,32
6850 :   NEXT I
6860 :
6890 :   RETURN
6899 :
6900 : REM +++ UNTERPROGRAMM 'LOESCHEN EINES TIEFFLIEGERS' +++
6901 :
6910 :    FOR I = 1 TO 14
6920 :      IF B(I) = P THEN B(I) = 0
6930 :    NEXT I
6940 :
6990 :    RETURN
6999 :
7000 REM +++ UNTERPROGRAMM 'BEWERTUNG' ++++++++++++++++++++++++++++++++
7001 :
7100 : IF AB > 0 THEN LET Z$=" VERLOREN!" : GOTO 7120
7110 :                LET Z$=" GEWONNEN!"
7120 : PRINT"SIE HABEN" Z$
7130 : PRINT"WOLLEN SIE EIN NEUES SPIEL WAGEN?"
7140 : GET Q$ : IF Q$ = "J" THEN RUN
7150 : IF Q$ <> "N" THEN 7140
7190 :
7900 : RETURN
```

Beispiel 6 : Labyrinth

Computer und Spieler suchen - im Wettstreit gegeneinander - möglichst schnell durch ein auf dem Bildschirm gegebenes Labyrinth zu gelangen.

Wir bauen das Programm in der bewährten Weise aus den Teilen (1) Ausgabe der Spielregeln (falls gewünscht), (2) Initialisierung, (3) Labyrinthbau, (4) Spiel (Marsch durchs Labyrinth) und (5) Bewertung auf. Interessant ist zunächst Teil zwei: der Aufbau des Labyrinths. Damit bei jeder einzelnen Partie ein anderes Labyrinth erscheint, müssen wir Zufallselemente hineinbringen. Zunächst werden waagerechte Balken

XXXXX XXXX XXXXXXXXXX XXXXX XXXXX XXXXXXXXX

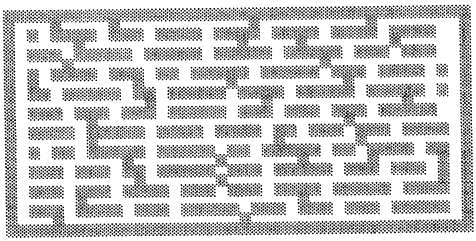
mit zufällig gesetzten Zwischenräumen gezeichnet; deren Dichte bestimmt die Variable D1 in folgender Weise:

```
4150  LET ZU = INT(D1 * RND(1) + 2)
```

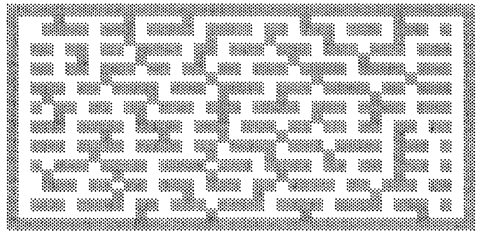
Dann werden zufällige Querverstrebungen eingefügt; ihre Dichte wird durch den Wert der Variablen D2 festgelegt:

```
4240  LET ZU = INT(D2 * RND(1) + 5)
```

Es können Labyrinthe folgender Art entstehen:



D1 = 3, D2 = 20



Das nebenstehende Labyrinth (D1 = 3, D2 = 6) besitzt keinen Durchgang von links oben (Start) nach rechts unten (Ziel); daher bauen wir in unser Programm einen Probelauf ein, der das erzeugte Labyrinth auf Durchlässigkeit prüft und gegebenenfalls einen Wegdurchbruch vornimmt.

Wie findet nun der Computer den Weg durch das Gewirr der Gänge? Er läuft immer rechts an der Wand entlang, d.h. zunächst versucht er einen Schritt nach unten ; ist dieser versperrt, weicht er nach rechts aus, dann nach oben und schließlich nach links.

Nach jedem Schritt des Computers erhält der Spieler Gelegenheit, seine 'Maus' mittels der Zahltasten 2,4,6,8 in der bekannten Weise zu steuern; nimmt er diese Gelegenheit nicht wahr, so tut der Computer einen weiteren Schritt.

Der große Vorzug des menschlichen Spielers vor dem Computer ist, daß er Wege mit einem Blick überschauen kann und Sackgassen sofort erkennt, während der Computer keinen Überblick hat, sondern jedes Winkelchen durchsucht. Sein Trumpf ist allein die Geschwindigkeit. Genau dieser Unterschied beherrscht auch die strategischen Spiele (wie z.B. Schach). Es ist eine der schwierigsten Aufgaben, dem Computer ein gewisses Maß an Voraussicht und die Fähigkeit zum Bewerten komplexer Spielsituationen beizubringen.-

Das Programm auf den folgenden Seiten ist schon recht umfangreich; ich habe versucht, es möglichst durchsichtig zu gestalten.

Manche Teile sind noch etwas umständlich gestaltet (z.B. das Unterprogramm 'Marsch durchs Labyrinth' oder die Steuerung der Maus durch den Spieler (Zeilen 8100 bis 8170)); hier gibt es viele Verbesserungsmöglichkeiten. Interessant ist die Cursorsteuerung mittels Zeichenketten (in den Zeilen 3020 bis 3070 und 4400, 4410, 4510, 4580): aus der ein für allemal definierten Kette der Steuerzeichen werden jeweils soviele herausgegriffen, wie man benötigt.

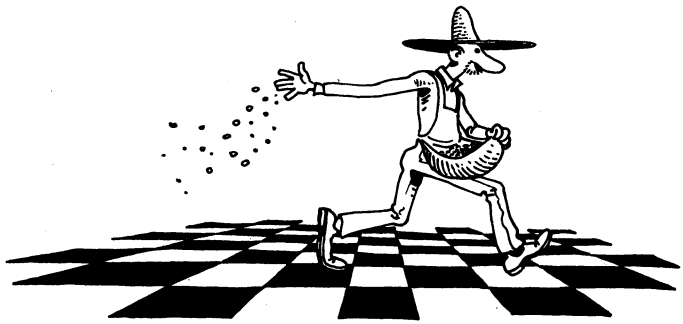
```
1000 : PRINT "■"
1010 : PRINT "            LABYRINTH
1011 : PRINT "            ---------
1020 :
1030 REM UEBERLEGUNGS- UND GESCHICKLICHKEITSSPIEL
1090 :
1099 :
1200 REM *** HAUPTPROGRAMM ***********************************************
1201 :
1220 : GOSUB 2000 : REM SPIELREGELN
1221 :
1230 : GOSUB 3000 : REM INITIALISIERUNG
1231 :
1240 : GOSUB 4000 : REM LABYRINTHBAU
1241 :
1250 : GOSUB 5000 : REM SPIEL
1251 :
1290 : GOSUB 9000 : REM BEWERTUNG UND ENDE
1291 :
1300 : END
1301 :
1900 REM *** ENDE HAUPTPROGRAMM ***************************************
1999 :
2000 REM +++ UNTERPROGRAMM 'SPIELREGELN' ++++++++++++++++++++++++++++
2001 :
2010 : PRINT
2020 : PRINT "SOLL ICH IHNEN DIE SPIELREGELN
2025 : PRINT "MITTEILEN?(J/N)
2030 : GET A$ : IF A$ = "" THEN 2030
2040 : IF A$ <> "J" THEN RETURN
2090 :
2100 : REM HIER FUEGEN SIE BITTE DIE SPIELREGELN EIN
2110 :
2900 : RETURN
2999 :
3000 REM +++ UNTERPROGRAMM 'INITIALISIERUNG' ++++++++++++++++++++++++
3001 :
3010 : REM --- ZEICHENKETTEN ZUR CURSORSTEUERUNG
3011 :
3020 :   LET CU$ = "■" : REM CURSOR NACH UNTEN
3030 :   LET CR$ = "■" : REM CURSOR NACH RECHTS
3040 :   FOR I = 1 TO 6
3050 :     LET CU$ = CU$ + CU$
3060 :     LET CR$ = CR$ + CR$
3070 :   NEXT I
3090 :
3100 : REM --- FESTLEGUNG DES SCHWIERIGKEITSGRADES
3101 :
3110 :   LET S = 20    : REM BESTIMMT DIE LAUFGESCHWINDIGKEIT
3120 :   LET D1 = 6    : REM BESTIMMT DIE DICHTE DES LABYRINTHS
3130 :   LET D2 = 12   : REM BESTIMMT DIE DICHTE DES LABYRINTHS
3190 :
3200 : REM --- START- UND ZIELKOORDINATEN
3201 :
3210 :  LET SP = 32809 : REM STARTPUNKT
3220 :  LET ZI = 33646 : REM ZIEL
3290 :
3900 : RETURN
3999 :
```

```
4000 REM +++ UNTERPROGRAMM 'LABYRINTHBAU' ++++++++++++++++++++++++++++
4001 :
4010 : PRINT "□"
4020 :
4030 : REM --- WAAGERECHTE RANDBALKEN
4031 :
4050 :   FOR I = 32768 TO 32807 : POKE I,102 : NEXT I
4070 :   FOR I = 33648 TO 33687 : POKE I,102 : NEXT I
4080 :
4090 : REM --- WAAGERECHTE MITTELBALKEN
4091 :
4100 :   FOR P = 32848 TO 33568 STEP 80
4110 :     LET ZU = 0
4120 :     FOR I = 0 TO 39
4130 :       IF ZU > 0 THEN 4160
4140 :       IF ZU = 0 THEN LET ZU = -1 : GOTO 4180
4150 :       LET ZU = INT(D1*RND(TI)+2)
4160 :       POKE P+I,102
4170 :       LET ZU = ZU-1
4180 :     NEXT I
4190 :   NEXT P
4195 :
4200 : REM --- SENKRECHTE QUERVERSTREBUNGEN
4201 :
4210 :   LET ZU = 4
4220 :   FOR P = 32808 TO 33608 STEP 80
4230 :     FOR I = 0 TO 39
4240 :       IF ZU = 0 THEN POKE P+I,102 : LET ZU = INT(D2*RND(TI)+5)
4260 :       LET ZU = ZU-1
4270 :     NEXT I
4280 :   NEXT P
4285 :
4290 : REM --- SENKRECHTE RANDBALKEN
4291 :
4300 :   FOR U = 32808 TO 33608 STEP 40
4310 :     POKE U,102
4320 :     POKE U+1,32
4340 :     POKE U+38,32
4350 :     POKE U+39,102
4360 :   NEXT U
4380 :
4390 : REM --- BESCHRIFTUNG
4391 :
4400 :   PRINT "■■START■
4410 :   PRINT "■" LEFT$(CU$,22) LEFT$(CR$,36) "■ZIEL■
4490 :
4500 : REM --- PROBEDURCHLAUF
4501 :
4510 :   PRINT "■" LEFT$(CU$,23);
4520 :   PRINT " ICH SUCHE EINEN WEG VOM START ZUM ZIEL
4530 :
4540 :   LET F$ = "PROBE"
4550 :   GOSUB 6000 : REM MARSCH DURCHS LABYRINTH
4560 :   LET F$ = "ERNST"
4570 :
4580 :   PRINT "■" LEFT$(CU$,23);
4581 :   PRINT "              WEG EXISTIERT!
4590 :
4900 : RETURN
4999 :
```

```
5000 REM +++ UNTERPROGRAMM 'SPIEL' ++++++++++++++++++++++++++++++++++
5001 :
5030 : POKE SP,0                    : REM COMPUTERMAUS AM STARTPUNKT
5040 : POKE SP+1,38                 : REM SPIELERMAUS DANEBEN
5050 :
5060 : FOR I = 1 TO 1000 : NEXT I : REM WARTESCHLEIFE
5090 :
5100 : PRINT "S" LEFT$(CU$,23);
5110 : PRINT "              LOS GEHT'S               ";
5120 :
5200 : GOSUB 6000                   : REM MARSCH DURCHS LABYRINTH
5210 :
5900 : RETURN
5999 :
6000 REM +++ UNTERPROGRAMM 'MARSCH DURCHS LABYRINTH' ++++++++++++++++++
6001 :
6010 :   LET PL = SP : LET PP = SP+1 : REM STARTPOSITIONEN
6020 :   GOTO 6400                   : REM ERST EIN SCHRITT NACH UNTEN
6070 :
6080 : REM --- SUCHSCHRITTE
6090 :
6100 :   LET FE = PL+1               : REM SCHRITT NACH RECHTS
6110 :   IF PP = ZI THEN LET FE = PP : GOTO 6800 : REM LAUF BEENDET
6120 :   IF FE = ZI THEN 6800
6130 :   IF PEEK(FE) <> 38 THEN 6170
6140 :   GOSUB 7000                 : REM UEBERSPRINGEN
6150 :   GOSUB 8000                 : REM SPIELERSTEUERUNG
6160 :   IF U = 0 THEN 6100
6170 :   IF PEEK(FE) = 32 THEN GOSUB 8000 : GOTO 6400 : REM NACH UNTEN
6190 :
6200 :   LET FE = PL-40              : REM SCHRITT NACH OBEN
6205 :   IF F$ = "ERNST" THEN 6210
6206 :   IF FE <= SP-40 THEN GOSUB 6900 : GOTO 6000
6210 :   IF PP = ZI THEN LET FE = PP : GOTO 6800
6230 :   IF PEEK(FE) <> 38 THEN 6270
6240 :   GOSUB 7000
6250 :   GOSUB 8000
6260 :   IF U = 0 THEN 6200
6270 :   IF PEEK(FE) = 32 THEN GOSUB 8000 : GOTO 6500 : REM NACH RECHTS
6290 :
6300 :   LET FE = PL-1               : REM SCHRITT NACH LINKS
6310 :   IF PP = ZI THEN LET FE = PP : GOTO 6800
6330 :   IF PEEK(FE) <> 38 THEN 6370
6340 :   GOSUB 7000
6350 :   GOSUB 8000
6360 :   IF U = 0 THEN 6300
6370 :   IF PEEK(FE) = 32 THEN GOSUB 8000 : GOTO 6200 : REM NACH OBEN
6390 :
6400 :   LET FE = PL+40              : REM SCHRITT NACH UNTEN
6410 :   IF PP = ZI THEN LET FE = PP : GOTO 6800
6420 :   IF FE = ZI THEN 6800
6430 :   IF PEEK(FE) <> 38 THEN 6470
6440 :   GOSUB 7000
6450 :   GOSUB 8000
6460 :   IF U = 0 THEN 6400
6470 :   IF PEEK(FE) = 32 THEN GOSUB 8000 : GOTO 6300 : REM NACH LINKS
6480 :   GOTO 6100
6490 :
```

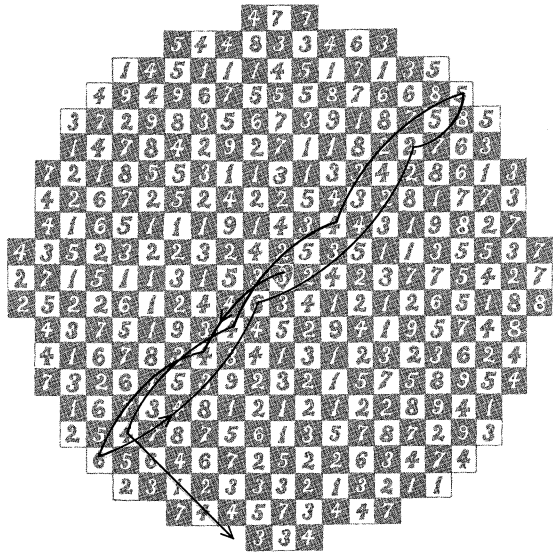
```
6500 :   LET FE = PL+1                 : REM SCHRITT NACH RECHTS
6510 :   IF PP = ZI THEN LET FE = PP : GOTO 6800
6520 :   IF FE = ZI THEN 6800
6530 :   IF PEEK(FE) <> 38 THEN 6570
6540 :   GOSUB 7000
6550 :   GOSUB 8000
6560 :   IF U = 0 THEN 6500
6570 :   IF PEEK(FE) = 32 THEN GOSUB 8000 : GOTO 6600 : REM NACH UNTEN
6580 :   GOTO 6200
6599 :
6600 :   LET FE = PL+40                : REM SCHRITT NACH UNTEN
6610 :   IF PP = ZI THEN LET FE = PP : GOTO 6800
6620 :   IF FE = ZI THEN 6800
6630 :   IF PEEK(FE) <> 38 THEN 6670
6640 :   GOSUB 7000
6650 :   GOSUB 8000
6660 :   IF U = 0 THEN 6600
6670 :   IF PEEK(FE) = 32 THEN GOSUB 8000 : GOTO 6300 : REM NACH LINKS
6680 :   GOTO 6500
6690 :
6800 : REM --- DURCHLAUF BEENDET
6801 :
6810 :    IF PP = FE THEN RETURN : REM DURCHLAUF BEENDET
6820 :   GOSUB 8000
6830 :
6890 :   RETURN
6899 :
6900 : REM +++ UNTERPROGRAMM 'WEG-DURCHBRUCH' +++
6901 :
6910 :   FOR K = 1 TO 10
6920 :     LET D = INT(840*RND(1))
6930 :     LET G = 32768 + D
6940 :     IF D < 41 OR D = 40*INT(D/40) THEN 6920
6950 :     IF ZI < G OR D+1 = 40*INT((D+1)/40) THEN 6920
6960 :     IF PEEK(G) = 32 THEN 6920
6970 :     POKE G,32 : REM HINDERNIS WEGRAEUMEN
6980 :   NEXT K
6985 :
6990 :   RETURN
6999 :
7000 : REM +++ UNTERPROGRAMM 'UEBERSPRINGEN' +++
7001 :
7010 :   LET U = 0
7020 :   IF PEEK(PP+1) <> 32 AND PEEK(PP-1) <> 32 THEN 7040
7030 :   GOTO 7070
7040 :   IF PEEK(PP+40) <> 32 AND PEEK(PP-40) <> 32 THEN 7060
7050 :   GOTO 7070
7060 :   LET U = 1
7070 :   LET FE = PL
7090 :
7900 :   RETURN
7999 :
```

```
8000 : REM +++ 'WEITERSCHALTEN UND SPIELERSTEUERUNG' +++
8001 :
8010 :   POKE PL,32                    : REM ALTE POSITION LOESCHEN
8020 :   LET PL = FE                   : REM NEUE POSITION WIRD ZUR ALTEN
8030 :   IF F$ = "PROBE"THEN RETURN    : REM PROBEDURCHLAUF
8090 :   POKE PL,0                     : REM COMPUTERMAUS ANZEIGEN
8100 :   GET P$ : IF P$ <> "" THEN LET R$ = P$ : REM TASTE GEDRUECKT
8120 :   IF R$ = "2" THEN LET FE = PP+40 : GOTO 8170
8130 :   IF R$ = "4" THEN LET FE = PP-1  : GOTO 8170
8140 :   IF R$ = "5" THEN RETURN
8150 :   IF R$ = "6" THEN LET FE = PP+1  : GOTO 8170
8160 :   IF R$ = "8" THEN LET FE = PP-40 : GOTO 8170
8170 :   IF PEEK(FE) = 32 THEN POKE PP,32 : LET PP = FE : POKE PP,38
8180 :
8200 :   IF F$ = "ERNST" THEN  FOR T = 1 TO S : NEXT T
8300 :
8900 :   RETURN
8999 :
9000 REM +++ UNTERPROGRAMM 'BEWERTUNG' ++++++++++++++++++++++++++++++++
9001 :
9100 : PRINT "■" LEFT$(CU$,23);  : REM CURSOR AUF UNTERSTE ZEILE
9110 : IF PP = ZI THEN PRINT "            SIE HABEN ";: GOTO 9130
9120 : PRINT "            ICH HABE ";
9130 : PRINT "GEWONNEN!            ";
9190 :
9200 : FOR Y = 1 TO 1000 : GET A$ : NEXT Y
9210 :
9300 : PRINT "□" : PRINT : PRINT
9310 : PRINT "NOCH EINMAL?(J/N)
9320 : GET A$ : IF A$ = "" THEN 9320
9330 : IF A$ = "J" THEN RUN
9390 :
9900 : RETURN
9999 :
```



Unter dem Titel "Zurück vom Klondike" veröffentlichte Sam Loyd, Amerikas berühmtester Rätsel-Spezialist und Spiele-Erfinder im Jahre 1907 folgende Aufgabe:

Beispiel 7 : Zahlenlabyrinth



Starten Sie beim Herzen in der Mitte. Gehen Sie dann in irgendeine der acht Himmelsrichtungen drei Schritte geradeaus, also nach Norden, Süden, Osten, Westen oder schräg, d.h. nach Nordosten, Nordwesten, Südosten oder Südwesten. Danach erreichen Sie ein Quadrat mit einer Nummer; sie gibt an, wieviel Schritte Sie am zweiten Tag der Reise tun müssen bzw. können. Von jedem neuen Punkt aus marschieren Sie in irgendeiner Richtung soviel Schritte weiter, wie die zugehörige Zahl angibt. Dies wiederholen sie solange, bis Sie auf ein Quadrat mit einer Zahl gelangen, von dem aus Sie nur einen Schritt über die Waldgrenze hinauskommen. Wenn Sie erst mal aus dem Wald heraussind, können Sie toben und schreien, soviel Sie wollen, denn dann haben Sie das Rätsel gelöst.

Statt mit dem Bleistift Linien auf Papier zu zeichnen, schreiben wir das Labyrinth auf den Bildschirm und geben unsere Marschrichtung über Tasten ein. Der Computer summiert die zurückgelegte Strecke und meldet Erfolg oder Mißerfolg.

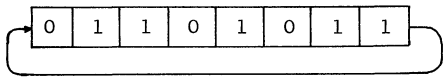
```
1000 : PRINT "■"
1010 : PRINT "          ZAHLENLABYRINTH
1011 : PRINT "          ---------------
1012 :
1020 REM SPIEL NACH EINER IDEE VON SAM LOYD
1021 :
1999 :
2000 REM *** HAUPTPROGRAMM ****************************************
2001 :
2300 : GOSUB 3000 : REM SPIELANLEITUNG
2301 :
2400 : GOSUB 4000 : REM INITIALISIERUNG
2401 :
2500 : GOSUB 6000 : REM SPIELFELD
2501 :
2600 : GOSUB 5000 : REM SPIEL
2601 :
2700 : GOSUB 7000 : REM WIEDERHOLUNG ODER VERABSCHIEDUNG
2701 :
2800 : END
2801 :
2900 REM *** ENDE DES HAUPTPROGRAMMS ********************************
2998 :
2999 :
3000 REM +++ UNTERPROGRAMM 'ANLEITUNG' ++++++++++++++++++++++++++++++
3001 :
3010 : PRINT
3020 : PRINT"■ SIE BEFINDEN SICH IN DER UNANGE-
3030 : PRINT"■ NEHMEN LAGE, AUS EINEM GROSSEN
3040 : PRINT"■ WALD HERAUS ZU MUESSEN. DABEI
3050 : PRINT"■ KOENNEN SIE DIE JEWEILIGE MARSCH-
3060 : PRINT"■ RICHTUNG DURCH DRUECKEN EINER TASTE
3070 : PRINT"■ VON 1 BIS 9 (OHNE 5) FESTLEGEN.
3080 : PRINT"■ SIE SIND JEDOCH ERST DANN ENT-
3090 : PRINT"■ KOMMEN, WENN SIE SICH GENAU EINEN
3100 : PRINT"■ SCHRITT AUSSERHALB DES WALDES (OHNE
3110 : PRINT"■ UMRANDUNG) BEFINDEN!
3120 :
3150 : GET A$ : IF A$ = "" THEN 3150
3190 :
3200 : PRINT"■■■■■■ DAS FELD AUF HELLEM GRUND ZEIGT
3210 : PRINT"■ IHNEN, WO SIE SICH BEFINDEN; DIE ZAHL
3220 : PRINT"■ AUF DEM FELD ZEIGT IHNEN DIE STRECKE,
3230 : PRINT"■ DIE SIE ZURUECKLEGEN KOENNEN (MUESSEN).
3240 : PRINT"■       V I E L  E R F O L G !
3250 :
3260 : GET A$ : IF A$ = "" THEN 3260
3290 :
3300 : PRINT"■"
3310 : PRINT"■■ DIE BEWEGUNGEN ERFOLGEN SO:
3320 : PRINT"■    8   :   NORD
3330 : PRINT"■    9   :   NORDOST
3340 : PRINT"■    6   :   OST
3350 : PRINT"■    3   :   SUEDOST
3360 : PRINT"■    2   :   SUED
3370 : PRINT"■    1   :   SUEDWEST
3380 : PRINT"■    4   :   SUEDWEST
3390 : PRINT"■    7   :   NORDWEST
3400 : PRINT"■ MIT TASTE '5' KOENNEN SIE AUFGEBEN!
3410 :
```

```
3420 : GET A$ : IF A$ = "" THEN 3420
3430 :
3900 : RETURN
3999 :
4000 REM +++ UNTERPROGRAMM 'INITIALISIERUNG' ++++++++++++++++++++++++++
4001 :
4010 :   LET AP = 33260  : REM AUFENTHALTSPUNKT
4020 :   LET S  =     0  : REM EINZELSTRECKE
4030 :   LET GS =     0  : REM GESAMTSTRECKE
4040 :   LET TZ =     0  : REM ANZAHL DER TAGE
4050 :
4100 : REM --- RICHTUNGEN
4101 :
4120 :   FOR I = 1 TO 9
4130 :     READ B(I) : READ B$(I)
4140 :   NEXT I
4145 :   RESTORE
4150 :
4200 DATA 39,SUEDWEST,40,SUED,41,SUEDOST
4210 DATA -1,WEST,,,1,OST,-41,NORDWEST
4220 DATA -40,NORD,-39,NORDOST
4300 :
4900 : RETURN
4999 :
5000 REM +++ UNTERPROGRAMM 'SPIEL' ++++++++++++++++++++++++++++++++++++
5001 :
5010 : LET S  = PEEK(AP)-176
5020 : LET GS = GS + S
5030 : LET TZ = TZ + 1
5040 :
5050 : POKE 32958,63
5060 : FOR I = 32959 TO 32967 : POKE I,32 : NEXT I
5090 :
5100 : GET R : IF R = 0 THEN 5100
5110 : IF B(R) = 0 THEN GOSUB 8000 : END : SPIELER GIBT AUF
5120 : IF PEEK(AP+B(R)*S)=102 AND PEEK(AP+B(R)*(S-1))<>102 THEN RETURN
5130 : IF PEEK(AP+B(R)*S) < 49  THEN 5100
5140 : IF PEEK(AP+B(R)*S) > 57  THEN 5100
5150 :
5200 : GOSUB 5400    : REM SPIELFELDAUSGABE
5210 :
5300 : POKE AP,S+48 : REM REVERTIEREN
5310 : LET AP = AP + (B(R)*S)
5320 : POKE AP,(PEEK(AP)+128)
5340 :
5350 : FOR I = 1 TO 1000 : NEXT I
5360 : GOTO 5000
5370 :
5400 : REM --- VARIABLE DRUCKEN
5401 :
5410 : PRINT"■■■■■";SPC(30);B$(R)
5420 : PRINT"■■■■";SPC(33);S
5430 : PRINT"■■■■■";SPC(35-LEN(STR$(GS)));GS
5450 : PRINT"■■■■■";SPC(37-LEN(STR$(TZ)));TZ
5500 :
5900 : RETURN
5999 :
```

```
6000 REM +++ UNTERPROGRAMM 'AUSGABE' ++++++++++++++++++++++++++++++++++
6001 :
6010 : PRINT"■"
6020 : PRINT"            ■■■■■"
6030 : PRINT"         ■■■■477■■■■"
6040 : PRINT"      ■■■544833463■■■"
6045 : PRINT"     ■■■1451114517135■■■"
6050 : PRINT"    ■■■4949675558766858■■■"
6060 : PRINT"    ■372983567391876858■"
6070 : PRINT"   ■■■1478429271182276■■■"
6080 : PRINT"   ■721855311313342861■■"
6090 : PRINT"   ■426725242254328177■■"
6100 : PRINT" ■■■4165111914344198278■■■"
6110 : PRINT" ■435232232425351355378■■"
6120 : PRINT" ■271511315332423775278■■"
6130 : PRINT" ■252281244634121265188■■"
6140 : PRINT" ■■■4375193445294195748■■■"
6150 : PRINT"  ■■41678343413123236248■■"
6160 : PRINT"  ■732615392321575895■■"
6170 : PRINT"  ■■■1673481212122894■■■"
6180 : PRINT"    ■254787561357872938■"
6190 : PRINT"    ■■■6564672522634748■■■"
6200 : PRINT"      ■■■2312333213211■■■"
6210 : PRINT"       ■■■■744573447■■■■"
6220 : PRINT"          ■■■■■334■■■■■"
6230 : PRINT"              ■■■■■"
6240 :
6300 : POKE 33260,179 : REM ANFANGSFELD
6310 :
6320 : PRINT"■■■";TAB(30);"RICHTUNG:"
6330 : PRINTTAB(30);"──────────"
6340 :
6350 : PRINT"■■■";TAB(31);"STRECKE:"
6360 : PRINTTAB(31);"──────────"
6370 : PRINTTAB(36);"KM"
6380 :
6390 : PRINT"■■";TAB(31);"GESAMT-"
6400 : PRINTTAB(31);"STRECKE:"
6410 : PRINTTAB(31);"──────────"
6420 : PRINTTAB(36);"KM"
6490 :
6500 : PRINT"■■■";TAB(31);"T A G E:"
6510 : PRINTTAB(31);"──────────"
6520 : PRINT"■"
6600 :
6900 : RETURN
6999 :
7000 REM +++ UNTERPROGRAMM 'WIEDERHOLUNG ODER VERABSCHIEDUNG' ++++++++
7001 :
7010 : PRINT"■"
7020 : PRINT"■       G R A T U L I E R E !
7030 : PRINT"■       ═══════════════════
7040 : PRINT"■■■ ENTGEGEN ALLEN ERWARTUNGEN HABEN
7050 : PRINT"■ SIE ES TATSAECHLICH GESCHAFFT, AUS
7060 : PRINT"■ DEM WALD HERAUSZUKOMMEN!
7070 : PRINT"■ DAZU BENOETIGTEN SIE";TZ;"TAGE.
7080 : PRINT"■ IM GANZEN LEGTEN SIE DIE STOLZE
7090 : PRINT"■ STRECKE VON";GS;" KILOMETERN
7100 : PRINT"■ ZURUECK!
7110 : PRINT"■■    WOLLEN SIE ES NOCHMAL VERSUCHEN?
7115 :
```

```
7120 : GET A$ : IF A$ = "" THEN 7120
7130 : IF A$ = "J" THEN RUN
7140 : IF A$ <>"N" THEN 7120
7150 : PRINT"■"
7190 :
7900 : RETURN
7999 :
8000 REM +++ UNTERPROGRAMM 'SPIELAUFGABE' ++++++++++++++++++++++++++++
8001 :
8010 : PRINT"■■■ SOSO! SIE WOLLEN ALSO AUFGEBEN.
8020 : PRINT"■■ DAS HABE ICH MIR GLEICH GEDACHT,
8030 : PRINT"■ ALS ICH SIE SPIELEN GESEHEN HABE.
8040 : PRINT"■ OBWOHL SIE";TZ;"TAGE UMHER-
8050 : PRINT"■ GEIRRT SIND , UND DABEI EINE
8060 : PRINT"■ STRECKE VON";GS;"KILOMETERN ZURUECK-
8070 : PRINT"■ GELEGT HABEN, STECKEN SIE NOCH
8080 : PRINT"■ IMMER IM TIEFSTEN BUSCH.
8090 : PRINT"■ MEIN BEILEID ZU DIESER LEISTUNG!
8100 :
8900 : RETURN
8999 :
```

Obwohl es nicht im engeren Sinne zur Graphik gehört, wollen wir am Ende dieses Kapitels noch einen Ausflug ins R e i c h d e r T ö n e unternehmen.
Zu besonderen Gelegenheiten (z.B. beim Einschalten oder bei einem Bedienungsfehler) geben viele Computer ein Tonsignal ab. Wie kommt dies zustande? Nun - jeder Mikroporzessor (das 'Herz' unseres Computers) enthält ein Bauteil, welches den zeitlichen Ablauf aller Schaltvorgänge nach einem festen Takt steuert: den sogenannten Taktgenerator.
Wie wir aus dem Physikunterricht wissen, wird ein Ton oder ein Geräusch einfach dadurch erzeugt, daß die Membran eines Lautsprechers in einer bestimmten Frequenz zum Schwingen gebracht wird.
Viele Mikrocomputer besitzen einen kleinen Lautsprecher (bzw. ein solcher ist anschließbar) - jetzt geht es also nur noch darum, die Frequenz des Taktgenerators geeignet umzuwandeln.
Commodore-Computer bieten hierfür folgende Möglichkeit: sie besitzen ein gewisses Schieberegister



in dem ein Bit-Muster im Kreis geschoben werden kann. Ob dies geschieht, bestimmt der Inhalt der Speicherzelle mit der Adresse 59467: steht in ihr eine 16, so darf das Register frei laufen (normalerweise steht eine Null darin). Der Inhalt der Speicherzelle Nr. 59464 bestimmt die Frequenz des Umlaufs und damit die Tonhöhe: bei jedem Umlauf wird an den eingebauten Lautsprecher (bzw. an den sogenannten User Port) ein Impuls abgegeben. Der Inhalt der Zelle Nr. 59466 schließlich bestimmt das im Schieberegister befindliche Bitmuster und damit die Klangfarbe des erzeugten Tons. (Diese Darstellung ist noch etwas vage; für genauere Darlegungen fehlt uns hier der Platz. Bitte sehen Sie bei den Literaturhinweisen in den Anmerkungen zu diesem Kapitel nach!)

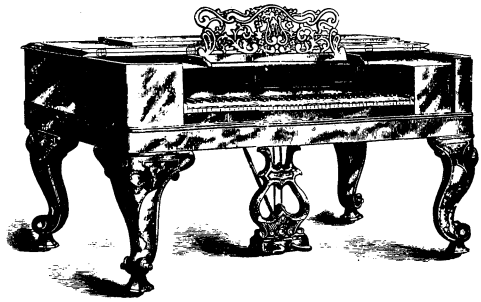
Wir haben also folgende Anweisungen zur Tonerzeugung:

POKE 59467,16	Ton an
POKE 59467,0	Ton aus
POKE 59464,h (0 $\leq$ h $\leq$ 255)	Tonhöhe
POKE 59466,k (0 $\leq$ k $\leq$ 255)	Klangfarbe

Erforschen Sie die Möglichkeiten mittels folgendem Programm!

```
100 : PRINT "□"
110 : PRINT "        TONGENERATOR
111 : PRINT "        ------------
112 :
120 REM DEMONSTRATIONSPROGRAMM FUER DIE TONERZEUGUNG
121 :
130 : PRINT
135 : PRINT "DAUERTON ..................... 1
140 : PRINT "SIRENE ....................... 2
150 : PRINT "ZUFALLSMUSIK ................. 3
160 : PRINT "ENDE ......................... 9
165 :
170 : GET A$ : IF A$ = "" THEN 170
175 : ON VAL(A$) GOSUB 200, 500, 600
180 :
185 : IF A$ <> "9" THEN 130
189 :
190 : POKE 59464,0
191 : POKE 59466,0
192 : POKE 59467,0
195 : END
199 :
200 REM +++ UNTERPROGRAMM 'DAUERTON' ++++++++++++++++++++++++++++++++
201 :
210 : PRINT "□"
220 : PRINT "SIE KOENNEN EINEN DAUERTON MIT VORGEGEBENER TONHOEHE
225 : PRINT "UND KLANGFARBE ERZEUGEN.
230 : PRINT
235 : INPUT "TONHOEHE (ZWISCHEN 1 UND 255)     "; H
240 : IF H < 1 OR H > 255 THEN 235
245 : INPUT "KLANGFARBE (ZWISCHEN 1 UND 255 ) "; K
250 : IF K < 1 OR K > 255 THEN 245
255 :
260 : PRINT
265 : PRINT "TON AN --> TASTE DRUECKEN!
270 : GET A$ : IF A$ = "" THEN 270
280 :
290 : POKE 59467,16 : REM TONGENERATOR ANSCHALTEN
300 : POKE 59464,H  : REM TONHOEHE WAEHLEN
310 : POKE 59466,K  : REM KLANGFARBE WAEHLEN
320 :
```

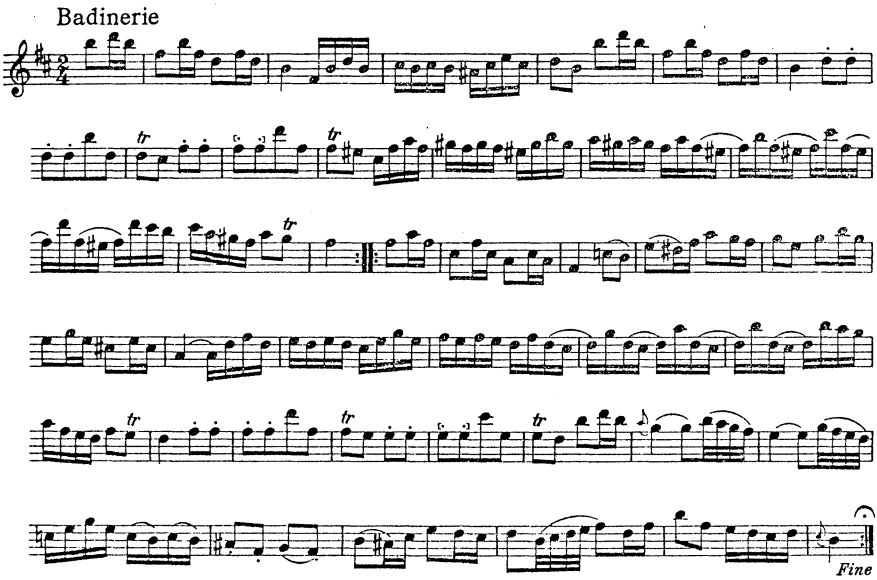
```
350 : PRINT
360 : PRINT "WENN SIE EINE ANDERE TONHOEHE ODER KLANGFARBE WUENSCHEN,
365 : PRINT "DRUECKEN SIE 'A'!
370 : PRINT "WENN SIE ZUM MENUE ZURUECKKEHREN WOLLEN, DRUECKEN SIE '@'!
375 :
380 : GET A$ : IF A$ = "" THEN 380
390 :
400 : IF A$ = "A" THEN 230 : REM ANDERE TONHOEHE ODER KLANGFARBE
410 : IF A$ = "@" THEN POKE 59467,0 : RUN
420 :
490 : GOTO 350
499 :
500 REM +++ UNTERPROGRAMM 'SIRENE' ++++++++++++++++++++++++++++++++++++
501 :
510 : PRINT "□"
520 : PRINT "           SIRENE
521 : PRINT "           ------
522 :
530 : PRINT "■■■■■■■■■■■ZUM BEENDEN TASTE DRUECKEN!
535 :
540 : POKE 59467,16                                  : REM TON AN
545 : POKE 59466,1                                   : REM KLANGFARBE
550 : FOR I = 10 TO 245 : POKE 59464,I : NEXT I : REM ANSCHWELLEN
555 : FOR I = 245 TO 10 STEP -1 : POKE 59464,I  : NEXT I
565 :
570 : GET A$ : IF A$ = "" THEN 540
575 :
580 : POKE 59467,0                                   : REM TON AUS
585 :
590 : RETURN
599 :
600 REM +++ UNTERPROGRAMM 'ZUFALLSMUSIK' ++++++++++++++++++++++++++++++
601 :
610 : PRINT "□"
620 : PRINT "            ZUFALLSMUSIK
621 : PRINT "            ------------
622 :
630 : PRINT "■■■■■■■■■■■ZUM BEENDEN TASTE DRUECKEN!
635 :
640 : POKE 59467,16                          : REM TON AN
645 : POKE 59466,50
650 : LET Z = INT(255*RND(TI)+1)             : REM ZUFALLSFREQUENZ
660 : POKE 59464,Z                           : REM ZUFALLSTON
665 :
670 : GET A$ : IF A$ = "" THEN 650
675 :
680 : POKE 59467,0                           : REM TON AUS
685 :
690 : RETURN
699 :
```



Bach

Beispiel 8 : Badinerie

Es soll ein Programm geschrieben werden, welches den Computer gegebene Musikstücke spielen läßt.



Jedes Musikstück besteht aus einer Folge von Paaren (Note, Tonlänge), wobei die Tonlänge die Zeitdauer des von der Note bezeichneten Ton ist. In dieser Form schreiben wir unser Musikstück in DATA-Zeilen:

```
a,2,z,2,c1,2,a,2,...      .
```

Diese Paare werden in ein Variablenpaar R$,T nacheinander eingelesen. Die Umwandlung von R$ in die zugehörige frequenz-bestimmende Zahl erfolgt in einem Unterprogramm; z.B.

```
510 IF R$ = "A" THEN LET R = 140 : RETURN .
```

Das heißt: in Speicherzelle Nr. 59464 wird R = 140 eingeschrieben, um den Ton A zu erzeugen. Seine Länge erhalten wir durch eine kleine Warteschleife

```
FOR I = 1 TO T1*T - T2 : NEXT I
```

wobei T1 und T2 geeignet festzulegende Konstanten sind.
Der Algorithmus lautet

MUSIKSTÜCK

```
WIEDERHOLE
  Lies Note R$ und Dauer T
  Übersetze R$ in die frequenz-bestimmende Zahl R
  Schreibe  R  in Speicherzelle 59464
  Durchlaufe Warteschleife der Dauer  T
BIS  R$ = "Ende"
```

```
100 : PRINT "■"
110 : PRINT "              BADINERIE
111 : PRINT "              ---------
112 :
120 REM BEISPIEL FUER EIN KOMPOSITIONSPROGRAMM
121 :
130 : PRINT "■■■■           LIEBEN SIE BACH?■■
135 : PRINT " DANN FREUEN SIE SICH SICHER UEBER DIE■
140 : PRINT "    BADINERIE AUS DER H-MOLL-SUITE■
150 : PRINT "      FUER FLOETE UND ORCHESTER.
160 : PRINT "■■■            TASTE DRUECKEN!
170 :
180 : GET A$ : IF A$ = "" THEN 180
190 :
```

```
200 REM *** HAUPTPROGRAMM ***********************************************
201 :
210 : LET T1 = 5                    : REM BESTIMMT TEMPO
215 :
220 : POKE 59467,16                 : REM TON AN
230 : POKE 59466,10                 : REM KLANGFARBE
240 : POKE 59464,0                  : REM FREQUENZ NULL
250 :
260 : READ R$,T                     : REM EINLESEN VON NOTE UND DAUER
265 :   IF R$ = "@" THEN 350        : REM STUECK ZU ENDE
270 :   GOSUB 400                   : REM UEBERSETZUNG NOTE -> FREQUENZ
280 :   POKE 59464,0                : REM FREQUENZ NULL
290 :   POKE 59464,R                : REM FREQUENZ R
295 :   FOR I = 1 TO T*T1 - 100 : NEXT I : REM DAUER T
300 : GOTO 260
320 :
350 : POKE 59464,0                  : REM FREQUENZ NULL
360 : POKE 59466,0                  : REM KLANGFARBE NULL
370 : POKE 59467,0                  : REM TON AUS
380 :
390 : END : REM *** ENDE DES HAUPTPROGRAMMS **************************
398 :
399 :
400 REM +++ UNTERPROGRAMM 'UEBERSETZUNG NOTE -> FREQUENZ' ++++++++++++
401 :
410 : IF R$ = "H0"  THEN LET R = 251 : RETURN
420 : IF R$ = "C"   THEN LET R = 237 : RETURN
430 : IF R$ = "C#"  THEN LET R = 224 : RETURN
440 : IF R$ = "D"   THEN LET R = 211 : RETURN
450 : IF R$ = "D#"  THEN LET R = 199 : RETURN
460 : IF R$ = "E"   THEN LET R = 188 : RETURN
470 : IF R$ = "F"   THEN LET R = 177 : RETURN
480 : IF R$ = "F#"  THEN LET R = 167 : RETURN
490 : IF R$ = "G"   THEN LET R = 157 : RETURN
500 : IF R$ = "G#"  THEN LET R = 149 : RETURN
510 : IF R$ = "A"   THEN LET R = 140 : RETURN
520 : IF R$ = "B"   THEN LET R = 132 : RETURN
530 : IF R$ = "H"   THEN LET R = 124 : RETURN
540 : IF R$ = "C1"  THEN LET R = 117 : RETURN
550 : IF R$ = "C1#" THEN LET R = 111 : RETURN
560 : IF R$ = "D1"  THEN LET R = 104 : RETURN
570 : IF R$ = "D1#" THEN LET R =  99 : RETURN
580 : IF R$ = "E1"  THEN LET R =  93 : RETURN
590 : IF R$ = "F1"  THEN LET R =  88 : RETURN
600 : IF R$ = "F1#" THEN LET R =  83 : RETURN
610 : IF R$ = "G1"  THEN LET R =  78 : RETURN
620 : IF R$ = "G1#" THEN LET R =  73 : RETURN
630 : IF R$ = "A1"  THEN LET R =  69 : RETURN
640 : IF R$ = "B1"  THEN LET R =  65 : RETURN
650 : IF R$ = "H1"  THEN LET R =  61 : RETURN
660 : IF R$ = "C2"  THEN LET R =  57 : RETURN
670 : IF R$ = "Z"   THEN LET R =   0 : RETURN
680 :
690 : RETURN
699 :
700 REM === MUSIKSTUECK ==============================================
701 :
710 DATA A1,2,Z,2,C2,2,A1,2,E1,2,Z,2,A1,2
720 DATA E1,2,C1,2,Z,2,E1,2,C1,2,A,60
730 :
740 : REM HIER BITTE DIE UEBRIGEN NOTEN EINFUEGEN
750 :
790 DATA @,0
```

Höhepunkt und Abschluß dieses Kapitels soll das berühmte 'Game of Life' bilden. Oktober 1970 und Februar 1971 veröffentlichte Martin Gardner in seiner Rubrik 'Mathematical Games' (Scientific American) die Beschreibung eines von John Horton Conway erfundenen Spiels namens 'Life', das Aufstieg, Zerfall und Veränderung von Populationen lebender Organismen darstellen soll.

Beispiel 9 : Das Lebensspiel

Spielfläche (Lebensraum) ist eine in quadratische Felder aufgeteilte Ebene. Jedes Feld kann zwei Zustände, nämlich 0 (unbesetzt, tot) und 1 (besetzt, lebendig) annehmen. Nachbarn von Feld j sind alle Felder im Quadrat der Seitenlänge drei mit j als Mittelpunkt.

o o o
o ● o
o o o

Es gelten folgende Übergangsregeln:

(1) Ein leeres Feld geht in ein besetztes über, wenn genau drei seiner Nachbarfelder besetzt sind.

(2) Ein besetztes Feld geht in ein leeres über, wenn weniger als zwei oder mehr als drei seiner Nachbarfelder besetzt sind, andernfalls bleibt es besetzt.

Wie kam Conway zu diesen Regeln? Er wollte gewährleisten, daß einerseits nicht zu viele Figuren ein unbeschränktes Wachstum zeigen, andererseits aber auch nicht zu viele Figuren nach kurzer Zeit verkümmern und absterben; d.h. er wünschte ein ausgewogenes Verhältnis zwischen Leben und Tod. Insbesondere sollte es Figuren mit folgendem Verhalten geben:

a) Völliges Verschwinden in überschaubarer Zeit,

b) Stabilisierung zu einer unveränderlichen Figur,

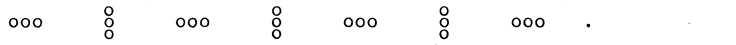
c) periodisches Oszillieren.

In der Tat liefern die genannten Regeln Figuren aller drei Klassen:

a) Eine einzelne lebende Zelle oder eine aus zwei lebenden Zellen bestehende Figur sterben offenbar sofort im ersten Schritt.

b) Unveränderlich ist folgender Viererblock ●● / ●● (beispielsweise).

c) Folgendes Tripel ('Blinker' genannt), oszilliert mit Periode 1:



Es gibt aber noch wesentlich interessanteres Verhalten und eine überraschende Fülle von Formen, wie wir mit Hilfe eines Programms sogleich sehen werden.

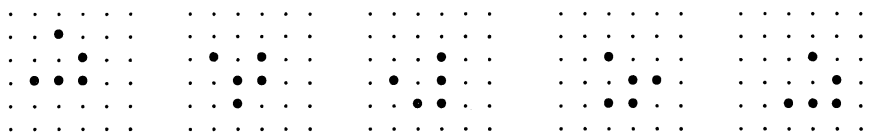
```
100 : PRINT "■"
110 : PRINT "          LEBENSSPIEL
111 : PRINT "          -----------
112 :
120 REM 'GAME OF LIFE' NACH J.H.CONWAY
121 :
130 : PRINT "DIES SPIEL SIMULIERT AUFSTIEG, ZERFALL  UND VERAENDERUNG
132 : PRINT "VON POPULATIONEN BELEBTER UND UNBELEBTER ORGANISMEN
140 :
150 : PRINT : PRINT "TASTE DRUECKEN!
160 :
170 : GET T$ : IF T$ = "" THEN 170
190 :
200 REM *** HAUPTPROGRAMM ***********************************************
201 :
230 : GOSUB 300 : REM INITIALISIERUNG
240 : GOSUB 400 : REM EINGABE
250 : GOSUB 500 : REM BESTIMMUNG DER FOLGEGENERATION
270 : GOSUB 800 : REM AUSGABE
271 :
280 : PRINT "NOCH EINE GENERATION?
281 : PRINT "DANN LEERTASTE DRUECKEN!
285 : GET T$ : IF T$ = "" THEN 285
290 : IF T$ = " " THEN 250
295 : END
297 :
298 REM *** ENDE DES HAUPTPROGRAMMS ************************************
299 :
300 REM +++ UNTERPROGRAMM 'INITIALISIERUNG' ++++++++++++++++++++++++++++
301 :
310 : LET N = 20                    : REM SEITENLAENGE DES LEBENSRAUMS
315 :
320 : DIM A(N+1,N+1), B(N+1,N+1) : REM ZWEI KOPIEN DES LEBENSRAUMS
340 :
350 : LET CR$ = "■" : LET CU$ = "■"
360 : FOR I = 1 TO 6
370 :   LET CR$ = CR$ + CR$ : LET CU$ = CU$ + CU$
380 : NEXT I
385 :
390 : RETURN
399 :
```

Punkt zwölf erscheint der Knochenmann
Und hält das Perpendikel an.

```
400 REM +++ UNTERPROGRAMM 'EINGABE' ++++++++++++++++++++++++++++++++++++
401 :
410 : GOSUB 800 : REM AUSGABE
415 :
420 : LET M = N*N
425 :
430 : PRINT "■" LEFT$(CU$,22);
435 : PRINT "ZEILE UND SPALTE EINGEBEN, ENDE MIT '--'";
440 :
450 : FOR K = 1 TO M
452 :   PRINT "■" LEFT$(CU$,23);
455 :   FOR I = 1 TO 39 : PRINT " ";: NEXT I : PRINT CHR$(13) "□";
456 :   INPUT "ZEILE: "; I$
460 :   IF I$ = "--" THEN RETURN
462 :   LET I = VAL(I$)
465 :   PRINT "□" TAB(15);
466 :   INPUT "SPALTE: "; J
470 :   LET A(I,J) = 1
475 :   PRINT "■" LEFT$(CU$,I) LEFT$(CR$,J-1) "●"
480 : NEXT K
485 :
490 : RETURN
499 :
500 REM +++ UNTERPROGRAMM 'BESTIMMUNG DER NAECHSTEN GENERATION' +++++++
501 :
510 :   FOR I = 1 TO N
515 :     FOR J = 1 TO N
520 :       GOSUB 700 : REM BESTIMMUNG DER ANZAHL DER NACHBARN
525 :
535 :       IF A(I,J) = 1 THEN 580 : REM ZELLE I,J IST BESETZT
540 :
550 :     REM --- GEBURT?
551 :
560 :       IF NZ = 3 THEN LET B(I,J) = 1 : GOTO 600
565 :       GOTO 590
570 :
580 :     REM --- TOD?
581 :
585 :       IF NZ = 2 OR NZ = 3 THEN B(I,J) = 1 : GOTO 600
590 :       LET B(I,J) = 0
595 :
600 :     NEXT J
610 :   NEXT I
620 :
630 : REM --- NEUE GENERATION WIRD ZUR ALTEN
631 :
640 :   FOR I = 1 TO N
645 :     FOR J = 1 TO N
650 :       LET A(I,J) = B(I,J)
660 :     NEXT J
670 :   NEXT I
680 :
690 : RETURN
699 :
```

```
700 REM +++ UNTERPROGRAMM 'ANZAHL DER NACHBARN' ++++++++++++++++++++++++
701 :
710 : LET NZ = 0
720 : FOR K = I-1 TO I+1
730 :   FOR L = J-1 TO J+1
740 :     LET NZ = NZ + A(K,L)
750 :   NEXT L
760 : NEXT K
770 : LET NZ = NZ - A(I,J)
780 :
790 : RETURN
799 :
800 REM +++ UNTERPROGRAMM 'AUSGABE' ++++++++++++++++++++++++++++++++++
801 :
810 : PRINT "□"
840 : FOR I = 1 TO N
850 :   FOR J = 1 TO N
860 :     IF A(I,J) = 1 THEN PRINT "●";: GOTO 870
865 :                    PRINT ".";
870 :   NEXT J
875 :   PRINT
880 : NEXT I
890 :
895 : RETURN
899 :
```

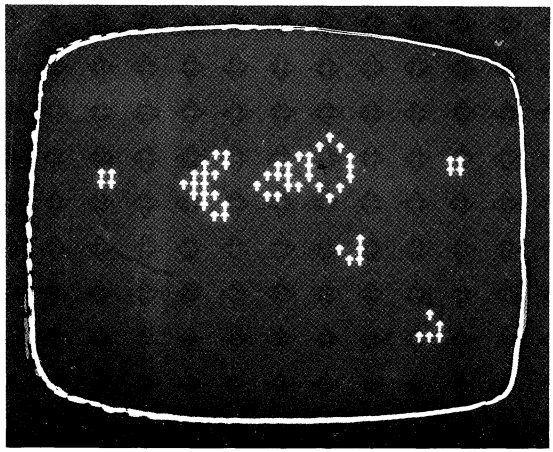
Dies Programm liefert uns z.B. den berühmten 'Gleiter', d.i. eine Figur, die sich in der 5. Generation selbst reproduziert hat, dabei aber um je ein Feld nach rechts unten weitergerückt ist:



Leider hat das Programm einen kleinen Fehler: es ist unerträglich langsam. Es läßt sich zwar noch etwas beschleunigen (siehe die Aufgaben) - wirksame Abhilfe schafft hier jedoch nur das 'Programmieren in Maschinen-Sprache'. Ich kann - aus Raumgründen - hier keine Einführung in die Maschinensprache geben (siehe dazu die Literaturhinweise in den Anmerkungen); doch soll an diesem Beispiel kurz dargestellt werden, worum es dabei geht. Vielleicht bekommen Sie Lust, sich näher damit zu beschäftigen: es ist ein faszinierendes Gebiet.

Wir verkehren mit unserem Rechner bekanntlich in der Sprache BASIC: sie

ist eine sogenannte höhere Programmiersprache; das heißt: sie ist an der Ausdrucksweise der Menschen, nicht an den Gegebenheiten der Maschine orientiert. Ihr Vorteil ist gute Verständlichkeit für den Menschen; ihr Nachteil, daß erst eine Übersetzung in die Sprache der Maschine vorgenommen werden muß. 'In Wirklichkeit versteht' ja der Rechner nur Bitmuster (Folgen der Art 10101001); um von BASIC in diese 0-1-Folgen übersetzen zu können, benutzt der Computer ein umfangreiches Programm ('Interpreter' genannt). Das Übersetzen einer Anweisung in die Maschinensprache kostet zwar im einzelnen nicht viel Zeit; wenn aber viele Anweisungen auszuführen sind, so kann - wie das Beispiel 'Lebensspiel' zeigt - der Zeitbedarf beträchtlich werden. Ein direkt in der Maschinensprache formuliertes Programm spart die Übersetzungszeit - sein Nachteil ist (wie schon gesagt) die schwere Verständlichkeit. Schauen Sie sich bitte die Zahlenfolgen in den DATA-Zeilen ab 700 im Programm auf der nächsten Seite an!



```
100 : PRINT "□"
110 : PRINT "                LIFE
111 : PRINT "                ----
112 :
120 REM CONWAYS 'GAME OF LIFE' IN MASCHINENSPRACHE
122 :
129 : PRINT : PRINT
130 : PRINT "DIESES SPIEL SIMULIERT AUFSTIEG, ZERFALL UND VERAENDERUNG
140 : PRINT "VON POPULATIONEN BELEBTER UND UNBELEBTER WESEN.
150 :
160 : PRINT : PRINT "           TASTE DRUECKEN!
170 :
180 : GET T$ : IF T$ = "" THEN 180
190 :
200 REM *** HAUPTPROGRAMM ***********************************************
201 :
230 : GOSUB 300 : REM INITIALISIERUNG
231 :
240 : GOSUB 400 : REM EINGABE
241 :
250 : GOSUB 500 : REM BERECHNUNG DER FOLGEGENERATION
251 :
280 : GOTO 250  : REM NICHTABBRECHENDE SCHLEIFE
285 :
290 REM *** ENDE HAUPTPROGRAMM ******************************************
299 :
300 REM +++ UNTERPROGRAMM 'INITIALISIERUNG' ++++++++++++++++++++++++++++
301 :
310 :   PRINT
320 :   PRINT "WIEVIELE SEKUNDEN SOLL EINE GENERATION
322 :   INPUT "AUF DEM BILDSCHIRM VERWEILEN ";T
340 :
350 : REM EINLESEN DES MASCHINENPROGRAMMS
351 :
360 :   FOR I = 826 TO 950 : READ C : POKE I,C : NEXT I
370 :
390 :   RETURN : REM ENDE DER INITIALISIERUNG ++++++++++++++++++++++++++
399 :
400 REM +++ UNTERPROGRAMM 'EINGABE' +++++++++++++++++++++++++++++++++++
401 :
410 : PRINT
415 : PRINT "GEBEN SIE NUN DIE ANFANGSFIGUR EIN,
420 : PRINT "INDEM SIE FUER DIE LEBEWESEN '●'
425 : PRINT "UND FUER IHRE LAGE DIE CURSORTASTEN
426 : PRINT "BENUTZEN.
428 : PRINT "■BEGINN DER REPRODUKTION MIT ■<RETURN>■
429 :
430 : PRINT"■■           TASTE DRUECKEN "
432 : GET A$ : IF A$ = "" THEN 432
436 : PRINT"□#■■";
437 :
```

```
440 : GET A$ : IF A$ = "" THEN 440
445 :
450 :   IF A$ = "■" OR A$ = "■" OR A$ = "■" OR A$ = "■" THEN 480
455 :   IF A$ = "●" OR A$ = CHR$(20) THEN 480
460 :
470 :   IF A$ = CHR$(13) THEN PRINT "  " : RETURN : REM RETURN-TASTE
475 :
480 :   PRINT " ■";A$;"#■";  : REM MARKE ZUR LOKALISIERUNG DER FIGUREN
485 :
490 : GOTO 440
495 :
497 : REM ENDE DER EINGABE ++++++++++++++++++++++++++++++++++++++++++++
499 :
500 REM +++ UNTERPROGRAMM 'BERECHNUNG DER FOLGEGENERATION' ++++++++++++
501 :
510 : SYS 887 : REM AUFRUF DES MASCHINENPROGRAMMS
520 :
530 : LET T0 = TI
540 : IF TI < T0 + 60*T THEN 540 : REM ZEIT ZWISCHEN ZWEI GENERATIONEN
550 :
590 : RETURN : REM ENDE DER BERECHNUNG DER FOLGEGENERATION ++++++++++++
599 :
700 REM === MASCHINENPROGRAMM ======================================
701 :
710 DATA 169,215,133,10,169,127,133
720 DATA 11,169,215,133,12,169,26,133
730 DATA 13,96,230,10,208,2,230,11
740 DATA 230,12,208,2,230,13,165,11
750 DATA 201,132,208,4,165,10,201,16
760 DATA 96,169,0,133,14,162,7
770 DATA 188,175,3,177,10,201
780 DATA 81,208,2,230,14,202,16,242,96
790 DATA 32,58,3,32,98,3,160,41
800 DATA 177,10,201,81,240,10,165
810 DATA 14,201,3,240,14,169,32,208
820 DATA 12,165,14,201,3,240,4,201
830 DATA 2,208,242,169,81,145,12,32
840 DATA 75,3,208,216,32,58,3,177
850 DATA 12,145,10,32,75,3,208,247
860 DATA 96,0,1,2,40,42,80,81,82
870 :
890 REM ENDE DES MASCHINENPROGRAMMS ================================
899 :
```

Will man ein Teilprogramm in Maschinensprache formulieren und ausführen lassen, so geht man (bei CBM-Rechnern) wie folgt vor:

1. Schritt: Entwicklung des Teilprogramms in der Assemblersprache.

Der Anfang des Assemblerprogramms für Life - und zwar dessen Hauptprogramm - lautet wie folgt (nächste Seite). Möchten Sie die Bedeutung der Befehle JSR, LDY, CMP u.s.w kennenlernen, so schlagen Sie bitte bei Sacht [26] oder Zaks [28] nach; wir haben leider hierzu keinen Platz.

```
887 :   32  JSR AB   826 : zum Unterprogramm 'Initialisierung'
890 :   32  JSR AB   866 : zum Unterprogramm 'Zähle die Nachbarn'
893 :  160  LDY #     41 : jeweiliges Feld besetzt ?
895 :  177  LDA INY   10 :
897 :  201  CMP #     81 :
899 :  240  BEQ R     10 : Sprung, wenn Zelle belegt ist
901 :  165  LDA ZP    14 : Anzahl der Nachbarn
903 :  201  CMP #      3 : drei Nachbarn?
905 :  240  BEQ R     14 : Sprung, wenn drei Nachbarn
907 :  169  LDA #     32 : Zelle leeren
909 :  208  BNE R     12 : Sprung, wenn nicht leer
911 :  165  LDA ZP    14 : Anzahl der Nachbarn
913 :  201  CMP #      3 : drei Nachbarn ?
915 :  240  BEQ R      4 : wenn ja, dann Geburt
917 :  201  CMP #      2 : zwei Nachbarn ?
919 :  208  BNE R     42 : wenn nicht, dann Rücksprung
921 :  169  LDA #     81 : erzeugt neues Individuum
923 :  145  STA INY   12 : Setzen in neues Feld
925 :   32  JSR AB   843 : Adressen inkrementieren
928 :  208  BNE R    216 : Rücksprung, wenn Feld nicht zuende
930 :   32  JSR AB   826 : zum Unterprogramm 'Initialisierung'
933 :  177  LDA INY   12 : neue Generation auf den Bildschirm
935 :  145  STA INY   10 :
937 :   32  JSR AB   843 : Feldadressen inkrementieren
940 :  208  BNE R    247 : Rücksprung
942 :   96  RTS          : Ende des Hauptprogramms
```

2. Schritt: Umwandlung des Assemblerprogramms in eine Folge von Maschinenbefehlen. Diese Folge lautet (in dezimaler Schreibweise) für unser Life-Hauptprogramm: 32, 58, 3, 32, 98, 3, 160, 41, ...

3. Schritt: Einschreiben des Maschinenprogramms als DATA-Zeilen im BASIC-Programm (bei uns ab den Zeilen 710; das Hauptprogramm beginnt mit 790). Das Einlesen in den Programmspeicher geschieht in Zeile 360 des BASIC-Programms: jeder Maschinenbefehl wird in die Variable C gelesen und dann mittels `POKE I,C` in die Speicherzelle Nr. I geschrieben. In den Zellen von 826 bis 950 steht nun das Maschinenprogramm.

4. Schritt: Aufruf des Maschinenprogramms vom BASIC-Programm aus: dies geschieht in unserem Beispiel mittels `SYS 887` (Zeile 510); in Zelle 887 beginnt das Maschinen-Hauptprogramm (siehe die Befehlsliste oben).

A u f g a b e n

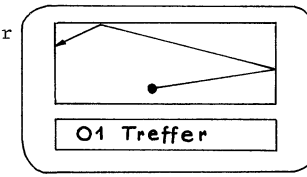
1. Es sollen Rechtecke mit vorgegebenen Eckpunkten auf den Bildschirm gezeichnet werden.

2. Eine schiefliegende Strecke zwischen zwei gegebenen Punkten soll auf den Bildschirm gezeichnet werden.

3. Schreiben Sie ein kleines Programm, mit dem man ein Dreieck mit vorgegebenen Eckpunkten auf den Bildschirm zeichnen kann.

4. Der auf den Rahmen prallende bewegliche Punkt soll - wie ein Ball, der von einer Wand abprallt - reflektiert werden.

5. In das Geschehen auf dem Bildschirm ist eine Zeile einzublenden, welche die Anzahl der Versuche, die Erfolge, die verflossene Zeit usw. angibt.



6. Ein Gegenstand, der aus m e h r e r e n Zeichen besteht, soll über den Bildschirm bewegt werden.

7. Hinter die Anfrage bei GET soll ein blinkender Cursor simuliert werden.

8. Schreiben Sie ein Programm, welches folgende einfache Spielidee realisiert: auf dem Bildschirm erscheinen zufällig verteilte Punkte, die von einem steuerbaren beweglichen Punkt 'gefressen' werden können. Dies soll möglichst schnell geschehen.

9. In das Programm 'Fledermaus' soll eine Überlebenszeit eingebaut werden; wenn der Spieler sie überstanden hat, gilt das Spiel als gewonnen.

10. Die Überlebenszeit im Programm 'Fledermaus' soll veränderlich sein: damit sind unterschiedliche Schwierigkeitsgrade wählbar.

11. Der Schwierigkeitsgrad im Programm 'Fledermaus' kann auch auf andere Weise veränderlich gemacht werden: z.B. durch Änderung des Zeitfaktors (etwa LET N = 0.98*N in Zeile 592).

12. Man kann Punkte vergeben, deren Anzahl von der erfolgreich überstandenen Zeit abhängt. Bauen Sie diese Möglichkeit ins Programm 'Fledermaus' ein!

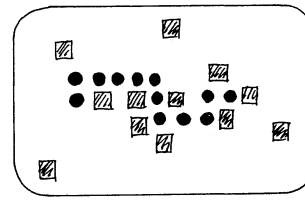
13. Die schon verflossene und die noch verbleibende Zeit sind ins Spielfeld einzublenden.

14. Gegen die vom Spieler gesteuerte Fledermaus soll der Computer als Gegenspieler antreten: programmieren Sie einen Zufallsweg des Com-

puters, der die Hindernisse umgeht. Machen Sie den Computer aber nicht zu perfekt, d.h. lassen Sie ihn auch gelegentlich auf ein Hindernis rennen und damit verlieren. Sonst verliert der Spieler den Mut.

15. Entwerfen Sie ein Spiel, bei dem analog zur 'Fledermaus' Hindernisse umgangen werden sollen, der Punkt aber in ein vorgegebenes Ziel zu lenken ist.

16. Programmieren Sie ein Spiel, bei dem die Hindernisse von einem Wurm aufgefressen werden; der Wurm wird dabei immer länger.
Wer den längsten Wurm erzeugt, hat gewonnen. Stößt der Wurm auf einen 'vergifteten' Brocken, von denen einige herumliegen, so gibt er seinen Geist auf und das Spiel ist verloren.



17. Beim Spiel 'Sterne jagen' soll der Schuß eine Leuchtspur hinterlassen: ➡ ..... ✸ .

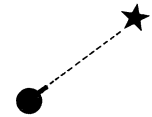
18. Die Sterne sollen wie beim Programm 'Fledermaus' mit wachsender Geschwindigkeit zunehmen, und der Spieler soll sich ihrer mit Schüssen erwehren.

19. Beim Spiel 'Sterne jagen' soll es Sterne unterschiedlicher Wertigkeit geben; die großen ergeben nach dem Abschuß mehr Punkte.

20. Analog zum Programm 'Fledermaus' sind beim 'Sterne jagen' unterschiedliche Schwierigkeitsgrade (vorher wählbar) einzubauen.

21. Ergänzen Sie das Spiel 'Eidechse' so, daß die Fliege nicht still in der Luft schwebt, sondern Zufallsbewegungen ausführt und Zufallsgeräusche erzeugt. Das Fangen und Verzehren einer Fliege soll akustisch untermalt werden.

22. Eine Spielidee: von der drehbaren Lafette aus kann geschossen werden.
(Tschuldigung für das viele Schießen. Ab jetzt geht es ganz friedlich zu.)



23. Die Suchstrategie der Computermaus im Spiel 'Labyrinth' kann verbessert werden, wenn der Computer es lernt, Sackgassen zu vermeiden. Allerdings wird er dann für den menschlichen Spieler unschlagbar. Versuchen Sie trotzdem diese Verbesserung.

24. Schreiben Sie ein Suchprogramm zum Entkommen aus Sam Loyds Zahlenlabyrinth (Beispiel 7). Wieviel Wege aus dem Wald gibt es? Sam Loyd hatte behauptet, es gäbe genau eine Lösung - stimmt diese Behauptung?

Sam's Lösung: SW auf 4, SW auf 6, NO auf 6, NO auf 2, NO auf 5, SW auf 4, SW auf 4, SW auf 4 und dann mit einem entschlossenen Schritt nach NW und in die Freiheit.

25. Eine weitere Lösung zum Entkommen aus dem Zahlenlabyrinth lautet: N auf 2, O auf 4, S auf 1, O auf 2, S auf 2, S auf 2, W auf 2, W auf 2, NW auf 4. Jetzt sind wir auf einem Punkt (Zeile 14, Element 7 von links), über den auch Loyds Weg führt.
Alle von Loyd abweichenden Lösungswege laufen über die 2 in der 6. Zeile von unten, 8. Element von links.
Zeigen Sie: wenn man diese 2 durch eine andere Ziffer $\neq 0$ ersetzt, so gibt es nur die Loyd'sche Lösung. (Vielleicht liegt hier ein Versehen des Graphikers vor, der das Labyrinth zeichnete, und Sam Loyd hatte Recht?)

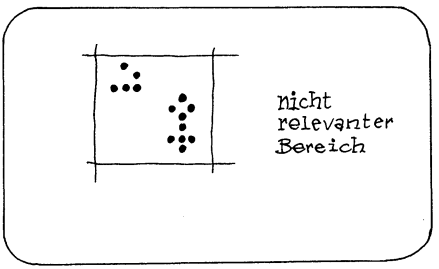
26. Schreiben Sie ein Programm, das den Rechner Zufallsschritte machen läßt. Wie groß ist die Wahrscheinlichkeit, auf diese Weise dem Labyrinth zu entkommen?

27. Setzen Sie andere Musikstücke nach dem Muster von Beispiel 8 Computermusik um!

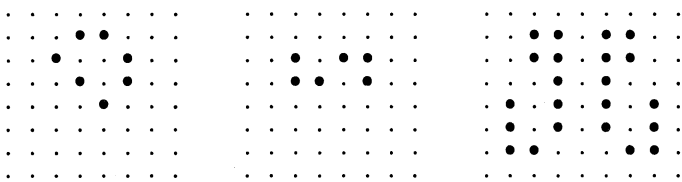
28. Erweitern Sie 'Zahlensenso' (Beispiel 2 in Kapitel 2) so, daß zu jeder auf dem Bildschirm erscheinenden Ziffer ein Ton erklingt.

29. Lassen Sie - analog zum 'Begriffe merken' (Beispiel 3 in Kapitel 2) den Spieler kleine Melodien erinnern.

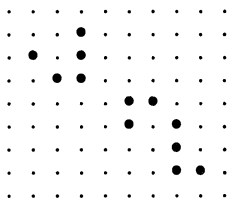
30. Machen Sie das Programm 'Lebensspiel' dadurch schneller, daß Sie vor der Bestimmung der Nachbarn den relevanten Bereich abstecken, innerhalb dessen überhaupt Veränderungen (Geburt bzw. Tod) zu erwarten sind.



31. Untersuchen Sie beim 'Lebensspiel' die Entwicklung folgender Figuren:

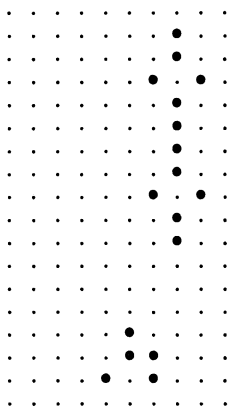


32. Der Esser verzehrt einen Gleiter:



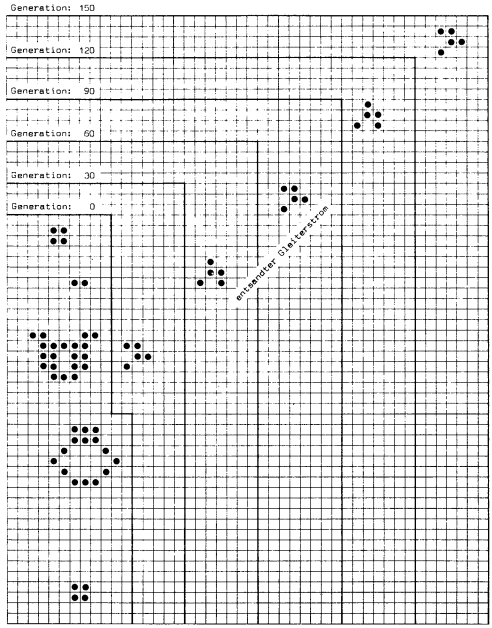
Probieren Sie es aus!

33. Auch hier wird ein Gleiter gegessen:



Lassen Sie diesen Vorgang auf dem Bildschirm ablaufen.

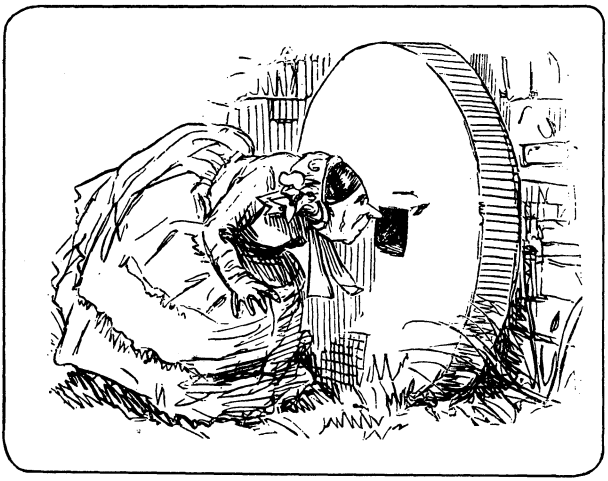
34. Dies ist die berühmte Gleiter-Kanone, die zu weitreichenden Folgerungen und Spekulationen auf dem Gebiet der theoretischen Informatik Anlaß gegeben hat:



35. Experimentieren Sie beim 'Lebensspiel' mit anderen Reproduktionsregeln; z.B. Geburt bei zwei oder drei lebenden Nachbarn o.ä.

36. Ein anderes Reproduktionsspiel stammt von dem Mathematiker S. Ulam: er definiert eine sogenannte orthogonale Nachbarschaft wie nebenstehend abgebildet, d.h. eine Zelle hat nur vier Nachbarn, die wie ein Kreuz angeordnet sind. Eine Zelle wird geboren, wenn genau ein lebender Nachbar existiert. Nach zwei Generationen stirbt jede Zelle! Dies Spiel erzeugt wunderhübsche (ständig wachsende) Figuren. Schreiben Sie ein Programm dafür!

```
. . . . . .
. . . o . . .
. . o ● o . .
. . . o . . .
. . . . . .
```



4
Such- und Ratespiele

Jeder von uns verbringt einen beträchtlichen Teil seiner Zeit damit, irgendetwas zu suchen. Man denke etwa an Kleidungsstücke oder andere Gebrauchsgegenstände in einem Kaufhaus, gute Sendungen im Fernsehen, verlegte oder verlorene Gegenstände in der Wohnung, die Suche nach einer passenden Formulierung oder die nach einem effizienten Algorithmus. Allgemeiner: jedes P r o b l e m l ö s e n kann als ein Suchen - nämlich nach der Lösung - aufgefaßt werden. Damit fällt alle zielgerichtete Tätigkeit unter das Stichwort 'Suchen'.
Im Spiel werden typische Problemsituationen vereinfacht dargestellt, und es geht stets darum, die Suche möglichst geschickt, d.h. mit möglichst wenigen Tests (Maßnahmen zur Beschaffung von Information über den gesuchten Gegenstand) durchzuführen.
Der Computer kann eine zwiefache Rolle übernehmen: einmal kann er die Aufgabe stellen und die Rateleistung bewerten; zum anderen kann er selbst Ratender sein, sofern ihm eine Rate- oder Suchstrategie gegeben wurde.-
In unserem ersten Beispiel soll der Computer eine vom Spieler gedachte Zahl erraten; das zunächst Verblüffende hieran ist, daß er mit scheinbar zu wenig Information auskommt. Daß die Information ausreicht, läßt sich mathematisch zeigen.

Beispiel 1: Nicomachus

Der Spieler denkt sich eine Zahl zwischen 1 und 1000. Er wird vom Computer aufgefordert, seine Zahl durch 7, 11 und 13 zu dividieren und die dabei entstehenden Reste anzugeben. Daraufhin nennt der Computer die gedachte Zahl.

Dialog:

```
Rest bei Division durch  7? 2
Rest bei Division durch 11? 3
Rest bei Division durch 13? 6

Lassen Sie mich etwas nachdenken ...

Haben Sie sich die Zahl 58 gedacht? j
Wollen Sie wissen, wie ich das
herausbekommen habe?
Dann ...
```

In der Tat ist $58 = 8 \cdot 7 + 2 = 5 \cdot 11 + 3 = 4 \cdot 13 + 6$. Wie kommt der Computer nun auf die 58? Er hat einfach

$$z = 715 \cdot 2 + 364 \cdot 3 + 924 \cdot 6 = 8066$$

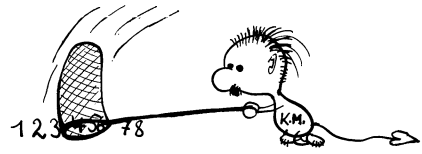
gerechnet und von z den Rest bei Division durch $1001 = 7 \cdot 11 \cdot 13$ gebildet: dies ist die zu ratende Zahl! Die Theorie hierzu ist - im Ansatz - schon in der Arithmetik des Nicomachus von Gerasa (um 100 n. Chr.) zu finden.

```
100 : PRINT "□"
110 : PRINT "             NICOMACHUS
111 : PRINT "             ----------
112 :
120 REM ZAHLENRATEN NACH DEM CHINESISCHEN RESTSATZ
129 :
130 : PRINT
140 : PRINT "DENKEN SIE SICH EINE ZAHL ZWISCHEN
145 : PRINT "1 UND 1000. ICH WERDE SIE ERRATEN,
150 : PRINT "WENN SIE MIR DREI KLEINE FRAGEN BEANTWORTEN.
160 : PRINT "UND ZWAR MUESSEN SIE JEWEILS DEN REST
165 : PRINT "BEI DIVISION IHRER ZAHL DURCH 7, DURCH 11
170 : PRINT "UND DURCH 13 NENNEN.
190 :
200 : PRINT
210 : INPUT "REST BEI DIVISION DURCH  7 "; R1
215 :   IF R1 < 0 OR R1 > 6 THEN 210
220 : INPUT "REST BEI DIVISION DURCH 11 "; R2
225 :   IF R2 < 0 OR R2 > 10 THEN 220
230 : INPUT "REST BEI DIVISION DURCH 13 "; R3
235 :   IF R3 < 0 OR R3 > 12 THEN 230
240 :
250 : PRINT
260 : PRINT "LASSEN SIE MICH ETWAS NACHDENKEN ...
270 : FOR I = 1 TO 2000 : NEXT I
290 :
300 : LET Z = 715*R1 + 364*R2 + 924*R3
310 : IF Z > 1001 THEN LET Z = Z-1001 : GOTO 310
315 :
320 : PRINT
330 : PRINT "HABEN SIE SICH DIE ZAHL ";Z;" GEDACHT?(J/N)
340 :
350 : GET A$ : IF A$ = "" THEN 350
360 : IF A$ = "J" THEN 400
370 : IF A$ = "N" THEN 450
380 : PRINT "ICH HABE WOHL NICHT RECHT VERSTANDEN.": GOTO 350
390 :
400 : PRINT
410 : PRINT "WOLLEN SIE WISSEN, WIE ICH DAS HERAUSBEKOMMEN HABE?
420 : PRINT "DANN SCHAUEN SIE BITTE BEIM ALTEN NICOMACHUS
430 : PRINT "VON GERASA (EINFUEHRUNG IN DIE ARITHMETIK) NACH!
440 : END
445 :
450 : PRINT
460 : PRINT "AN IHRER ARITHMETIK STIMMT ETWAS NICHT.
470 : PRINT "VIELLEICHT MACHEN SIE NOCH EINEN VERSUCH!
475 :
480 : FOR I = 1 TO 2000 : NEXT I
490 : RUN
```

Nach diesem Ausflug in die Zahlentheorie wollen wir uns den Ratespielen im eigentlichen Sinn zuwenden.

Beispiel 2 : Intervallsuche

Eine zwischen 1 und m zufällig gewählte natürliche Zahl soll durch wiederholte Angabe eines Intervalls, in dem sie vermutet wird, bestimmt werden.



D i a l o g :

```
Ich denke mir eine Zahl zwischen 1 und 352.

Geben Sie die Intervallgrenzen ein!

Untere Grenze? 1
Obere  Grenze? 170

Die gesuchte Zahl liegt im Intervall.

Untere Grenze? 85
Obere  Grenze? 170

Die gesuchte Zahl liegt nicht im Intervall.

Untere Grenze?
.......................

Untere Grenze? 99
Obere  Grenze? 99

Leider daneben.

Untere Grenze? 69
Obere  Grenze? 69

                ERRATEN!

Die gesuchte Zahl ist 69. Auf Wiedersehen.
```

Das Programm auf der nächsten Seite dürfte unmittelbar verständlich sein.

```
100 : PRINT "■"
110 : PRINT "          INTERVALLSUCHE
111 : PRINT "          --------------
112 :
120 REM ZAHLENRATEN DURCH ANGABE VON SUCHINTERVALLEN
121 :
140 : LET M = INT(10000*RND(TI)+1)
150 :
160 : PRINT
165 : PRINT "ICH DENKE MIR EINE ZAHL ZWISCHEN 1 UND ";M;".
169 : PRINT
170 : PRINT "SIE SOLLEN DIE ZAHL DURCH WIEDERHOLTE ANGABE
175 : PRINT "EINES INTERVALLS, D.H. EINER UNTEREN UND
180 : PRINT "OBEREN GRENZE FUER DIE ZAHL, AUFFINDEN.
190 :
200 : LET X = INT(M*RND(TI)+1)
210 :
220 : PRINT
230 : PRINT "GEBEN SIE NUN DIE INTERVALLGRENZEN EIN!
235 : PRINT
240 : INPUT "UNTERE GRENZE "; U
250 : INPUT "OBERE  GRENZE "; O
260 : IF O < U THEN PRINT : PRINT "NICHT ZULAESSIG!" : GOTO 235
270 : IF O = U THEN 360    : REM  BEIDE INTERVALLGRENZEN GLEICH
290 :
300 : PRINT : PRINT
310 : PRINT "DIE GESUCHTE ZAHL LIEGT ";
320 :
330 : IF X < U OR X > O THEN PRINT "NICHT ";
340 : PRINT "IM INTERVALL." : GOTO 235
350 :
360 : PRINT "■"
370 : IF U <> X THEN PRINT "■■■■■■ LEIDER DANEBEN!" : GOTO 235
380 : PRINT "■■■■■■■■■■■■■■■ERRATEN!■
390 : PRINT "■■■DIE GESUCHTE ZAHL IST "; X;".
395 :
399 : END
```

Beispiel 3 : Begriffe raten

Ein vom Computer zufällig gewähltes Wort soll erraten werden. Der Rater kann im Wort vermutete Buchstaben nennen; diese werden dann - falls vorhanden - an der richtigen Stelle angezeigt. Es kann auch das ganze Wort auf einmal genannt werden.

Dialog:

```
Gesucht wird: . . . . . .
Ihr Lösungsbuchstabe oder -wort? E

Gesucht wird: E . . E . . . . .
Ihr Lösungsbuchstabe oder -wort? A

Gesucht wird: E . . E . . A . .
Ihr Lösungsbuchstabe oder - wort? EISENHANS

Spitze! Sie haben richtig geraten.
```

Das Programm auf der nächsten Seite ist wie folgt aufgebaut: Zunächst wird eine zufällige Stelle z zwischen 1 und 14 (Anzahl der im Programm verankerten Worte; Sie werden natürlich noch wesentlich mehr Worte hinzunehmen!) bestimmt und dann das Wort mit dieser Nummer eingelesen. Nachdem seine Länge L festgestellt ist, wird als erstes 'Hilfswort' eine Zeile aus L Punkten ausgegeben. Nachdem der Spieler seinen Lösungsbuchstaben L$ eingegeben hat, wird das unbekannte Wort U$ Buchstabe für Buchstabe durchgegangen und mit L$ verglichen. Zugleich wird damit das neue Hilfswort N$ aufgebaut: kommt L$ im unbekannten Wort vor, wird er in das Hilfswort eingefügt. Dies geschieht im Teil 'Aufbau des Hilfsworts' ab Zeile 600. Gibt der Spieler jedoch ein ganzes Lösungswort ein (Bedingung LEN(L$) > 1 in Zeile 540), so wird es mit dem unbekannten Wort verglichen und eine entsprechende Meldung ausgegeben.

```
100 : PRINT "■"
110 : PRINT "              BEGRIFFE RATEN
111 : PRINT "              --------------
112 :
120 REM SPIEL FUER MAERCHENFREUNDE
190 :
200 REM === SPIELREGELN ================================================
201 :
210 : PRINT "■■ICH, DER COMPUTER DENKE MIR EIN WORT -
220 : PRINT "UND SIE SOLLEN ES ERRATEN.
230 : PRINT "SIE KOENNEN EINEN BUCHSTABEN EINGEBEN:
240 : PRINT "DANN ERSCHEINEN DIE GERATENEN BUCHSTABEN
250 : PRINT "AN DER RICHTIGEN STELLE.
260 : PRINT "WENN SIE DAS WORT KENNEN, GEBEN SIE ES GANZ EIN.
290 :
300 REM === FESTLEGEN DES UNBEKANNTEN WORTS =============================
301 :
310 : LET Z = INT(14*RND(TI))+1          : REM ZUFALLSZAHL WAEHLEN
320 : FOR I = 1 TO Z : READ U$ : NEXT I  : REM EINLESEN BIS STELLE Z
390 :
400 REM === AUFBAU DES ERSTEN HILFSWORTS ================================
401 :
410 : LET L = LEN(U$)            : REM ERMITTLUNG DER BUCHSTABENANZAHL
420 : LET H$ = ""
430 : FOR I = 1 TO L : LET H$ = H$ + "." : NEXT I
490 :
500 REM === RATEVERSUCH ================================================
501 :
510 : PRINT "■GESUCHT WIRD: "; H$
520 : PRINT : INPUT "IHR LOESUNGSBUCHSTABE ODER -WORT "; L$
530 : IF L$ = U$ THEN 700            : REM WORT ERRATEN
540 : IF LEN(L$) > 1 THEN 500        : REM WORT FALSCH GERATEN
590 :
600 REM === AUFBAU DES HILFSWORTES ======================================
601 :
610 : LET N$ = ""                    : REM NEUES HILFSWORT
620 : FOR I = 1 TO L
630 :   LET B$ = MID$(U$,I,1)        : REM BUCHSTABE DES UNBEKANNTEN WORTS
640 :   LET C$ = MID$(H$,I,1)        : REM BUCHSTABE DES HILFSWORTS
650 :   IF L$ = B$ THEN N$ = N$+L$ : GOTO 670 : REM BUCHSTABE EINGESETZT
660 :   LET N$ = N$+C$               : REM HILFSWORTBUCHSTABE KOPIERT
670 : NEXT I
680 : LET H$ = N$                    : REM NEUES HILFSWORT WIRD ZUM ALTEN
690 : GOTO 500                       : REM NAECHSTER VERSUCH
699 :
700 REM === WIEDERHOLUNG ODER VERABSCHIEDUNG ============================
701 :
710 : PRINT "■■SPITZE! SIE HABEN RICHTIG GERATEN.
720 : PRINT "■■NOCH EIN VERSUCH?(J/N)
730 : GET A$ : IF A$ = "" THEN 730
740 : IF A$ = "J" THEN RUN
790 :
800 REM === ZU RATENDE BEGRIFFE =========================================
801 :
810 DATA MAERCHEN, FROSCHKOENIG, DAUMESDICK, FUNDEVOGEL, GEVATTER
820 DATA DRACHE, GLUECKSKIND, RUMPELSTILZCHEN, RAPUNZEL, SCHNEIDERLEIN
830 DATA EISENHANS, GOLDKINDER, DUMMLING, STADTMUSIKANTEN
840 :
899 END
```

Beispiel 4 : Verwürfelt

Ein Wort, dessen Buchstaben - verwürfelt - kurz auf dem Bildschirm erscheinen, soll erraten werden. Dazu wird ein Stichwort gegeben.

D i a l o g :

```
Stichwort: Unser blauer Planet

          D E E R

Ihre Antwort ? Erde

Richtig!
```

V e r f a h r e n :

Zunächst wird eines der Worte, die am Ende des Programms in DATA-Zeilen gespeichert sind, mit zugehörigem Stichwort ausgewählt. Dann wird eine zufällige Permutation p(1), p(2), ..., p(L) der Zahlen 1,2,...,L erzeugt, wobei L die Länge (Buchstabenanzahl) des zu ratenden Worts ist. Das geht so: für alle i = 1,2,...L ist p(i) eine Zufallszahl zwischen 1 und L; wenn eine solche Zahl schon früher einmal vorkam, wird eine neue erzeugt:

```
430 FOR J = 1 TO I-1
440   IF P(I) = P(I-J) THEN 430
450 NEXT J
```

Dann wird

MID$(W$, P(I), 1)

der zufällig ausgewählte Buchstabe des zu ratenden Worts W$, der an i-ter Stelle auf dem Bildschirm erscheint.

Durch puren Zufall könnte es geschehen, daß das verwürfelte Wort gleich dem zu ratenden Wort ist; die Wahrscheinlichkeit hierfür ist aber extrem klein (nämlich $\frac{1}{n!}$, wobei $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$): bei 8 Buchstaben 0,000025, sofern diese paarweise verschieden sind.

```
100 : PRINT "□"
110 : PRINT "            VERWUERFELT
111 : PRINT "            -----------
112 :
120 REM WORTRATESPIEL; DIE BUCHSTABEN DES GESUCHTEN WORTS
121 REM WERDEN VERWUERFELT (PERMUTIERT)
129 :
200 REM === ANLEITUNG UND INITIALISIERUNG ============================
201 :
210 : REM HIER GEHOERT DIE ANLEITUNG HIN
220 :
260 : LET N = 25 : REM HOECHSTMOEGLICHE LAENGE EINES WORTS
270 : DIM P(N)   : REM TABELLE ZUM PERMUTIEREN
299 :
300 REM === ERZEUGEN DES ANGEBOTENEN WORTS ===========================
301 :
305 : REM --- FESTLEGEN DES ZU RATENDEN WORTS
306 :
310 :   LET Z = INT(3*RND(TI)+1)
320 :   FOR I = 1 TO Z : READ D$,D$ : NEXT I : REM UEBERLESEN
330 :   READ W$,ST$                          : REM EINLESEN
340 :
350 :   PRINT "■STICHWORT: "; ST$
360 :
370 :   FOR K = 1 TO 500 : NEXT K  : REM MERKZEIT FUER DAS STICHWORT
380 :
400 : REM --- VERWUERFELN (ZUFALLSPERMUTATION)
401 :
410 :   LET L = LEN(W$)
420 :   FOR I = 1 TO L
430 :     LET P(I) = INT(L*RND(TI)+1)
440 :     FOR J = 1 TO I-1
450 :       IF P(I) = P(I-J) THEN 430  : REM ZAHL SCHON EINMAL ERZEUGT
455 :     NEXT J
460 :   NEXT I
470 :
500 : REM --- VERWUERFELTE BUCHSTABEN AUSGEBEN
501 :
510 : PRINT "■■■■■";
520 : FOR I = 1 TO L : PRINT MID$(W$,P(I),1)"  ";: NEXT I
530 :
540 : FOR K = 1 TO 1000: NEXT K         : REM BEDENKZEIT
590 :
600 REM === RATEVERSUCH ==============================================
601 :
610 : PRINT "■■■■■IHRE ANTWORT?
620 : INPUT A$
630 :
640 : IF A$ = W$ THEN PRINT "■■■■RICHTIG GERATEN!": GOTO 690
650 :                  PRINT "■■■■LEIDER FALSCH.
690 : END
699 :
700 REM === BEGRIFFE UND STICHWOERTER ================================
701 :
710 DATA KARL MAY, BEDEUTENDER REISESCHRIFTSTELLER
720 DATA OLD SHATTERHAND, STRECKT GEGNER MIT JAGDHIEB NIEDER
730 DATA WINNETOU, BERUEHMTER HAEUPTLING - HAT NIE GELEBT
740 DATA HADSCHI HALEF OMAR, TRAEGER VON SIEBEN BARTHAAREN
750 :
760 REM HIER FUEGEN SIE BITTE NOCH DREISSIG WEITERE NAMEN EIN
790 :
```

Beispiel 5 : Knack' den Code

Dieses auch als 'Superhirn' bekannte Spiel verlangt, daß der Spieler eine vier- oder fünfstellige Zahl errät, wobei ihm folgende Information zur Verfügung steht: ist eine Ziffer erraten, und zwar an der richtigen Stelle, heißt es 'Volltreffer!'; stimmt die Ziffer, aber nicht die Stelle, heißt es 'Halbtreffer!'.
Der Computer wählt die Ziffern, der Spieler rät.

D i a l o g :

```
1. Versuch: Wie lautet Ihre Zahl? 1234
Sie erreichten 0 Volltreffer und 1 Halbtreffer.

2. Versuch: Wie lautet Ihre Zahl? @

Die unbekannte Zahl lautete  5619 .
```

V e r f a h r e n : Zunächst wird - nach einem etwas anderen Verfahren als im vorigen Beispiel - eine zufällige Permutation der Ziffern von 1 bis 9 erzeugt.
Dann wird die vom Spieler eingegebene Zahl (bzw. Ziffernfolge) auf Treffer untersucht. Dazu kopieren wir die in der Tabelle u(1),..., u(4) gegebenen unbekannten Ziffern in eine Tabelle c(1),..., c(4) und schreiben die Lösungsziffern des Spielers in eine Tabelle L(1),..., L(4). Nun stellen wir fest, wie oft L(i) = c(i) vorkommt: dies ist die Anzahl der Volltreffer (Zeile 760 - 810). Die Anzahl der Halbtreffer stellen wir dadurch fest, daß wie ermitteln, wie oft L(i) = c(j) für ein Paar i,j ε {1,2,3,4} auftritt (Zeilen 830 - 890 des Programms auf der folgenden Seite).
In der Anleitung hätte noch gesagt werden müssen, daß der Spieler durch Drücken der Taste " @ " (Affen-A) aufgeben, d.h. die unbekannte Zahl sich nennen lassen kann (siehe den Dialog oben). Fügen Sie bitte in Zeile 280 diesen Hinweis ein!

```
100 : PRINT "■            ■KNACK DEN CODE
110 :
120 REM DER COMPUTER SPIELT DAS BEKANNTE RATESPIEL 'SUPERHIRN'
121 :
200 REM === SPIELREGELN ================================================
201 :
210 : PRINT
220 : PRINT "ICH, DER COMPUTER DENKE MIR EINE VIERSTELLIGE ZAHL AUS
230 : PRINT "LAUTER VERSCHIEDENEN ZIFFERN, DIE SIE ERRATEN SOLLEN.
240 : PRINT "WENN EINE ZIFFER AN DER RICHTIGEN STELLE SITZT, IST DIES
250 : PRINT "EIN VOLLTREFFER; WENN EINE ZIFFER ERRATEN IST, ABER NICHT
260 : PRINT "AN DER RICHTIGEN STELLE SITZT, IST ES EIN HALBTREFFER.
290 :
300 REM === BESTIMMUNG DER ZU RATENDEN ZAHL ============================
301 :
310 : FOR I = 1 TO 9 : LET U(I) = I : NEXT I
320 : FOR I = 9 TO 2 STEP -1
330 :   LET K = INT(I*RND(TI))
340 :   LET H = U(I) : LET U(I) = U(K) : LET U(K) = H
350 : NEXT I
390 :
400 REM === DIALOG =====================================================
401 :
410 : LET N = 0               : REM ANZAHL DER VERSUCHE
420 :
430 : LET N = N+1
435 :   PRINT : PRINT
440 :   PRINT N;". VERSUCH:  ";
450 :   INPUT "WIE LAUTET IHRE ZAHL  "; L$
455 :   IF L$ = "0" THEN 620 : REM SPIELER GIBT AUF
460 :   LET L$ = LEFT$(L$,4)
470 :
480 :   GOSUB 700 : REM ZUR TREFFERBESTIMMUNG
490 :
500 :   IF V = 4 THEN 600    : REM ZAHL ERRATEN
510 :
520 :   PRINT
530 :   PRINT " SIE ERREICHTEN ";V;" VOLLTREFFER ";
540 :   PRINT "UND ";H;" HALBTREFFER."
550 : GOTO 430
590 :
600 : PRINT : PRINT
610 : PRINT "SIE HABEN DIE UNBEKANNTE ZAHL IN ";N;" VERSUCHEN ERRATEN.
620 : PRINT
630 : PRINT "DIE UNBEKANNTE ZAHL LAUTETE: ";
640 : FOR I = 1 TO 4 : PRINT U(I);: NEXT I
650 :
690 : END : REM === ENDE DES DIALOGS ==================================
699 :
```

```
700 REM +++ UNTERPROGRAMM TREFFERBESTIMMUNG ++++++++++++++++++++++++++++
710 : FOR I = 1 TO 4
720 :   LET L(I) = VAL(MID$(L$,I,1))
730 :   LET C(I) = U(I)
740 : NEXT I
750 :
760 : LET V = 0  : REM ANZAHL DER VOLLTREFFER
770 : FOR I = 1 TO 4
780 :   IF L(I) <> C(I) THEN 810
790 :     LET V = V+1
800 :     LET L(I) = -1 : LET C(I) = -2
810 : NEXT I
820 :
830 : LET H = 0  : REM ANZAHL DER HALBTREFFER
840 : FOR I = 1 TO 4
850 :   FOR J = 1 TO 4
860 :     IF L(I) <> C(J) THEN 880
865 :       LET L(I) = -1 : LET C(J) = -2
870 :       LET H = H+1
880 :   NEXT J
890 : NEXT I
895 :
899 : RETURN
```

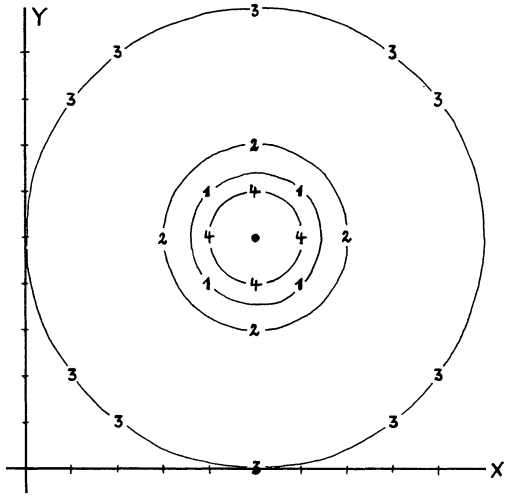
Beispiel 6 : Spionsuche

Innerhalb eines 10 x 10 - Gitters sind vier 'Spione' versteckt, die aufgefunden werden sollen. Der Computer nennt die Standorte der enttarnten Spione und er gibt die Entfernung zu den nicht-enttarnten an.

D i a l o g :

```
Die Spione sind jetzt gut versteckt.

1. Versuch: wo sitzt ein Spion?

x = ? 5
y = ? 5

Die Entfernung zu Spion 1 beträgt 1.4 Längeneinheiten
Die Entfernung zu Spion 2 beträgt  2  Längeneinheiten
Die Entfernung zu Spion 3 beträgt  5  Längeneinheiten
Die Entfernung zu Spion 4 beträgt  1  Längeneinheit.
```

Hieraus läßt sich über die möglichen Verstecke der Spione folgendes entnehmen:



Der Dialog geht damit wie folgt weiter:

```
2. Versuch: wo sitzt ein Spion?
x = ? 5
y = ? 4

Die Entfernung zu Spion 1 beträgt  1  Längeneinheit
Die Entfernung zu Spion 2 beträgt  3  Längeneinheiten
Die Entfernung zu Spion 3 beträgt 5.8 Längeneinheiten
Sie haben Spion 4 erwischt!

3. Versuch: wo sitzt ein Spion?
x = ?
```

Das Programm ist - wie üblich - in die Teile (1) Anleitung, (2) Setzen der Anfangswerte, hier: Verstecken der Spione, (3) Spiel und (4) Wiederholung oder Verabschiedung gegliedert. Der dritte Teil verläuft wie folgt:

SPIEL

```
FÜR i VON 1 BIS 10 WIEDERHOLE
  Unterprogramm 'Rateversuch'
  WENN alle Spione gefunden DANN
     Z$ := 'Partie gewonnen'
     Unterprogramm 'Spiel' wird verlassen
  ENDE-WENN
ENDE-WIEDERHOLE
Unterprogramm 'Partie verloren'
```

Das vollständige BASIC-Programm findet sich auf den folgenden Seiten.

```
100 : PRINT "□"
110 : PRINT "            SPIONSUCHE
111 : PRINT "            ----------
112 :
120 REM EINFACHES SUCHSPIEL IN ZWEI DIMENSIONEN
190 :
200 REM *** HAUPTPROGRAMM **********************************************
201 :
230 : GOSUB 300 : REM SPIELREGELN
240 : GOSUB 400 : REM SPIONE VERSTECKEN
250 : GOSUB 500 : REM PARTIE
270 : GOSUB 990 : REM WIEDERHOLUNG ODER VERABSCHIEDUNG
280 : END
295 :
297 REM *** ENDE HAUPTPROGRAMM ******************************************
298 :
299 :
300 REM ### UNTERPROGRAMM 'SPIELREGELN' ################################
301 :
310 : PRINT "■■ MOECHTEN SIE DIE SPIELREGELN ERFAHREN? (J/N)
315 : GET A$ : IF A$ = "" THEN 315
320 : IF A$ <> "J" THEN RETURN
325 :
330 : PRINT "■AUF EINEM 10 MAL 10 FELD HALTEN SICH 4 SPIONE VERSTECKT.
335 : PRINT "SIE SOLLEN SIE FINDEN, INDEM SIE DIE POSITION
340 : PRINT "JEDES SPIONS DURCH ZWEI ZAHLEN ZWISCHEN 0 UND 9 ";
345 : PRINT "KENNZEICHNEN.
350 : PRINT "ICH SAGE IHNEN DANN, OB SIE ERFOLGREICH WAREN;
355 : PRINT "ANDERNFALLS NENNE ICH IHNEN DIE ENTFERNUNG ZUM SPION.
360 : PRINT "■SIE HABEN INSGESAMT 10 VERSUCHE.
365 : PRINT "■■     ■TASTE DRUECKEN!
370 :
380 : GET T$ : IF T$ = "" THEN 380
385 :
390 : RETURN
399 :
400 REM ### UNTERPROGRAMM 'SPIONE VERSTECKEN' ########################
401 :
410 : FOR I = 1 TO 4
420 :   LET X(I) = INT(10*RND(TI))
430 :   LET Y(I) = INT(10*RND(TI))
440 : NEXT I
450 :
490 : RETURN
499 :
500 REM ### UNTERPROGRAMM 'PARTIE' ####################################
501 :
510 : FOR J = 1 TO 10
520 :   GOSUB 600                        : REM RATEVERSUCH
530 :   GOSUB 700                        : REM PARTIE GEWONNEN?
535 :   IF Z$ = "GEWONNEN" THEN RETURN : REM PARTIE ZU ENDE
540 :   PRINT "■     ■TASTE DRUECKEN!" : REM BEDENKZEIT
545 :   GET T$ : IF T$ = "" THEN 545   : REM WEITER GEHT'S
550 : NEXT J
555 :
560 : GOSUB 800                        : REM PARTIE VERLOREN
570 :
590 : RETURN : REM ENDE 'PARTIE' ###################################
599 :
```

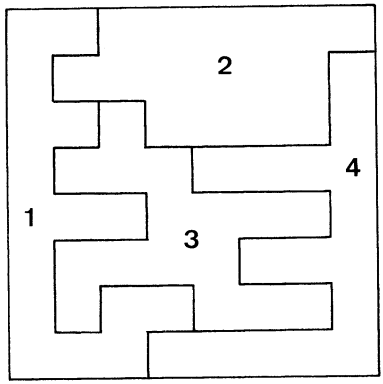
```
600 : REM +++ UNTERPROGRAMM 'RATEVERSUCH' ++++++++++++++++++++++++++++
601 :
610 :   PRINT "■■■DIE SPIONE HABEN SICH JETZT GUT VERSTECKT.
620 :   PRINT "■■";J;". VERSUCH: WO SITZT EIN SPION? "
621 :
625 :   INPUT " X = "; X
626 :   IF X < 0 OR X > 9 THEN 625
630 :   INPUT " Y = "; Y
631 :   IF Y < 0 OR Y > 9 THEN 630
635 :
640 :   FOR I = 1 TO 4
645 :     IF X(I) = - 1 THEN 695          : REM HIER SASS FRUEHER EINER
650 :     IF X(I) <> X OR Y(I) <> Y THEN 680      : REM SPION VERFEHLT
655 :     PRINT "■SIE HABEN SPION NR.";I;" ERWISCHT!
660 :     LET X(I) = -1                   : REM FUNDSTELLE KENNZEICHNEN
665 :     GOTO 695                        : REM NAECHSTER SPION
670 :
680 :     LET D = SQR((X(I)-X)↑2 + (Y(I)-Y)↑2) : REM ENTFERNUNG
685 :     PRINT "■DIE ENTFERNUNG ZU SPION NR.";I;" BETRAEGT";
690 :     PRINT INT(10*D + 0.5)/10;" LAENGENEINHEITEN.
695 :   NEXT I
696 :
697 :   RETURN : REM ENDE 'RATEVERSUCH' ++++++++++++++++++++++++++++++
699 :
700 : REM +++ UNTERPROGRAMM 'PARTIE GEWONNEN?' ++++++++++++++++++++++
701 :
710 :   FOR I = 1 TO 4
720 :     IF X(I) = -1 THEN 740
730 :     RETURN : REM NOCH NICHT ALLE ERWISCHT, NEUER VERSUCH
740 :   NEXT I
750 :
760 :   PRINT "■■SIE HABEN ALLE SPIONE IN";J;" VERSUCHEN GEFUNDEN!
770 :   LET Z$ = "GEWONNEN"                                    ●
780 :
790 :   RETURN : REM ENDE 'PARTIE GEWONNEN?' ++++++++++++++++++++++++++
799 :
800 : REM +++ UNTERPROGRAMM 'PARTIE VERLOREN' ++++++++++++++++++++++++
801 :
810 :   PRINT "■■■
820 :   PRINT "SIE HABEN DIE SPIONE NACH 10 VERSUCHEN ";
825 :   PRINT "LEIDER NICHT ERWISCHT.
830 :
840 :   PRINT "■■MOECHTEN SIE WISSEN, WO DIE SPIONE SASSEN?(J/N)
845 :   GET A$ : IF A$ = "" THEN 845
850 :   IF A$ <> "J" THEN RETURN
855 :
860 :   PRINT "■■
865 :   FOR I = 1 TO 4
870 :     IF X(I) = -1 THEN 880
875 :     PRINT "SPION NR.";I;" SASS IM PUNKT (";X(I);",";Y(I);")"
880 :   NEXT I
885 :
890 :   RETURN : REM ENDE 'PARTIE VERLOREN' ++++++++++++++++++++++++++++
899 :
900 REM ### UNTERPROGRAMM 'WIEDERHOLUNG ODER VERABSCHIEDUNG' ##########
901 :
910 : PRINT "■■■■ NOCH EINE PARTIE? (J/N)
920 : GET A$ : IF A$ = "" THEN 920
930 : IF A$ = "J" THEN RUN
940 :
990 : RETURN
```

Als letztes Beispiel dieses Kapitels wollen wir ein Spiel kennenlernen, welches die Züge von 'Superhirn' und 'Schiffe versenken' in sich vereint:

Beispiel 7: LAP (L.A.Pijanowski)

Ein 8 x 8 - Gitter ist - dem Spieler unbekannt - in vier gleichgroße, zusammenhängende Regionen aufgeteilt. Sein Ziel ist es, die genauen Grenzen der Regionen herauszufinden. Dazu darf er Fragen folgender Art stellen: er nennt ein 2 x 2 - Quadrat; der Spielgegner gibt daraufhin an, wieviele der vier Felder in jeder der Regionen liegen.

Angenommen, das Feld ist in folgende vier Regionen eingeteilt:



Dann spielt sich folgender D i a l o g ab:

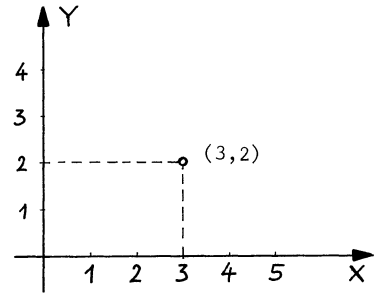
```
Geben Sie das Testquadrat ein, und zwar
nur die Koordinaten seiner linken oberen Ecke!
Zeile ? 1
Spalte? 1

3 Felder liegen in Region 1
1 Felder liegen in Region 2
0 Felder liegen in Region 3
0 Felder liegen in Region 4 .

Wünschen Sie alle Regionen zu sehen?(j/n) n

Geben Sie das Testquadrat ein ...
```

Bevor wir mit dem Programmentwurf beginnen, sollten wir uns einmal klarmachen, welche Möglichkeiten es gibt, ein zweidimensionales Gitter auf dem Computer zu realisieren.

1. Koordinatenmethode:

   Ihr Vorteil ist, daß sie an die üblichen mathematischen Bezeichnungen anknüpft.

   Jeder Punkt ist durch ein Zahlenpaar (x,y) gekennzeichnet.



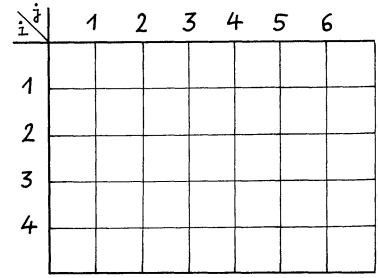
2. Matrizenmethode:

   Man gibt Zeile i und Spalte j ein; ihr Vorteil ist leichte Programmierbarkeit.
   Die vier Felder eines Quadrats haben einfach

   i,j i,j+1

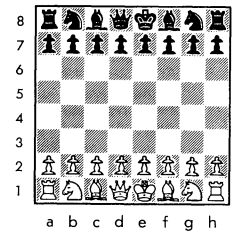
   i+1,j i+1,j+1

   als Zeile und Spalte.



3. Schachbrettmethode:

   Sie werden wir im Zusammenhang mit den Brettspielen anwenden. Jeder Punkt ist durch ein Paar (Buchstabe, Ziffer) gekennzeichnet.



Beim Spiel LAP verwenden wir die Matrizenmethode: dann können wir eine gegebene Aufteilung in Regionen ganz leicht in DATA-Zeilen festhalten und einlesen:

```
FOR I = 1 TO 8
  FOR J = 1 TO 8 : READ F(I,J) : NEXT J
NEXT I
```

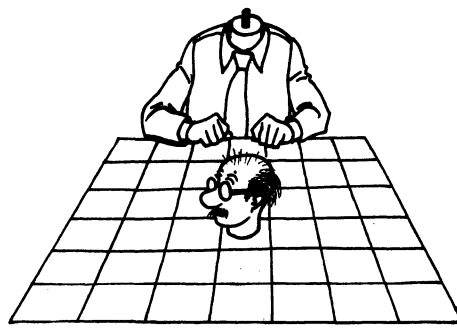
In der Tabelle M(1),..., M(4) zählen wir, wieviele Felder des Testquadrats in Region K liegen. Dies geht ganz leicht:

```
FOR Z1 = 0 TO 1
  FOR S1 = 0 TO 1
    LET K = F(Z+Z1, S+S1)
    LET M(K) = M(K) + 1
  NEXT S1
NEXT Z1
```

Hat nämlich der Spieler das Paar Z,S als Zeile und Spalte der linken oberen Ecke seines Testquadrats eingegeben, so durchläuft die Schleife die Felder Z,S Z,S+1 Z+1,S Z+1,S+1. Es ist K = F(Z+Z1, S+S1) die Zahl, welche die jeweilige Region bezeichnet, und damit wird in der zugehörigen Variablen M(K) um eins weitergezählt. Ist beispielsweise Z,S = 1,1, so ist K = F(1,1) = 1 und damit gilt M(1) = M(1) + 1. Das folgende Programm ist noch sehr ausbaufähig!

```
100 : PRINT "
110 : PRINT "                    LAP
111 : PRINT "                    ---
112 :
120 REM SPIEL VON L. PIJANOWSKI, COPYRIGHT HUGENDUBEL-VERLAG MUENCHEN
121 :
200 REM === ANLEITUNG ================================================
201 :
210 : REM --- HIER BITTE DIE SPIELREGELN EINFUEGEN ---
290 :
300 REM === INITIALISIERUNG ===========================================
301 :
310 : FOR I = 1 TO 8
320 :   FOR J = 1 TO 8 : READ F(I,J) : NEXT J
330 : NEXT I
340 :
361 DATA 1,1,2,2,2,2,2,2
362 DATA 1,2,2,2,2,2,2,4
363 DATA 1,1,3,2,2,2,2,4
364 DATA 1,3,3,3,4,4,4,4
365 DATA 1,1,1,1,3,3,3,4
366 DATA 1,3,3,3,3,4,4,4
367 DATA 1,3,1,1,3,3,3,4
368 DATA 1,1,1,4,4,4,4,4
369 :
```

```
400 REM === SPIEL ====================================================
401 :
410 : PRINT
420 : PRINT "GEBEN SIE DAS TESTQUADRAT EIN, UND ZWAR
425 : PRINT "NUR DIE KOORDINATEN SEINER LINKEN OBEREN ECKE!
430 : PRINT
440 : INPUT "ZEILE  "; Z
445 : INPUT "SPALTE "; S
450 :
460 : PRINT
470 : PRINT "ZUM TESTQUADRAT GEHOEREN DIE FELDER ";
475 : PRINT Z;S;"  ";Z;S+1;"  ";Z+1;S;"  ";Z+1;S+1
480 :
500 : FOR K = 1 TO 4 : LET M(K) = 0 : NEXT K
510 :
520 : FOR Z1 = 0 TO 1
530 :   FOR S1 = 0 TO 1
540 :     LET K = F(Z+Z1,S+S1)
550 :     LET M(K) = M(K)+1
560 :   NEXT S1
570 : NEXT Z1
590 :
600 : PRINT
610 : FOR K = 1 TO 4
620 :   PRINT M(K); "FELDER LIEGEN IN REGION "; K
630 : NEXT K
640 :
700 : PRINT
710 : PRINT "WUENSCHEN SIE ALLE REGIONEN ZU SEHEN?(J/N)
720 : GET A$ : IF A$ = "" THEN 720
730 : IF A$ <> "J" THEN 400 : REM NEUER VERSUCH
740 :
800 REM === ENTHUELLUNG ==============================================
801 :
810 : PRINT "□"
820 : FOR I = 1 TO 8
830 :   FOR J = 1 TO 8 : PRINT F(I,J);: NEXT J
840 :   PRINT
850 : NEXT I
880 :
890 : END
```



Aufgaben

1. Um hinter das Wunder des Nicomachus in Beispiel 1 zu kommen, unternehmen Sie bitte folgendes:

   a) Geben Sie die Reste 0,0,1; 0,1,0; 1,0,0 ein und schauen Sie sich zur Erklärung Zeile 300 des Programms an.

   b) Lassen Sie sich eine Tabelle der Siebener, Elfer- und Dreizehnerreste ausdrucken; etwa so:

Zahl	7er-Rest	11er-Rest	13er-Rest
500	3	5	6
501	4	6	7
502	5	7	8
503	6	8	9
504	0	9	10
505	1	10	11
506	2	0	12
.....			

   Wieviele solcher Reste-Kombinationen gibt es?

   c) Experimentieren Sie auch mit anderen Zahlen (anstelle von 7,11,13)!

2. Geben Sie dem Ratenden bei der Intervallsuche (Beispiel 2) mehr Information, indem Sie ihm mitteilen, ob die zu erratende Zahl rechts von der oberen Grenze oder links von der unteren Grenze des Suchintervalls liegt.

3. Bauen Sie in das Programm zur Intervallsuche einen Zähler der Versuche ein und bewerten Sie die jeweilige Rateleistung. Sie müssen dazu überlegen, wieviel Versuche - bei optimaler Suchstrategie - erforderlich sind, um die unbekannte Zahl zu finden.
   Vergleichen Sie auch mit dem Ratespiel 'Zu groß - zu klein' (Beispiel 3 von Kapitel 1).

4. Beim Programm 'Begriffe raten' sollen die einzelnen Begriffe vorher umschrieben werden; z.B.: "Wer riß sich selbst mitten entzwei?" als Hinweis auf 'Rumpelstilzchen'. Erweitern Sie das Programm in diesem Sinne.

5. Das Programm 'Begriffe raten' soll die Anzahl der Rateversuche protokollieren und die Rateleistung kommentieren.

6. Galgenmännchen.
   Erweitern Sie das Wortratespiel von Beispiel 3 so, daß sich bei jedem Rateversuch - Schritt für Schritt - ein Galgen oder ein Monster aufbaut, dem der Ratende anheimfällt, wenn er nicht rechtzeitig zu Potte kommt.

7. Bauen Sie das Programm 'Verwürfelt' so aus, daß nach der Anzahl der zu ratenden Worte gefragt wird und Schwierigkeitsgrade wählbar sind. Eine Bewertung der Rateleistung soll sich anschließen.

8. Lassen Sie die verwürfelten Buchstaben von Beispiel 4 an zufälligen Stellen des Bildschirms erscheinen!

9. Analog zu Beispiel 4 sollen S p r i c h w o r t e geraten werden, deren Bestandteile an zufälligen Stellen des Bildschirms kurz zu sehen sind.

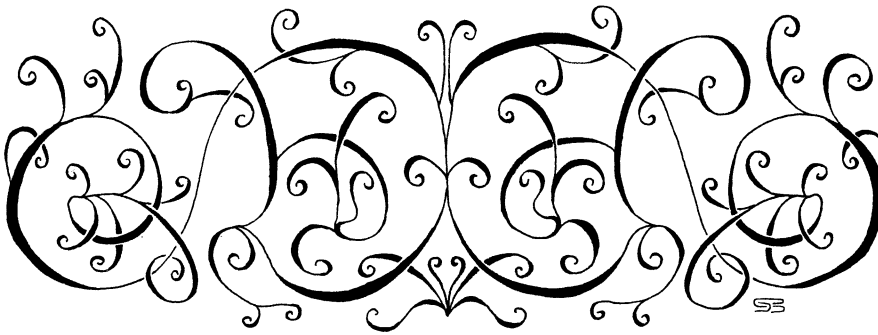
10. Erweitern Sie das Programm 'Knack den Code' so, daß der Spieler die Länge der zu ratenden Ziffernfolge (4 oder 5) selbst vorher wählen kann.

11. Wir betrachten eine vereinfachte Form von 'Knack den Code': es seien nur die Ziffern 1,2,3,4 zugelassen. Für diesen Fall soll eine optimale Ratestrategie entworfen und dem Computer mitgegeben werden (d.h. der Computer ist jetzt Ratender).
Hinweis: es gibt eine Strategie, bei der man in mehr als der Hälfte aller Fälle die richtige Ziffernfolge (Anordnung) bereits nach drei Versuchen kennt.

12. Ergänzen Sie das Programm 'Spionsuche' durch geeignete - zufällig zu wählende - Trostantworten für den Fall des Mißerfolgs.

13. Schatzsuche.
Ein Schatz ist auf einem n x n - Gitter verborgen. Der Ratende erhält als Information die Himmelsrichtung; 5 Versuche stehen ihn im 10 x 10 - Gitter zur Verfügung. Entwerfen Sie ein Programm!

14. Das Programm 'LAP' ist dergestalt zu erweitern, daß der Computer zufällige Einteilungen des Gitters in vier Regionen (jede mit 16 Feldern und zusammenhängend!) vornimmt. Ferner sind auch hier die Versuche zu zählen, und es soll die Leistung des Spielers bewertet werden.



5
Glücksspiele

Beim sogenannten Geschicklichkeitsspiel bestimmen körperliche und/oder geistige Fähigkeiten der Spieler den Spielausgang; beim G l ü c k s s p i e l dagegen hängen Gewinn oder Verlust ausschließlich oder vorwiegend vom Zufall ab. Die Abgrenzung beider Spieltypen voneinander ist schwierig; die im vorliegenden Kapitel betrachteten Spiele weisen alle das Merkmal auf, daß Aktionen der Spieler den Ausgang beeinflussen - und sei es auch nur aufgrund der Entscheidung, wann das Spiel abgebrochen wird. -
Jedes Glücksspiel benötigt einen 'Zufallsgenerator': die ältesten Zufallsgeräte sind Würfel, die ältesten Glücksspiele Würfelspiele; sie wurden schon von den alten Ägyptern, Griechen und Römern leidenschaftlich betrieben. Eingeführt wurde das Würfeln wohl von Priestern und Zauberern, um den Willen der Götter zu ergründen; dann hat es sich als Spiel verselbständigt. Der Pharao Rhampsinitos, so erzählt Herodot, stieg in die Unterwelt, wo er mit der Göttin Demeter Würfel spielte. "Teils gewann er, teils verlor er; als Geschenk der Demeter brachte er ein goldenes Handtuch mit" (Historien II, 122).
Gewürfelt wurde ursprünglich so ziemlich mit allem, was irgendwie rollt und dann in einer definierten Stellung liegen bleibt (Muscheln, Stäbchen, Steine, Knochen). Die frühesten Würfel trugen 1 und 2, 3 und 4, 5 und 6 auf entgegengesetzten Seiten; seit etwa 1400 v. Chr. kennt man die heute übliche Form (die Augenzahlen gegenüberliegender Seiten ergänzen sich zu 7).

Beispiel 1: Die böse Sechs

Bei diesem Spiel für n Personen werden jeweils zwei Würfel geworfen. Wer am Zug ist, kann solange werfen, wie er möchte. Hört er nach einer Anzahl von Würfen auf, so wird ihm die erzielte Augensumme gutgeschrieben - sofern er keine 6 gewürfelt hat. Beim Auftreten einer 6 (auf einem oder beiden Würfeln) jedoch wird ihm für diese Runde nichts gutgeschrieben und er muß an den nächsten Spieler übergeben. Wer zuerst die Punktzahl 100 erreicht oder übertrifft, hat gewonnen.-
Der Reiz des Spiels besteht im Abwägen der Chance einer Erhöhung der Punktzahl gegenüber dem Risiko des Auftretens einer 6 bei fortgesetztem Werfen.

Für den Spieler erhebt sich nach jedem Wurf die Frage, ob er noch einmal werfen, oder ob er aufhören und an den nächsten Spieler übergeben soll. Die Entscheidung muß er von der bereits erreichten Gesamtpunktzahl, von der in der laufenden Runde erzielten Augensumme und von den Gesamtpunktzahlen der Mitspieler abhängig machen. Diese Entscheidungssituation ist so komplex, daß wir sie nicht unmittelbar durchschauen können. Um uns an das Problem heranzutasten, betrachten wir zunächst eine vereinfachte Version des Spiels. Bei ihr kümmern wir uns nicht um die Mitspieler - der Spieler steht allein vor der Entscheidung "soll ein weiterer Wurf (mit zwei Würfeln) gewagt werden?".
Zufallsgerät ist der Computer, der D i a l o g mit ihm lautet:

```
Wollen Sie noch einmal würfeln?(j/n) j

Sie haben 1 und 4 geworfen.
Erreichte Zwischensumme: 5

Wollen Sie noch einmal würfeln?(j/n) j

Sie haben 2 und 4 geworfen.
Erreichte Zwischensumme: 11

Wollen Sie noch einmal würfeln?(j/n) n


Ihre Punktzahl: 11
Sie hätten noch 23 Punkte mehr erreichen können!


Noch eine Partie?  n

Auf Wiedersehen.
```

Woher weiß nun der Computer, wieviel Punkte der Spieler "hätte mehr erreichen können"? Nun, er spielt die gesamte Serie - bis zum Auftreten einer Sechs - vorweg durch und merkt sich die Ergebnisse in zwei Tabellen W1 (Wurf 1) und W2 (Wurf 2). Das eigentliche Spiel besteht nur im Abrufen dieser gespeicherten Ergebnisse, und im richtigen Aufhören. Im letzteren Fall gibt der Computer die Differenz S - S1 zwischen der vom Spieler erreichten Summe S1 und der im Vorlauf erzielten Summe S bekannt. Hat der Spieler zu spät aufgehört, d.h. ist er bis zum Ende der gespeicherten Ergebnisfolge vorgedrungen, so heißt es "Verloren!" (siehe Zeile 470 des folgenden Programms).

```
100 : PRINT "■
110 : PRINT "          DIE BOESE SECHS
111 : PRINT "          ---------------
112 :
120 REM VEREINFACHTE VERSION DES GLUECKSSPIELS NAMENS PIG
121 :
200 REM === SPIELREGELN ================================================
201 :
210 : PRINT "SIE WERFEN SO OFT SIE MOEGEN ZWEI WUERFEL;
215 : PRINT "DIE AUGENZAHLEN WERDEN ADDIERT.
220 : PRINT "ERSCHEINT ALLERDINGS (BEI EINEM ODER BEIDEN WUERFELN)
225 : PRINT "EINE SECHS, SO GEHEN SIE DER BISHER ERREICHTEN
230 : PRINT "AUGENSUMME VERLUSTIG.
290 :
300 REM === SPEICHERUNG EINER WURFSERIE ================================
301 :
310 : DIM W1(30), W2(30)            : REM TABELLEN FUER DIE WUERFE
315 :
320 : LET S = 0                     : REM ANFANGSWERT DER AUGENSUMME
325 :
330 : LET I = 1                     : REM ZAEHLER DER WUERFE
335 :
340 : LET W1 = INT(6*RND(TI)+1)     : REM ERSTER  WUERFEL FAELLT
345 : LET W2 = INT(6*RND(TI)+1)     : REM ZWEITER WUERFEL FAELLT
350 :
355 : LET W1(I) = W1
360 : LET W2(I) = W2
365 :
370 : IF W1 = 6 OR W2 = 6 THEN 400 : REM DIE BOESE 6 IST ERSCHIENEN
375 :
380 : LET S = S + W1 + W2
385 :
390 : LET I = I+1 : GOTO 340       : REM NEUER WURF
395 :
```

```
400 REM === WURFENTSCHEIDUNG ========================================
401 :
410 : LET S1 = 0                        : REM AUGENSUMME DES SPIELERS
415 :
420 : LET I = 1                         : REM ZAEHLER DER WUERFE
425 :
430 : PRINT "■WOLLEN SIE (NOCH) EINMAL WUERFELN?(J/N)
435 : GET A$ : IF A$ = "" THEN 435
440 : IF A$ = "N" THEN 500              : REM SPIELER HAT GENUG
445 :
450 : LET W1 = W1(I)
455 : LET W2 = W2(I)
460 :
465 : PRINT "■SIE HABEN ";W1;" UND ";W2;" GEWORFEN.
470 : IF W1 = 6 OR W2 = 6 THEN PRINT "■■■VERLOREN!": GOTO 600
475 :
480 : LET S1 = S1 + W1 + W2
485 : PRINT "ERREICHTE ZWISCHENSUMME: "; S1
490 :
495 : LET I = I+1 : GOTO 430
499 :
500 REM === BILANZ ===================================================
501 :
510 : PRINT "■■IHRE PUNKTZAHL: "; S1
520 : PRINT "SIE HAETTEN "; S-S1;" PUNKTE MEHR ERREICHEN KOENNEN!
590 :
600 REM === WIEDERHOLUNG ODER VERABSCHIEDUNG ========================
601 :
610 : PRINT "■■■NOCH EINE PARTIE?
620 : GET A$ : IF A$ = "" THEN 620
630 : IF A$ = "J" THEN RUN
640 :
650 : PRINT "■■■AUF WIEDERSEHEN.
660 :
690 : END
```

Wenn Sie mit diesem Programm spielen, werden Sie etwa herausfinden, bei welcher jeweils erreichten Augensumme es geraten ist, abzubrechen. Wir wollen dies Ergebnis durch eine leichte Rechnung theoretisch herleiten. Angenommen, Sie haben bereits s Punkte erreicht (der Computer sagt: 'erreichte Zwischensumme: ...'). Wenn Sie nun ein weiteres Mal würfeln, so können Sie die Augenzahlen

$$2 \ , \ 3 \ , \ 4 \ , \ 5 \ , \ 6 \ , \ 7 \ , \ 8 \ , \ 9 \ , \ 10$$

mit den Wahrscheinlichkeiten

$$\frac{1}{36} \ , \frac{2}{36} \ , \frac{3}{36} \ , \frac{4}{36} \ , \frac{5}{36} \ , \frac{4}{36} \ , \frac{3}{36} \ , \frac{2}{36} \ , \frac{1}{36}$$

erzielen,

und sie können die bereits gewonnenen s Punkte mit der Wahrscheinlichkeit $\frac{11}{36}$ verlieren. Der Erwartungswert ist demnach

$$\frac{1\cdot 2 + 2\cdot 3 + 3\cdot 4 + 4\cdot 5 + 5\cdot 6 + 4\cdot 7 + 3\cdot 8 + 2\cdot 9 + 1\cdot 10 - 11\cdot s}{36}$$

$$= \frac{150 - 11s}{36} \quad .$$

Der Wert dieses Terms ist genau dann positiv, wenn $150 > 11s$ oder $s < 14$ ist. Das heißt: sind bereits 14 oder mehr Punkte erzielt, so sollte man mit dem Werfen aufhören.-

Dieses Ergebnis gilt allerdings nur, wenn wir uns nicht darum kümmern, wie weit wir vom gesteckten Ziel (100 Punkte) entfernt sind, und wieviel Punkte die Mitspieler bereits auf ihrem Konto haben.

Nun wollen wir einige Überlegungen zum 'Endspiel' anstellen. Angenommen, Sie haben nur einen Gegenspieler, dessen erreichte Gesamtpunktzahl liegt nahe bei 100, und Sie sind an der Reihe. In diesem Fall ist es besser, die eben aufgestellte Regel nicht anzuwenden, sondern weiterzuwürfeln in der Hoffnung, als erster das Ziel zu erreichen. Wenn Sie z.B. bereits 98 oder 99 Punkte haben, und Sie werfen 2 oder mehr, so übertreffen bzw. erreichen Sie die 100. Werfen Sie 11 oder 12, bleiben Sie auf dem bisher erreichten 'Plateau' stehen und Ihr Gegner ist an der Reihe. Die Wahrscheinlichkeit, das Spiel jetzt zu gewinnen, beträgt $\frac{25}{36}$.

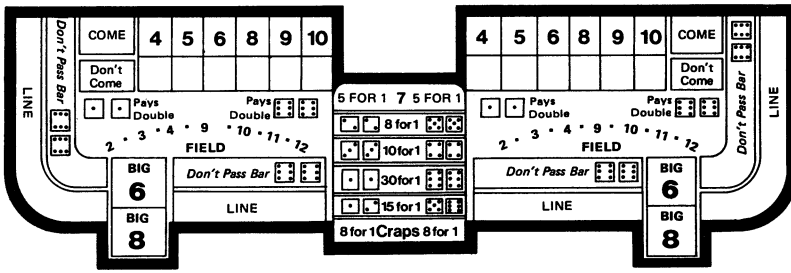
Auf diese Weise können Sie eine Tabelle der Gewinnwahrscheinlichkeiten in Abhängigkeit von Ihrem Plateau aufstellen.-

Bei mehr als zwei Mitspielern ist die Analyse schwieriger. In den Aufgaben finden Sie weitere Anregungen und Hinweise zur Beschäftigung mit diesem interessanten Spiel.

Das - insbesondere in den Vereinigten Staaten - wohl am weitesten verbreitete Würfelspiel nennt sich 'Craps'. Es stammt direkt von 'Hazard' ab, welches im 12. Jahrhundert von englischen Kreuzfahrern erfunden worden und im 19. Jahrhundert in England und Frankreich sehr populär war. Die Bezeichnung leitet sich vermutlich von dem englischen 'Crabs' ('Krabben', der Bezeichnung für die Fehlwürfe mit den Augenzahlen 2,3 oder 12) her.

Beispiel 2 : Craps

Das Spiel wird mit zwei Würfeln gespielt. Erscheint als Augensumme eine 7 oder eine 11, so hat der Spieler gewonnen (diese Würfe heißen 'Naturals'); erscheint 2,3 oder 12, so hat der Spieler verloren (dies sind die schon erwähnten 'Craps'). Erscheint 4,5,6,8,9 oder 10 ('Punkte' genannt), so fährt der Spieler mit Werfen fort, bis er entweder seinen Punkt erneut erreicht oder eine 7 geworfen hat; letzteres bedeutet Verlust, ersteres Gewinn der Partie.
Gespielt wird gegen die Bank, die im Fall des Verlusts den Einsatz einstreicht, im Fall des Gewinns den Einsatz draufzahlt.-
Craps wird vorzugsweise von am Boden hockenden Spielern praktiziert, die ihre Würfel gegen eine Wand klatschen. Im Casino spielt man es auf einem Tisch folgender Art:



Hinten ist eine Wand angebracht, von der die Würfel auf den Tisch zurückfallen. Die geheimnisvollen Inschriften auf dem Tisch deuten die Setzmöglichkeiten an. 'Pass-Line' bedeutet das Wetten auf die oben geschilderten Ausfälle; 'Big 6 or big 8' bedeutet das Wetten darauf, daß 6 oder 8 vor Erscheinen der 7 auftritt u.s.w.

Das Programm auf der folgenden Seite simuliert das Wetten auf 'Pass-Line':

```
100 : PRINT "□
110 : PRINT "                  CRAPS
111 : PRINT "                  -----
112 :
120 REM BEKANNTES AMERIKANISCHES WUERFELSPIEL
121 :
200 REM === ANFANGSKAPITAL UND EINSATZ ================================
201 :
210 : LET K = 10  : REM KAPITAL DES SPIELERS
215 : LET M = 100 : REM MAXIMALEINSATZ
220 :
230 : PRINT "■■SIE HABEN ";K;" DM SPIELKAPITAL. ";
235 :
240 : INPUT "WIEVIEL SETZEN SIE "; E
245 : IF E <= 0 OR E > K THEN 240
250 : IF E > M THEN PRINT "EINSATZ ZU HOCH!" : GOTO 240
290 :
300 REM === ERSTER WURF ================================================
301 :
310 : LET W = INT(6*RND(TI)) + INT(6*RND(TI)) + 2
315 :
320 : FOR J = 1 TO 500 : NEXT J : REM WARTESCHLEIFE
325 :
330 : PRINT "■SIE HABEN IM ERSTEN WURF ";W;" AUGEN ERZIELT.
335 :
340 : IF W = 7 OR W = 11           THEN 510 : REM GEWONNEN
350 : IF W = 2 OR W = 3 OR W = 12 THEN 560 : REM VERLOREN
390 :
400 REM === WEITERE WUERFE =============================================
401 :
405 : LET I = 1 : REM ZAEHLER DER WÜRFE
410 : LET P = W : REM DER ZU MERKENDE 'PUNKT'
415 :
420 : PRINT "■ZWEITE WURFSERIE (IHR PUNKT IST ";P;"):
425 :
430 : LET I = I+1
435 :
440 :   LET W = INT(6*RND(TI)) + INT(6*RND(TI)) + 2
445 :
450 :   FOR J = 1 TO 300 : NEXT J : REM WARTESCHLEIFE
455 :
460 :   PRINT I;". WURF: ";W;" AUGEN
465 :
470 :   IF W = P THEN PRINT "DER PUNKT WURDE GETROFFEN. ";: GOTO 510
475 :   IF W = 7 THEN 560   : REM VERLOREN
480 :
485 : GOTO 430              : REM NOCH EIN WURF
490 :
```

```
500 REM === AUSWERTUNG ==================================================
501 :
510 : PRINT "DAMIT IST DIE PARTIE GEWONNEN!
520 : LET K = K+E          : REM NEUES KAPITAL
530 : PRINT "■IHR KAPITAL BETRAEGT JETZT ";K;" DM."
540 : GOTO 600
550 :
560 : PRINT "■■■DAMIT IST DIE PARTIE LEIDER VERLOREN.
570 : LET K = K-E          : REM NEUES KAPITAL
580 : IF K <= 0 THEN 650 : REM SPIELER IST RUINIERT
590 :
600 REM === AUFGABE ODER NEUE PARTIE ====================================
601 :
610 : PRINT "■■■■WOLLEN SIE EINE NEUE PARTIE WAGEN?
620 : GET A$ : IF A$ = "" THEN 620
630 : IF A$ = "J" THEN PRINT "■" : GOTO 230 : REM NEUE PARTIE
640 :
650 : PRINT "■■■■ES IST BESSER FUER SIE, AUFZUHOEREN.
660 :
690 : END
```

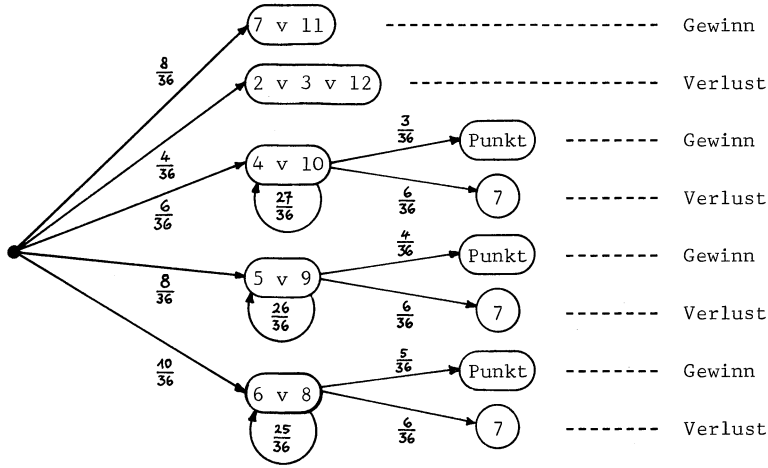
Und so läuft das Programm:

```
Sie haben 10 DM Spielkapital. Wieviel setzen Sie? 2

Sie haben im ersten Wurf 11 Augen erzielt.
Damit ist die Partie gewonnen!

Wollen Sie eine neue Partie wagen? j

Sie haben 12 DM Spielkapital. Wieviel setzen Sie? 5

Sie haben im ersten Wurf 4  Augen erzielt.
Zweite Wurfserie (Ihr Punkt ist 4):
2. Wurf: 9
3. Wurf: 6
4. Wurf: 4
Der Punkt wurde getroffen.
Damit ist die Partie gewonnen!

Wollen Sie eine neue Partie wagen? j

Sie haben 17 DM Spielkapital. Wieviel setzen Sie? 17

Sie haben im ersten Wurf 2 Augen erzielt.
Damit ist die Partie leider verloren.

Es ist besser für Sie, aufzuhören.
```

Wir wollen nun die Gewinnwahrscheinlichkeit für das Wetten auf die 'Pass-Line' berechnen. Für den Doppelwurf mit zwei Würfeln gilt die Verteilung

Augensumme	2	3	4	5	6	7	8	9	10	11	12
Wahrscheinlichkeit	$\frac{1}{36}$	$\frac{2}{36}$	$\frac{3}{36}$	$\frac{4}{36}$	$\frac{5}{36}$	$\frac{6}{36}$	$\frac{5}{36}$	$\frac{4}{36}$	$\frac{3}{36}$	$\frac{2}{36}$	$\frac{1}{36}$

Damit läßt sich folgendes Baumdiagramm aufstellen:



Die Wahrscheinlichkeit, nach dem ersten Wurf zu gewinnen, beträgt demnach $\frac{6}{36} + \frac{2}{36}$. Trat 4 oder 10 auf, so beträgt die Wahrscheinlichkeit, den Punkt zu erzielen, $\frac{1}{3}$; trat 5 oder 9 auf, so beträgt sie $\frac{2}{5}$; trat endlich 6 oder 8 auf, so ist sie gleich $\frac{5}{11}$.
Also ist die Gewinnwahrscheinlichkeit insgesamt

$$\frac{8}{36} + \frac{6}{36}\cdot\frac{1}{3} + \frac{8}{36}\cdot\frac{2}{5} + \frac{10}{36}\cdot\frac{5}{11} = \frac{244}{495} = 0{,}4\overline{92} \approx 49{,}3\% \ ;$$

das Spiel ist nahezu fair. Setzt man bei jeder Partie 1 DM, so gewinnt man im Mittel

$$1\cdot\frac{244}{495} + (-1)\cdot\frac{251}{495} = -\frac{7}{495} = -\ 0{,}0\overline{14}$$

DM pro Partie, d.h. man verliert im Mittel pro Partie etwa 1,4 Pf.
An dem Baum auf der vorigen Seite liest man als mittlere Spieldauer (= mittlere Anzahl der Würfe pro Partie) die Zahl $\frac{557}{165} \approx 3{,}4$ ab.

Nach den Würfeln wollen wir uns nunmehr den M ü n z e n als fast ebenso beliebten Hilfsmitteln für Glücksspiele zuwenden.

Beispiel 3 : Ein Münzwurfspiel

Wird eine Münze dreimal geworfen, gibt es 8 mögliche Ausfälle, nämlich KKK, KKZ, KZK, KZZ, ZKK, ZKZ, ZZK, ZZZ. ('K' bedeutet 'Kopf' und 'Z' bedeutet 'Zahl'). Spieler A wählt einen dieser Ausfälle, Spieler B einen der 7 übrigen. Nun wird die Münze solange geworfen, bis einer dieser Ausfälle auftritt; der Spieler, dessen 'Muster' als erstes auftritt, hat gewonnen.
Durch Computersimulation soll die Frage beantwortet werden, ob die Chancen für A und B gleich stehen - oder ob B , unter Beachtung der Wahl von A , seine Gewinnchancen verbessern kann.

Wir entwickeln ein Programm, das folgenden Dialog ermöglicht:

```
Wieviel Wurfserien sollen es sein? 3
Welches Muster wählen Sie?   KKZ
Ich wähle dagegen das Muster ZKK

1. Serie: ZKK    MEIN MUSTER!

2. Serie: KKKKZ  IHR  MUSTER!

3. Serie: ZZKK   MEIN MUSTER!


Ergebnis nach 3 Wurfserien:
1-mal kam Ihr Muster zuerst,
2-mal kam mein Muster zuerst.
```

```
100 : PRINT "■
110 : PRINT "          EIN MUENZWURFSPIEL
111 : PRINT "          ------------------
112 :
120 REM SOLL ZUM STUDIUM VON MARKOFF-KETTEN ANREGEN
121 :
200 REM === ANLEITUNG ================================================
201 :
210 : PRINT "■■ANGENOMMEN, EINE MUENZE WIRD DREIMAL GEWORFEN.
215 : PRINT "UNTER DEN ACHT MOEGLICHEN AUSFAELLEN
220 : PRINT "KKK, KKZ, KZK, KZZ, ZKK, ZKZ, ZZK, ZZZ
225 : PRINT "KOENNEN SIE EINEN WAEHLEN.
230 : PRINT "ICH WAEHLE EBENFALLS EINEN AUS.
235 : PRINT "NUN WIRD WIEDERHOLT GEWORFEN (WURFSERIE).
240 : PRINT "DER, DESSEN MUSTER ZUERST ERSCHEINT, HAT GEWONNEN.
245 : PRINT
290 :
300 REM === INITIALISIERUNG ==========================================
301 :
310 : FOR I = 1 TO 8 : READ B$(I), C$(I) : NEXT I
315 :
320 : INPUT "■■WIEVIEL WURFSERIEN SOLLEN ES SEIN "; N
325 :
330 : PRINT : INPUT "WELCHES MUSTER WAEHLEN SIE  "; B$
335 :
340 : FOR I = 1 TO 8
345 :   IF B$ = B$(I) THEN LET C$ = C$(I) : LET I = 8 : GOTO 370
350 : NEXT I
355 :
360 : PRINT "FALSCHE EINGABE!": GOTO 330
365 :
370 : PRINT "ICH WAEHLE DAGEGEN DAS MUSTER "; C$
375 :
380 : PRINT "■■        ■TASTE DRUECKEN!
381 : GET T$ : IF T$ = "" THEN 381
385 :
390 DATA KKK, ZKK, ZZZ, KZZ, KKZ, ZKK, ZZK, KZZ
391 DATA KZK, KKZ, ZKZ, ZZK, KZZ, KKZ, ZKK, ZZK
399 :
400 REM === PARTIE ====================================================
401 :
410 : FOR J = 1 TO N : REM SCHLEIFENANFANG ----------------------------
415 :
420 :     PRINT "■";J;". SERIE: ";
425 :
430 :     LET W$ = ""
435 :
440 :   REM --- MUENZWURF
441 :
450 :     IF RND(TI) < 0.5 THEN LET A$ = "K" : GOTO 460
455 :                      LET A$ = "Z"
460 :     PRINT A$;
470 :     LET W$ = W$ + A$
490 :
```

```
500 :   REM --- VERGLEICH
501 :
505 :     LET M$ = RIGHT$(W$,3) : REM LETZTE DREI WUERFE
510 :     IF M$ = B$ THEN 610   : REM MUSTER DES SPIELERS ERSCHEINT
520 :     IF M$ = C$ THEN 650   : REM MUSTER DES COMPUTERS KOMMT
530 :     GOTO 450              : REM NEUER VERSUCH
590 :
600 :   REM --- GEWINNENTSCHEID
601 :
610 :     LET B = B+1
620 :     PRINT "   ■IHR MUSTER!
625 :     PRINT
630 :     GOTO 700
640 :
650 :     LET C = C+1
660 :     PRINT "    MEIN MUSTER.
665 :     PRINT
690 :
700 :   REM --- WARTESCHLEIFE
701 :
710 :     PRINT "       ■TASTE DRUECKEN!
715 :     GET T$ : IF T$ = "" THEN 715
720 :
730 : NEXT J : REM SCHLEIFENENDE ------------------------------------
790 :
800 REM === AUSWERTUNG ================================================
801 :
810 : PRINT "■■■ ERGEBNIS NACH ";N;" WURFSERIEN:
815 : PRINT
820 : PRINT B;"-MAL KAM IHR  MUSTER "; B$;" ZUERST.
830 : PRINT C;"-MAL KAM MEIN MUSTER "; C$;" ZUERST.
890 :
```

E r l ä u t e r u n g e n zum Programm:

Zunächst werden 8 Paare B$(I), C$(I) aus den DATA-Zeilen 390, 391 eingelesen; sie drücken die Zuordnung des vom Computer gewählten Musters zu dem des Spielers aus. Nach Festlegung der Anzahl der Wurfserien wird der Spieler nach seinem Muster B$ gefragt. Anschließend wählt der Computer sein Muster C$ (Zeilen 340 - 350).
Während des eigentlichen Spiels wird zuerst der Münzwurf simuliert (450) und das ermittelte Zeichen ('K' oder 'Z') an die schon bestehende Zeichenkette W$ angefügt; galt z.B. W$ = KKZ und fiel K, so heißt es nun W$ = KKZK. Durch

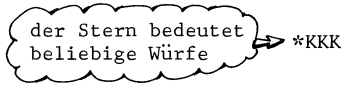
$$M\$ = \text{RIGHT}\$(W\$,3)$$

werden aus der Zeichenkette W$ die letzten drei Zeichen herausgegriffen: sie kennzeichnen die letzten drei Würfe. Gilt M$ = B$, so ist das Muster des Spielers aufgetreten und die Serie ist beendet, gilt M$ = C$, so erschien das Muster des Computers und dieser hat gewonnen. Andernfalls wird die Münze erneut geworfen.

*

Wenn Sie mit diesem Programm spielen, werden Sie bemerken, daß der Computer häufig gewinnt. Wie macht er das - d.h. nach welchem Plan wählt er sein Muster?

Angenommen, Sie wählen das Muster KKK. Die Wahrscheinlichkeit, daß es nach drei Würfen erscheint, beträgt $\frac{1}{8}$ (wir nehmen an, die Münze sei nicht gefälscht). Wenn nun Ihr Gegenspieler sein Muster geschickt wählt, kann er verhindern, daß Ihres noch einmal erscheint: er muß nur dafür sorgen, daß beim vierten (oder einem späteren) Wurf der Fall

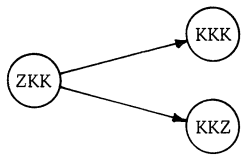


n i c h t eintritt. Dies erreicht er, wenn er für * ein Z setzt, d.h. wenn er zu Beginn das Muster ZKK wählt. Genau dies tut auch der Computer (siehe Zeile 390 DATA KKK, ZKK, ...)!

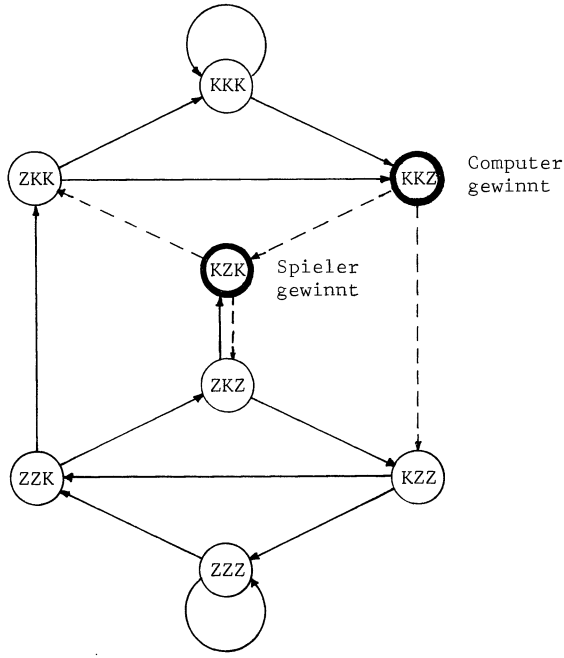
Ihre Gewinnwahrscheinlichkeit beträgt also $\frac{1}{8}$, die des Gegenspielers $\frac{7}{8}$, d.h. Sie werden auf lange Sicht 87,5% aller Wurfserien verlieren.-

Was passiert, wenn Sie z.B. das Muster KZK wählen? Der Gegner wird so wählen, daß der Anfang Ihres Musters (das ist KZ) bei ihm am Ende steht: also nimmt er KKZ oder ZKZ als eigenes Muster. Der Computer entscheidet sich für ersteres (siehe DATA-Zeile 391). Mit welcher Wahrscheinlichkeit gewinnt er nun?

Um diese Frage zu beantworten, zeichnen wir einen sogenannten Übergangsgraphen (nächste Seite). Er gibt die möglichen Übergänge von einem Muster zum nächsten an, wenn die Münze geworfen wurde. So bedeutet z.B.



daß nach Erscheinen des Musters ZKK jeweils mit Wahrscheinlichkeit $\frac{1}{2}$ entweder das Muster KKK oder das Muster KKZ erscheint: aus *ZKK kann entweder *ZKKK oder *ZKKZ werden. (Die drei unterstrichenen Buchstaben bilden das Muster nach dem Wurf.)



Die gestrichelten Pfeile deuten n i c h t mögliche Übergänge an: KKZ und KZK sind ja Endzustände. (Damit soll die Symmetrie der Figur hervorgehoben werden, bevor die Spieler ihre Muster ausgewählt haben.) Alle Wahrscheinlichkeiten sind $\frac{1}{2}$.

An dem Übergangsgraphen liest man unmittelbar ab: Beginnt das Spiel mit den Mustern KKZ, ZKK oder KKK, so wird der Computer gewinnen. Erscheint zuerst das Muster ZZK, ZZZ oder KZZ, so muß, bevor einer der Spieler gewinnt, das Muster ZZK aufgetreten sein. Also ist die Wahrscheinlichkeit dafür, daß der Computer gewinnt, wenn das Spiel mit ZZZ oder mit KZZ beginnt, gleich der, wenn es mit ZZK beginnt. Benutzen wir die Abkürzung

P(C|KKK) für "Wahrscheinlichkeit, daß C gewinnt, wenn zuerst Muster KKK erschien",

so können wir unsere bisherigen Erkenntnisse wie folgt ausdrücken:

(1) $P(C|KKZ) = P(C|ZKK) = P(C|KKK) = 1$

(2) $P(C|KZK) = 0$

(3) $P(C|ZZK) = P(C|ZZZ) = P(C|KZZ)$.

Es bleiben nur noch die Wahrscheinlichkeiten $P(C|ZKZ)$ und $P(C|ZZK)$ zu bestimmen. Ist Muster ZZK erschienen, so kommt nach dem nächsten Wurf entweder ZKK oder ZKZ, und zwar mit jeweils gleicher Wahrscheinlichkeit; also gilt

(4) $P(C|ZZK) = \frac{1}{2}P(C|ZKZ) + \frac{1}{2}P(C|ZKK)$,

und entsprechend

(5) $P(C|ZKZ) = \frac{1}{2}P(C|KZZ) + \frac{1}{2}P(C|KZK)$.

Hieraus folgt nach Berücksichtigung von (1), (2) und (3):

$$P(C|ZZK) = \frac{2}{3} \quad \text{und} \quad P(C|ZKZ) = \frac{1}{3} .$$

Die Gewinnwahrscheinlichkeit für C, den Computer, ist demnach

$$\frac{1}{8}(1+1+1+0+\frac{2}{3}+\frac{2}{3}+\frac{2}{3}+\frac{1}{3}) = \frac{2}{3} .$$

Das heißt: wählen Sie Muster KZK, und der Computer wählt KKZ, so wird er auf lange Sicht in zwei von drei Fällen (Wurfserien) gewinnen.-
Die übrigen Wahlen und Wahrscheinlichkeiten lassen sich ganz entsprechend begründen bzw. herleiten. In den Aufgaben erhalten Sie weitere Anregungen für die Beschäftigung mit diesem interessanten Spiel.

Nach Würfeln und Münzen greifen wir nun zu den Karten. Interessant ist das amerikanische Casino-Spiel 'Blackjack' (französisch: vingt-et-un, d.h. 21), weil hier der Spieler zuweilen eine positive Gewinnerwartung hat. Es kostet jedoch viel Zeit, die Berechnungen für die Strategie zu verstehen, und man benötigt erhebliche Konzentrationsfähigkeit, um sie erfolgreich anwenden zu können; daher sind die gewöhnlichen Berufstätigkeiten einträglicher. Die Spielregeln von Blackjack sind kompliziert, und sie variieren auch von Ort zu Ort. Wir wollen uns mit einer einfachen deutschen Version begnügen.

Beispiel 4 : Siebzehn und vier

Bei diesem Spiel mit 32 Karten sind nicht die Farben, sondern nur ihre Werte von Bedeutung; diese sind: Bube = 2; Dame = 3; König = 4; 7,8,9, 10 wie die Zahlen angeben; As = 11. Nachdem der Bankhalter sein Spielkapital als Bank aufgelegt hat, werden die Karten gemischt und abgehoben. Jeder Spieler entscheidet über seinen Einsatz und erhält dann zwei Karten (verdeckt), der Bankhalter eine (offen). Nunmehr kann der Spieler weitere Karten anfordern. Er bemüht sich dabei, möglichst genau auf 21 Punkte (oder nahe daran heran) zu kommen; keinesfalls darf er die 21 überschreiten: in diesem Fall scheidet er aus und verliert seinen Einsatz. Haben alle Spieler gezogen, nimmt sich der Bankhalter ebenfalls so viele Karten vom Stapel, wie er für nötig erachtet. Kommt er über 21, muß er allen, die noch im Spiel sind, den doppelten Einsatz auszahlen. Andernfalls gewinnt die höhere Punktzahl (≤ 21); bei gleicher Punktzahl gewinnt der Bankhalter.

Wir wollen, um das wesentliche herauszuarbeiten, zunächst ein recht einfaches Programm entwerfen, und nehmen daher nur einen Spieler sowie den Bankhalter, der vom Computer dargestellt wird. Auch werden keine Einsätze getätigt; das Ergebnis ist einfach 'gewonnen' oder 'verloren'.
Ein D i a l o g sollte sich so abspielen:

```
                Siebzehn und vier

Soll ich Ihnen die Spielregeln nennen? n

                DAS SPIEL BEGINNT!

Ich habe Bube bekommen.
Sie haben Dame und 9 bekommen.

Möchten Sie eine Karte ziehen? j
Sie haben 8 gezogen.

Möchten Sie eine Karte ziehen? n

Der Gesamtwert Ihrer Karten beträgt 20.

Ich habe Dame gezogen.
Ich habe As gezogen.
Ich habe Dame gezogen.

Der Gesamtwert meiner Karten beträgt 19.

Sie haben gewonnen!


Noch eine Partie? j

                DAS SPIEL BEGINNT!

Ich habe 8 bekommen.
Sie haben Dame und 10 bekommen.

Möchten Sie eine Karte ziehen? j
Sie haben König gezogen.

Möchten Sie eine Karte ziehen? j
Sie haben 8 gezogen.

Der Gesamtwert Ihrer Karten
beträgt 25. Sie haben verloren.

Noch eine Partie? n
```

17 + ?

... + 8 = 25
Pech ...

```
1000 : PRINT "■
1010 : PRINT "            SIEBZEHN UND VIER
1011 : PRINT "            ----------------
1012 :
1020 REM GLUECKSSPIEL MIT KARTEN, DER COMPUTER HAELT DIE BANK
1098 :
1099 :
2000 REM *** HAUPTPROGRAMM *****************************************
2001 :
2300 : GOSUB 3000 : REM ANLEITUNG
2301 :
2400 : GOSUB 4000 : REM INITIALISIERUNG
2401 :
2500 : GOSUB 5000 : REM PARTIE
2501 :
2700 : GOSUB 7000 : REM VERABSCHIEDUNG ODER WIEDERHOLUNG
2701 :
2800 : END
2801 :
2900 REM *** ENDE HAUPTPROGRAMM *************************************
2998 :
2999 :
3000 REM ### UNTERPROGRAMM 'ANLEITUNG' ###############################
3001 :
3010 : PRINT : PRINT
3020 : PRINT "SOLL ICH IHNEN DIE SPIELREGELN NENNEN? (J/N)
3030 : GET A$ : IF A$ = "" THEN 3030
3040 : IF A$ <> "J" THEN RETURN
3050 :
3070 : PRINT "SEHEN SIE BITTE IM BUCH AUF SEITE 136 NACH!
3080 :
3090 : FOR I = 1 TO 500 : NEXT I
3100 :
3900 : RETURN : REM ENDE ANLEITUNG ###################################
3999 :
4000 REM ### UNTERPROGRAMM 'INITIALISIERUNG' ##########################
4001 :
4100 : REM === BLATT EINRICHTEN
4101 :
4110 :   DIM B(32) : REM TABELLE DES 32-ER BLATTS
4120 :
4130 :   LET T = 0
4140 :   FOR I = 1 TO 4
4150 :     FOR J = 1 TO 8
4160 :       LET T = T+1
4170 :       LET B(T) = J
4180 :     NEXT J
4190 :   NEXT I
4199 :
4200 : REM === ANFANGSWERTE
4201 :
4210 :   LET N = 32 : REM ANZAHL DER KARTEN AUF DEM STAPEL
4220 :
4230 :   LET WB = 0 : REM WERT DER KARTEN DER BANK
4240 :   LET WS = 0 : REM WERT DER KARTEN DES SPIELERS
4290 :
4900 : RETURN : REM ENDE INITIALISIERUNG ############################
4999 :
```

```
5000 REM ### UNTERPROGRAMM 'PARTIE' ###################################
5001 :
5010 : PRINT "■■            ■DAS SPIEL BEGINNT
5090 :
5100 : REM === KARTE AN BANK AUSTEILEN
5101 :
5120 :   GOSUB 6000 : REM KARTE ZIEHEN
5130 :   PRINT "■■■
5140 :   PRINT "ICH HABE ";W$;" BEKOMMEN.
5150 :   LET WB = WB + W
5190 :
5200 : REM === KARTEN AN SPIELER AUSTEILEN
5201 :
5210 :   GOSUB 6000 : REM KARTE ZIEHEN
5220 :   PRINT "■SIE HABEN ";W$;" UND ";
5230 :   LET WS = WS + W
5240 :   GOSUB 6000 : REM KARTE ZIEHEN
5250 :   LET WS = WS + W
5260 :   PRINT W$;" BEKOMMEN.
5290 :
5300 : REM === SPIELER ZIEHT KARTEN
5301 :
5320 :   PRINT
5330 :   PRINT "MOECHTEN SIE EINE KARTE ZIEHEN? (J/N)
5335 :   GET A$ : IF A$ = "" THEN 5335
5340 :   IF A$ <> "J" THEN 5400
5350 :   GOSUB 6000 : REM KARTE ZIEHEN
5360 :   PRINT "SIE HABEN ";W$;" GEZOGEN.
5370 :   LET WS = WS + W
5380 :   GOTO 5320 : REM NOCH EINE KARTE?
5390 :
5400 : REM === AUSWERTUNG DER SPIELERKARTEN
5401 :
5420 :   PRINT
5430 :   PRINT "DER GESAMTWERT IHRER KARTEN BETRAEGT "; WS
5440 :   IF WS > 21 THEN 5800: REM SPIELER VERLIERT
5490 :
5500 : REM === BANK ZIEHT KARTEN
5501 :
5510 :   PRINT "■■
5520 :   GOSUB 6000 : REM KARTE ZIEHEN
5530 :   PRINT "ICH HABE ";W$;" GEZOGEN.
5540 :   LET WB = WB + W
5550 :   IF WB < 17 THEN 5520 : REM WEITERE KARTE
5590 :
5600 : REM === AUSWERTUNG DER KARTEN DER BANK
5601 :
5620 :   PRINT "■DER GESAMTWERT MEINER KARTEN BETRAEGT: "; WB
5690 :
5700 : REM === SPIELENTSCHEID
5701 :
5720 :   IF WB > 21 THEN 5850: REM SPIELER GEWINNT
5730 :
5740 :   IF WB = 21 THEN 5800: REM BANK GEWINNT
5750 :
5780 :   IF WB < WS THEN 5850: REM SPIELER GEWINNT
5790 :
5800 :   PRINT "■■ICH HABE GEWONNEN.": GOTO 5900
5850 :   PRINT "■■SIE HABEN GEWONNEN!
5890 :
5900 :   RETURN : REM ### ENDE PARTIE ###################################
5999 :
```

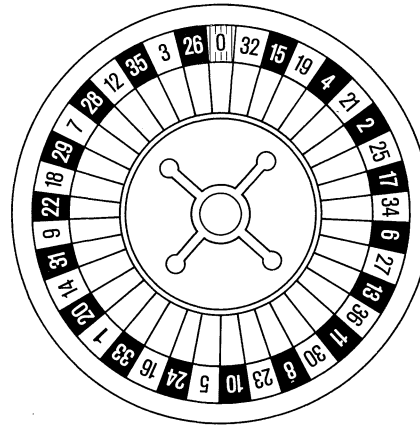
```
6000 : REM +++ UNTERPROGRAMM 'KARTE ZIEHEN' ++++++++++++++++++++++++++
6001 :
6010 : REM --- ZIEHEN
6020 :
6100 :   LET R = INT(N*RND(TI)+1)
6110 :   LET K = B(R)
6120 :   LET B(R) = B(N)
6130 :   LET N = N-1 : REM AUF DEM STAPEL LIEGT EINE KARTE WENIGER
6190 :
6200 : REM --- WERTANGABE
6201 :
6220 :   IF K = 1 THEN LET W$ = "AS"      : LET W = 11
6230 :   IF K = 2 THEN LET W$ = "BUBE"    : LET W = 2
6240 :   IF K = 3 THEN LET W$ = "DAME"    : LET W = 3
6250 :   IF K = 4 THEN LET W$ = "KOENIG" : LET W = 4
6260 :   IF K > 4 THEN LET W$ = STR$(K+2): LET W = K+2
6270 :
6290 :   RETURN : REM ENDE KARTE ZIEHEN ++++++++++++++++++++++++++++++
6299 :
7000 REM ### UNTERPROGRAMM 'VERABSCHIEDUNG ODER WIEDERHOLUNG' ##########
7001 :
7100 : PRINT
7200 : PRINT "NOCH EINE PARTIE? (J/N)
7210 : GET A$ : IF A$ = "" THEN 7210
7220 : IF A$ = "J" THEN RUN
7260 :
7290 : RETURN : REM ### ENDE VERABSCHIEDUNG ODER WIEDERHOLUNG ##########
7299 :
```

Das berühmteste und schillerndste aller Glücksspiele ist zweifellos R o u l e t t e . Es gilt geradezu als Inbegriff des Glücksspiels; bei keinem anderen Spiel folgen Gewinn und Verlust so rasch aufeinander, und kein Spiel ist so eindringlich literarisch verewigt worden (man denke an Dostojewski's "Spieler"). Es wird zuweilen behauptet, das Spiel sei von Blaise Pascal erfunden worden; dies ist jedoch Unsinn. Pascal hat zwar mehrere Abhandlungen mit dem Titel 'La Roulette' geschrieben, meint damit aber eine Kurve, die wir heute als 'Zykloide' bezeichnen (sie entsteht beim Abrollen eines Rades, daher ihr Name). Jedenfalls wurde das Spiel erst im Jahr 1765 von einem Gabriel de Sartine in Paris eingeführt; Vorformen gab es natürlich schon früher.

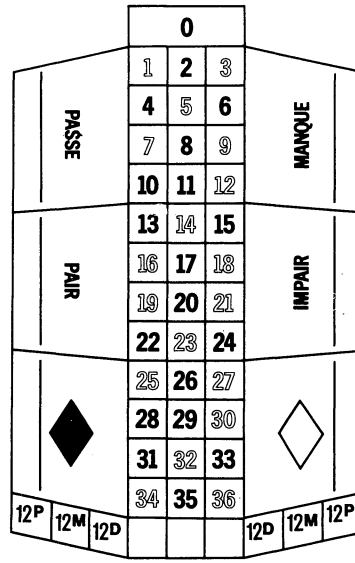
Jansenistisches Tourniquet-Spiel (eine Art Roulett) aus dem 18. Jahrhundert.

Ehe man aus Neugier, aus Leidenschaft oder aus Gewinnsucht in einem Kasino va banque spielt, sollte man sich mit dem Spiel, d.h. seinen Regeln und seinen Chancen gründlich vertraut machen. Man kann sich ein Spielzeug-Roulette kaufen und damit üben - ein besserer Weg ist die Simulation des Spiels auf dem Computer (dieser ist eine universelle Maschine, die jedes Glücksspiel nachahmen kann).

Beispiel 5: Roulette



In der Roulette-Maschine, dem sogenannten Kessel, ist ein von Hand drehbarer Zylinder auf Kugeln gelagert. Während der bewegliche Teil aus Metall besteht, ist der feststehende Kessel selbst aus einem edlen Holz gefertigt. Der Griff zur Bewegung des Zylinders ist in Form eines Kreuzes (Drehkreuz) gestaltet, um den Croupiers die Handhabung bei jeder Stellung leichtzumachen.
Rund um den Zylinder sind kreisförmig die Zahlen 1 bis 36 und die 0 (Zéro genannt) in sinnfälliger Weise angeordnet. Zu jeder Zahl gehört ein Fach, das mit Filz unterlegt ist.
Eine Kugel, die vom Croupier geworfen wird und unter dem Rand des Kessels kreist, bis sie schließlich in ein Fach fällt, entscheidet über die Gewinnzahlen.
An den Roulette-Kessel schließt sich der Roulette-Tisch an. Er ist mit grünem Filz belegt, auf dem die Setzmöglichkeiten aufgezeichnet sind. Diese sind:



1. Einzelne Zahl (plein); Gewinn: das 35fache des Einsatzes.
2. Zwei Zahlen (cheval); Gewinn: das 14fache des Einsatzes.
3. Drei Zahlen (transversale plein); Gewinn: das 11fache des Einsatzes.
4. Vier Zahlen (carré); Gewinn: das 8fache des Einsatzes.
5. Sechs Zahlen (transversale simple); Gewinn: das 5fache des Einsatzes.
6. Dutzende; Gewinn: das Doppelte des Einsatzes.

7. Kolonnen; Gewinn: das Doppelte des Einsatzes.

8. Einfache Chancen (Manque, Passe, Pair, Impair, Rouge, Noir); Gewinn: das Einfache des Einsatzes.

Ferner gilt (natürlich): der Einsatz gehört dem Gewinner.

Wird Zéro getroffen, so sind alle einfachen Chancen g e s p e r r t, die Jetons werden vom Croupier auf die Sperrlinie geschoben. Der Spieler kann jetzt verlangen, daß sein Einsatz geteilt wird. In diesem Fall verfällt die Hälfte des Einsatzes der Bank, die andere Hälfte bekommt der Spieler zurück. Verzichtet er auf die Teilung, so wird sein Einsatz dann wieder freigegeben (und vom Croupier von der Sperrlinie abgezogen), wenn als nächstes die Chance, auf die er gesetzt hatte, eintritt. Fällt erneut Zero, so sind alle (noch) gesperrten Einsätze der Bank verfallen.

Unser Programm soll zunächst nur das Spiel eines einzigen Spielers gegen die Bank simulieren; die komplizierte Zéro-Regelung bauen wir noch nicht ein. Auch lassen wir die Setzmöglichkeiten 2,3,4,5 weg.

D i a l o g :

```
                    Roulette

Folgende Einsätze können Sie tätigen:

Zahl (plein) ..................  1 bis 36
Erstes Dutzend ..................... 37
Zweites Dutzend .................... 38
Drittes Dutzend .................... 39
.....................................
Rot (Rouge) ........................ 47
Schwarz (Noir)  .................... 48

Wählen Sie! Art Ihres Einsatzes? 15
Wieviel setzen Sie? 20
Noch ein Einsatz? j

Wählen Sie! Art Ihres Einsatzes? 47
Wieviel setzen Sie? 30
Noch ein Einsatz? n

                  RIEN NE VA PLUS

Die Kugel blieb auf  14  liegen.
Sie haben beim 1. Einsatz (15 plein) 20 DM verloren.
Sie haben beim 2. Einsatz (Rouge)    30 DM gewonnen.
```

```
1000 : PRINT "■
1010 : PRINT "              ROULETTE
1011 : PRINT "              --------
1012 :
1020 REM ES WIRD DAS BEKANNTE CASINO-SPIEL FUER 1 SPIELER SIMULIERT
1021 :
2000 REM === INITIALISIERUNG ========================================
2001 :
2100 : REM   VARIABLEN:
2120 : REM     N ............... ANZAHL DER EINSAETZE
2130 : REM     I ............... NUMMER DES EINSATZES
2140 : REM     T, T(I) ......... TYP DES EINSATZES NR.I
2150 : REM     E, E(I) ......... EINSATZ NR.I
2160 : REM     A ............... AUSZAHLUNG BZW. VERLUST
2190 :
2200 : LET M = 5                 : REM MINDESEINSATZ
2210 : LET H = 1000              : REM HOECHSTEINSATZ
2220 :
2230 : LET KB = 10000            : REM KAPITAL DER BANK
2240 :
2250 : PRINT "■■WIE HOCH IST IHR SPIELKAPITAL? ";
2260 : INPUT KS
2270 : IF KS < M  THEN PRINT "DAS REICHT NICHT!" : GOTO 2250
2999 :
3000 REM === SETZEN ==================================================
3001 :
3002 : LET I = 0    : REM NUMMER DES EINSATZES
3003 : LET KL=KS    : REM LAUFENDES SPIELERKAPITAL
3004 :
3005 : LET I = I+1
3006 :
3100 :   PRINT
3110 :   PRINT "FOLGENDE EINSAETZE KOENNEN SIE TAETIGEN:
3120 :   PRINT
3130 :   PRINT "ZAHL (PLEIN) ..................... 1 BIS 36
3140 :   PRINT "ERSTES  DUTZEND ..................... 37
3150 :   PRINT "ZWEITES DUTZEND ..................... 38
3160 :   PRINT "DRITTES DUTZEND ..................... 39
3170 :   PRINT "ERSTE  KOLONNE ..................... 40
3180 :   PRINT "ZWEITE KOLONNE ..................... 41
3190 :   PRINT "DRITTE KOLONNE ..................... 42
3200 :   PRINT "1-18 (MANQUE) ..................... 43
3210 :   PRINT "19-36 (PASSE) ..................... 44
3220 :   PRINT "GERADE (PAIR) ..................... 45
3230 :   PRINT "UNGERADE (IMPAIR) ................. 46
3240 :   PRINT "ROT (ROUGE) ....................... 47
3250 :   PRINT "SCHWARZ (NOIR) .................... 48
3290 :
3300 :   PRINT "■■WAEHLEN SIE! ";
3400 :   INPUT "ART IHRES EINSATZES "; T
3410 :   IF T < 1 OR T > 49 OR T <> INT(T) THEN 3300
3420 :   LET T(I) = T
3440 :
3500 :   INPUT "WIEVIEL SETZEN SIE "; E
3510 :   IF E < M  THEN PRINT "MINDESTEINSATZ IST "; M : GOTO 3500
3520 :   IF E > H  THEN PRINT "HOECHSTEINSATZ IST "; H : GOTO 3500
3530 :   IF E > KL THEN PRINT "SO VIEL HABEN SIE NICHT!":GOTO 3500
3540 :   LET E(I) = E
3550 :
```

```
3600 :   PRINT "■■NOCH EIN EINSATZ? (J/N)
3610 :   GET A$ : IF A$ = "" THEN 3610
3620 :
3630 : IF A$ = "J" THEN KL = KL-E : GOTO 3005 : REM WEITERER EINSATZ
3690 :
4000 REM === DAS GLUECKSRAD DREHT SICH ================================
4001 :
4100 : PRINT "■■            ■RIEN NE VAS PLUS
4110 : PRINT "        DAS GLUECKSRAD DREHT SICH
4120 : FOR J = 1 TO 1000 : NEXT J : REM WARTESCHLEIFE
4190 :
4200 : LET K = INT(37*RND(TI)) : REM ZUFALLSZAHL ZWISCHEN 0 UND 36
4210 :
4220 : PRINT "■■DIE KUGEL BLIEB AUF ■";K;"■_IEGEN.
4900 :
5000 REM === ERMITTLUNG VON GEWINN UND VERLUST ========================
5001 :
5010 : LET N = I : REM ANZAHL DER GETAETIGTEN EINSAETZE
5020 :
5080 : PRINT "■■
5090 :
5100 : FOR I = 1 TO N
5104 :
5105 :   PRINT "SIE HABEN BEIM ";I;". EINSATZ ";
5106 :
5110 :   LET E = E(I)
5115 :
5120 :   IF K = 0 THEN LET A = -E : GOTO 5230
5125 :
5130 :   IF T(I) <= 36 THEN GOSUB 6100 : GOTO 5230
5135 :   ON INT((T(I)-36)/7)+1 GOTO 5140,5150 : REM VORVERTEILER
5140 :   ON T(I)-36 GOSUB 6200, 6300, 6400, 6500, 6600, 6700 : GOTO 5230
5150 :   ON T(I)-42 GOSUB 6800, 6900, 7000, 7100, 7200, 7300 : GOTO 5230
5160 :   PRINT "SPRUNGVERTEILER FUNKTIONIERT NICHT"
5170 :
5230 :   IF A > 0 THEN PRINT  A;" DM GEWONNEN.": GOTO 5250
5240 :                 PRINT -A;" DM VERLOREN.
5250 :   LET KS = KS + A : REM NEUES KAPITAL DES SPIELERS
5260 :   LET KB = KB - A : REM NEUES KAPITAL DER BANK
5270 :
5280 : NEXT I : REM NAECHSTER EINSATZ WIRD AUSGEWERTET
5290 :
5300 : PRINT
5310 : PRINT "IHR SPIELKAPITAL BETRAEGT NUN "; KS;" DM.
5320 :
5330 : IF KS < M  THEN PRINT "SIE GEHEN JETZT BESSER NACH HAUSE.":END
5340 : IF KS > KB THEN PRINT "SIE HABEN DIE BANK GESPRENGT!!!":END
5390 :
5400 : PRINT "■■NOCH EIN SPIEL?
5410 : GET A$ : IF A$ = "" THEN 5410
5420 : IF A$ = "J" THEN 3000
5430 :
5900 : END
5999 :
```

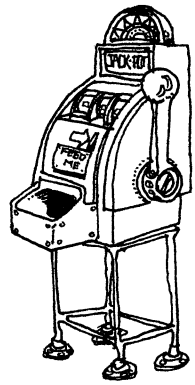
```
6000 REM ### UNTERPROGRAMME ZUR AUSWERTUNG DES GLUECKSRADS ###########
6001 :
6100 :   REM +++ ZAHL +++
6101 :
6110 :     PRINT "(";T(I);" PLEIN) ";
6120 :     IF K = T(I) THEN LET A = 35*E : RETURN
6130 :                      LET A = -E   : RETURN
6190 :
6200 :   REM +++ ERSTES DUTZEND +++
6201 :
6210 :     PRINT "(12P)         ";
6220 :     IF 1 <= K AND K <= 12 THEN LET A = 2*E : RETURN
6230 :                           LET A = -E   : RETURN
6290 :
6300 :   REM +++ ZWEITES DUTZEND +++
6301 :
6310 :     PRINT "(12M)         ";
6320 :     IF 13 <= K AND K <= 24 THEN LET A = 2*E : RETURN
6330 :                            LET A = -E   : RETURN
6390 :
6400 :   REM +++ DRITTES DUTZEND +++
6401 :
6410 :     PRINT "(12D)         ";
6420 :     IF 25 <= K AND K <= 36 THEN LET A = 2*E : RETURN
6430 :                            LET A = -E   : RETURN
6490 :
6500 :   REM +++ ERSTE KOLONNE +++
6501 :
6510 :     PRINT "(1. KOLONNE) ";
6520 :     IF K = 3*INT(K/3) = 1 THEN LET A = 2*E : RETURN
6530 :                           LET A = -E   : RETURN
6590 :
6600 :   REM +++ ZWEITE KOLONNE +++
6601 :
6610 :     PRINT "(2. KOLONNE) ";
6620 :     IF K - 3*INT(K/3) = 2 THEN LET A = 2*E : RETURN
6630 :                           LET A = -E   : RETURN
6690 :
6700 :   REM +++ DRITTE KOLONNE +++
6701 :
6710 :     PRINT "(3. KOLONNE) ";
6720 :     IF K - 3*INT(K/3) = 0 THEN LET A = 2*E : RETURN
6730 :                           LET A = -E   : RETURN
6790 :
6800 :   REM +++ MANQUE +++
6801 :
6810 :     PRINT "(MANQUE)      ";
6820 :     IF 1 <= K AND K <= 18 THEN LET A =  E : RETURN
6830 :                           LET A = -E : RETURN
6890 :
6900 :   REM +++ PASSE +++
6901 :
6910 :     PRINT "(PASSE)       ";
6920 :     IF 19 <= K AND K <= 36 THEN LET A =  E : RETURN
6930 :                            LET A = -E : RETURN
6990 :
```

```
7000 :   REM +++ GERADE +++
7001 :
7010 :     PRINT "(PAIR)        ";
7020 :     IF K - 2*INT(K/2) = 0 THEN LET A =  E : RETURN
7030 :                           LET A = -E : RETURN
7090 :
7100 :   REM +++ UNGERADE +++
7101 :
7110 :     PRINT "(IMPAIR)      ";
7120 :     IF K - 2*INT(K/2) = 1 THEN LET A =  E : RETURN
7130 :                           LET A = -E : RETURN
7190 :
7200 :   REM +++ ROT +++
7201 :
7210 :    PRINT "(ROUGE)       ";
7220 :    GOSUB 7400
7230 :    IF  R    THEN LET A =  E : RETURN
7240 :                  LET A = -E : RETURN
7290 :
7300 :  REM +++ SCHWARZ +++
7301 :
7310 :    PRINT "(NOIR)        ";
7320 :    GOSUB 7400
7330 :    IF NOT R THEN LET A =  E : RETURN
7390 :
7400 :  REM +++ UNTERPROGRAMM 'ROT' +++
7401 :
7410 :    LET R1 = (K=1  OR K=3  OR K=5  OR K=7  OR K=9  OR K=12)
7420 :    LET R2 = (K=14 OR K=16 OR K=18 OR K=19 OR K=21 OR K=23)
7430 :    LET R3 = (K=25 OR K=27 OR K=30 OR K=32 OR K=34 OR K=36)
7440 :    LET R  = R1 OR R2 OR R3
7460 :    RETURN
```

Zum Schluß dieses Kapitels wollen wir als Beitrag Amerikas zu den Glücksspielen einen Spielautomaten betrachten.

Beispiel 6 : Einarmiger Bandit

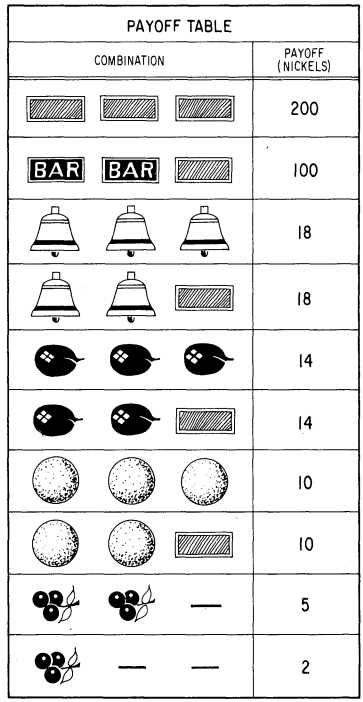
Darunter versteht man einen münzschluckenden Automaten mit einem Hebel, welcher drei Räder in Bewegung setzt, auf denen Symbole wie Glocken, Kirschen, Orangen etc. angebracht sind. Je nach der im Fenster sichtbaren Kombination nach dem Anhalten der Räder werden Groschen ausgeworfen.
Ein Programm soll geschrieben werden, das einen solchen Automaten simuliert.



Dialog:

```
Werfen Sie Ihren Groschen ein!

    DIE RÄDER DREHEN SICH

          Kirsche
          Glocke
          Glocke

Sie gewannen leider nichts.
Noch ein Spiel? (j/n) j

Werfen Sie Ihren Groschen ein!

    DIE RÄDER DREHEN SICH

          Orange
          Orange
          Orange

Sie gewannen 2 DM!
Noch ein Spiel? (j/n) n
```

Folgende Tabelle gibt eine mögliche Auszahlungsfunktion für den Automaten an:



Unser Programm soll aber nur drei Symbole, nämlich Glocke, Orange und Kirsche enthalten. Ein Gewinn tritt auf, wenn drei gleiche Symbole, d.h. wenn Glocke-Glocke-Glocke oder Orange-Orange-Orange oder Kirsche-Kirsche-Kirsche nach dem Stillstehen der Räder im Fenster erscheinen.
Im ersten Fall sollen 3 DM, im zweiten Fall 2 DM und im dritten Fall 1 DM ausgezahlt werden; der Einsatz beträgt 1 Groschen. -
(Im Programm auf der folgenden Seite mußte eine besondere Maßnahme ergriffen werden: beim wiederholten Abziehen des Einsatzes treten Rundungsfehler auf; lassen Sie einmal Zeile 615 weg!)

```
100 : PRINT "■
110 : PRINT "       EINARMIGER BANDIT
111 : PRINT "       ----------------
112 :
120 REM SIMULATION EINES GLUECKSSPIEL-AUTOMATEN
121 :
200 REM === ANLEITUNG ================================================
201 :
210 : PRINT
220 : PRINT "SIE WERFEN EINEN GROSCHEN EIN
222 : PRINT "UND DRUECKEN DEN HEBEL RECHTS HERUNTER.
224 : PRINT "ERSCHEINEN DREI GLOCKEN,  ERHALTEN SIE 3 DM,
226 : PRINT "ERSCHEINEN DREI ORANGEN,  ERHALTEN SIE 2 DM,
228 : PRINT "ERSCHEINEN DREI KIRSCHEN, ERHALTEN SIE 1 DM.
230 :
240 : PRINT
250 : INPUT "WIEVIEL SPIELKAPITAL HABEN SIE "; SK
260 :
270 : PRINT "WERFEN SIE DEN GROSCHEN EIN!
280 :
300 REM === DIE GLUECKSRAEDER DREHEN SICH ============================
301 :
310 : PRINT
320 : PRINT "            ■DIE RAEDER DREHEN SICH
330 : FOR I = 1 TO 1000 : NEXT I
340 :
350 : LET GL = 0 : LET OG = 0 : LET KI = 0
360 :
370 : FOR J = 1 TO 3
380 :
390 :   LET Z = INT(3*RND(TI)+1)
400 :   IF Z = 1 THEN PRINT "GLOCKE": LET GL = GL+1 : GOTO 440
410 :   IF Z = 2 THEN PRINT "ORANGE": LET OG = OG+1 : GOTO 440
420 :                 PRINT "KIRSCHE";LET KI = KI+1
430 :
440 : NEXT J
490 :
500 REM === AUSWERTUNG ===============================================
501 :
510 : PRINT
520 : PRINT "SIE GEWANNEN ";
530 :
540 : IF GL = 3 THEN PRINT "3 DM.": LET SK = SK+3 : GOTO 600
550 : IF OG = 3 THEN PRINT "2 DM.": LET SK = SK+2 : GOTO 600
560 : IF KI = 3 THEN PRINT "1 DM.": LET SK = SK+1 : GOTO 600
570 : PRINT "LEIDER NICHTS."
590 :
600 REM === NOCH EIN SPIEL? ==========================================
601 :
610 : LET SK = SK - 0.1
615 : LET SK = INT(100*SK+0.5)/100
620 :
630 : PRINT
640 : PRINT "IHR SPIELKAPITAL BETRAEGT JETZT ";SK;"DM.
650 :
660 : PRINT "■■■■■NOCH EIN SPIEL? (J/N)
670 : GET A$ : IF A$ = "" THEN 670
680 : IF A$ = "J" THEN 300 : REM NEUES SPIEL
690 :
700 : PRINT "SIE HANDELN WEISE! AUF WIEDERSEHEN.
710 :
790 : END
```

Aufgaben

1. Bauen Sie das Programm "Die böse Sechs" zu einem Spiel für n Personen aus (der Computer soll Spielmeister sein).

2. Entwerfen Sie Spiel und Programm für andere 'böse Zahlen' (z.B. "die böse Eins"). Welche Zahl böse ist, soll vorher wählbar sein.

3. Erweitern Sie das Spiel "Die böse Sechs" derart, daß nicht mit zwei, sondern mit k Würfeln zugleich geworfen werden kann. Zeigen Sie, daß die optimale Strategie so aussieht: man würfelt weiter, wenn die erreichte Punktzahl kleiner als

$$3k\left(\frac{5}{6}\right)^k \;/\; \left(1-\left(\frac{5}{6}\right)^k\right)$$

ist (das Endspiel nicht berücksichtigt).

4. Gerade-ungerade
Eines der ältesten Würfelspiele hat folgende Regeln: ein Würfel wird solange geworfen, bis zum ersten Mal eine ungerade Augenzahl erscheint. Man wettet auf die Anzahl der Würfe, bis dieser Fall eintritt. Schreiben Sie ein Programm für dies Spiel!

5. Wie groß ist die Wahrscheinlichkeit, daß beim gleichzeitigen Wurf von 13 guten Würfeln die Serie 1,2,3,4,5,6 erscheint? Beantworten Sie die Frage mit Hilfe einer Simulation und durch Rechnung.
(Lösung:

$$1 - \frac{6\cdot 5^{13} - 15\cdot 4^{13} + 20\cdot 3^{13} - 15\cdot 2^{13} + 6}{6^{13}}\ )$$

6. Nicolas Bernoulli erfand das sogenannte St.Petersburg-Spiel: ein Würfel wird solange geworfen, bis zum ersten Mal 6 erscheint; tritt dieser Fall nach n Würfen ein, erhält der Spieler n Geldeinheiten ausbezahlt. Wieviel bekommt man 'auf lange Sicht' ausbezahlt, und wie hoch muß demnach der Einsatz sein, damit das Spiel fair ist?
Lösen Sie die Aufgabe durch Simulation und durch Rechnung!

7. Würfel auf Eis
Es werden 5 Würfel zugleich geworfen. Nach jedem Wurf wird einer oder werden mehrere 'auf Eis' gelegt; mit den übrigen wird solange weitergewürfelt, bis alle auf Eis gelegt sind. Ziel ist die Maximierung der Augensumme der auf Eis gelegten Würfel. Ist s die erzielte Augensumme, so erhält der Spieler von jedem Mitspieler s - 24 Punkte; ist $s < 24$, so muß er an jeden Mitspieler 24 - s Punkte abgeben.
Entwickeln Sie ein Programm mit optimaler Spielstrategie.

8. Alle Sechse
Es werden 5 Würfel geworfen; diejenigen, welche eine 6 zeigen, werden beiseite gelegt. Mit den übrigen wird solange weitergespielt, bis alle beiseite gelegt sind. Wie lange muß man im Mittel würfeln, bis dieser Zustand erreicht ist?

9. Dreiundzwanzig
Es werden 4 Würfel gleichzeitig geworfen. Das Ziel ist, die 4 Augenzahlen mittels der 4 Grundrechenarten zur Zahl 23 (oder möglichst wenig darunter) zu verknüpfen; dabei muß jede Augenzahl genau einmal benutzt werden. Beispiel:

a) (6:1)·4 - 5 = 24 - 5 = 19 b) 4·6 - (5-1) = 24 - 4 = 20
c) 5·(6-1) - 4 = 25 - 4 = 21 d) 4·(5-1) + 6 = 16 + 6 = 22
e) (4-1)·6 + 5 = 18 + 5 = 23 f) (6+1)·4 - 5 = 28 - 5 = 23 .

Schreiben Sie ein Programm für dies Spiel.

10. Ziel 27
Bei diesem Zweipersonen-Spiel würfeln beide Spieler abwechselnd. Die jeweils erzielte Augenzahl kann entweder zum eigenen Gesamtergebnis addiert oder von dem des Gegners subtrahiert werden. Gewonnen hat derjenige, welcher mit seinem Ergebnis zuerst einen Wert zwischen 26 und 29 erreicht. Schreiben Sie ein Programm für den Computer als Spielgegner.

11. Teutonenpoker
Gespielt wird mit 6 Würfeln, jeder Spieler hat drei Würfe pro Runde, wobei er jeweils entscheiden kann, welche Würfe er stehenläßt. Nach dem dritten Wurf werden ihm die Punkte gutgeschrieben. Wer zuerst 10 000 Punkte erreicht, hat gewonnen. Die Eins zählt 100 Punkte, die Fünf zählt 50 Punkte, die übrigen (allein) nichts. Ferner zählt dreimal 6 : 600 Punkte, dreimal 5 : 500 Punkte, dreimal 4 : 400 P., dreimal 3 : 300 Punkte, dreimal 2 : 200 Punkte und dreimal 1 zählt 1000 Punkte.

12. Karnevalistisches Würfeltreiben
Sam Loyd beschreibt folgendes Spiel: auf dem Tisch befinden sich 6 Felder, die mit den Ziffern von 1 bis 6 bezeichnet sind. Die Spieler können auf jedes Feld soviel Geld legen, wie sie möchten. Es wird mit drei Würfeln gespielt. Zeigt einer der Würfel Ihre Feldzahl an, so erhalten Sie Ihr Geld zurück und darüber hinaus die gleiche Menge noch einmal. Zeigen zwei Würfel Ihre Feldzahl, dann erhalten Sie Ihr Geld zurück und zusätzlich das Doppelte der eingezahlten Menge. Erscheint Ihre Feldziffer auf allen drei Würfeln, so erhalten Sie außer dem Einsatz den dreifachen Betrag. Zeigt jedoch keiner der Würfel Ihre Feldzahl an, so kassiert der Spielmacher Ihren Einsatz. Für wen ist das Spiel günstig? (Schreiben Sie ein Simulationsprogramm oder beantworten Sie die Frage durch Rechnung.)

13. Bestätigen Sie die theoretisch hergeleiteten Werte für die Gewinnwahrscheinlichkeiten und für die mittlere Spieldauer bei 'Craps' durch ein Simulationsprogramm.

14. Angenommen, ein Spieler beginnt mit 10 DM Kapital beim Craps-Spiel, und er setzt 1 DM pro Partie solange, bis er sein gesamtes Kapital verspielt hat. Bestätigen Sie durch eine Simulation, daß er bis zum Ruin

$$10 \cdot \frac{495}{7} \approx 707$$

DM setzen kann. Ihr Programm sollte eine Liste folgender Gestalt ausdrucken:

Nr.	Anzahl der Partien bis zum Ruin	Anzahl der Würfe	Maximal erreichtes Spielkapital
Mittelwert	707	2390	35

15. Unser 'Craps'-Programm simulierte das Spiel eines einzigen gegen die Bank. Schreiben Sie ein Spielprogramm für mehrere Personen, unter denen eine den Bankhalter ('shooter') spielen muß. Dieser wird zu Beginn entweder durch Stechen ermittelt (jeder wirft einmal und 'hoch' übernimmt den Posten), oder man läßt die erste Bank versteigern, wobei der Meistbietende den Zuschlag erhält.
Diesen Betrag setzt er auf seinen Sieg; die übrigen Spieler setzen dagegen, und zwar setzen sie insgesamt gleich viel wie der Shooter. Nun würfelt der Shooter. Gewinnt er durch ein Natural, so streicht er alle Einsätze ein und ist auch - wenn er es wünscht - in der nächsten Runde der Shooter. Verliert er durch einen Crap, muß er jedem Mitspieler den doppelten Betrag von dessen Einsatz aus der Kasse auszahlen. Er kann - wenn er will - Shooter bleiben. Verliert er nach einem 'Punkt-Wurf', so muß er wie vorher auszahlen und die Würfel abgeben.

16. Bestätigen Sie durch eine Simulation, daß die Gewinnwahrscheinlichkeit beim Setzen auf 'big 6 or big 8' (d.h. Erscheinen der 6 oder der 8 vor der 7)

$$\frac{5}{11} \approx 45{,}5\%$$

beträgt.

17. Wie für jedes Glücksspiel gibt es auch für 'Craps' zahlreiche Wettsysteme. Das S t r e i c h u n g s s y s t e m geht wie folgt: man schreibt (z.B.) die Zahlen 1,2,3,4,5,6,7,8,9,10 nebeneinander und setzt die Summe der ersten und letzten Zahl (Einsatz also 11 DM). Hat man gewonnen, so werden diese Zahlen gestrichen; es bleibt also die Folge

2,3,4,5,6,7,8,9 .

Nach einer verlorenen Partie fügt man den verlorenen Betrag als letzte Zahl an: 1,2,3,4,5,6,7,8,9,10,11.

Nun setzt man wieder die Summe aus erster und letzter Zahl usf. Die 'Theorie' dahinter ist: da es etwa gleichviel Gewinne wie Verluste auf lange Sicht gibt, so wird irgendeinmal die ganze Liste gestrichen sein, d.h. der Spieler hat die Summe der ursprünglich angeschriebenen Zahlen gewonnen. Zeigen Sie mit Hilfe eines Simulationsprogramms, daß das System auf die Dauer nicht zum Erfolg führt. Wo steckt der Trugschluß?

18. Schreiben Sie das Programm 'Münzwurf' so um, daß es zum Schätzen der Gewinnwahrscheinlichkeiten bei Wahl und Gegenwahl der Muster taugt. (Frage: wer gewinnt bei den Muster KZK und ZKZ - und mit welcher Wahrscheinlichkeit?)

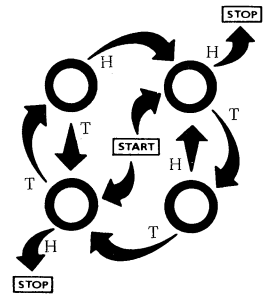
19. Prüfen Sie, ob die in den DATA-Zeilen 390, 391 des Programms 'Münzwurfspiel' verankerte Gewinnstrategie auch für gefälschte Münzen gilt (das sind solche, bei denen die Wahrscheinlichkeit für 'Kopf' nicht 1/2 ist).

20. Schreiben Sie das Programm 'Münzwurf' so um, daß der Computer sein Muster zufällig wählt und der Benutzer dann anschließend wählen kann.

21. Verallgemeinern Sie das Münzwurfspiel von Beispiel 3 auf Muster mit vier Würfen (z.B. KKKK, KKKZ, KKZK, ...).

22. Schreiben Sie das Programm 'Münzwurfspiel' für drei Spieler um. Bleibt die bisherige Strategie optimal?

23. Vier Münzen liegen auf dem Tisch, Zahl oben. Nun wird jede Münze solange geworfen, bis Kopf oben liegt. Wie lange dauert es im Mittel, bis alle vier Münzen Kopf zeigen?



24. Erweitern Sie das Programm '17 + 4' so, daß mehrere Personen mitspielen können und lassen Sie Einsätze zu.

25. Geben Sie beim Programm '17 + 4' dem Spieler die Möglichkeit, das As wahlweise 1 oder 11 Punkte zählen zu lassen.

26. Erweitern Sie das Roulette-Programm so, daß auch auf 'Cheval', auf die Transversalen und auf 'Carré' gesetzt werden kann.

27. Ermöglichen Sie das Spiel mehrerer Personen beim Roulette-Programm.

28. Bauen Sie ins Roulette-Programm die Möglichkeit ein, daß beim Erscheinen von Zéro der Satz geteilt werden kann.

29. Beim Roulette-Programm muß die Eingabekontrolle noch verbessert werden; z.B. ist zu verhindern, daß der Spieler aus Versehen mehr als einmal auf die gleiche Chance setzt.

30. Das Roulette-Programm eignet sich dazu, die verschiedenen Wettsysteme zu testen (um festzustellen, daß sie nichts taugen).
    a) Das bekannteste ist das sogenannte d'Alembert-System: der Spieler setzt auf die einfachen Chancen und erhöht nach jedem Verlust den Einsatz um einen festen Betrag, nach jedem Gewinn verringert er den Einsatz um diesen Betrag.
    Dahinter steht die (unsinnige) Vorstellung, daß die weiße Kugel sich daran erinnert, daß (beispielsweise) mehrfach hintereinander Rot gefallen ist, und daß sie diese Farbe nun zu meiden sucht (um des Ausgleichs willen).
    b) Beim Martingale-System wird der Einsatz nach jedem Verlust verdoppelt (evtl. noch zusätzlich + 1). Überlegen Sie, warum auch dies System scheitern muß.
    c) Das Streichungssystem haben wir schon in Aufgabe 17 kennengelernt.
    d) Das kubanische System beruht auf der Tatsache, daß die dritte Kolonne (3,6,9,...,36) acht rote und nur vier schwarze Zahlen trägt. Man setzt auf Schwarz und auf Kolonne drei zugleich. Probieren Sie es aus! (Dann überlegen Sie bitte, warum das System auf einem Trugschluß beruht.)
    Jedes dieser Systeme ist ja ein Algorithmus und kann daher unschwer in ein Programm eingebaut werden.

31. Erweitern Sie das Programm 'Einarmiger Bandit' durch Berücksichtigung der auf Seite 149 gezeigten Kombinationen.

32. Unser Programm 'Einarmiger Bandit' ist für den Spieler äußerst günstig: er wird ansehnliche Beträge einstreichen. Wie muß der Einsatz festgelegt werden, damit der Automat seinem Eigentümer einen Verdienst abwirft?

33. Lustige Sieben
    Auf ein Blatt Papier wird nebenstehende Figur gezeichnet. Die Spieler können auf die Zahlen beliebig setzen. Würfelt der Bankhalter eine Zahl der rechten Seite, so kassiert er alle Einsätze dieser Seite, muß aber alle Einsätze der linken Seite verdoppeln. Entsprechend, wenn er eine Zahl der linken Seite würfelt.

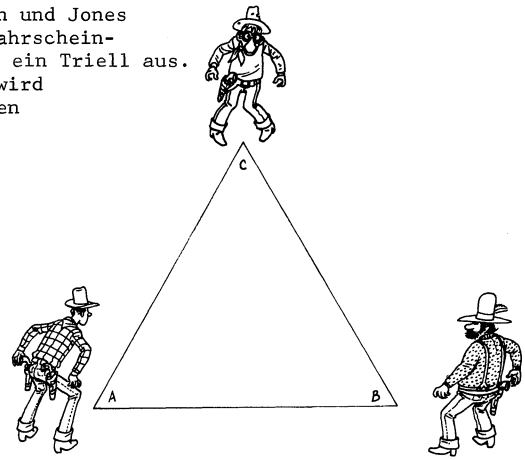
7	
2	3
4	5
6	8
9	10
11	12

    Würfelt er eine 7, so kassiert er alle Einsätze, muß aber die auf der 7 verdreifachen.
    Programmieren Sie dies Spiel!

34. Variante der Lustigen Sieben
    Der Bankhalter zahlt den Einsatz der getroffenen Nummer doppelt aus, die in gleicher Reihe stehende einfach. Die Sieben erhält den dreifachen Betrag. Alle anderen Einsätze zieht er ein.

35. Zeigen Sie durch Simulation und durch Rechnung, daß der Bankhalter bei der Lustigen Sieben langfristig im Vorteil ist. Weiter: wer auf die Sieben setzt, verliert auf lange Sicht ein Drittel seines Einsatzes (Variante von Aufgabe 34). Wie ist es im Spiel der Aufgabe 33?

36. Die Herren Smith, Brown und Jones fechten mit den Treffwahrscheinlichkeiten 1, 0.8, 0.5 ein Triell aus. Die Reihenfolge 1,2,3 wird ausgelost. Dann schießen sie zyklisch in dieser Reihenfolge, wobei Tote übersprungen werden, bis nur noch einer am Leben ist. Jeder darf sich sein Ziel frei wählen. Wie groß sind ihre Überlebenschancen, wenn sie sich optimal verhalten? Ergibt sich aus dem Triell eine Lektion, die sich vielleicht auf dem Gebiet der internationalen Beziehungen anwenden ließe?



37. Wir beginnen mit 12 Urnen und legen in jede eine rote Kugel. Der Bankhalter fügt dann jeder Urne eine gewisse Anzahl (die auch Null sein darf) weißer Kugeln hinzu. Nun werden die Urnen zufällig angeordnet und der Spieler zieht eine Urne zufällig, dann nimmt er eine Kugel heraus. Ist die gezogene Kugel rot, so bekommt er den verdoppelten Einsatz zurück, ist sie weiß, so verliert er seinen Einsatz. Wieviele weiße Kugeln legt der Bankhalter in die Urnen, damit er auf lange Sicht einen Vorteil hat, damit aber andererseits das Spiel attraktiv bleibt (d.h. der Vorteil darf nicht zu groß sein)? Schreiben Sie ein Simulationsprogramm.

38. Verhexte Würfel
Rolf Müller schreibt in TETRA: "Zugegeben, ganz fair ist die Geschichte nicht, aber verblüffend und reizvoll allemal. Beschriften Sie doch vier Würfel auf eigene Art: der erste Würfel soll nur die Zahl 3 tragen - sechsmal. Der zweite soll viermal die 2 und zweimal die 6 erhalten, der dritte dreimal die 1 und dreimal die 5, der vierte schließlich soll viermal die 4 und zweimal die 0 bekommen.

Nun brauchen Sie nur noch ein Opfer für das folgende Spielchen: Ihr Partner möge sich einen Würfel aussuchen; Sie nehmen anschließend einen anderen. Jetzt wird um die Wette gewürfelt: beide werfen zugleich; wer die höhere Augenzahl hat, erhält einen Pluspunkt.
Nach einiger Zeit beginnt Ihre Punkteanzahl die Ihres Mitspielers deutlich zu übertreffen - in etwa werden Sie doppelt soviel Punkte haben wie er.
Ihr Gegner wird unzufrieden, begehrt besseres Spielgerät, vielleicht sogar Ihren Würfel. Geben Sie ihn getrost her, nehmen Sie selbst einen der anderen drei, und dann kann's wieder losgehen. Bald zeigt sich erneut Ihr Würfelglück, das Ihren Spielgegner beunruhigt.
Er wird nun wohl den noch besseren Würfel wollen, den Sie gerade benutzen. Seien Sie großzügig, trennen Sie sich von ihm! Auch in der dritten Runde sind Sie auf die Dauer etwa doppelt so gut mit Ihrer neuen Wahl wie Ihr Gegenspieler.
Nun ist er aber wohl entdeckt, d e r Würfel, der der beste von allen ist: es muß der sein, den Sie jetzt benutzen!
Trennen Sie sich von ihm - auch mit diesem 'besten aller Würfel' kann Ihnen Ihr Gegner nicht das Wasser reichen ...
Sein Fluch ist, daß er die Wahl hat, und damit die Qual. Denn stets ist Ihre Gegenwahl die bessere."
Schreiben Sie ein Programm, in dem der Computer die Gegenwahl hat und damit gewinnt.

39. Eine Variante zu Bradley Efron - Müllers Spiel: Würfel A trage die Ziffern 1,2,3,3,6,6 und Würfel B die Ziffern 1,1,2,5,6,6.
Zeigen Sie, daß A gegen B mit Wahrscheinlichkeit 16/36 gewinnt.
Würfel C trage die Ziffern 2,2,4,4,4,5.
Dann gewinnt B gegen C mit Wahrscheinlichkeit 17/36 - aber es gewinnt C gegen A mit Wahrscheinlichkeit 18/36!
Bestätigen Sie dies mit einem Simulationsprogramm und rechnen Sie es nach.
Finden Sie weitere verhexte Würfel!

"Ehrlich - die alten Spiele
gefielen mir besser!"

6
Strategiespiele

Nirgendwo hat der Mensch
mehr Scharfsinn an den Tag gelegt
als in seinen Spielen.
LEIBNIZ

In diesem Kapitel werden wir uns mit Spielen beschäftigen, die sich folgendermaßen kennzeichnen lassen: es geht um einen Wettkampf zwischen zwei Personen, der

- nach einer endlichen Anzahl von Spielhandlungen beendet ist, in dem
- keinerlei Zufallselemente (wie z.B. Würfeln oder Karten austeilen) eine Rolle spielen, der Spielverlauf also ausschließlich durch die Entscheidungen der Spieler bestimmt ist, und bei dem
- beide Spieler in die Züge des Gegners und die entstandenen Spielstellungen ungehinderten Einblick genießen.

Eine einzelne Realisierung des Spiels, d.h. eine tatsächlich durchlaufene Folge von Spielstellungen, heißt Partie; die Entscheidungen der Spieler im Verlauf einer Partie heißen Züge. Unter einer Strategie des Spielers A versteht man einen Plan, welcher zu jeder Spielstellung, in der A am Zuge ist, eine Anweisung enthält, welcher Zug als nächster auszuführen ist.

*

Das Nim-Spiel (mit seinen verschiedenen Varianten) ist eines der einfachsten nicht-trivialen strategischen Spiele, das vollständig analysiert und für Computer programmiert worden ist. Schon im Jahr 1612 beschreibt Bachet de Méziriac in seinen "Problèmes plaisants

et delectables, qui se font par les nombres" ein Spiel, das wir heute als "Nim mit einem Haufen" bezeichnen würden (siehe das Beispiel 4 der Einleitung). Ende des 19. Jahrhunderts war das Spiel unter amerikanischen College-Studenten als "Fan-tan" bekannt; die Bezeichnung 'Nim' erhielt es von dem kanadischen Mathematiker Ch.L. B o u t o n , der es im Jahr 1902 mathematisch analysierte.

Beispiel 1 : Nim mit mehreren Schalen

Mehrere Schalen, mit Früchten gefüllt, stehen auf dem Tisch. Zwei Spieler dürfen abwechselnd jeweils einer Schale soviele Früchte entnehmen, wie sie mögen - mindestens aber jeweils eine. Wer nicht mehr ziehen kann (weil alle Schalen leer sind), hat verloren.
Einer der Spieler soll der Computer sein.

Um den Computer zu einem ernstzunehmenden Spielgegner zu machen, müssen wir ihm eine aussichtsreiche Spielstrategie beibringen. Wir betrachten zunächst den Fall, daß eine der Schalen (schon) leer ist und spielen eine Partie (zu Ende); Spieler A sei am Zug:



Spielstellung	Zug
⁘	A nimmt 2 Früchte aus Schale zwei
⁙	B nimmt 1 Frucht aus Schale eins
∴	A nimmt 1 Frucht aus Schale zwei
:	B nimmt 1 Frucht aus Schale zwei
•	A nimmt 1 Frucht aus Schale eins
	B kann nicht mehr ziehen und hat damit verloren.

Welche Strategie hat A dabei verfolgt? Nun, er nahm immer soviele Früchte weg, daß er B jeweils gleichviele Früchte in beiden Schalen überließ. Was B auch unternehmen mochte - Spieler A konnte die ausgewogene Situation "in beiden Schalen gleichviele Früchte" stets wiederherstellen. Damit besaß A - bei obiger Anfangsstellung - eine Gewinnstrategie. (Wären zu Beginn allerdings gleichviele Früchte in beiden Schalen gewesen, so hätte der Nachziehende B eine Gewinnstrategie besessen, denn mit dem ersten Zug hätte A eine ungleiche Verteilung der Früchte herstellen müssen, und dann wäre B verfahren wie vorher A.) -

Eine Spielstellung läßt sich durch Anzahl der in jeder Schale befindlichen Früchte kennzeichnen; kürzen wir obige Anfangsstellung durch das Zahlentripel (2, 4, 0) ab, so läßt sich sagen: jede Stellung

$$(n, m, 0) \quad \text{mit} \quad n \neq m$$

ist für den Anziehenden eine Gewinnstellung; jede Stellung

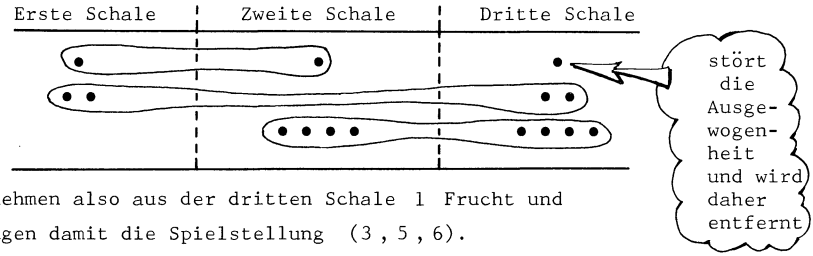
$$(n, n, 0)$$

ist für den Anziehenden eine Verluststellung, d.h. er muß die Partie verlieren, wenn der Gegner Bescheid weiß.

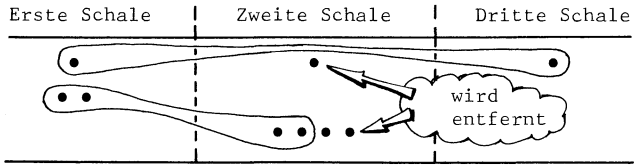
Wie sieht dies nun im Fall dreier nicht-leerer Schalen aus? Nehmen wir den Fall (3, 5, 7)!



Gibt es wieder einen Zug, der eine 'ausgewogene' Situation herbeiführt; eine Situation, die unser Spielgegner zerstören muß, und die wir anschließend wieder herstellen? Was heißt jetzt 'ausgewogen'? Es kann nicht bedeuten: "gleichviele Früchte in den Schalen" - das wäre gegen die Spielregeln. Folgende Interpretation bietet sich an:



Wir nehmen also aus der dritten Schale 1 Frucht und erzeugen damit die Spielstellung (3 , 5 , 6).
Die Ausgewogenheit wird nun unser Spielgegner mit seinem nächsten Zug zerstören (müssen) - egal, was er anstellt. Angenommen, er nimmt 5 Früchte aus Schale drei und hinterläßt damit die Stellung (3 , 5 , 1): die Ausgewogenheit ist jetzt gründlich zerstört. Wir stellen sie wie folgt wieder her:



Wir nehmen also aus der zweiten Schale 3 Früchte und erzeugen damit die Spielstellung (3 , 2 , 1). Dies ist - wie leicht zu sehen - eine Verluststellung für unseren Gegner. Was will er noch machen ?! (Der Leser möge die verbleibenden Möglichkeiten durchspielen.)

Damit können wir die auftretenden Spielstellungen allgemein so beschreiben:

> Jede Spielstellung (gegeben durch die Früchteverteilung in den drei Schalen) ist entweder eine G-Stellung oder eine V-Stellung. Es gilt:
> (1) Die Endstellung (0 , 0 , 0) ist eine V-Stellung.
> (2) Eine Stellung ist eine G-Stellung, wenn es einen Zug gibt derart, daß die resultierende Stellung eine V-Stellung ist.
> (3) Eine Stellung ist eine V-Stellung, wenn jeder Zug sie in eine G-Stellung überführt.

Damit ist die Strategie klar: man muß versuchen, dem Gegner immer V-Stel-

lungen vorzulegen. Dann ist er nämlich gemäß Eigenschaft (3) gezwungen, uns wieder eine G-Stellung zu überlassen. Das heißt: ist die Anfangsstellung eine G-Stellung, so gewinnt der Anziehende, wenn er die genannte Strategie befolgt, sicher. Er muß, wenn er dran ist, einen der nach Eigenschaft (2) stets existierenden Züge finden, der seine G-Stellung in eine V-Stellung überführt: darin besteht die Kunst (gut zu spielen). Da sich bei jedem Zug die Anzahl der Früchte vermindert, endet das Spiel nach endlich vielen Zügen in der Stellung (0,0,0).-

Woran erkennen wir nun die G- und die V-Stellungen?

E i n e Möglichkeit wäre folgende: wir schreiben, ausgehend von der V-Stellung (0,0,0), alle Stellungen auf, die durch einen Spielzug in (0,0,0) überführt werden können (Vorgänger), und markieren Sie als G-Stellungen. Von jeder so gefundenen G-Stellung ausgehend, suchen wir alle Vorgänger und markieren unter ihnen die V-Stellungen als diejenigen, deren sämtliche Folgestellungen G-Stellungen sind; die übrigen als G-Stellungen. Dies wiederholen wir solange, bis alle Stellungen markiert sind. (Das Verfahren läßt sich auch mittels Computer durchführen!)

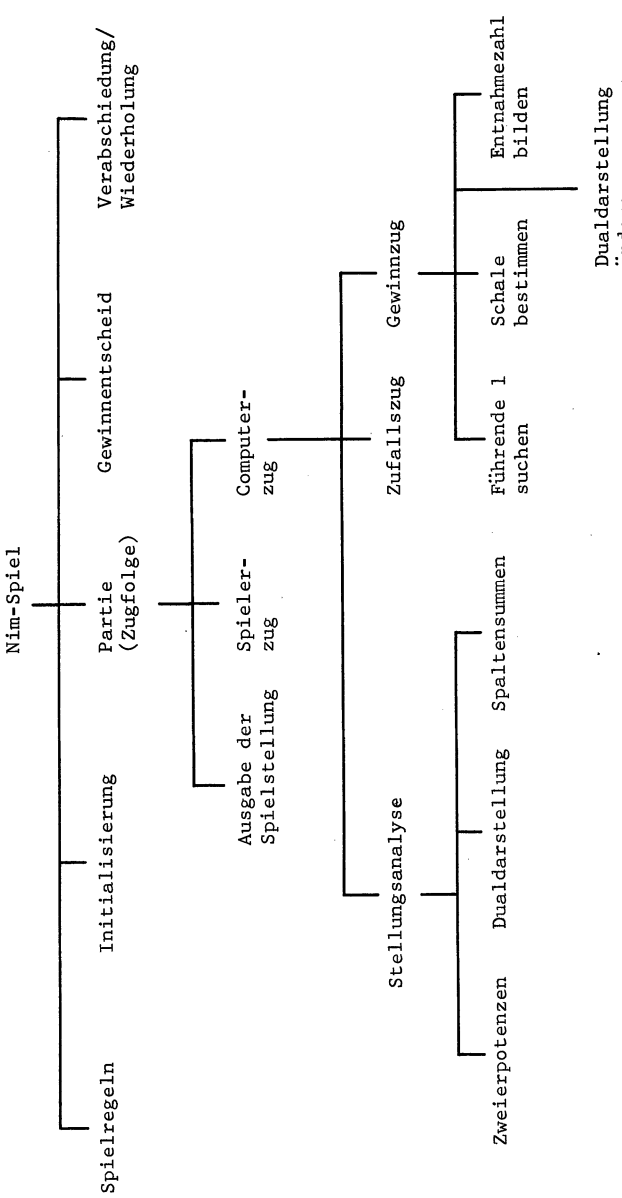
Eine z w e i t e Möglichkeit knüpft an unsere vorigen Überlegungen an. Wir hatten die Früchteanzahlen in Potenzen von 2 zerlegt und dann die 'ausgewogenen' Stellungen (= Verluststellungen) daran erkannt, daß die Anzahl der auftretenden Zweierpotenzen g e r a d e ist. [Im Fall (3,5,7) ergab sich $3 = 2 + 1$, $5 = 4 + 1$ und $6 = 4 + 2$, d.h. die Anzahl der Einsen, Zweien und Vieren war gerade.] Etwas formaler: die Früchteanzahlen werden als Dualzahlen geschrieben, dann wird deren N i m - S u m m e gebildet (0+0 = 0, 0+1 = 1+0 = 1, 1+1 = 0, stellenweise Addition); ist sie Null, liegt eine V-Stellung, andernfalls eine G-Stellung vor.

[In obigem Beispiel sieht das so aus:

```
3 =  II
5 = IOI
6 = IIO
    000
```

, also ist (3,5,6) eine V-Stellung.] Es läßt sich leicht zeigen, daß die so gekennzeichneten G- und V-Stellungen die obigen Bedingungen (1), (2) und (3) für eine Gewinn-Verlust-Zerlegung erfüllen. Dies ist Boutons berühmte Methode; wir wollen sie nun in ein Programm übersetzen.

Unser Programm wird folgende Struktur aufweisen:



Dabei werden wir darauf achten, daß die Unterprogramme nicht im Gesamtprogramm verstreut sind, sondern daß sie 'innerhalb' des jeweils übergeordneten Programms auftreten.

Das Kennzeichnende des Nim-Spiels zeigt im wesentlichen das folgende Unterprogramm:

PARTIE (ZUGFOLGE)

```
Ausgabe: Spielstellung
WIEDERHOLE
  Spielerzug
  letzter_Zug := "Mensch"
  Ausgabe: Spielstellung
  WENN  Früchteanzahl = 0  DANN ABBRUCH
  Computerzug
  letzter_Zug := "Computer"
  Ausgabe: Spielstellung
  WENN  Früchteanzahl = 0  DANN ABBRUCH
ENDE-WIEDERHOLE
```

Das heißt: die Partie besteht in einem abwechselnden Ziehen von Mensch und Computer, dabei merkt sich das Programm in der Variablen letzter Zug, wer jeweils zuletzt am Zuge war. Dies ist für den Gewinnentscheid notwendig; wer nach dem Abbruch den letzten Zug gemacht hatte, ist Sieger:

GEWINNENTSCHEID

```
WENN  letzter_Zug = "Mensch" DANN
  Ausgabe: "Sie haben gewonnen!"
SONST
  Ausgabe: "Ich (der Computer) habe gewonnen."
ENDE-WENN
```

Die oben entwickelte Bouton-Strategie äußert sich im Programmteil "Computerzug": wenn der Computer Anziehender in einer Verluststellung ist, so macht er einen Zufallszug, der darin besteht, daß er aus einer zufällig gewählten Schale eine Frucht nimmt; andernfalls macht er einen 'Gewinnzug', d.h. einen Zug, wie ihn die Gewinnstrategie vorschreibt.

COMPUTERZUG

```
Stellungsanalyse
WENN  spaltensumme = 0  DANN
  Zufallszug
SONST
  Gewinnzug
ENDE-WENN
Ausgabe: "Ich nehme ... Früchte"
Berechnung der neuen Früchteanzahl
```

Vorauszugehen hat dem eine Stellungsanalyse, in welcher die Früchteanzahlen zunächst in Dualdarstellung übersetzt und dann die Dualziffern spaltenweise addiert werden (nach der Formel $a \oplus b = a + b - a*b$).
Der Gewinnzug läßt sich wie folgt beschreiben: man beginne von links, ändere die Spalten mit einer ungeraden Anzahl von I durch Hinzufügen oder Entfernen einer I so, daß eine gerade Anzahl entsteht. Dies geht immer, indem man nur die Anzahl einer einzigen Schale ändert. Demgemäß zerfällt diese Prozedur in die Teile "Führende I suchen", "Schale bestimmen", "Dualdarstellung ändern" und "Entnahmezahl bilden".
Beispiel:

	Dualstelle j						
Schale i	5	4	3	2	1	0	Früchteanzahl F(i)
1	0	0	0	0	I	I	3
2	0	0	0	I	0	I	5
3	0	0	I	I	0	I	13
Summe S(j)	0	0	I	0	I	I	

In diesem Beispiel werden der dritten Schale 7 Früchte entnommen. -

Das Programm beginnt auf der folgenden Seite.

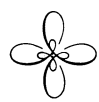
A XORB NOT ((A AND B) OR NOT A AND NOT B)

```
1000 : PRINT "□
1010 : PRINT "                          NIM-SPIEL
1011 : PRINT "                          ---------
1020 :
1030 REM DER COMPUTER SPIELT GEMAESS DER BOUTON-STRATEGIE
1040 REM NIM MIT DREI HAUFEN (BZW. SCHALEN) GEGEN DEN BENUTZER
1090 :
1099 :
2000 REM *** HAUPTPROGRAMM *****************************************
2010 :
2300 : GOSUB 3000 : REM SPIELREGELN
2301 :
2400 : GOSUB 4000 : REM INITIALISIERUNG
2401 :
2500 : GOSUB 5000 : REM PARTIE (ZUGFOLGE)
2501 :
2600 : GOSUB 6000 : REM GEWINNENTSCHEID
2601 :
2700 : GOSUB 7000 : REM VERABSCHIEDUNG ODER WIEDERHOLUNG
2701 :
2900 : END
2950 :
2990 REM *** ENDE DES HAUPTPROGRAMMS ********************************
2998 :
2999 :
3000 REM ### UNTERPROGRAMM 'SPIELREGELN' ##############################
3010 :
3100 : PRINT
3110 : PRINT "SOLL ICH DIE SPIELREGELN NENNEN? (J/N)
3120 : GET A$
3130 : IF A$ = "" THEN 3120
3140 : IF A$ <> "J" THEN RETURN
3190 :
3195 : PRINT
3200 : PRINT "AUF DEM TISCH STEHEN DREI SCHALEN MIT FRUECHTEN.
3300 : PRINT "SIE UND ICH NEHMEN IM WECHSEL AUS JE EINER SCHALE
3400 : PRINT "MINDESTENS EINE FRUCHT.
3500 : PRINT "WER NICHTS MEHR NEHMEN KANN, WEIL ALLE SCHALEN
3600 : PRINT "LEER SIND, HAT VERLOREN.
3900 :
3990 : RETURN : REM ENDE SPIELREGELN ###################################
3999 :
4000 REM ### UNTERPROGRAMM 'INITIALISIERUNG' ##########################
4010 :
4100 : PRINT : PRINT
4150 : PRINT "VORGABE DER ANZAHLEN:
4300 : PRINT
4400 : LET N = 0          : REM GESAMTANZAHL DER FRUECHTE
4500 : FOR I = 1 TO 3
4550 :   PRINT "WIEVIELE FRUECHTE LIEGEN IN DER";I;". SCHALE ";
4555 :   INPUT F
4556 :   IF F < 1 OR F > 50 THEN 4550
4560 :   LET F(I) = F : LET N = N+F
4580 : NEXT I
4590 :
4900 : RETURN : REM ENDE DER INITIALISIERUNG ##########################
4999 :
```

```
5000 REM ### UNTERPROGRAMM 'PARTIE' ###################################
5010 :
5100 : GOSUB 5200              : REM AUSGABE DER SPIELSTELLUNG
5105 :
5110 : GOSUB 5300              : REM SPIELERZUG
5120 :
5130 : GOSUB 5200              : REM AUSGABE DER SPIELSTELLUNG
5135 :
5140 : IF N <= 0 THEN RETURN : REM ALLE SCHALEN LEER, PARTIE ZUENDE
5145 :
5150 : GOSUB 5400              : REM COMPUTERZUG
5160 :
5165 : GOSUB 5200              : REM AUSGABE DER SPIELSTELLUNG
5169 :
5170 : IF N <= 0 THEN RETURN : REM ALLE SCHALEN LEER, PARTIE ZUENDE
5175 :
5180 : GOTO 5110               : REM RUECKSPRUNG SPIELERZUG
5185 :
5190 : REM ENDE DER PARTIE ############################################
5199 :
5200 : REM +++ UNTERPROGRAMM 'AUSGABE DER SPIELSTELLUNG' ++++++++++++++
5201 :
5205 :   PRINT
5210 :   PRINT "MOMENTANE SPIELSTELLUNG:
5215 :   PRINT
5220 :   FOR I = 1 TO 3
5225 :     PRINT I;". SCHALE: ";
5230 :     IF F(I) = 0 THEN 5260
5240 :     FOR J = 1 TO F(I) : PRINT "●    "; : NEXT J
5260 :     PRINT
5270 :   NEXT I
5280 :
5290 :   RETURN : REM ENDE AUSGABE DER SPIELSTELLUNG ++++++++++++++++++
5299 :
5300 : REM +++ UNTERPROGRAMM 'SPIELERZUG' ++++++++++++++++++++++++++++
5301 :
5310 :   PRINT
5320 :   PRINT "SIE SIND DRAN! ";
5330 :   INPUT "AUS WELCHER SCHALE NEHMEN SIE "; A
5340 :   IF A < 1 OR INT(A) <> A OR A > 3 THEN 5330
5350 :   IF F(A) = 0 THEN PRINT "DIE SCHALE IST LEER!": GOTO 5330
5355 :
5360 :   INPUT "WIEVIEL FRUECHTE "; F
5370 :   IF F < 1 OR INT(F) <> F OR F > F(A) THEN 5360
5380 :   LET F(A) = F(A)-F
5385 :
5390 :   LET N = F(1)+F(2)+F(3)
5393 :
5394 :   LET L$ = "MENSCH"
5395 :
5397 :   RETURN : REM ENDE SPIELERZUG ++++++++++++++++++++++++++++++++
5399 :
```

```
5400 : REM +++ UNTERPROGRAMM COMPUTERZUG ++++++++++++++++++++++++++++++
5401 :
5410 :   GOSUB 5500                                  : REM STELLUNGSANALYSE
5420 :   IF S = 0 THEN GOSUB 5600 : GOTO 5440 : REM ZUFALLSZUG
5430 :   GOSUB 5700                                  : REM GEWINNZUG
5440 :   PRINT
5450 :   PRINT "ICH NEHME AUS DER ";I0;". SCHALE ";Z;" STUECK."
5470 :
5475 :   LET N = F(1)+F(2)+F(3)
5480 :
5485 :   LET L$ = "COMPUTER"
5490 :
5495 :   RETURN : REM ENDE COMPUTERZUG
5499 :
5500 :   REM +++ UNTERPROGRAMM STELLUNGSANALYSE +++
5501 :
5505 :     REM ZWEIERPOTENZEN
5506 :
5510 :       LET P(0) = 1
5520 :       FOR J = 1 TO 5 : LET P(J) = 2*P(J-1) : NEXT J
5530 :
5535 :     REM DUALDARSTELLUNG
5536 :
5540 :       FOR I = 1 TO 3
5545 :         LET M = F(I)
5550 :         FOR J = 5 TO 0 STEP -1
5555 :           LET P = P(J) : LET Z = INT(M/P)
5560 :           LET D(I,J) = Z : LET M = M - Z*P
5565 :         NEXT J
5570 :       NEXT I
5575 :
5580 :     REM SPALTENSUMMEN BILDEN
5581 :
5583 :       LET S = 0
5584 :       FOR J = 0 TO 5
5586 :         LET S(J) = 0
5588 :         FOR I = 1 TO 3
5590 :           LET D = D(I,J)
5592 :           LET S(J) = S(J)+D - 2*S(J)*D
5594 :         NEXT I
5595 :         LET S = S + S(J)
5596 :       NEXT J
5597 :
5598 :       RETURN : REM ENDE STELLUNGSANALYSE
5599 :
5600 :   REM +++ UNTERPROGRAMM ZUFALLSZUG +++
5601 :
5610 :     LET I0 = INT(3*RND(TI))+1   : REM WAHL DER SCHALE
5615 :     IF F(I0) = 0 THEN 5610      : REM SCHALE LEER
5620 :     LET Z = 1                   : REM ENTNAHMEZAHL
5630 :     LET F(I0) = F(I0) - Z       : REM NEUE FRUECHTEZAHL
5640 :
5650 :     RETURN : REM ENDE ZUFALLSZUG
5690 :
5700 :   REM +++ UNTERPROGRAMM GEWINNZUG +++
5701 :
5710 :     REM FUEHRENDE 1 SUCHEN
5711 :
5720 :       FOR J = 5 TO 0 STEP -1
5730 :         IF S(J) > 0 THEN LET J0 = J : LET J = 0
5740 :       NEXT J
5745 :
```

```
5750 :      REM SCHALE BESTIMMEN
5751 :
5755 :        FOR I = 1 TO 3
5760 :          IF D(I,J0) > 0 THEN LET I0 = I : LET I = 3
5765 :        NEXT I
5770 :
5775 :      REM DUALDARSTELLUNG AENDERN
5776 :
5777 :        LET D(I0,J0) = 0
5778 :        IF J0 = 0 THEN 5790  : REM AENDERUNG BEREITS ERFOLGT
5779 :        FOR J = J0-1 TO 0 STEP -1
5781 :          IF S(J) > 0 THEN LET D(I0,J) = 1 - D(I0,J)
5785 :        NEXT J
5788 :
5790 :      REM ENTNAHMEZAHL BILDEN
5791 :
5792 :        LET K = 0
5793 :        FOR J = 0 TO 5 : LET K = K+P(J)*D(I0,J) : NEXT J
5794 :        LET Z = F(I0)-K     : REM ENTNAHMEZAHL
5795 :        LET F(I0) = K
5796 :
5798 :        RETURN : REM ENDE GEWINNZUG
5999 :
6000 REM ### UNTERPROGRAMM 'GEWINNENTSCHEID' ##########################
6010 :
6100 : PRINT
6200 : IF L$ = "COMPUTER" THEN PRINT "ICH HABE GEWONNEN." : RETURN
6300 :                         PRINT "SIE HABEN GEWONNEN."
6800 :
6900 : RETURN : REM ENDE GEWINNENTSCHEID ##############################
6999 :
7000 REM ### UNTERPROGRAMM 'VERABSCHIEDUNG ODER WIEDERHOLUNG' ########
7010 :
7050 : PRINT : PRINT
7100 : PRINT "NOCH EINE PARTIE? (J/N)
7110 : GET A$
7120 : IF A$ = "" THEN 7110
7130 : IF A$ = "J" THEN RUN
7140 :
7200 : PRINT : PRINT : PRINT
7210 : PRINT "AUF WIEDERSEHEN."
7800 :
7900 : RETURN : REM ENDE VERABSCHIEDUNG ODER WIEDERHOLUNG #############
7999 :
```

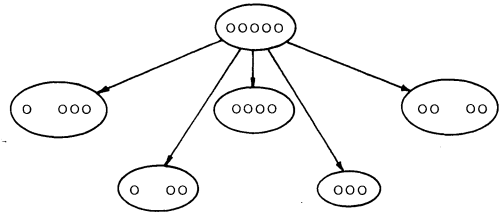


Beispiel 2 : Kegelspiel

Auf Fußboden oder Tisch stehen, linear in Gruppen angeordnet, n Kegel:

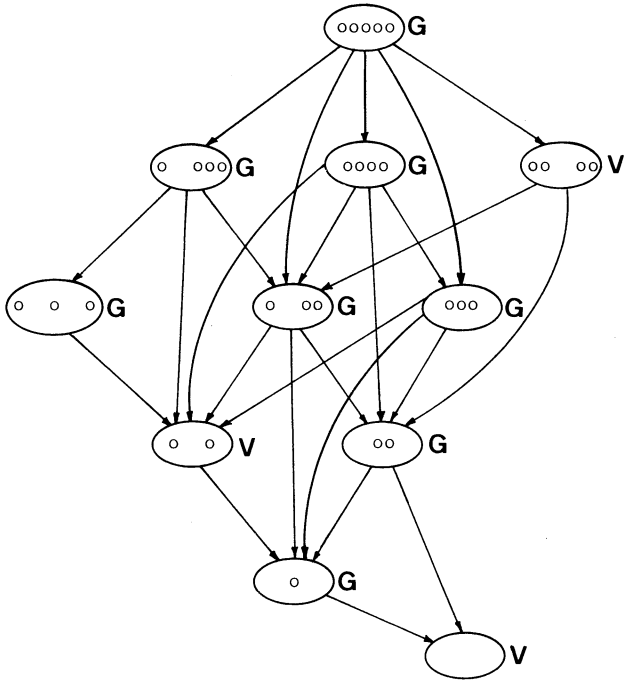
Man kann einen Ball gegen sie rollen und damit einen einzigen oder zwei benachbarte Kegel umwerfen (wegnehmen); Kegel in verschiedenen Gruppen sind nicht benachbart. Verloren hat, wer nicht mehr ziehen (werfen) kann, weil keine Kegel mehr da sind. Eine Gewinnstrategie ist gesucht.

Angenommen, zu Beginn stehen 5 Kegel in einer Gruppe auf dem Tisch (wir deuten die Kegel ab jetzt durch kleine Kringel an): ooooo .
Was kann der anziehende Spieler A nun tun? Er kann einen oder zwei Kegel wegnehmen, und zwar jeweils am Rand oder aus dem Innern; diese Möglichkeiten deuten wir so an:



Von diesen 5 möglichen Zügen führt nur einer zum Erfolg (wenn unser Gegner keinen Fehler macht), nämlich der Zug zur Position (oo oo) ! Was Spieler B nun auch unternehmen mag - A kann es ihm nachtun und hat damit den letzten Zug. Wir sagen - wie beim Nim-Spiel: (oo oo) ist eine Verluststellung (für den am Zug befindlichen Spieler). Um zu zeigen, daß die übrigen vier Stellungen Gewinnstellungen sind, vervollständigen wir das Diagramm. (Man nennt ein solches Diagramm einen Graphen; die Kringel ( ) heißen Knoten und die Pfeile ⟶ heißen gerichtete Kanten.) Das neue Diagramm findet sich auf der folgenden Seite.
An ihm läßt sich folgendes erkennen:
(1) Der Endknoten ist ein V-Knoten (d.h. er kennzeichnet eine V-Stellung).
(2) Von jedem G-Knoten führt mindestens ein Pfeil zu einem V-Knoten.

(3) Jeder Pfeil, der von einem V-Knoten ausgeht, führt zu einem G-Knoten.



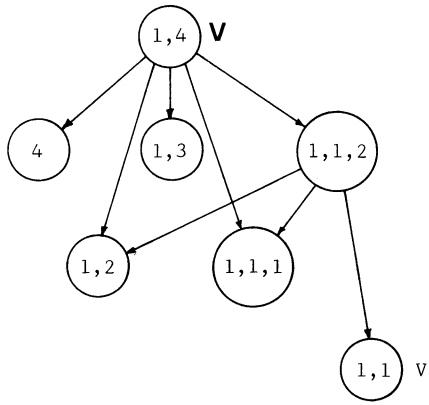
Eine Partie (Zugfolge) können wir uns so vorstellen, daß die Spieler abwechselnd einen Spielstein von Knoten zu Knoten entlang den Kanten (in Pfeilrichtung) schieben; verloren hat, wer nicht mehr ziehen kann, weil der Stein auf dem Endknoten liegt. Der Spieler, welcher in einem G-Knoten beginnt, besitzt eine Gewinnstrategie: er braucht den Stein nur in einen V-Knoten zu schieben (was nach Eigenschaft (2) immer geht); der Nachziehende muß anschließend den Stein in einen G-Knoten bewegen (nach (3)).

Können wir nun wieder - wie beim Nim-Spiel - an den Spielstellungen unmittelbar erkennen, ob sie G- oder V-Stellungen sind? Zweifellos lassen sich (beispielsweise) die V-Stellungen (o o) und (oo oo) in unserem

früheren Sinne als 'ausgewogen' bezeichnen; die Ausgewogenheit erkennen wir wieder an den Nim-Summen

$$\begin{array}{r} \mathrm{I} \\ \underline{\mathrm{I}} \\ 0 \end{array} \qquad \begin{array}{r} \mathrm{IO} \\ \underline{\mathrm{IO}} \\ 00 \end{array}$$

Aber auch z.B. (o oooo) ist eine V-Stellung, denn alle von dieser Stellung ausgehenden Pfeile führen auf G-Stellungen, wie der folgende Graph zeigt (wir schreiben künftig 1,4 statt o oooo etc.):



Auch ist (beispielsweise) die Stellung (2,4,6) eine V-Stellung (wie man leicht durch vollständige Fallunterscheidung sieht). Wenn wir unser Ausgewogenheitskriterium ("Nim-Summe null") retten wollen, dürfen wir nicht die Dualdarstellungen der Kegel-Anzahlen nehmen, sondern müssen andere die Stellungen kennzeichnende Zahlen finden.

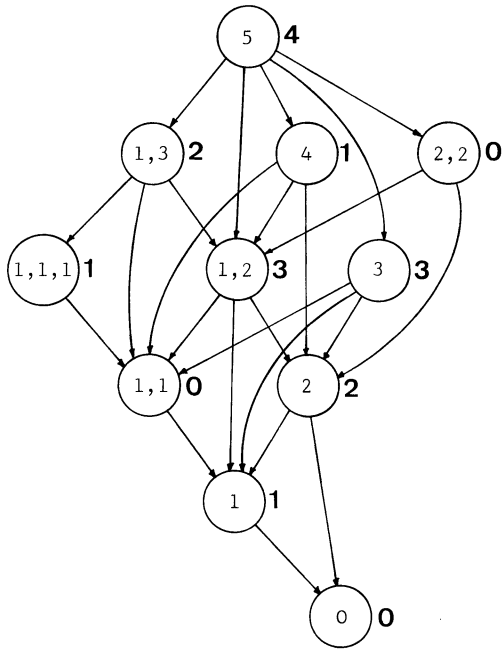
Die entscheidende Idee hierfür hatte der Berliner Mathematiker Roland Sprague (1936, und unabhängig von ihm P.M. Grundy 1939); er ordnet jeder Stellung (jedem Knoten des Spielgraphen) eine natürliche Zahl als 'Rang' folgendermaßen zu:

(1) Der (oder die) Endknoten erhält (erhalten) den Rang null.

(2) Der Rang eines Knotens ist die kleinste natürliche Zahl unter den Zahlen, die nicht Rang eines unmittelbaren Nachfolgers dieses Knotens sind.

(Dabei heißt Knoten Q ein unmittelbarer Nachfolger von P, wenn ein Pfeil von P zu Q führt.)

Insbesondere gilt: der Rang eines Knotens ist verschieden vom Rang aller seiner unmittelbaren Nachfolger.
Mit Hilfe der Spragueschen Regeln (1) und (2) lassen sich nun den Knoten unseres Spielgraphen wie folgt Ränge zuordnen (sie sind jeweils rechts neben den Knoten geschrieben):



Hieran erkennen wir: genau dann ist ein Knoten ein V-Knoten, wenn er den Rang null hat. Denn

(1) Endknoten haben den Rang null.

(2) Von einem Knoten mit Rang > 0 existiert stets ein Pfeil zu einem Knoten mit Rang 0. (Gäbe es nämlich keinen solchen Pfeil, müßte der Knoten gemäß Sprague's Regel selbst den Rang 0 bekommen.)

(3) Jeder Pfeil von einem Knoten mit Rang 0 führt zu einem Knoten mit Rang > 0 (denn der Rang eines Knotens und irgendeines seiner unmit-

telbaren Nachfolger sind verschieden).

Wir finden: rang(1) = 1, rang(2) = 2, rang(3) = 3, rang(4) = 1, rang(5) = 4, rang(6) = 3.
Am Spielgraphen fällt uns aber noch etwas auf: es ist

$$\text{rang}(1,3) = 2, \quad \text{rang}(1) = 1, \quad \text{rang}(3) = 3 \quad \text{und} \quad 1 \oplus 3 = 2, \quad \text{also}$$

$$\text{rang}(1,3) = \text{rang}(1) \oplus \text{rang}(3).$$

Ferner:

$$\text{rang}(2,2) = 0, \quad \text{rang}(2) = 2 \quad \text{und} \quad 2 \oplus 2 = 0, \quad \text{also}$$

$$\text{rang}(2,2) = \text{rang}(2) \oplus \text{rang}(2).$$

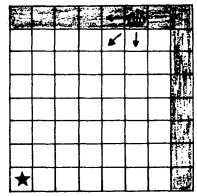
Diese Beziehung gilt allgemein (was wir hier nicht beweisen können): der Rang einer Spielstellung ist gleich der Nim-Summe der Ränge ihrer Komponenten. Eine Verluststellung ist also dadurch gekennzeichnet, daß die Nim-Summe der Ränge ihrer Komponenten null ist.
Wegen $1 \oplus 2 \oplus 3 = 0$ ist also (2,4,6) eine Verluststellung (beispielsweise). Um die Verluststellungen zu identifizieren, benötigen wir also nur eine Tabelle der Ränge folgender Art:

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
rang(n)	0	1	2	3	1	4	3	2	1	4	2	6	4	1	2	7	1	4

Die Ränge heißen auch Sprague-Grundy-Nummern und die Funktion, welche einer Stellung ihre Sprague-Grundy-Nummer zuordnet, heißt Sprague-Grundy-Funktion.
Wir wissen nun, daß es (beispielsweise) zur Stellung (2,4,5) eine Stellung mit Rang null gibt, die in einem Zuge erreichbar ist, und die wir unserem Spielgegner vorlegen müssen, wollen wir gewinnen - aber wie man sie findet,wissen wir noch nicht. So einfach wie beim Nim-Spiel geht es leider nicht; ich kann dem Leser im Moment nur folgendes Verfahren vorschlagen: man erzeuge alle unmittelbaren Nachfolger einer gegebenen Stellung und durchsuche sie nach einer Stellung vom Rang null. Die Theorie sagt uns, daß das Verfahren immer abbricht; das Programm sei dem Leser zur Übung überlassen!

Beispiel 3 : Treib' die Dame in die Ecke

Spieler A setzt die Dame auf eines der Felder der obersten Zeile oder der am weitesten rechts befindlichen Spalte (schraffiert). Die Dame kann wie üblich bewegt werden, aber nur in die Richtungen West, Südwest, Süd.
B hat den ersten Zug, dann wird abwechselnd gezogen. Wer nicht mehr ziehen kann, weil die Dame im Eck ★ steht, hat verloren.
Eine Gewinnstrategie ist gesucht.



Wir bauen von der Endstellung (0,0) her die Sprague-Grundy-Funktion und eine Gewinn-Verlust-Zerlegung auf. Zunächst stellen wir fest, daß alle Positionen (0,1), (0,2), (0,3), ... , (1,0), (2,0), (3,0), ... sowie die Diagonale (1,1), (2,2), (3,3), ... Gewinnstellungen sind (für den Spieler, der am Zuge ist), ihre Sprague-Grundy-Nummern sind:

5	-					
4	-					
3	-					
2	[0]	-				
1	2	[0]	-	-	-	
0	1	2	3	4	5	

Die beiden ersten Verluststellungen sind (2,1) und (1,2); ihre Sprague-Grundy-Nummern sind 0 (siehe oben). Gehen wir rekursiv weiter, so erhalten wir folgendes Tableau (die Verluststellungen sind eingerahmt):

8	6	7	10	1	2	5	3	4
7	8	6	9	[0]	1	4	5	3
6	7	8	1	9	10	3	4	5
5	3	4	[0]	6	8	10	1	2
4	5	3	2	7	6	9	[0]	1
3	4	5	6	2	[0]	1	9	10
2	[0]	1	5	3	4	8	6	7
1	2	[0]	4	5	3	7	8	6
0	1	2	3	4	5	6	7	8

Hätten wir das Tableau weiter fortgeführt, so hätten wir folgende Verluststellungen x(n), y(n) gefunden:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14
x(n)	1	3	4	6	8	9	11	12	14	16	17	19	21	22
y(n)	2	5	7	10	13	15	18	20	23	26	28	31	34	36

Erkennen wir eine mathematische Regel? Offensichtlich gilt

(1) $$y(n) = x(n) + n$$

(2) $$x(y(n)) = x(n) + y(n) \ ;$$

beispielsweise ist $y(2) = 5$, $x(5) = 8 = x(2) + y(2)$, also

$$x(y(2)) = x(2) + y(2).$$

Bemerkung: Wir haben nur die Koordinaten der Verluststellungen links der Diagonale angeschrieben; die übrigen liegen symmetrisch zu ihr, d.h. x(n) und y(n) sind vertauscht.-

Damit lassen sich die Verluststellungen numerisch leicht berechnen:

$x(1) = 1$

$x(n) =$ kleinste natürliche Zahl, die nicht unter den $x(k)$, $y(k)$ mit $k < n$ vorkommt

$y(n) = x(n) + n$

An den Folgen x(n) und y(n) ließen sich noch viele überraschende mathematische Entdeckungen machen; z.B. erkennt man Glieder der berühmten Fibonacci-Folge darin. Leider können wir aus Platzgründen nicht darauf eingehen; in den Aufgaben werden Ihnen jedoch weitere Hinweise und Anregungen gegeben werden.

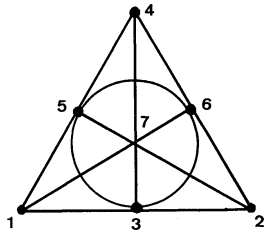
Aufgaben

1. Erweitern Sie das Programm für das Nim-Spiel auf eine beliebige Anzahl von Schalen.

2. Ändern Sie das Programm 'Nim-Spiel' so ab, daß a) die Anfangszahl der Früchte in den Schalen durch Zufall bestimmt wird und b) der anziehende Spieler zufällig bestimmt wird.

3. Zeigen Sie, daß die Nim-Summe der Früchteanzahlen beim Nim-Spiel die Sprague-Grundy-Nummer der jeweiligen Spielstellung ist.

4. Zeigen Sie, daß in nebenstehendem Dreieck bei allen Tripeln, deren Ziffern auf einer Geraden liegen, jede Ziffer die Nim-Summe der beiden übrigen ist (Beispiel: $2 \oplus 5 = 7$, $2 \oplus 7 = 5$ und $5 \oplus 7 = 2$). Dies Wissen ist beim Nim-Spiel nützlich - warum?



5. Zeigen Sie, daß die Menge der natürlichen Zahlen bezüglich der Nimaddition eine Gruppe bildet (!). Stellen Sie insbesondere eine Verknüpfungstabelle auf; ihr Anfang sieht so aus:

$\oplus$	0	1	2	3	4	5	6	7	8
0	0	1	2	3	4	5	6	7	8
1	1	0	3	2	5	4	7	6	9
2	2	3	0	1	6	7	4	5	10
3	3	2	1	0	7	6	5	4	11
4	4	5	6	7	0	1	2	3	12
5	.....								

6. Collier
Auf dem Tisch liegen n Münzen im Kreis; jede Münze berührt ihre beiden Nachbarn. Die beiden Spieler nehmen abwechselnd aus dem Kreis eine oder zwei benachbarte Münzen weg; wer nicht mehr ziehen kann (weil keine Münzen mehr da sind), hat verloren. Zeigen Sie, daß es für dieses Spiel eine 'Symmetriestrategie' gibt, die in folgendem besteht: beim zweiten Zug verwandelt der nachziehende Spieler

die offene Kette in zwei gleichlange Teilketten. Dann macht er stets an der anderen Teilkette das nach, was der erste Spieler an seiner Teilkette vormacht. Wer gewinnt folglich (bei dieser Strategie)? Schreiben Sie ein Programm!

7. Nim ohne Wiederholung
Dies Spiel hat die gleichen Spielregeln wie Nim (mit e i n e m Haufen), jedoch mit der Einschränkung, daß der jeweils letzte Zug des Gegners nicht wiederholt werden darf. Stellen Sie die Sprague-Grundy-Funktion auf und entwickeln Sie eine Gewinnstrategie!

8. Schreiben Sie ein Programm für das Kegelspiel.

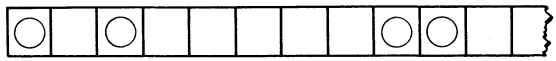
9. Schreiben Sie ein Programm für das Spiel "Treib' die Dame in die Ecke"!

10. Steine nehmen
Ein altes chinesisches Spiel heißt "Steine nehmen" (Tsian-schi-dzi) und geht wie folgt: gegeben sind zwei Haufen mit Spielsteinen; die Spieler dürfen abwechselnd vom einen oder vom anderen Haufen soviele Steine nehmen, wie sie möchten, oder von beiden Haufen zugleich, aber jeweils gleichviel Steine.
Zeigen Sie, daß dies Spiel im wesentlichen (d.h. bis auf die Bezeichnungen) mit "Treib' die Dame in die Ecke" übereinstimmt.
(Das Spiel "Steine nehmen" wurde im Jahr 1907 von dem Holländer W.A. Wythoff wieder erfunden und heißt daher auch "Wythoffs Nim".)

11. Ein Haufen mit n Hölzchen (Steinen oder Früchten) ist gegeben. Der anziehende Spieler nimmt soviele weg, wie er möchte (außer dem ganzen Haufen). Von nun an wechseln die Spieler ab, indem sie eines oder mehrere Hölzchen nehmen - aber jeweils höchstens doppelt so viele, wie der Gegenspieler soeben weggenommen hat.
Stellen Sie die Sprague-Grundy-Funktion auf und entwickeln Sie für den Computer eine Gewinnstrategie.

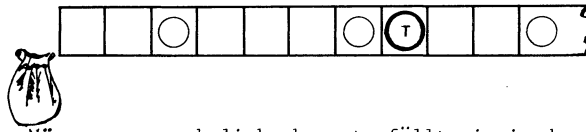
12. Laskers Nim
Der langjährige Schachweltmeister Emanuel Lasker hat auch ein Buch "Brettspiele der Völker"(Berlin 1931) verfaßt; darin beschreibt er folgende Variante des Nim-Spiels: "Der am Zuge Befindliche darf irgendeinen der Haufen in zwei Haufen zerteilen oder aber, nach seinem freien Ermessen, verkleinern." Gewonnen hat, wer den letzten Zug tut. Lasker vermochte einige Verluststellungen zu berechnen, und schreibt dann: "Das Gesetz der Verluststellungen habe ich nicht gefunden; es scheint mir aber, daß es mit dem des Nim-Spiels verwandt ist."
Sie kennen das mächtige Hilfsmittel der Sprague-Grundy-Funktion und können das 'Gesetz der Verluststellungen' aufstellen und eine Gewinnstrategie entwickeln.

13. Auf einer beliebig langen Felderreihe liegen getrennt n Steine. Zwei Personen ziehen abwechselnd jedesmal einen beliebigen Stein auf irgend ein freies, dem Anfang der Reihe näheres Feld.
Überspringen ist a) nicht gestattet b) gestattet.
Wer nicht mehr ziehen kann, weil die Steine auf den ersten n Feldern liegen, hat verloren.



Dies Spiel heißt "die wandernden Steine"; im Fall b) auch "Welters' Spiel". Zeigen Sie, daß es eine Variante des Nim-Spiels ist, stellen Sie die Sprague-Grundy-Funktion auf und geben Sie eine Gewinnstrategie an, die einem Computer beigebracht werden kann.

14. Geldsack
Dies Spiel geht wie die "wandernden Steine" - es wird jedoch mit Münzen und einem Silbertaler gespielt. Links ist ein Geldsack angebracht; gezogen wird wie in Aufgabe 13.



Wurde eine Münze ganz nach links bewegt, fällt sie in den Sack; gewonnen hat, wem es gelingt, den Silbertaler im Geldsack unterzubringen.

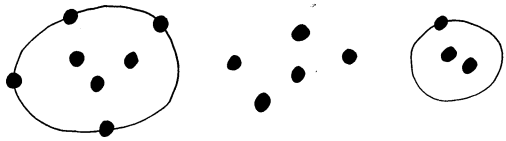
15. Prim-Nim, Prim und Dim
Prim-Nim ist Nim mit der Einschränkung, daß die Haufen nur durch Primzahlen verändert werden dürfen (die Anzahlen der wegzunehmenden Hölzchen, Steine oder Früchte müssen Primzahlen sein); auch die 1 gilt hier als Primzahl.
Bei Prim müssen die Anzahlen der wegzunehmenden Gegenstände zur Haufenzahl relativ prim (teilerfremd) sein. Beispiel: liegen auf einem Haufen 8 Gegenstände, so dürfen nur 1, 3, 5 oder 7 davon weggenommen werden.
Beim Spiel "Dim" dagegen muß die Anzahl der wegzunehmenden Gegenstände ein Teiler (engl./lat.: divisor) der Haufenanzahl sein.

Ermitteln Sie die Sprague-Grundy-Funktionen sowie jeweils eine Gewinnstrategie und schreiben Sie das zugehörige Programm.

16. Nim mit Schranke
Von E.H. Moore stammt die Einschränkung des Nim-Spiels, daß die Anzahl der wegzunehmenden Gegenstände eine Zahl k nicht überschreiten darf. Zeigen Sie, daß eine Stellung V-Stellung ist, wenn ihre Nim-Summe bei Division durch k+1 den Rest null läßt. Bauen Sie darauf eine Gewinnstrategie und ein Programm auf.

17. Distichon (Grundys Spiel)
Wie beim Nim ist ein(e) (oder sind mehrere) Haufe(n) mit Hölzchen (Schalen mit Früchten) gegeben. Ein Spielzug besteht darin, einen der Haufen in zwei u n g l e i c h e Teile zu zerlegen. Verloren hat, wer nicht mehr ziehen kann (weil es nichts mehr zu teilen gibt). Stellen Sie die Sprague-Grundy-Funktion auf und entwickeln Sie für den Computer eine Gewinnstrategie.

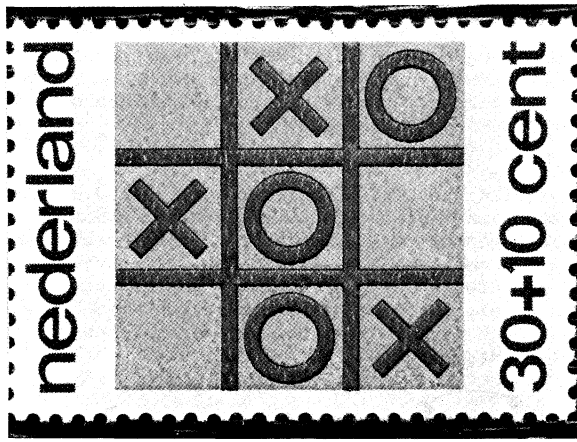
18. Rims
Die Anfangsstellung besteht bei diesem Spiel von J.H. Conway aus einigen Punkten auf Papier. Ein Spielzug besteht darin, einen einfach geschlossenen Bogen durch einige dieser Punkte zu ziehen; der Bogen darf weder sich selbst noch einen anderen Bogen kreuzen oder berühren. Zeigen Sie, daß dies Spiel zum Nim-Spiel äquivalent ist - mit der zusätzlichen Regel, daß durch Herausnahme von Hölzchen aus dem Innern eines Haufens auch neue Haufen gebildet werden können.
Beispielsweise entspricht die Rims-Stellung



der Stellung (3,5,2) im Nim-Spiel.
Wenn jeder Bogen durch genau einen oder zwei der Punkte gehen muß, so haben wir eine zum Kegelspiel (Beispiel 2) äquivalente Fassung.

19. Zweidimensionales Nim
Von D. Gale stammt folgende Nim-Version: die Spielsteine liegen auf einem Rechteck mit m Zeilen und n Spalten. Ein Spielzug besteht darin, zunächst eine Position io, jo zu wählen (auf ihr muß sich ein Stein befinden) und dann alle Steine mit Positionen $i \geq io$ und $j \geq jo$ zu entfernen. Wer die Position 1,1 wählen muß, verliert!

20. Tac-tix
Der Däne Piet Hein erfand folgende Abwandlung des Nim-Spiels: wie beim zweidimensionalen Nim liegen die wegzunehmenden Steine auf einem m x n - Rechteck. Die Spieler nehmen abwechselnd Steine einer Zeile (waagerecht) oder eine Spalte (senkrecht) weg - mit der einzigen Einschränkung, daß die wegzunehmenden Steine b e n a c h b a r t sein müssen.
Zeigen Sie, daß das normale Spiel (verloren hat, wer nichts mehr wegnehmen kann) eine einfache Gewinnstrategie besitzt und somit trivial ist. Analysieren Sie die abweichende Spielform (Misere-Form) für den Fall eines 3 x 3 - Quadrats und entwickeln Sie eine Gewinnstrategie!

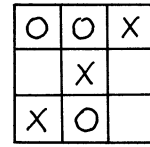


Die bisher untersuchten Spiele hatten die Eigenschaft, daß jeder Spielzug auch prinzipiell dem Gegenspieler offenstand; bei Mühle, Dame oder Schach (beispielsweise) ist dies jedoch nicht der Fall: hier darf der Spieler nur Figuren der eigenen Farbe bewegen. Damit trifft die bisher entwickelte Theorie nicht mehr zu.-
Eines der ältesten Spiele heißt 'Kringel und Kreuze' (engl.: Tic-tac-toe). Man hat es auf ägyptischen Grabmalereien und griechischen Vasen gefunden; König Tut-ench-Amon, Sokrates und Elisabeth I hätten alle gleichermaßen die Figur oben auf der Briefmarke erkannt und damit zu spielen gewußt. Charles Babbage, einer der Väter der Computerei, entwickelte sechs verschiedene Maschinen zum Tic-tac-toe-Spielen; es existieren zahllose Strategien und Computerprogramme dafür. Obwohl das Spiel sehr leicht zu begreifen und zu spielen ist, stellt uns seine Programmierung vor keine einfache Aufgabe.

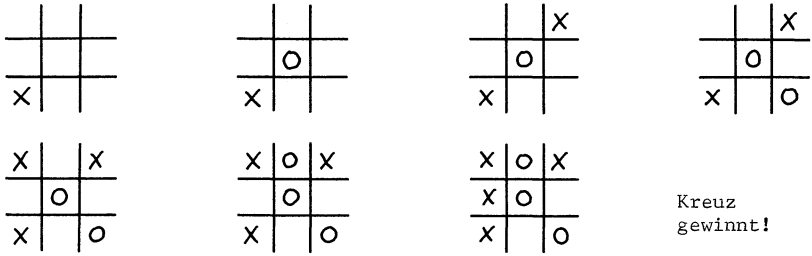
Schauen wir uns nun die Regeln an!

Beispiel 4 : Kringel und Kreuze

Die Spieler 'Kringel' (O) und 'Kreuz' (X) belegen abwechselnd die Felder eines 3 x 3 - Quadrats mit dem Ziel, zuerst drei Exemplare der eigenen Figur in eine Zeile (waagerecht), Spalte (senkrecht) oder Diagonale (schräg) zu bringen.
Der Computer soll Spieler sein.



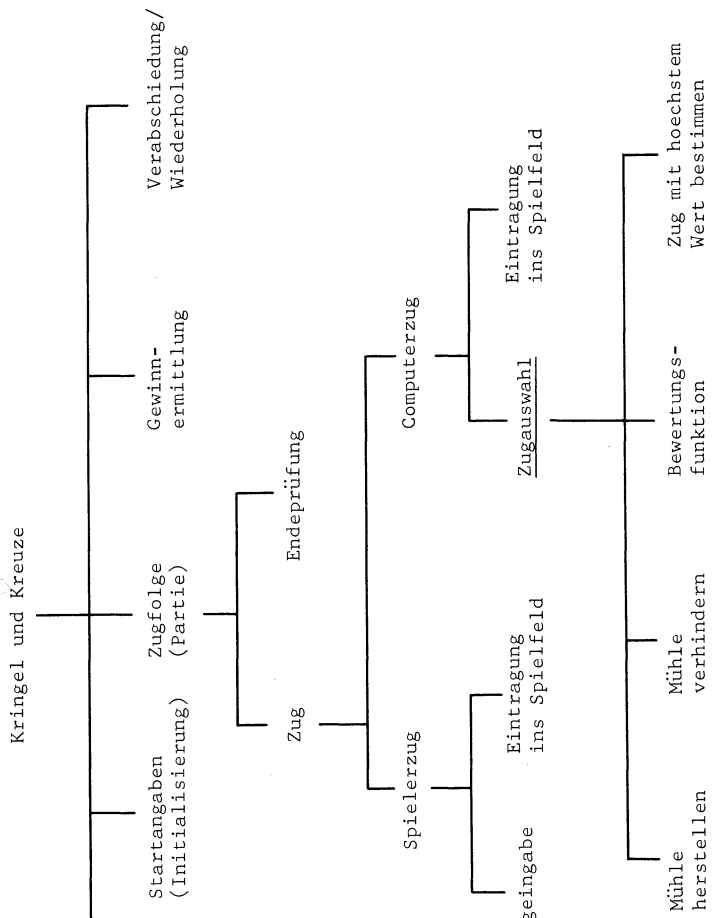
Eine mögliche Partie sieht so aus:



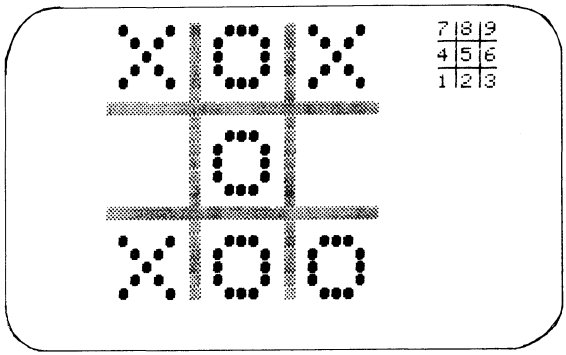
Auf der folgenden Seite findet sich die Grobstruktur des zu entwerfenden Programms; sie sieht wie bei jedem Spiel aus. Interessant ist allein der Teil "Zugauswahl", denn er enthält die vom Computer anzuwendende Strategie. Es geht darum, jeder Spielstellung einen Computerzug zuzuordnen.
Zu diesem Zweck muß man die Ziele (nach Dringlichkeit geordnet) aufstellen:

(1) Eine Mühle erzeugen, d.h. eine Reihe (Zeile, Spalte oder Diagonale) mit eigenen Figuren zu besetzen (drei Kringeln).
(2) Eine Mühle des Gegenspielers verhindern.
(3) Eine Reihe mit zwei eigenen Figuren zu besetzen.
(4) Dies beim Gegenspieler verhindern.

Die ersten beiden Teilziele lassen sich leicht programmieren: wir prüfen, ob es eine Reihe mit 'Loch' (d.h. zwei gleiche Figuren, ein leeres Feld) gibt und füllen dies Loch dann mit einer eigenen Figur (dem Kringel des Computers). Die weiteren Ziele bringen wir in eine sogenannte Bewertungsfunktion ein, d.i. eine Vorschrift, die jeder Spielstellung eine Zahl (ihren 'Wert' im Hinblick auf die Ziele) zuordnet.


Kringel und Kreuze
Begrüßung und Spielbeschreibung
Startangaben (Initialisierung)
Zugfolge (Partie)
Gewinn-ermittlung
Verabschiedung/ Wiederholung
Zug
Endeprüfung
Spielerzug
Computerzug
Zugeingabe
Eintragung ins Spielfeld
Zugauswahl
Eintragung ins Spielfeld
Mühle herstellen
Mühle verhindern
Bewertungs-funktion
Zug mit hoechstem Wert bestimmen

Wir lassen den Computer denjenigen Zug ausführen, bei dem die Bewertungsfunktion ihren höchsten Wert annimmt.
Woran erkennt der Rechner nun, ob eine Mühle droht bzw. erreichbar ist? Ganz einfach: wir bilden eine Tabelle T, in der alle 8 Mühlen gespeichert sind; ferner eine Tabelle U mit weiteren interessanten Spielstellungen. Spielen Sie mit dem Programm, ändern Sie die Tabelle U ab und studieren Sie, wie sich die Bewertung des Computers ändert!
Die Besetzung des Spielfelds wird in einer Tabelle B mit 9 Feldern festgehalten; der Stellung



entspricht die Tabelle

1	2	3	4	5	6	7	8	9	
-1	1	1	0	1	0	-1	1	-1	B

Nach dem Zug des Spielers bzw. des Computers wird 1 oder -1 in die Tabelle eingetragen; den Wert bestimmt die 'Flagge' F : sie wird auf -1 gesetzt, wenn der menschliche Spieler am Zuge und auf 1, wenn der Computer dran ist.-
Das Programm ist (hoffentlich) so übersichtlich aufgebaut, daß Sie es verstehen werden; die Bewertungsfunktion können Sie sich leicht an ausgewählten Beispielen klarmachen.

```
1000 : PRINT "□
1010 : PRINT "          KRINGEL UND KREUZE
1011 : PRINT "          ------------------
1012 :
1020 REM DER COMPUTER SPIELT TIC-TAC-TOE (MIT KRINGEL)
1021 :
1022 :
1200 REM *** HAUPTPROGRAMM *********************************************
1201 :
1220 : GOSUB 2000 : REM BEGRUESSUNG UND SPIELBESCHREIBUNG
1221 :
1230 : GOSUB 3000 : REM INITIALISIERUNG
1231 :
1240 : GOSUB 4000 : REM SPIELFELDAUSGABE
1241 :
1250 : GOSUB 5000 : REM ZUGFOLGE
1251 :
1280 : GOSUB 8000 : REM GEWINNERMITTLUNG
1281 :
1290 : GOSUB 9000 : REM VERABSCHIEDUNG ODER WIEDERHOLUNG
1291 :
1900 : END
1901 :
1990 REM *** ENDE DES HAUPTPROGRAMMS ***********************************
1991 :
1998 :
1999 :
2000 REM ### PROZEDUR 'BEGRUESSUNG UND SPIELBESCHREIBUNG' ##############
2001 :
2100 : PRINT : PRINT
2110 : INPUT "WIE LAUTET IHR VORNAME "; N$
2120 : PRINT
2130 : PRINT "GUTEN TAG, "; N$;" !
2140 : PRINT "■MOECHTEN SIE DIE SPIELREGELN KENNENLERNEN ? (J/N)
2150 : GET A$ : IF A$ = "" THEN 2150
2160 : IF A$ <> "J" THEN RETURN
2190 :
2200 : REM HIER SOLLTEN DIE SPIELREGELN STEHEN
2201 :
2900 : RETURN
2901 :
2990 REM ### ENDE DER BEGRUESSUNG UND SPIELBESCHREIBUNG ###############
2998 :
2999 :
3000 REM ### PROZEDUR 'INITIALISIERUNG' ################################
3001 :
3100 : REM --- DIMENSIONIERUNG DER TABELLEN
3101 :
3110 :   DIM T(8,3) : REM TABELLE DER MUEHLEN
3120 :   DIM U(9,4) : REM TABELLE SPEZIELLER SPIELSTELLUNGEN
3130 :   DIM C(9)   : REM ZAEHLER FUER FIGUREN GLEICHER ART
3140 :   DIM B(9)   : REM BESETZT-TABELLE (1 = KREUZ, -1 = KRINGEL)
3160 :   DIM V(9)   : REM BEWERTUNGSTABELLE
3170 :   DIM C$(2)  : REM KREUZ UND KRINGEL
3180 :   DIM L(9)   : REM KOORDINATEN DER FELDER
3190 :
```

```
3200 : REM --- EINLESEN VON SPIELKONSTELLATIONEN
3201 :
3210 :   FOR I = 1 TO 8
3220 :     FOR J = 1 TO 3 : READ T(I,J) : NEXT J
3230 :   NEXT I
3240 :
3250 DATA 7,8,9,4,5,6,1,2,3 : REM WAAGERECHTE MUEHLE
3260 DATA 7,4,1,8,5,2,9,6,3 : REM SENKRECHTE MUEHLE
3270 DATA 7,5,3,1,5,9       : REM DIAGONALE MUEHLE
3290 :
3310 :   FOR I = 1 TO 9
3320 :     FOR J = 1 TO 4 : READ U(I,J) : NEXT J
3330 :   NEXT I
3340 :
3350 DATA 3,4,8,0,3,5,0,0,3,6,7,0
3360 DATA 2,4,0,0,2,5,7,8,2,6,0,0
3370 DATA 1,4,7,0,1,5,0,0,1,6,8,0
3390 :
3400 : REM --- KREUZ UND KRINGEL
3401 :
3410 :   LET C$(1) = "●▣●▣●▣●▣●█████●□●□●□●□"
3420 :   LET C$(2) = "▣●█●██ ▣●▣●▣●▣●●●□●█□●█□●██□● "
3490 :
3500 : REM --- KOORDINATEN DER NEUN FELDER
3501 :
3510 :   FOR I = 1 TO 9 : READ L(I) : NEXT I
3520 :
3530 DATA 1608,1616,1624,808,816,824,8,16,24
3590 :
3600 : REM --- SPIELFELDBALKEN
3601 :
3610 :   LET B1$ = "               ▒       ▒"
3620 :   LET B2$ = "         ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
3690 :
3700 : REM ---- SPIELFELD LEEREN
3701 :
3710 :   FOR I = 1 TO 9 : LET B(I) = 0 : LET C(I) = 0 : NEXT I
3720 :
3790 :
3800 : REM --- ZEICHENKETTE ZUR CURSORSTEUERUNG
3801 :
3810 :   LET D$ = "▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣"
3820 :
3900 : RETURN
3901 :
3990 REM ### ENDE DER INITIALISIERUNG ################################
3998 :
3999 :
4000 REM ### PROZEDUR 'SPIELFELDAUSGABE' ############################
4001 :
4100 : REM --- BALKEN
4101 :
4110 :   PRINT "□";
4111 :
4200 :   FOR J = 1 TO 6 : PRINT B1$ : NEXT J
4220 :   PRINT B2$
4240 :   FOR J = 1 TO 7 : PRINT B1$ : NEXT J
4260 :   PRINT B2$
4280 :   FOR J = 1 TO 6 : PRINT B1$ : NEXT J
4290 :   PRINT B1$
4299 :
```

```
4400 :    PRINT "S";
4401 :
4500 : REM --- KLEINES SPIELFELD RECHTS OBEN
4501 :
4510 :    FOR J = 1 TO 34 : PRINT "U";: NEXT J
4520 :    PRINT "7|8|9S-----H-|-|-S-----H4|5|6S-----H-|-|-S-----H1|2|3"
4530 :
4540 :    PRINT "S"; D$;
4590 :
4900 : RETURN
4901 :
4990 REM ### ENDE DER SPIELFELDAUSGABE ################################
4998 :
4999 :
5000 REM ### PROZEDUR 'ZUGFOLGE' ######################################
5001 :
5100 : REM = PROZEDURRUMPF 'ZUGFOLGE' ================================
5101 : :
5110 : : FOR N = 1 TO 5
5120 : :
5130 : :    GOSUB 5400                    : REM SPIELERZUG
5140 : :
5150 : :    GOSUB 7000                    : REM ENDEPRUEFUNG
5160 : :
5170 : :    IF E$ = "FERTIG" THEN RETURN  : REM PARTIE BEENDET
5180 : :
5200 : :    GOSUB 6000                    : REM COMPUTERZUG
5210 : :
5220 : :    GOSUB 7000                    : REM ENDEPRUEFUNG
5230 : :
5240 : :    IF E$ = "FERTIG" THEN RETURN  : REM PARTIE BEENDET
5250 : :
5260 : : NEXT N
5270 : :
5280 : : RETURN
5281 : :
5290 : REM = ENDE PROZEDURRUMPF 'ZUGFOLGE' ===========================
5299 :
5400 : REM +++ UNTERPROZEDUR 'SPIELERZUG' ++++++++++++++++++++++++++++
5401 :
5410 :    REM - PROZEDURRUMPF 'SPIELERZUG' ----------------------------
5411 :    :
5415 :    :  LET F = -1 : REM FLAGGE FUER SPIELER
5420 :    :  GOSUB 5500 : REM ZUGEINGABE
5430 :    :  GOSUB 6800 : REM EINTRAGUNG INS SPIELFELD
5450 :    :  RETURN
5460 :    :
5490 :    REM - ENDE PROZEDURRUMPF 'SPIELERZUG' ------------------------
5499 :
```

```
5500 :   REM +++ UNTERPROZEDUR 'ZUGEINGABE' ++++++++++++++++++++++++++
5501 :
5510 :     PRINT "■"; D$; "   IHR ZUG ? (ZIFFER 1 BIS 9)
5520 :     GET Z$ : IF Z$ = "" THEN 5520
5525 :
5530 :     IF Z$ < "1" OR Z$ > "9" THEN 5510
5535 :
5540 :     LET Z = VAL(Z$)
5560 :     IF B(Z) <> 0 THEN 5510 : REM FELD IST SCHON BESETZT
5570 :     RETURN
5580 :
5590 :   REM +++ ENDE DER ZUGEINGABE ++++++++++++++++++++++++++++++++++
5599 :
5690 : RETURN : REM +++ ENDE DES SPIELERZUGES ++++++++++++++++++++++++
5999 :
6000 : REM +++ UNTERPROZEDUR 'COMPUTERZUG' ++++++++++++++++++++++++++
6001 :
6100 :   REM - PROZEDURRUMPF 'COMPUTERZUG' --------------------------
6101 :   :
6110 :   :  LET F = 1   : REM FLAGGE FUER COMPUTER
6120 :   :  GOSUB 6200 : REM ZUGAUSWAHL
6130 :   :  GOSUB 6800 : REM EINTRAGUNG INS SPIELFELD
6140 :   :  RETURN
6150 :   :
6160 :   REM - ENDE PROZEDURRUMPF 'COMPUTERZUG' ----------------------
6190 :
6200 :   REM +++ UNTERPROZEDUR 'ZUGAUSWAHL' ++++++++++++++++++++++++++
6201 :
6210 :     REM --- MUEHLE HERSTELLEN
6211 :
6220 :       FOR I = 1 TO 8
6230 :         IF C(I) = 2 THEN LET I1 = I : GOTO 6400
6240 :       NEXT I
6250 :
6260 :     REM --- MUEHLE VERHINDERN
6261 :
6270 :       FOR I = 1 TO 8
6280 :         IF C(I) = -2 THEN LET I1 = I : GOTO 6400
6290 :       NEXT I
6291 :
6300 :       GOTO 6500 : REM ZUR BEWERTUNG
6301 :
6400 :     REM --- LOCH FUELLEN
6401 :
6410 :       FOR J = 1 TO 3
6420 :         LET Z = T(I1,J)
6430 :         IF B(Z) = 0 THEN RETURN : REM LOCH GEFUNDEN
6440 :       NEXT J
6480 :
6500 :     REM --- BEWERTUNGSFUNKTION
6501 :
6510 :       FOR I = 1 TO 9
6520 :         LET V(I) = 0
6530 :         IF B(I) <> 0 THEN 6580
6540 :         FOR J = 1 TO 4
6550 :           LET P = U(I,J)
6560 :           IF P > 0 THEN LET V(I) = V(I) + ABS(C(P)) + 1
6570 :         NEXT J
6580 :       NEXT I
6590 :
```

```
6600 :      REM --- ZUG MIT HOECHSTEM WERT BESTIMMEN
6601 :
6610 :        LET V = 0
6620 :        FOR I = 1 TO 9
6630 :          IF V(I) <= V THEN 6660
6640 :          LET V = V(I) : REM HOECHSTER WERT
6650 :          LET Z = I    : REM ZUG MIT HOECHSTEM WERT
6660 :        NEXT I
6670 :
6790 : RETURN : REM ENDE DER ZUGAUSWAHL +++++++++++++++++++++++++++++++++
6799 :
6800 : REM +++ UNTERPROZEDUR 'EINTRAGUNG INS SPIELFELD' ++++++++++++++++
6801 :
6810 :    LET B(Z) = F : REM BESETZEN DES FELDES NR.Z MIT 1 ODER -1
6811 :
6820 :    PRINT "■";
6830 :    LET I = INT(L(Z)/100) : REM ZEILE
6840 :    LET J = L(Z) - 100*I  : REM SPALTE
6850 :    FOR I1 = 1 TO I : IF I > 0 THEN PRINT "■";: NEXT I1
6860 :    FOR J1 = 1 TO J : PRINT "■";: NEXT J1
6870 :    PRINT C$((B(Z)+3)/2)  : REM KRINGEL ODER KREUZ
6880 :
6890 : RETURN : REM ENDE DER EINTRAGUNG INS SPIELFELD +++++++++++++++++
6899 :
7000 : REM +++ UNTERPROZEDUR 'ENDEPRUEFUNG' ++++++++++++++++++++++++++
7001 :
7200 :    REM --- MUEHLE GEFUNDEN?
7201 :
7210 :      FOR J = 1 TO 4
7220 :        LET P = U(Z,J)
7230 :        IF P = 0 THEN 7270
7240 :        LET C(P) = C(P) + F
7250 :        LET C = C(P)
7260 :        IF C = 3 OR C = -3 THEN LET E$ = "FERTIG": RETURN
7270 :      NEXT J
7290 :
7300 :    REM --- ALLE FELDER BESETZT?
7301 :
7310 :      FOR I = 1 TO 9
7320 :        IF B(I) = 0 THEN RETURN : REM NOCH FREIES FELD DA
7330 :      NEXT I
7340 :
7350 :      LET E$ = "FERTIG"
7360 :
7390 :      RETURN
7391 :
7900 : REM ENDE DER ENDEPRUEFUNG ++++++++++++++++++++++++++++++++++++++
7901 :
7990 REM ### ENDE DER ZUGFOLGE ##########################################
7998 :
7999 :
```

```
8000 REM ### PROZEDUR 'GEWINNERMITTLUNG' ##############################
8001 :
8100 : PRINT "■"; D$;
8101 :
8200 : IF C = -3 THEN PRINT "      ■*** SIE HABEN GEWONNEN ***■":RETURN
8210 : IF C =  3 THEN PRINT "      ■*** SIE HABEN VERLOREN ***■":RETURN
8310 :                PRINT "         ■*** UNENTSCHIEDEN ***■
8900 : RETURN
8901 :
8990 REM ### ENDE DER GEWINNERMITTLUNG ###################################
8998 :
8999 :
9000 REM ### VERABSCHIEDUNG ODER WIEDERHOLUNG ###########################
9001 :
9100 : FOR I = 1 TO 5000 : NEXT I : REM WARTESCHLEIFE
9101 :
9200 : PRINT "■"
9210 : PRINT "■■■       NOCH EINE PARTIE?
9220 :
9230 : GET A$ : IF A$ = "" THEN 9230
9240 : IF A$ = "J" THEN RUN
9250 :
9260 : PRINT "■■■  ES WAR NETT, MIT IHNEN ZU SPIELEN.
9270 :
9900 : RETURN
9901 :
9990 REM ### ENDE DER VERABSCHIEDUNG ###################################
9998 :
9999 :
```

*

Aufgaben

22. Wenn Sie bei unserem 'Kringel-u.-Kreuze'-Programm zur Abwechslung einmal gegen den Computer gewinnen wollen, so ändern Sie die Bewertungsfunktion in Zeile 6560 ab; zum Beispiel, indem Sie die 1 streichen. Studieren Sie die Auswirkungen solcher Änderungen auf die Spielweise des Programms.

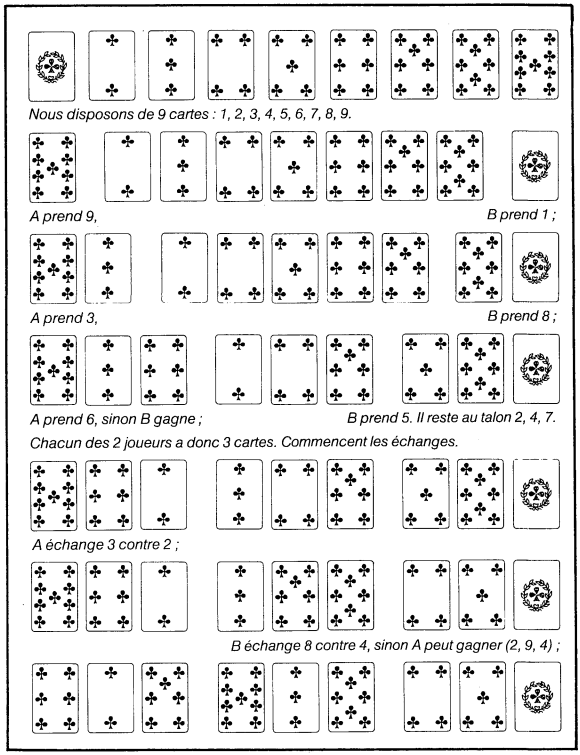
23. Erweitern Sie das Programm 'Kringel und Kreuze' so, daß der anziehende Spieler a) durch Zufall bestimmt wird, b) vom Benutzer bestimmt werden kann.

24. Schreiben Sie ein Programm zum Spielen von 'Kringel und Kreuze' für zwei Personen; der Computer ist dabei lediglich der Spielmeister.

25. Das auf der folgenden Seite notierte Tic-tac-toe-Programm ist nicht gut strukturiert und daher kaum verständlich. Versuchen Sie trotzdem, durchzusteigen und geben Sie dem Programm eine übersichtliche Struktur (Gliederung nach den Phasen eines Spiels).

```
100 PRINT "□
110 PRINT "            TIC-TAC-TOE
111 PRINT "            -----------
112 :
120 REM SCHLECHT STRUKTURIERTES PROGRAMM,
121 REM - ALS ABSCHRECKUNG GEDACHT
129 :
130 PRINT : PRINT
140 PRINT "COMPUTER SPIELT MIT '0', SIE SPIELEN MIT '11',
145 PRINT "JEDES FELD WIRD WAHLWEISE MIT EINER ZIFFER
150 PRINT "VON 1 BIS 9 GEKENNZEICHNET.
155 PRINT "DER COMPUTER SPIELT ALS ERSTER.
160 PRINT "IHR SPIELZUG BESTEHT IN DER EINGABE DER
165 PRINT "NUMMER DES FELDES, DAS SIE BESETZEN WOLLEN.
170 PRINT : PRINT
190 :
200 FOR I = 1 TO 9 : LET T(I) = I : NEXT I
210 LET B = 9
220 DEF FN A(X) = X - 8*INT((X-1)/8)
230 GOSUB 500
240 GOSUB 600
250 LET J = 0
260 LET J = J+1
270 IF J = 4 THEN 350
280 LET B = FNA(P+1)
290 GOSUB 500
300 GOSUB 600
310 IF P = FNA(B+4) THEN 260
320 LET B = FNA(B+4)
330 GOSUB 500
340 GOTO 370
350 LET B = FNA(P+5)
360 PRINT : PRINT "UNENTSCHIEDEN!": GOTO 380
370 PRINT : PRINT "DER COMPUTER HAT GEWONNEN."
380 PRINT : PRINT "AUF EIN NEUES? (J/N)
390 GET A$ : IF A$ = "" THEN 390
400 IF A$ = "J" THEN 200
410 PRINT
420 PRINT "         SCHADE!
430 END
490 :
500 REM UNTERPROGRAMM 1
510 LET T(B) = 0
520 PRINT TAB(18); T(1); TAB(24); T(2); TAB(30); T(3)
530 PRINT TAB(18); T(8); TAB(24);   0 ; TAB(30); T(4)
540 PRINT TAB(18); T(7); TAB(24); T(6); TAB(30); T(5)
550 RETURN
590 :
600 REM UNTERPROGRAMM 2
610 PRINT : INPUT "IHR ZUG "; P
620 IF P < 1 OR P > 9 THEN 610
630 LET T(P) = 11
640 PRINT
650 RETURN
```

26. Fünfzehn gewinnt
Auf dem Tisch liegen Spielkarten mit den Werten von 1 bis 9 (Farbe gleichgültig). Die beiden Spieler nehmen abwechselnd je drei Karten; das Spielziel ist, daß die Summe der eigenen Kartenwerte genau 15 beträgt. Nach der Aufnahme der drei Karten können die Spieler abwechselnd jeweils eine Karte mit den auf dem Tisch verbliebenen drei Karten austauschen. Wer zuerst die 15 erreicht, ist Sieger.
Eine typische Partie sieht so aus:



2	9	4
7	5	3
6	1	8

Es gewinnt der Spieler A.-
a) Zeigen Sie, daß dies Spiel mit 'Kringel und Kreuze' äquivalent ist. (Eine wesentliche Rolle spielt dabei das magische Quadrat oben rechts; vergleichen Sie Kapitel 8)
b) Schreiben Sie ein Spielprogramm!

27. Mosers Spiel
Gegeben sind die englischen Worte

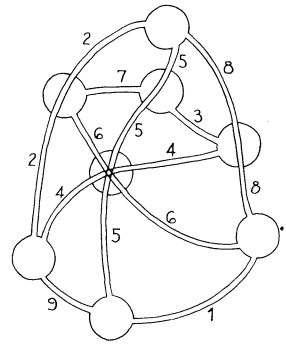
HOT TANK TIED FORM HEAR BRIM WOES WASP SHIP .

Die zwei Spieler streichen abwechselnd je ein Wort aus (und kennzeichnen es mit ihrem Namen). Wer als erster drei Worte mit genau einem gemeinsamen Buchstaben ausgestrichen hat, ist Sieger.

a) Zeigen Sie, daß es sich wieder um ein zu 'Kringel und Kreuze' äquivalentes Spiel handelt!

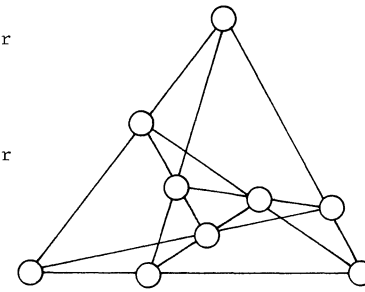
b) Finden Sie entsprechende deutsche Worte und schreiben Sie ein Programm.

28. Jam
Gegeben ist eine Straßenkarte, wie nebenstehend gezeichnet. Ein Spielzug besteht darin, eine der Straßen zu sperren, indem sie mit der dem Spieler eigenen Farbe in ihrer ganzen Länge angemalt wird. Wer als erster drei Straßen, die in eine Stadt einmünden mit seiner Farbe gekennzeichnet hat, ist Sieger.
Auch dies ist - in verkleideter Form - 'Kringel und Kreuze'!
Schreiben Sie ein Programm dafür.

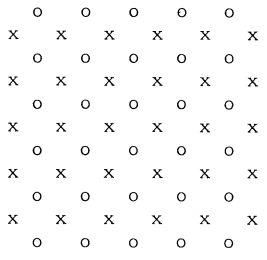


29. Wildes Tic-tac-toe
Es wird wie beim Tic-tac-toe gespielt - mit der Abweichung, daß jeder Spieler sowohl O als auch X spielen kann. Zeigen Sie, daß der Anziehende immer gewinnt.
Beim 'umgekehrten wilden Tic-tac-toe' v e r l i e r t derjenige Spieler, der eine Reihe (Zeile, Spalte oder Diagonale) aus drei gleichen Zeichen vollendet. Entwickeln Sie Spielstrategie und Programm.

30. Tri-Hex
Statt eines 3 x 3 - Quadrats haben wir jetzt nebenstehendes Spielfeld.
Gewonnen hat wieder, wer als erster drei eigene Zeichen in einer Reihe (auf der gleichen Geraden) hat.
Zeigen Sie, daß der Anziehende immer gewinnen kann, wenn er zuerst auf einen der Seitenmitten des äußeren Dreiecks setzt.
Benutzen Sie dies zur Entwicklung eines optimal spielenden Programms.

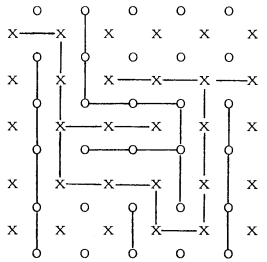


Beispiel 5: Brückenspiel



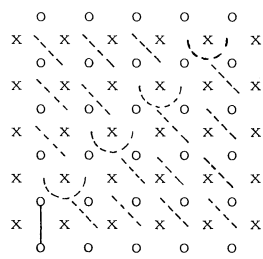
Das Spielfeld sieht wie nebenstehend abgebildet aus (die Anzahl der Zeilen bzw. Spalten kann natürlich noch größer gewählt werden).
Spieler A verbindet zwei benachbarte Kreise entweder durch eine senkrechte oder durch eine waagerechte Strecke; sein Ziel ist ein durchgehender Streckenzug vom nördlichen zum südlichen Ufer.
Spieler B tut das Entsprechende mit den Kreuzen; sein Ziel ist ein durchgehender Streckenzug zwischen dem östlichen und dem westlichen Ufer. Gegnerische Verbindungen dürfen nicht gekreuzt werden.
Wer sein Ziel als erster erreicht hat, ist Sieger.

Am Schluß einer typischen Partie sieht das Spielfeld etwa so aus:



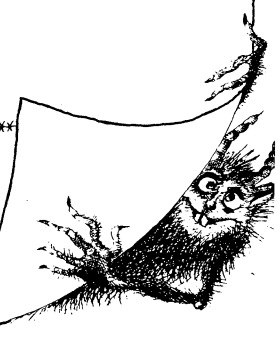
Der zwischen West und Ost vermittelnde Spieler B hat gewonnen.-

Unser Programm soll das Spiel auf dem Bildschirm ermöglichen und darüber wachen, daß korrekt gezogen wird. Die Ermittlung des Siegers nimmt es nicht vor; dies ist eine interessante Programmieraufgabe, welche dem Leser überlassen wird.
Eine überraschend einfache Spielstrategie ist vor kurzem gefunden worden; sie lautet: "Kreuz eröffnet mit dem unten links gezeigten Zug. Jedesmal, wenn Kreis anschließend eine der punktierten Linien an einem Ende kreuzt, kreuzt Kreuz (!) sie am anderen."



Probieren Sie die Strategie aus; Sie werden sicher gegen jeden Gegner gewinnen (die Lage der punktierten Linien müssen Sie sich vorher gut einprägen)!
Und dies ist das Programm:

```
1000 : PRINT "□
1010 : PRINT "            BRUECKENSPIEL
1011 : PRINT "            -------------
1012 :
1020 REM TOPOLOGISCHES SPIEL VON DAVID GALE
1021 :
1022 :
1200 REM *** HAUPTPROGRAMM *****************************************
1201 :
1220 : GOSUB 2000 : REM ANLEITUNG
1221 :
1230 : GOSUB 3000 : REM INITIALISIERUNG
1231 :
1240 : GOSUB 4000 : REM SPIELFELDAUSGABE
1241 :
1250 : GOSUB 5000 : REM ZUGFOLGE
1251 :
1300 : END
1301 :
1990 REM *** ENDE DES HAUPTPROGRAMMS ***************
1998 :
1999 :
```



```
2000 REM ### PROZEDUR 'ANLEITUNG' ######################################
2001 :
2010 : PRINT
2020 : PRINT "ANLEITUNG ERWUENSCHT ? (J/N)
2030 :
2040 : GET A$ : IF A$ = "" THEN 2040
2050 : IF A$ <> "J" THEN RETURN
2090 :
2120 : PRINT"■"
2130 : PRINT "IN DIESEM SPIEL FUER ZWEI PARTNER GEHT"
2140 : PRINT "ES DARUM, EIN FELD ZU UEBERQUEREN UND"
2150 : PRINT "GLEICHZEITIG DEN GEGENSPIELER DARAN"
2160 : PRINT "ZU HINDERN, DAS GLEICHE ZU ERREICHEN."
2170 : PRINT "■ES GIBT KREIS ('o') UND KREUZ ('x')."
2180 : PRINT "KREIS MUSS VERSUCHEN, DAS FELD VON"
2190 : PRINT "NORDEN NACH SUEDEN ZU UEBERQUEREN;"
2200 : PRINT "KREUZ VERSUCHT ES VON WESTEN NACH OSTEN."
2210 : PRINT "BEIDE SPIELER SPRINGEN JEWEILS AUF DEN"
2220 : PRINT "FUER SIE GEKENNZEICHNETEN FELDERN."
2230 : PRINT "■BEI JEDEM SPRUNG WIRD EINE LINIE"
2240 : PRINT "HINTER DEM SPRUNG GEZOGEN, DIE ZU"
2250 : PRINT "UEBERQUEREN UNMOEGLICH IST."
2260 : PRINT "■DIE EINGABE ERFOLGT DURCH KOORDINATEN;"
2270 : PRINT "DIE RICHTUNG WIRD DURCH DEN ERSTEN"
2280 : PRINT "BUCHSTABEN DER HIMMELSRICHTUNG GENANNT."
2285 :
2290 : PRINT "■  ■*** LEERTASTE DRUECKEN ***■
2300 : GET A$ : IF A$ <> " " THEN 2300
2301 :
2900 : RETURN
2901 :
2990 REM ENDE DER ANLEITUNG ############################################
2998 :
2999 :
3000 REM ### PROZEDUR 'INITIALISIERUNG' ##################################
3001 :
3100 : LET H$(0) = "KREIS"
3110 : LET H$(1) = "KREUZ"
3111 :
3200 : LET P(0) = 87 : REM CODEZAHL FUER KREIS
3210 : LET P(1) = 86 : REM CODEZAHL FUER KREUZ
3211 :
3220 : LET B(0) = 64 : REM CODE FUER WAAGERECHTEN ZUG
3230 : LET B(1) = 93 : REM CODE FUER SENKRECHTEN ZUG
3231 :
3240 : LET SZ(0)= 40 : REM SIEGRICHTUNG KREIS
3250 : LET SZ(1)=  1 : REM SIEGRICHTUNG KREUZ
3260 :
3900 : RETURN
3901 :
3990 REM ENDE DER INITIALISIERUNG ######################################
3998 :
3999 :
```

```
4000 REM ### PROZEDUR 'SPIELFELDAUSGABE' ##############################
4001 :
4100 :   PRINT "■";
4101 :
4200 : REM --- KREISE
4201 :
4210 :   FOR I = 0 TO 5
4220 :     FOR J = 0 TO 4
4230 :       POKE 32812 + 160*I + 4*J, P(0)
4240 :     NEXT J
4250 :   NEXT I
4260 :
4300 : REM --- KREUZE
4301 :
4310 :   FOR I = 0 TO 4
4320 :     FOR J = 0 TO 5
4330 :       POKE 32890 + 160*I + 4*J, P(1)
4340 :     NEXT J
4350 :   NEXT I
4360 :
4400 : REM --- BESCHRIFTUNG
4401 :
4410 :   FOR I = 75 TO 65 STEP -1
4420 :     PRINT "■■" CHR$(I)
4430 :   NEXT I
4440 :
4450 : PRINT"■■  A B C D E F G H I J K "
4460 :
4900 : RETURN
4901 :
4990 REM ENDE DER SPIELFELDAUSGABE ###################################
4998 :
4999 :
5000 REM ### PROZEDUR 'ZUGFOLGE' ########################################
5001 :
5010 : REM = PROZEDURRUMPF 'ZUGFOLGE' ================================
5011 :
5100 :   REM --- BESTIMMUNG DES ANZIEHENDEN
5101 :
5110 :     PRINT "■" TAB(27) "■WER BEGINNT?
5120 :     PRINT TAB(27) "■KREIS ... 0
5130 :     PRINT TAB(27) "KREUZ ... 1
5140 :
5150 :     GET A$ : IF A$ <> "0" AND A$ <> "1" THEN 5150
5160 :     LET W = VAL(A$)
5170 :
5200 :   REM --- EINGABE DES SPIELZUGES
5201 :
5210 :     GOSUB 6000 : REM LOESCHEN DER ANZEIGE
5220 :     GOSUB 7000 : REM EINGABE
5230 :
5300 :   REM --- EINGABEKONTROLLE
5301 :
5310 :     LET PK = 33610 + 2*X - 80*Y
5320 :
5330 :     IF PEEK(PK+4*V) <> P(W) AND V <> SZ(W) THEN 5200
5340 :     IF PEEK(PK+2*V) <> 32 THEN 5200
5350 :
```

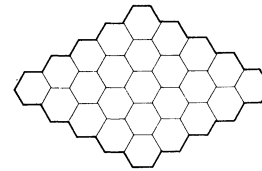
```
5400 :    REM --- EINTRAGUNG INS SPIELFELD
5401 :
5410 :      FOR I = 1 TO 3
5420 :        POKE PK+V*I, B(C)
5430 :      NEXT I
5440 :
5500 :    REM --- NAECHSTER SPIELER
5501 :
5510 :      LET W = 1-W
5520 :
5530 :      GOTO 5200 : REM NEUER ZUG
5590 :
5600 : REM ENDE DES PROZEDURRUMPFES 'ZUGFOLGE' ======================
5601 :
6000 : REM +++ UNTERPROZEDUR 'LOESCHEN DER ANZEIGE' ++++++++++++++++++
6001 :
6010 :    PRINT "S
6020 :    FOR I = 0 TO 13
6030 :      PRINT TAB(27) "              "
6040 :    NEXT I
6050 :
6090 :    RETURN
6100 :
6190 : REM      ENDE DES LOESCHENS DER ANZEIGE ++++++++++++++++++++++++
6199 :
7000 : REM +++ UNTERPROZEDUR 'EINGABE' ++++++++++++++++++++++++++++++++
7001 :
7100 :    REM ANGABE DES AM ZUG BEFINDLICHEN SPIELERS
7101 :
7110 :      PRINT "S"
7120 :      PRINT TAB(27) "AM ZUG: "
7140 :      PRINT TAB(27) "  " H$(W)" "
7160 :
7200 :    REM REM EINGABE DER X-KOORDINATE
7201 :
7210 :      PRINT TAB(28) "X := ";
7220 :      GET A$ : IF A$ = "" THEN 7220
7230 :      LET A = ASC(A$)
7240 :      IF A < 65 OR A > 75 THEN 7220
7250 :      IF W = 0 THEN IF A <> 2*INT(A/2) THEN PRINT "?";: GOTO 7220
7260 :      IF W = 1 THEN IF A = 2*INT(A/2)  THEN PRINT "?";: GOTO 7220
7270 :      PRINT A$
7280 :      LET X = A-65
7290 :
7300 :    REM EINGABE DER Y-KOORDINATE
7301 :
7310 :      PRINT TAB(28) "Y := ";
7320 :      GET A$ : IF A$ = "" THEN 7320
7330 :      LET A = ASC(A$)
7340 :      IF A < 65 OR A > 75 THEN 7320
7350 :      IF W = 0 THEN IF A = 2*INT(A/2)  THEN PRINT "?";: GOTO 7320
7360 :      IF W = 1 THEN IF A <> 2*INT(A/2) THEN PRINT "?";: GOTO 7320
7370 :      PRINT A$
7380 :      LET Y = A-65
7390 :
```

```
7400 :   REM EINGABE DER RICHTUNG
7401 :
7410 :      PRINT TAB(27) "■■RICHTUNG:
7420 :      PRINT TAB(27) "(O,S,W,N)
7430 :      PRINT TAB(27) "■R:= ";
7440 :      GET R$
7450 :      IF R$ = "O" THEN V =   1 : PRINT R$ : C = 0 : GOTO 7500
7460 :      IF R$ = "S" THEN V =  40 : PRINT R$ : C = 1 : GOTO 7500
7470 :      IF R$ = "W" THEN V =  -1 : PRINT R$ : C = 0 : GOTO 7500
7480 :      IF R$ = "N" THEN V = -40 : PRINT R$ : C = 1 : GOTO 7500
7490 :      GOTO 7440
7499 :
7500 :      RETURN
7510 :
7590 : REM ENDE DER EINGABE ++++++++++++++++++++++++++++++++++++++++++++
7591 :
7990 REM ENDE DER ZUGFOLGE ##############################################
7998 :
7999 :
```

*

Aufgabe

31. Von Piet Hein stammt auch das Spiel Hex. Es wird auf Bienenwaben (aneinandergereihten regelmäßigen Sechsecken) gespielt; die Anzahl der Sechsecke kann verschieden sein (am gebräuchlichsten sind 11 x 11). Bei 5 x 5 Sechsecken spricht man von 'Mini-Hex' (siehe nebenstehend).



Ziel des Spieles ist es, eine ununterbrochene Verbindung zweier gegenüberliegender Seiten zu erreichen. Dazu setzen die Spieler abwechselnd weiße und schwarze Steine aus einem Damespiel - oder sie markieren die Felder abwechselnd mit verschiedenen Farben.
Schreiben Sie ein Programm für Hex mit dem Computer als Spielmeister. Es ist keine allgemeine Gewinnstrategie bekannt; bei kleinen Spielfeldern erkennt man jedoch leicht, daß der Anziehende immer gewinnen kann. Ermöglichen Sie dies dem Computer.

Die Krone aller strategischen Spiele ist zweifellos S c h a c h .
Da es unter den Spielen schon immer als das bevorzugte Anwendungsgebiet menschlicher Intelligenz angesehen wurde, hat kein Spiel die Computerfachleute so sehr fasziniert und herausgefordert. Sollte es gelingen, einer Maschine das Schachspielen beizubringen, so könnte man ihr das Attribut 'intelligent' nicht länger absprechen und hätte damit bewiesen, daß künstliche Intelligenz möglich ist.

Schon v o r der Computer-Ära ist immer wieder versucht worden, Schach spielende Maschinen zu erfinden bzw. ihre Erfindung vorzutäuschen.
In seinem Essay "Maelzels Schachspieler" schildert E.A. Poe einen 1769 vom Baron von Kempelen ("Edelmann aus Preßburg in Ungarn") konstruierten "automatischen Schachspieler", welcher von einem gewissen Maelzel zu Paris, Wien, London und in den Vereinigten Staaten vorgeführt worden war. Interessant ist der Versuch Poes, die Fähigkeiten eines Schachcomputers mit einer Rechenmaschine, wie er sie vom Computer-Pionier Babbage kannte, zu vergleichen:

"Nun sind aber arithmetische oder algebraische Kalkulationen schon von Natur aus fixiert und eindeutig determiniert. Gewisse Daten vorausgesetzt, erhalten wir unausweichlich und mit zwingender Notwendigkeit die entsprechenden Resultate. ... Ganz anders hingegen ist's um unseren Schachspieler bestellt. Bei ihm gibt es keinerlei vorherbestimmten festumrissenen Ablauf. Kein Schachzug ist ja die notwendige Folge eines ihm vorausgegangenen. ... Selbst wenn wir annehmen wollten, daß die Züge der Schachautomate streng vorherbestimmt sind, so würde ihre Folgerichtigkeit doch unterbrochen und durcheinandergebracht durch die nicht vorherbestimmbaren Entschlüsse des jeweiligen Gegenspielers. So gibt's denn keinerlei Analogie zwischen den Operationen von Maelzels Schach- und Mr. Babbages Rechenautomate - und sind wir willens, die erstere für eine von sich aus funktionierende Maschine anzusehen, so müssen wir wohl oder übel auch zugeben, daß wir's in ihr mit der wunderbarsten, über jedweden Vergleich turmhoch erhabene Erfindung der Menschheit zu tun haben."

"Nicht immer bleibt der Schachtürke Sieger. Wäre die Maschine jedoch ein Apparat, der einzig und allein von sich aus funktioniert, so könnte dies nimmermehr der Fall sein - sie würde j e d e s Spiel gewinnen. Ist das Prinzip erst einmal entdeckt, nach welchem man eine Maschine dazu bringen kann, Schach zu spielen, so bedarf es bloß einer Erweiterung solchen Prinzips, sie das Spiel auch gewinnen, und einer neuerlichen Erweiterung, sie j e d e s Spiel gewinnen zu lassen. Die Schwierigkeit, welche darin besteht, eine Maschine zum Gewinnen sämtlicher Schachpartien zu bringen, kann um nichts größer sein denn diejenige, die es zu meistern gilt, um solche Maschine auch nur eine einzige Partie gewinnen zu machen.
Sähen wir den Schachtürken für eine Maschine an, so müßten wir etwas im höchsten Maße Unwahrscheinliches glauben: nämlich, daß der Erfinder es vorgezogen, seinem Kinde n i c h t die letzte Vollendung zu geben ...".

*

Wir wollen im folgenden ein Schachprogramm entwickeln, beanspruchen dabei aber nicht, daß es mit bereits existierenden Programmen hinsichtlich der Spielstärke wetteifern kann. Vielmehr geht es uns nur darum, die grundlegenden Ideen und Methoden zu verdeutlichen, nach denen Schachprogramme aufgebaut sind. Der Leser mag dann in der angegebenen Richtung weiterforschen, um sein eigenes spielstarkes Programm zu entwerfen.

Beispiel 6 : 6 x 6 - Mikroschach

Es soll ein auf einem 6 x 6 - Quadrat spielendes Schachprogramm - ohne Springer - entworfen werden. Der Computer führt die weißen Figuren.

Das Schachbrett stellen wir im Rechner als zweidimensionale Tabelle F[0..7 , 0..7] dar; die Figuren werden wie folgt codiert:

	weiß	schwarz
Bauer	1	-1
Läufer	2	-2
Turm	3	-3
Dame	4	-4
König	5	-5

Die Anfangsstellung sieht für den Rechner demnach so aus (nächste Seite):

i \ j	0	1	2	3	4	5	6	7
0	10	10	10	10	10	10	10	10
1	10	3	2	4	5	2	3	10
2	10	1	1	1	1	1	1	10
3	10	0	0	0	0	0	0	10
4	10	0	0	0	0	0	0	10
5	10	-1	-1	-1	-1	-1	-1	10
6	10	-3	-2	-4	-5	-2	-3	10
7	10	10	10	10	10	10	10	10

Der Zeilenindex ist i, der Spaltenindex j; F(1,4) = 5 bedeutet, daß in der 1. Zeile und 4. Spalte der Tabelle F der weiße König steht. Die 10 in der 0. und 7. Zeile und Spalte markiert den Brettrand; leere Felder werden durch 0 gekennzeichnet.

Auf dem Bildschirm erscheint das Spielfeld in folgender Gestalt:

Großbuchstaben:
weiß (Computer)

Kleinbuchstaben:
schwarz (Mensch)

F	E	D	C	B	A	
T	L	D	K	L	T	1
B	B	B	B	B	B	2
						3
						4
b	b	b	b	b	b	5
t	l	d	k	l	t	6

Das Gesamtprogramm gliedert sich, wie üblich, in die Teile

- Anleitung, Initialisierung, Zugfolge und Verabschiedung/Wiederholung.

Die eigentliche Schachpartie (Zugfolge) gliedert sich in die Teile

- o Spielfeldausgabe
- o Computerzug
- o Spielfeldausgabe
- o Spielerzug,

welche bis zum Abbruch ("Ich bin matt") wiederholt werden. Die Prozedur 'Computerzug' läßt sich wie folgt beschreiben:
Der Computer geht jedes Feld (i,j) mit der Frage durch, ob sich dort eine eigene (d.h. weiße) Figur befindet; das Kriterium hierfür lautet 'F(i,j) > 0'. Ist dies der Fall, so stellt er zunächst fest,welche Figur (Bauer, Läufer, Turm, Dame oder König) auf diesem Feld steht; dann führt er - probeweise - alle Züge aus, die dieser Figur eigentümlich sind.

```
30  FOR I = 1 TO 6
40    FOR J = 1 TO 6
50      IF F(I,J) < 0 THEN 70
60      ON F(I,J) GOSUB 100, 200, 300, 400, 500
70    NEXT J
80  NEXT I
```

Das weitere Geschehen spielt sich nun in den Prozeduren 'Bauernzüge', 'Läuferzüge' (200), 'Turmzüge' (300), 'Damezüge' (400) und 'Königszüge' ab. Bei jedem Zug, den er zu machen gedenkt, muß der Computer zunächst prüfen, ob dieser überhaupt l e g a l ist, d.h. ob auf dem Feld, das die Figur betreten soll, nicht schon eine eigene Figur steht; ferner prüft er, ob jener Zug nicht etwa den eigenen König einem Schachgebot durch den Gegner aussetzt. Züge dieser Art und nicht-legale Züge betrachtet er als unzulässig, d.h. er übergeht sie. Um nun unter allen zulässigen Zügen denjenigen auszuwählen, den er tatsächlich ausführen wird, muß der Computer die durch den Probezug entstandene Spielstellung analysieren und bewerten. Seine Bewertungsgesichtspunkte sind:

(1) Ist die Figur von gegnerischen Figuren bedroht bzw. von eigenen Figuren gedeckt?
(2) Ist die Figur wirksam eingesetzt - gemessen an der Anzahl bestrahlter Felder?

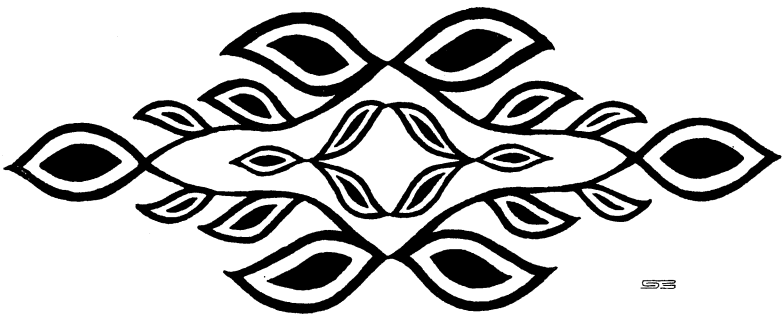
Ein weiterer Gesichtspunkt ist natürlich, ob eine Figur des Gegners geschlagen werden kann. Wir müssen diese Frage leider unberücksichtigt lassen, weil sie ohne eine Analyse möglicher Züge des Gegners nicht zufriedenstellend beantwortet werden kann. Unsere Bewertungsfunktion ist also nur 'statisch'.

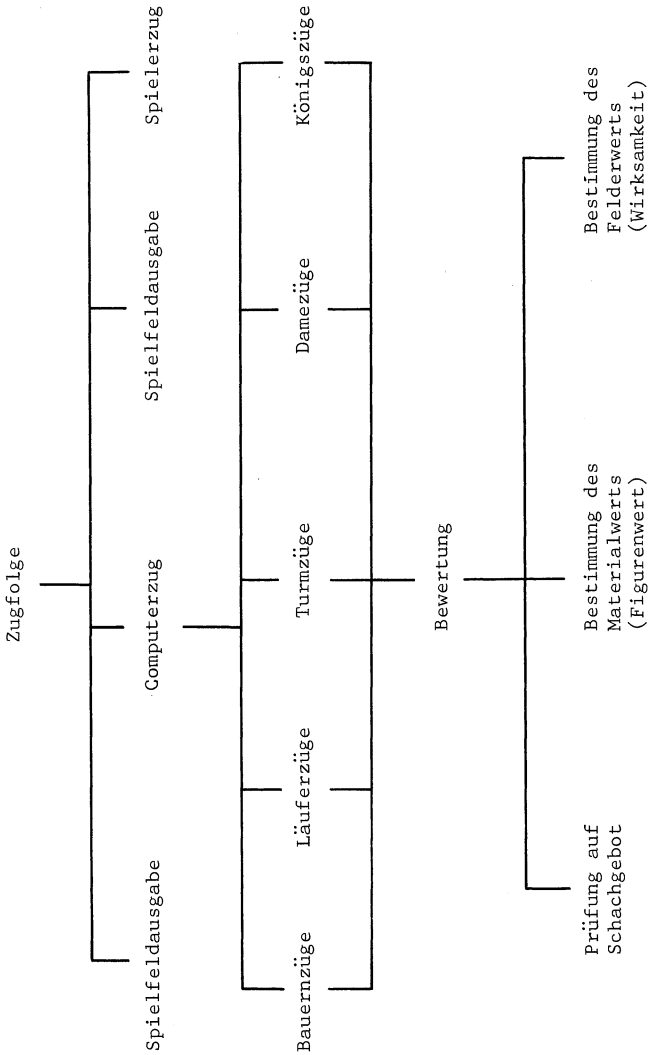
Die Prozedur 'Bewertung' prüft zunächst, ob der ins Auge gefaßte Zug den eigenen König einem Schachgebot aussetzt; ist dies der Fall, wird die Prozedur verlassen und der nächste Zug vorgenommen. Sodann wird der Felderwert (die Wirksamkeit) der Figur ermittelt. Zu diesem Zweck werden die Prozeduren 'mögliche Läuferzüge', 'mögliche Turmzüge' und 'mögliche Damezüge' aufgerufen: sie liefern als Beitrag zur schon erreichten Bewertungssumme die Anzahl der von der Figur aus erreichbaren freien Felder.

Am schwierigsten zu programmieren ist die Prozedur, welche den Materialwert bestimmen soll. Zunächst werden in eine Tabelle DE(..) diejenigen Figuren eingetragen, welche die eigene Figur decken; sodann in eine Tabelle DR(..) die Figuren, welche die eigene Figur bedrohen. Der Wert der drohenden Figuren wird vom Wert der deckenden Figuren abgezogen (Bauer: 10, Läufer: 30, Turm: 50, Dame: 100) und die Differenz dem Gesamtwert zugeschlagen.

Dies wird für jedes betrachtete Feld (i,j) durchgeführt; dabei merkt sich der Computer den jeweiligen vorläufigen Höchstwert und den zugehörigen Zug.

Den Zug mit dem insgesamt höchsten Wert führt er dann tatsächlich aus. - Damit der Leser die Suche nach dem besten Zug am Bildschirm verfolgen kann, werden alle probeweise vorgenommenen Züge angezeigt.




Zugfolge
Spielfeldausgabe
Computerzug
Spielfeldausgabe
Spielerzug
Bauernzüge
Läuferzüge
Turmzüge
Damezüge
Königszüge
Bewertung
Prüfung auf Schachgebot
Bestimmung des Materialwerts (Figurenwert)
Bestimmung des Felderwerts (Wirksamkeit)

```
10000 : PRINT "■
10010 : PRINT "          6X6-SCHACH
10011 : PRINT "          ----------
10012 :
10020 REM SCHACH-DEMONSTRATIONSPROGRAMM AUF 6X6-FELD OHNE SPRINGER
10040 REM DIE BEWERTUNGSPROZEDUR MUSS NOCH EINGEFUEGT WERDEN
10041 REM DAS PROGRAMM IST NOCH NICHT LAUFFAEHIG!!
10042 :
11000 :
12000 REM *** HAUPTPROGRAMM ****************************************
12001 :
12200 : GOSUB 20000 : REM ANLEITUNG
12201 :
12300 : GOSUB 30000 : REM INITIALISIERUNG
12301 :
12400 : GOSUB 40000 : REM ZUGFOLGE
12401 :
12600 : GOSUB 60000 : REM VON VORNE?
12601 :
12900 : END
12901 :
12990 REM ENDE DES HAUPTPROGRAMMS ************************************
12998 :
12999 :
20000 REM ### PROZEDUR 'ANLEITUNG' ###################################
20001 :
20100 : PRINT "■■DIES IST EIN SCHACHPROGRAMM, BEI DEM
20200 : PRINT "AUS GRUENDEN DER GESCHWINDIGKEIT DAS
20300 : PRINT "SPIELFELD NUR EIN 6X6-QUADRAT IST.
20400 : PRINT "AUF DIE SPRINGER WIRD VERZICHTET,
20500 : PRINT "IM UEBRIGEN GELTEN DIE BEKANNTEN REGELN.
20600 : PRINT "DIE ZUGEINGABE ERFOLGT IN LANGER
20700 : PRINT "NOTATION, Z.B.: 'LA3-B2'.
20800 : PRINT "SIE FUEHREN DIE SCHWARZEN FIGUREN
20900 : PRINT "(KLEINBUCHSTABEN), DER COMPUTER DIE
21000 : PRINT "WEISSEN (GROSSBUCHSTABEN).
21100 :
21200 : PRINT "■■     ■***TASTE DRUECKEN***■
21300 :
21400 : GET T$ : IF T$ = "" THEN 21400
21500 :
21600 : PRINT "■
21700 :
21800 : RETURN
21900 :
29000 REM ENDE DER ANLEITUNG ########################################
29998 :
29999 :
30000 REM ### PROZEDUR 'INITIALISIERUNG' ##############################
30001 :
30100 : REM --- DATENSAETZE
30101 :
30200 DATA 3,-3,2,-2,4,-4,5,-5,2,-2,3,-3      : REM FIGURENCODES
30300 DATA K,D,T,L,B," ","|","L","|","-","." : REM FIGURENZEICHEN
30900 :
31000 : REM --- TABELLEN
31001 :
31100 :   DIM F(7,7) : REM SPIELFELD
31200 :   DIM Z$(11) : REM FIGURENZEICHEN
31900 :
```

```
33000 : REM --- BAUERN SETZEN
33001 :
33100 :   FOR I = 1 TO 6 : LET F(2,I) = 1 : LET F(5,I) = -1 : NEXT I
33200 :
34000 : REM --- FIGUREN SETZEN
34001 :
34100 :   FOR I = 1 TO 6 : READ F(1,I) : READ F(6,I) : NEXT I
34200 :
35000 : REM --- FIGURENZEICHEN LESEN
35001 :
35100 :   FOR I = 1 TO 11 : READ Z$(I) : NEXT I
35200 :
37000 : REM --- KLEINSCHREIBUNG SETZEN
37001 :
37100 :   POKE 59468,14
37200 :
38000 : REM --- ZEICHENKETTE ZUR CURSORSTEUERUNG
38001 :
38010 :   LET D$ = "
38090 :
39000 :   RETURN
39001 :
39900 REM ENDE DER INITIALISIERUNG ######################################
39998 :
39999 :
40000 REM ### PROZEDUR 'ZUGFOLGE' ###################################
40001 :
40100 : REM = PROZEDURRUMPF 'ZUGFOLGE' ================================
40101 : :
40110 : : GOSUB 45000 : REM SPIELFELDAUSGABE
40111 : :
40120 : : GOSUB 50000 : REM COMPUTERZUG
40121 : :
40130 : : GOSUB 45000 : REM SPIELFELDAUSGABE
40131 : :
40140 : : GOSUB 59000 : REM SPIELERZUG
40141 : :
40150 : : GOTO  40110 : REM NEUER ZUG
40151 : :
40190 : : RETURN
40200 : :
40900 : REM ENDE DES PROZEDURRUMPFES 'ZUGFOLGE' =======================
40998 :
40999 :
45000 : REM +++ UNTERPROZEDUR 'SPIELFELDAUSGABE' ++++++++++++++++++++++
45001 :
45100 :   PRINT "
45101 :
45150 :   FOR I = 1 TO 6
45200 :     PRINT "
45250 :     FOR J = 1 TO 6
45300 :       PRINT "|" Z$(F(I,J)+6) "|";
45400 :     NEXT J
45450 :     PRINT I
45500 :   NEXT I
45600 :   PRINT "
45700 :   PRINT " F  E  D  C  B  A "
45900 :   PRINT : PRINT
45950 :
49000 :   RETURN
49900 :
```

```
50000 : REM +++ UNTERPROZEDUR 'COMPUTERZUG' ++++++++++++++++++++++++++++
50001 :
50010 :    REM - PROZEDURRUMPF 'COMPUTERZUG' --------------------------
50020 :    :
50030 :    : FOR I = 1 TO 6
50040 :    :   FOR J = 1 TO 6
50050 :    :     IF F(I,J) <= 0 THEN 50070
50060 :    :     ON F(I,J) GOSUB 50100, 50200, 50300, 50400, 50500
50070 :    :   NEXT J
50080 :    : NEXT I
50081 :    :
50098 :    REM ENDE PROZEDURRUMPF 'COMPUTERZUG' ----------------------
50099 :
50100 :    REM +++ UNTERPROZEDUR 'BAUERNZUEGE' ++++++++++++++++++++++++
50101 :
50110 :      IF F(I+1,J)   = 0 THEN X = I+1 : Y = J   : GOSUB 51000
50120 :      IF F(I+1,J+1) < 0 THEN X = I+1 : Y = J+1 : GOSUB 51000
50130 :      IF F(I+1,J-1) < 0 THEN X = I+1 : Y = J-1 : GOSUB 51000
50140 :
50180 :      RETURN
50181 :
50190 :    REM ENDE DER BAUERNZUEGE ++++++++++++++++++++++++++++++++++++
50199 :
50200 :    REM +++ UNTERPROZEDUR 'LAEUFERZUEGE' ++++++++++++++++++++++++
50201 :
50210 :      FOR K1 = -1 TO 1 STEP 2
50220 :        FOR K2 = -1 TO 1 STEP 2
50230 :          LET X = I : LET Y = J
50235 :          LET X = X + K1 : LET Y = Y + K2
50240 :          IF F(X,Y) < 0 THEN GOSUB 51000 : GOTO 50280
50245 :          IF F(X,Y) > 0 THEN 50280 : REM EIGENE FIGUR
50250 :          GOSUB 51000 : GOTO 50235 : REM BEWERTUNG UND ZURUECK
50260 :        NEXT K2
50270 :      NEXT K1
50275 :
50280 :      RETURN
50281 :
50290 :    REM ENDE DER LAEUFERZUEGE ++++++++++++++++++++++++++++++++++++
50299 :
50300 :    REM +++ UNTERPROZEDUR 'TURMZUEGE' ++++++++++++++++++++++++++++
50301 :
50310 :      LET Y = J
50315 :      FOR K = -1 TO 1 STEP 2
50320 :        LET X = I
50325 :        LET X = X + K
50330 :        IF F(X,Y) > 0 THEN 50345
50335 :        IF F(X,Y) < 0 THEN GOSUB 51000 : GOTO 50345
50340 :        GOSUB 51000 : GOTO 50325
50345 :      NEXT K
50349 :
50350 :      LET X = I
50355 :      FOR K = -1 TO 1 STEP 2
50360 :        LET Y = Y + K
50365 :        IF F(X,Y) > 0 THEN 50375
50370 :        IF F(X,Y) < 0 THEN GOSUB 51000 : GOTO 50375
50372 :        GOSUB 51000 : GOTO 50360
50375 :      NEXT K
50379 :
50380 :      RETURN
```

```
50399 :
50400 :   REM +++ UNTERPROZEDUR 'DAMENZUEGE' ++++++++++++++++++++++++++
50401 :
50410 :     GOSUB 50200 : REM LAEUFERZUEGE
50420 :     GOSUB 50300 : REM TURMZUEGE
50430 :
50480 :     RETURN
50481 :
50490 :   REM ENDE DER DAMENZUEGE ++++++++++++++++++++++++++++++++++++++
50491 :
50500 :   REM +++ KOENIGSZUEGE ++++++++++++++++++++++++++++++++++++++++
50501 :
50510 :     FOR K1 = -1 TO 1
50520 :       FOR K2 = -1 TO 1
50530 :         IF F(I+K1,J+K2) < 1 THEN X=I+K1: Y=J+K2 : GOSUB 51000
50540 :       NEXT K2
50550 :     NEXT K1
50560 :
50580 :     RETURN
50581 :
50590 :   REM ENDE DER KOENIGSZUEGE ++++++++++++++++++++++++++++++++++++
50599 :
51000 :   REM +++ UNTERPROZEDUR 'BEWERTUNG' ++++++++++++++++++++++++++
51001 :
51010 :     REM - PROZEDURRUMPF 'BEWERTUNG' ----------------------------
51011 :     :
51031 :     : LET H1 = 32849+(X-1)*80+(Y-1)*3 : REM ZUR ANZEIGE
51032 :     : LET H2 = 32849+(I-1)*80+(J-1)*3 : REM ZUR ANZEIGE
51033 :     : LET H3 = PEEK(H1) : POKE H1,PEEK(H2) : POKE H2,32
51100 :     :
51120 :     : REM ERMITTLUNG DES MATERIALWERTS UND DER WIRKSAMKEIT
51121 :     : REM MUSS NOCH EINGETRAGEN WERDEN
51123 :     :
51180 :     : RETURN
51190 :     :
51900 :     REM ENDE PROZEDURRUMPF 'BEWERTUNG' ------------------------
51999 :
52000 :     REM +++ UNTERPROZEDUR 'MOEGLICHE LAEUFERZUEGE' ++++++++++++
52001 :
52100 :       REM MUSS NOCH EINGETRAGEN WERDEN
52101 :
52900 :       RETURN
52901 :
52990 :     REM ENDE DER MOEGLICHEN LAEUFERZUEGE ++++++++++++++++++++++
52999 :
53000 :     REM +++ UNTERPROZEDUR 'MOEGLICHE TURMZUEGE' +++++++++++++++
53001 :
53100 :       REM MUSS NOCH EINGETRAGEN WERDEN
53101 :
53900 :       RETURN
53901 :
53990 :     REM ENDE DER MOEGLICHEN TURMZUEGE ++++++++++++++++++++++++++
53999 :
54000 :     REM UNTERPROZEDUR 'MOEGLICHE DAMENZUEGE' +++++++++++++++++++
54001 :
54030 :       GOSUB 52000       : REM MOEGLICHE LAEUFERZUEGE
54050 :       GOSUB 53000       : REM MOEGLICHE TURMZUEGE
54060 :
54900 :       RETURN
54901 :
```

```
58000 :      REM +++ UNTERPROZEDUR 'SCHACHPRUEFUNG' ++++++++++++++++++++
58001 :
58010 :        FOR II = 1 TO 6
58020 :          FOR JJ = 1 TO 6
58030 :            IF F(II,JJ) = 5 THEN LET KX = II : LET KY = JJ
58040 :          NEXT JJ
58050 :        NEXT II
58090 :
58100 :        LET H$ = "SCHACH"
58101 :
58110 :        IF F(KX+1,KY+1) = -1 THEN RETURN
58120 :        IF F(KX-1,KY+1) = -1 THEN RETURN
58190 :
58200 :        LET U = KX  : LET V = KY
58210 :        LET U = U+1 : LET V = V+1 : IF F(U,V) = 0 THEN 58210
58220 :        IF F(U,V) = -2 THEN RETURN: REM LAEUFER BIETET SCHACH
58230 :        IF F(U,V) = -4 THEN RETURN: REM DAME BIETET SCHACH
58240 :
58250 :        LET U = KX  : LET V = KY
58260 :        LET U = U+1 : LET V = V-1 : IF F(U,V) = 0 THEN 58260
58270 :        IF F(U,V) = -2 OR F(U,V) = -4 THEN RETURN
58280 :
58290 :
58300 :        LET U = KX  : LET V = KY
58310 :        LET U = U-1 : LET V = V+1 : IF F(U,V) = 0 THEN 58320
58320 :        IF F(U,V) = -2 OR F(U,V) = -4 THEN RETURN
58330 :
58350 :        LET U = KX  : LET V = KY
58360 :        LET U = U-1 : LET V = V-1 : IF F(U,V) = 0 THEN 58360
58370 :        IF F(U,V) = -2 OR F(U,V) = -4 THEN RETURN
58380 :
58400 :        LET U = KX  : LET V = KY
58410 :        LET U = U+1 : IF F(U,V) = 0 THEN 58410
58420 :        IF F(U,V) = -3 THEN RETURN : REM TURM BIETET SCHACH
58430 :        IF F(U,V) = -4 THEN RETURN : REM DAME BIETET SCHACH
58440 :
58450 :        LET U = KX
58460 :        LET U = U-1 : IF F(U,V) = 0 THEN 58460
58470 :        IF F(U,V) = -3 OR F(U,V) = -4 THEN RETURN
58480 :
58500 :        LET V = KY
58510 :        LET V = V+1 : IF F(U,V) = 0 THEN 58510
58520 :        IF F(U,V) = -3 OR F(U,V) = -4 THEN RETURN
58530 :
58550 :        LET V = KY
58560 :        LET V = V-1 : IF F(U,V) = 0 THEN 58560
58570 :        IF F(U,V) = -3 OR F(U,V) = -4 THEN RETURN
58590 :
58600 :        FOR M = -1 TO 1
58610 :          FOR N = -1 TO 1
58620 :            IF F(KX+M,KY+N) = -5 THEN RETURN
58630 :          NEXT N
58640 :        NEXT M
58690 :
58700 :        LET H$ = "KEIN SCHACH"
58701 :
58800 :        RETURN
58801 :
58990 : REM ENDE DES COMPUTERZUGS ++++++++++++++++++++++++++++++++++++
58999 :
```

```
59000 : REM +++ UNTERPROZEDUR 'SPIELERZUG' ++++++++++++++++++++++++++++
59001 :
59010 : REM --- EINGABE
59011 :
59020 :   PRINT "■";D$;"                                    "
59025 :   PRINT "                                  "
59030 :   INPUT "□□IHR ZUG: "; Z$
59040 :
59050 :   IF LEN(Z$) <> 6 THEN 59500 : REM FEHLERMELDUNG
59060 :   IF MID$(Z$,4,1) <> "-" THEN 59500 : REM FEHLERMELDUNG
59070 :
59100 : REM --- EINTRAGUNG INS SPIELFELD
59101 :
59110 :   LET VX = 7 - (ASC(MID$(Z$,2,1))-64)
59120 :   LET VY = VAL(MID$(Z$,3,1))
59130 :
59140 :   LET NX = 7 - (ASC(MID$(Z$,5,1))-64)
59150 :   LET NY = VAL(MID$(Z$,6,1))
59160 :
59200 :   IF Z$(F(VY,VX)+6) <> LEFT$(Z$,1) OR F(NY,NX) < 0 THEN 59500
59210 :
59220 :   LET F(NY,NX) = F(VY,VX)
59230 :   LET F(VY,VX) = 0
59290 :
59300 :   GOTO 40150 : REM ZURUECK ZUM PROZEDURRUMPF 'ZUGFOLGE'
59301 :
59500 : REM --- FEHLERMELDUNG
59501 :
59510 :    PRINT "□
59520 :    PRINT "□■       FEHLER!          ■ ,
59530 :
59540 :    FOR I = 1 TO 1000 : NEXT I
59550 :
59560 :    GOTO 59020
59561 :
59590 : REM ENDE DES SPIELERZUGS +++++++++++++++++++++++++++++++++++++
59591 :
59900 REM ENDE DER ZUGFOLGE ##############################################
```

Aufgaben

32. Fügen Sie die fehlende Prozedur 'Bewertung' ins Mikroschach-Programm ein!

33. Entwerfen Sie ein 5 x 5 - Schachprogramm (mit Läufer, Springer, Turm, Dame, König und der entsprechenden Anzahl Bauern).

34. Ergänzen Sie das 6 x 6 - Schachprogramm durch zwei Springer zu einem 8 x 8 - Programm.

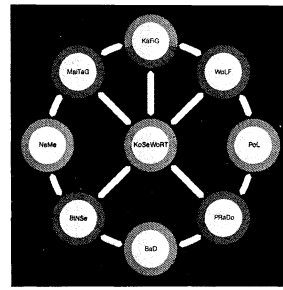
35. Blockwenden
Dies Spiel wird auf einem 3 x 3 - Brett mit Reversi-Steinen (eine Seite O, eine Seite X) gespielt.
Die Anfangsstellung ist nebenstehend abgebildet. Ein Spielzug besteht im Umwenden aller Steine eines Rechtecks ('Blocks'); die Rechtecke haben die Seitenlängen 1 x 1, 1 x 2, 1 x 3, 2 x 2, 2 x 3, 3 x 3. Das Blockwenden ist jedoch nur zulässig, wenn die linke obere Ecke des Blocks einen Stein mit einem X trägt (beide Spieler sitzen auf der gleichen Seite des Spielfelds). Wem es gelingt, alle Steine auf die O - Seite zu wenden, hat gewonnen. Die Spieler ziehen abwechselnd; wer an der Reihe ist, muß einen Block wenden.

X	O	X
O	X	O
X	O	X

Entwickeln Sie von der Endstellung her rekursiv eine Gewinnstrategie.

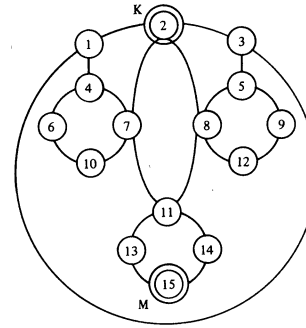
36. Euklids Spiel
Gegeben ist ein Rechteck mit ganzzahligen Seitenlängen. Zwei Personen A und B spielen folgendermaßen gegeneinander: ein Zug besteht darin, vom Rechteck eines oder mehrere Quadrate abzuschneiden; dabei muß ein Rechteck als Rest bleiben. Wer nicht mehr ziehen kann, weil kein Rechteck mehr da ist, hat verloren. (Der Gewinner hat also das Rechteck vollständig in Quadrate zerlegt.) Entwickeln Sie eine Strategie und programmieren Sie dieselbe.

37. Hermann und Gerhard benutzen zu einem Spiel, das über mehrere Runden geht, folgende Wörter: Bad, Binse, Käfig, Kosewort, Maitag, Name, Pol, Parade, Wolf. Zwei Wörter heißen miteinander verträglich, wenn sie genau einen Konsonanten gemeinsam haben.
Zu Beginn bestimmt Hermann für sich und für Gerhard je eines der neun Wörter. Dann nennt jeder Spieler im Wechsel mit dem anderen ein Wort, das mit dem von ihm selbst zuletzt genannten Wort verträglich ist.

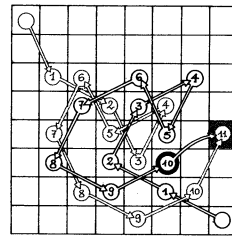


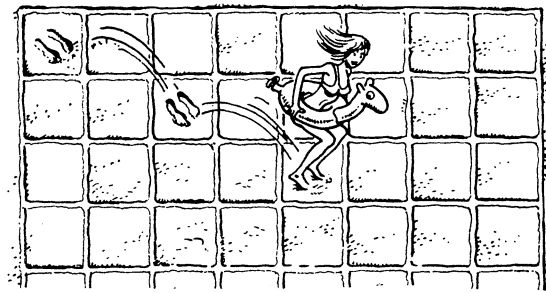
Gerhard hat gewonnen, wenn sein Wort mit dem soeben von Hermann genannten Wort übereinstimmt. Gewinnstrategie gesucht!

38. Gegeben ist ein Spielfeld wie nebenstehend abgebildet. Die Spieler sind 'Katz' und 'Maus'; zu Beginn steht die Katze auf Feld 2, die Maus auf Feld 15. Die Katze, welche anzieht, soll die Maus fangen, d.h. den gleichen Platz wie diese besetzen. Erlaubt sind nur Züge auf Nachbarfelder über die eingezeichneten Bögen. Entwickeln Sie eine Spielstrategie.



39. Auf einem Schachbrett stehen zwei Springer; der eine verfolgt den anderen. Sein Ziel ist es, auf das Feld zu springen, das der Verfolgte gerade einnimmt. Zu Beginn stehen die Springer auf zwei gegenüberliegenden Eckfeldern. Gewinnstrategie und Programm gesucht!





Zum Abschlus dieses Kapitels möchte ich den Lesern noch ein schönes Spiel vorstellen:

Gegenbeispiel : Vier gewinnt

```
10 DIMRR(11)
20 DIMRE(10,10),A3(2,100),A2(2,100)
30 PRINT"□"
40 PRINT"■■■■■■■■■■■■■■                                                    ■"
42 PRINT"■        V I E R   G E W I N N T          "
44 PRINT"■                                                                 ■"
50 FORZ=1TO1000:NEXTZ
60 PRINT"■■■■■■  KENNEN SIE DIE SPIELREGELN ?"
70 GETZ$:IFZ$<>"J"ANDZ$<>"N"THEN70
80 IFZ$="N"THENGOSUB50000
100 PRINT"□"
110 FORZ=1TO400:NEXTZ
120 PRINT" |   |   |   |   |   |   |   |   |   |"
140 PRINT" |---+---+---+---+---+---+---+---+---+---|"
150 PRINT" |   |   |   |   |   |   |   |   |   |   |"
170 PRINT" |---+---+---+---+---+---+---+---+---+---|"
180 PRINT" |   |   |   |   |   |   |   |   |   |   |"
200 PRINT" |---+---+---+---+---+---+---+---+---+---|"
210 PRINT" |   |   |   |   |   |   |   |   |   |   |"
230 PRINT" |---+---+---+---+---+---+---+---+---+---|"
240 PRINT" |   |   |   |   |   |   |   |   |   |   |"
260 PRINT" |---+---+---+---+---+---+---+---+---+---|"
270 PRINT" |   |   |   |   |   |   |   |   |   |   |"
290 PRINT" |---+---+---+---+---+---+---+---+---+---|"
300 PRINT" |   |   |   |   |   |   |   |   |   |   |"
320 PRINT" |---+---+---+---+---+---+---+---+---+---|"
340 PRINT" |   |   |   |   |   |   |   |   |   |   |"
360 PRINT" |---+---+---+---+---+---+---+---+---+---|"
380 PRINT" |   |   |   |   |   |   |   |   |   |   |"
400 PRINT" |---+---+---+---+---+---+---+---+---+---|"
420 PRINT" |   |   |   |   |   |   |   |   |   |   |"
450 PRINT" '---+---+---+---+---+---+---+---+---+---'"
470 PRINT"   1   2   3   4   5   6   7   8   9"
475 PRINT:PRINT" WOLLEN SIE BEGINNEN ?"
480 GETC$:IFC$<>"N"ANDC$<>"J"THEN480
490 PRINT"□                                □"
495 IFC$="N"THENPRINT:GOTO 800
500 PRINT"    BEGINNEN SIE !"
600 GETA$
610 IFA$<>"1"ANDA$<>"2"ANDA$<>"3"ANDA$<>"4"ANDA$<>"5"ANDA$<>"6"THEN630
620 GOTO700
630 IFA$<>"7"ANDA$<>"8"ANDA$<>"9"THEN600
700 A=VAL(A$)
710 IFRR(A)>0THEN IFLOG(RR(A))/LOG(10)>9THEN600
720 SP=1
750 GOSUB5400:GOSUB5000
780 PRINT"□                          "
790 GOSUB12000:GOSUB20000 : IFSG$="SIEG"THEN900
800 GOSUB2000
```

```
810 IFSG$="VERLOREN"THEN850
820 PRINT"□   SIE SIND DRAN !        "
830 POKE158,0:GOTO600
850 PRINT" SIE HABEN LEIDER  V E R L O R E N .":GOTO910
900 PRINT"  S I E  HABEN  G E W O N N E N  !"
910 FORY=1TO200:GETA$:NEXTY
920 GETA$:IFA$=""THEN920
999 :  RUN
2000 REM --- RECHNER SETZT ---
2080 A=0
2090 SP=2:IFM(SP)=0THEN3050
2100 FORR=1TOM(SP)
2120 B=INT(A3(SP,R)):C=(A3(SP,R)-B)*10
2125 GOSUB30000
2130 IFC=0THEN3000
2140 IFRE(B,C-1)<>0THEN 3000
2200 NEXTR :GOTO3050
3000 A=B:IFA>0THEN3500
3050 SP=1:IFM(SP)=0THEN3150
3060 FORR=1TOM(SP)
3070 B=INT(A3(SP,R)):C=(A3(SP,R)-B)*10
3075 GOSUB30000
3080 IFC=0ANDB<>0THEN3900
3085 IFC=0THEN3100
3090 IFRE(B,C-1)<>0THEN3900
3100 NEXTR
3150 SP=2:IFN(SP)=0THENA=0:GOTO3285
3160 FORR=1TON(SP)
3170 B=INT(A2(SP,R)):C=(A2(SP,R)-B)*10
3175 GOSUB30000
3180 IFC=0THEN3250
3190 IFRE(B,C-1)<>0THEN3250
3200 GOTO3280
3250 GOSUB40000
3265 IFDF$="NEIN"THEN3280
3270 IFB<>0 GOTO3900
3280 NEXTR
3285 SP=1:IFN(1)<2THEN3305
3287 FORR=N(SP)-1TO1STEP-1
3289 B=INT(A2(SP,R)):C=(A2(SP,R)-B)*10
3291 GOSUB30000
3293 IFC=0THEN3297
3294 IFRE(B,C-1)<>0THEN3297
3296 GOTO3301
3297 GOSUB40000
3299 IFDF$="NEIN"THEN3301
3300 IFB<>0 GOTO3900
3301 NEXTR
3305 IFA<>0THENB=A
3310 IFA=0THENB=INT(RND(1)*9+1)
3313 IFRR(B)=0THENC=0:GOTO3320
3315 C=INT(LOG(RR(B))/LOG(10))+2:IFC>9THENA=0:GOTO3310
3320 GOSUB40000
3330 IFDF$="NEIN"THENA=0:GOTO 3310
3400 GOTO3900
3500 SP=2:GOSUB5400:SG$="VERLOREN":GOTO4000
3900 A=B:SP=2:GOSUB5400:GOSUB12000
4000 GOSUB20000:GOSUB5000: RETURN
```

```
5000 REM --- SETZEN ---
5010 ZE=86
5020 IFSP=2THENZE=ZE-5
5050 LE=32768
5060 FORU=0TOA*4-2
5070 POKELE+U,ZE
5080 FORYY=1TO20:NEXTYY :  REM --FALLVERZOEGERUNG--
5090 POKELE+U,32
5100 NEXTU
5200 U=U+40
5210 ZV=PEEK(LE+U)
5220 IFPEEK(LE+U+40)=81ORPEEK(LE+U+40)=86THEN5350
5230 POKELE+U,ZE
5240 FORYY=1TO20:NEXTYY :  REM --FALLVERZOEGERUNG--
5250 IFU>760THEN5380
5260 POKELE+U,ZV
5280 GOTO5200
5350 POKELE+U-40,ZE
5380 RETURN
5400 IFRR(A)=0THENRR(A)=SP:GOTO5450
5410 RR(A)=VAL(STR$(RR(A))+STR$(SP))
5450 LO=LEN(STR$(RR(A)))-2
5460 RE(A,LO)=SP
5500 RETURN
12000 IFSP=2THENZW=-2
12003 ZW=-ZW
12005 D=0:I=1 :FORQ=0TO3
12010 B=A:C=LO+1-Q
12013 IFB<10RC<00RB>90RC>9THEN12190
12015 Y=RE(B,C)
12020 IFQ=0THENAX(I)=B+C/10:I=I+1
12030 IFY=SPTHEND=D+1
12040 IFY<>SPANDY<>0THEN12100
12050 GOTO12190
12100 IFD=3THENFORT=1TOI-1:M(SP)=M(SP)+1:A3(SP,M(SP))=AX(T):GOSUB15000:NEXTT
12120 IFD=ZWTHENFORT=1TOI-1:N(SP)=N(SP)+1:A2(SP,N(SP))=AX(T):GOSUB16000:NEXTT
12140 IFD=4THENSG$="SIEG":GOTO13000
12180 IFQ=4THEN12200
12190 NEXTQ
12195 GOTO12100
12200 GOTO12400
12202 FORW=0TO3
12205 D=0:I=1 :FORQ=0TO3
12210 B=A-3+W+Q:C=LO
12213 IFB<10RC<00RB>90RC>9THEN12380
12215 Y=RE(B,C)
12220 IFY=0THENAX(I)=B+C/10:I=I+1
12230 IFY=SPTHEND=D+1
12240 IFY<>SPANDY<>0THEN12300
12250 GOTO12380
12300 IFD=3THENFORT=1TOI-1:M(SP)=M(SP)+1:A3(SP,M(SP))=AX(T):GOSUB15000:NEXTT
12320 IFD=2THENFORT=1TOI-1:N(SP)=N(SP)+1:A2(SP,N(SP))=AX(T):GOSUB16000:NEXTT
12340 IFD=4THENSG$="SIEG":GOTO13000
12350 D=0:I=1
12360 IFQ=4THEN12390
12380 NEXTQ
12385 GOTO12300
12390 NEXTW
12399 GOTO13000
```

```
12400 FORW=0TO3
12405 D=0:I=1 :FORQ=0TO3
12410 B=A-3+W+Q:C=LO+3-Q-W
12411 GOSUB30000
12413 IFB<1ORC<00RB>90RC>9THEN12580
12415 Y=RE(B,C)
12420 IFY=0THENAX(I)=B+C/10:I=I+1
12430 IFY=SPTHEND=D+1
12440 IFY<>SPANDY<>0THEN12500
12450 GOTO12580
12500 IFD=3THENFORT=1TOI-1:M(SP)=M(SP)+1:A3(SP,M(SP))=AX(T):GOSUB15000:NEXTT
12520 IFD=ZWTHENFORT=1TOI-1:N(SP)=N(SP)+1:A2(SP,N(SP))=AX(T):GOSUB16000:NEXT
12540 IFD=4THENSG$="SIEG":GOTO13000
12550 D=0:I=1
12560 IFQ=4THEN12590
12580 NEXTQ
12585 GOTO12500
12590 NEXTW
12600 FORW=0TO3
12605 D=0:I=1 :FORQ=0TO3
12610 B=A-3+W+Q:C=LO-3+Q+W
12611 GOSUB30000
12613 IFB<1ORC<00RB>90RC>9THEN12780
12615 Y=RE(B,C)
12620 IFY=0THENAX(I)=B+C/10:I=I+1
12630 IFY=SPTHEND=D+1
12640 IFY<>SPANDY<>0THEN12700
12650 GOTO12780
12700 IFD=3THENFORT=1TOI-1:M(SP)=M(SP)+1:A3(SP,M(SP))=AX(T):GOSUB15000:NEXTT
12720 IFD=ZWTHENFORT=1TOI-1:N(SP)=N(SP)+1:A2(SP,N(SP))=AX(T):GOSUB16000:NEXT
12740 IFD=4THENSG$="SIEG":GOTO13000
12750 D=0:I=1
12760 IFQ=4THEN12790
12780 NEXTQ
12785 GOTO12700
12790 NEXTW
12800 GOTO12202
13000 RETURN
15000 :
15003 IFI=1THENM(SP)=M(SP)-1:GOTO15500
15008 IFM(SP)=1THEN15070
15009 FORR=1TOM(SP)-1
15010 IFA3(SP,R)=A3(SP,M(SP))THEN15050
15020 NEXTR
15030 N=INT(A3(SP,M(SP))):M=INT((A3(SP,M(SP))-N)*10+.2)
15040 IFRE(N,M)<>0THENM(SP)=M(SP)-1:GOTO15500
15045 GOTO15070
15050 M(SP)=M(SP)-1:A3(SP,M(SP)+1)=0:GOTO15500
15070 FORR=1TON(SP)
15080 IFA2(SP,R)=A3(SP,M(SP))THEN15100
15090 NEXTR:GOTO15500
15100 N(SP)=N(SP)-1:FORE=RTON(SP)
15110 A2(SP,E)=A2(SP,E+1)
15120 NEXTE:A2(SP,N(SP)+1)=0
15500 RETURN
16000 IFI=1THENN(SP)=N(SP)-1:GOTO16500
16002 IFN(SP)=1THEN16200
16004 N=INT(A2(SP,N(SP))):M=INT(A2(SP,N(SP))-N+.2)
16006 IFRE(N,M)<>0THEN16100
16010 FORR=1TON(SP)-1
16020 IFA2(SP,N(SP))=A2(SP,R)THEN16100
16040 NEXTR
```

```
16060 GOTO16200
16100 N(SP)=N(SP)-1:A2(SP,N(SP)+1)=0
16150 GOTO16500
16200 FORR=1TOM(SP)
16220 IFA2(SP,N(SP))=A3(SP,R)THEN16300
16240 NEXTR
16260 GOTO16500
16300 N(SP)=N(SP)-1:A2(SP,N(SP)+1)=0
16500 RETURN
20000 ZX=A+LO/10
20020 FORR=1TOM(1)
20040 IFA3(1,R)=ZXTHEN20080
20060 NEXTR:GOTO20200
20080 FORE=RTOM(1)-1
20100 A3(1,E)=A3(1,E+1)
20120 NEXTE:A3(1,M(1))=0: M(1)=M(1)-1
20200 FORR=1TON(1)
20240 IFA2(1,R)=ZXTHEN20280
20260 NEXTR:GOTO20400
20280 FORE=RTON(1)-1
20300 A2(1,E)=A2(1,E+1)
20320 NEXTE:A2(1,N(1))=0: N(1)=N(1)-1
20400 FORR=1TON(2)
20440 IFA2(2,R)=ZXTHEN20480
20460 NEXTR:GOTO20600
20480 FORE=RTON(2)-1
20500 A2(2,E)=A2(2,E+1)
20520 NEXTE:A2(2,N(2))=0: N(2)=N(2)-1
20600 FORR=1TOM(2)
20640 IFA3(2,R)=ZXTHEN20680
20660 NEXTR:GOTO21000
20680 FORE=RTOM(2)-1
20700 A3(2,E)=A3(2,E+1)
20720 NEXTE:A3(2,M(2))=0: M(2)=M(2)-1
21000 RETURN
30000 C=INT(C+.2):RETURN
40000 REM ---KEIN SIEG FUER GEGNER---
40050 DF$=""
40070 IFRR(B)=0THENC=1:GOTO40200
40100 C=INT(LOG(RR(B))/LOG(10))+2
40200 FORF=1TOM(1)
40300 IFA3(1,F)=B+C/10THENDF$="NEIN"
40400 NEXTF
40500 IFC>9THENDF$="NEIN"
41000 RETURN
50000 PRINT"■":PRINT:PRINT
50005 PRINT"■                                          "
50010 PRINT"■     VIER GEWINNT  -  SPIELREGELN         ■"
50015 PRINT"■                                          "
50030 PRINT:PRINT:PRINT:PRINT
50050 PRINT"AUF DEM BILDSCHIRM ERSCHEINT GLEICH DAS "
50060 PRINT"SPIELFELD. ES BESTEHT AUS 9 NUMERIERTEN "
50070 PRINT"SPALTEN. DER COMPUTER UND SIE LASSEN    "
50080 PRINT"ABWECHSELND IHRE 'SPIELSTEINE' IN DIESE "
50090 PRINT"SPALTEN FALLEN. DABEI MUESSEN SIE       "
50100 PRINT"VERSUCHEN, ALS ERSTER  V I E R  STEINE  "
50110 PRINT"IN EINER  R E I H E  ZU BEKOMMEN."
50150 GETZ$:IFZ$=""THEN50150
```

```
50200 PRINT"■":PRINT:PRINT
50220 PRINT"DIESE REIHE KANN DIAGONAL, HORIZOTAL     "
50230 PRINT"ODER VERTIKAL LIEGEN."
50250 PRINT:PRINT:PRINT
50300 PRINT"IHR SPIELSTEIN:  ' x '"
50310 PRINT
50320 PRINT"DER SPIELSTEIN DES COMPUTERS:  ' ● '"
50330 PRINT:PRINT:PRINT"IHR SPIELSTEIN FAELLT IN EINE SPALTE,    "
50340 PRINT"SOBALD SIE DEREN NUMMER GEDRUECKT HABEN."
50350 FORZZ=1TO5:PRINT:NEXTZZ
50500 PRINT"  VIEL  GLUECK !!"
50600 GETZ$:IFZ$=""THEN50600
52000 RETURN
```

Geben Sie das Programm ein und spielen Sie damit: Sie werden Spannung und Freude erleben. Nur einen Anspruch erheben Sie bitte nicht, nämlich den, das Programm verstehen zu wollen.
Es soll noch einmal - als Gegenbeispiel, d.h. als Beispiel dafür, wie man es nicht machen soll - das Anliegen dieses Buchs verdeutlichen:

Verständlichkeit.

Verständlich programmieren heißt: strukturiert programmieren - so, wie es die voraufgegangenen Programme vorgemacht haben. Wenn Sie ein schönes Programm fertiggestellt haben, dann nehmen Sie sich bitte noch einmal die d o p p e l t e Zeit und sagen sich: "so, jetzt schreibe ich das ganze Programm noch einmal völlig neu - und zwar so, daß ich selbst nach vier Wochen noch, und daß jeder andere das Programm ohne Mühe verstehen kann." Dies ist eine schwere Arbeit, vielleicht schwerer als die Erstellung der ersten Fassung. Aber ohne sie ist das Programm wertlos.

7

Gemischte Strategien

In diesem Kapitel lernen wir einige Spiele kennen, die zur Klasse der Spiele mit sogenannter g e m i s c h t e r S t r a t e g i e gehören: es handelt sich nicht um reine Glücksspiele (wie in Kapitel 5), aber auch nicht um reine Strategiespiele (wie in Kapitel 6), sondern um eine Zwischenform. Sowohl der Zufall als auch die Entscheidungen der Spieler beeinflussen den Spielausgang; doch ist die Rolle des Zufalls hier eine andere: während er bei den Glücksspielen sozusagen als Schicksal auftritt, dem die Spieler mehr oder weniger ausgeliefert sind, wird er hier als Teil der Strategie konstruktiv eingesetzt.

Beispiel 1: Einfaches Pfennigknobeln

Armin und Bettina spielen nach folgenden Regeln: sie strecken gleichzeitig jeweils einen oder zwei Finger aus. Armin gibt Bettina soviele Pfennige, wie insgesamt Finger gezeigt wurden, wenn diese Anzahl eine g e r a d e Zahl ist; ist die Anzahl der insgesamt gezeigten Finger ungerade, so zahlt Bettina diese Anzahl Pfennige an Armin.
Der Part Armins soll vom Computer übernommen, d.h. dieser soll zu einem möglichst intelligenten Spieler ausgebildet werden.

Folgende vier Fälle sind möglich:

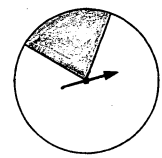
1. Beide zeigen einen Finger: Armin zahlt an Bettina 2 Pfennige.
2. Armin zeigt einen, Bettina zwei Finger: Bettina zahlt an Armin 3 Pfennige.

3. Armin zeigt zwei, Bettina einen Finger: Bettina zahlt an Armin 3 Pfennige.
4. Beide zeigen zwei Finger: Armin zahlt an Bettina 4 Pfennige.

Dieser Sachverhalt läßt sich durch folgende Auszahlungstabelle übersichtlich darstellen (Auszahlungen an Armin positiv, Auszahlungen von Armin an Bettina negativ):

		Bettina	
	F	1	2
Armin	1	-2	3
	2	3	-4

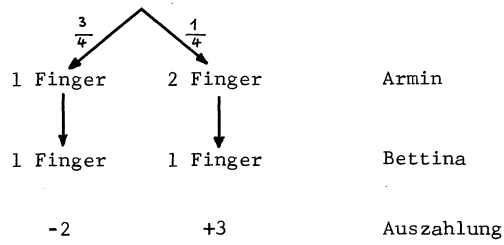
Das Spiel scheint f a i r zu sein, denn die Summe der Auszahlungen von A an B ist gleich der Summe der Auszahlungen von B an A.-
Wie muß nun jeder spielen, um möglichst gut dazustehen?
Sehr schnell wird Armin erkennen, daß er nicht stets einen Finger zeigen darf; denn sobald dies Bettina durchschaut hätte, würde sie dem ebenfalls einen Finger entgegensetzen: Armin hätte bei jeder Partie einen Verlust von zwei Pfennigen. Umgekehrt wird auch Bettina nicht stets einen und denselben Finger ausstrecken, weil dann Armin auf die entsprechende, für ihn günstige Weise reagiert.
Armin wird also zwischen dem Zeigen eines und dem Zeigen zweier Finger irgendwie abwechseln müssen. Etwa so: 1 2 1 2 1 2 1 2 1 2 ... - oder so: 1 1 2 1 1 2 1 1 2 1 1 2 ... - oder ... ? Auch dies würde Bettina nach einiger Zeit heraushaben und geeignet reagieren. In Armins 'Fingerfolge' darf also k e i n M u s t e r erkennbar sein, es muß r e g e l l o s aussehen. Dies erreicht er am besten, indem er die Anzahl der Einsen und der Zweien vom Z u f a l l bestimmen läßt; etwa durch ein Glücksrad, das er vor jeder Partie dreht.
Analog geht Bettina vor.
Bleibt folgende Frage: wie ist das Glücksrad aufzuteilen, d.h. in welchem Verhältnis sind 1 und 2



zu mischen? Wir vermuten: es sind etwas mehr Einsen als Zweien zu nehmen, weil hier der etwaige Verlust geringer ist.
Um die Frage zu beantworten, simulieren wir das Spiel in folgender Weise; das Programm auf der folgenden Seite liefert den Dialog:

```
        Simulation des Pfennigknobelns

Wieviele Partien? 100
Häufigkeit, mit der Spieler A einen Finger zeigt? 0.75
Häufigkeit, mit der Spieler B einen Finger zeigt? 1
Mittlere Auszahlung für Spieler A pro Partie: - 0.6

Wieviele Partien? 100
Häufigkeit, mit der Spieler A einen Finger zeigt? 0.5
Häufigkeit, mit der Spieler B einen Finger zeigt? 0
Mittlere Auszahlung für Spieler A pro Partie: - 0.64

Wieviele Partien? 500
Häufigkeit, mit der Spieler A einen Finger zeigt? 0.6
Häufigkeit, mit der Spieler B einen Finger zeigt? 0.5
Mittlere Auszahlung für Spieler A pro Partie: 0.11
```

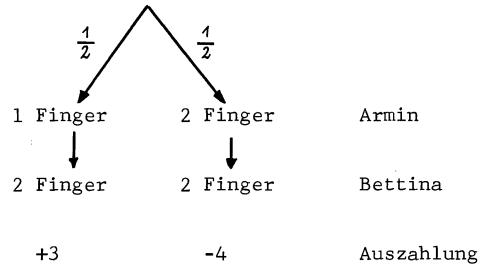
Wir wollen diese Ergebnisse nun durch Rechnung erhärten. Angenommen, Armin wählt in $\frac{3}{4}$ der Fälle einen Finger und in $\frac{1}{4}$ der Fälle zwei Finger (in zufälliger Reihenfolge). Dann kann ihm Bettina die reine Strategie "immer einen Finger zeigen" entgegensetzen, wie das Baumdiagramm zeigt:



```
100 : PRINT "
110 : PRINT "  SIMULATION DES PFENNIGKNOBELNS
111 : PRINT "  ------------------------------
112 :
120 REM DER COMPUTER SPIELT N PARTIEN DES PFENNIGKNOBELNS NACH,
121 REM UM DIE OPTIMALE MISCHUNG VON 1 UND 2 ZU ERMITTELN
122 :
200 REM === EINGABEN UND ANFANGSWERTE ================================
201 :
210 : PRINT
220 : INPUT "WIEVIELE PARTIEN "; N
230 :
240 : INPUT "HAEUFIGKEIT, MIT DER SPIELER A EINEN FINGER ZEIGT "; PA
245 : IF PA < 0 OR PA > 1 THEN PRINT "ZAHL ZWISCHEN 0 UND 1!": GOTO 240
250 : INPUT "HAEUFIGKEIT, MIT DER SPIELER B EINEN FINGER ZEIGT "; PB
255 : IF PB < 0 OR PB > 1 THEN PRINT "ZAHL ZWISCHEN 0 UND 1!": GOTO 240
260 :
270 : LET A(1,1) = -2 : LET A(1,2) =  3 : REM AUSZAHLUNGSTABELLE
275 : LET A(2,1) =  3 : LET A(2,2) = -4
280 :
290 : LET ZZ = RND(-TI) : REM INITIALISIERUNG DES ZUFALLSGENERATORS
299 :
300 REM === SPIELDURCHFUEHRUNG =======================================
301 :
310 : LET A = 0 : REM AUSZAHLUNG AN SPIELER A
320 :
330 : FOR I = 1 TO N
340 :
350 :     LET ZA = RND(1) : LET ZB = RND(1)
360 :
370 :     IF ZA > PA THEN 440
380 :
390 :   REM --- SPIELER A ZEIGT EINEN FINGER
400 :
410 :     IF ZB <= PB THEN LET A = A + A(1,1) : GOTO 490
420 :                      LET A = A + A(1,2) : GOTO 490
430 :
440 :   REM --- SPIELER A ZEIGT ZWEI FINGER
450 :
460 :     IF ZB <= PB THEN LET A = A + A(2,1) : GOTO 490
470 :                      LET A = A + A(2,2)
480 :
490 : NEXT I
499 :
500 REM === AUSWERTUNG ==============================================
501 :
510 : PRINT
520 : LET M = A/N                    : REM MITTLERE AUSZAHLUNG
530 : LET M = INT(M*100 + 0.5)/100 : REM RUNDEN
540 : PRINT "MITTLERE AUSZAHLUNG FUER A PRO PARTIE: "; M
590 :
599 : END
```

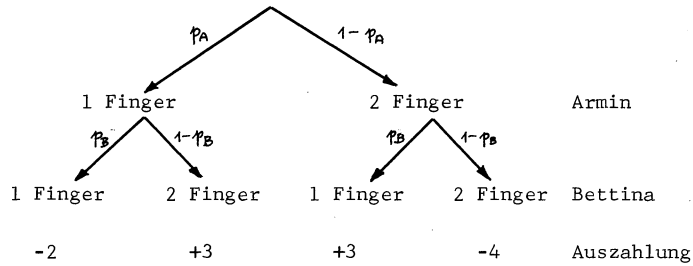
Bettina gewinnt also wegen $-2\cdot\frac{3}{4} + 3\cdot\frac{1}{4} = -\frac{3}{4}$ bei dieser Spielweise auf lange Sicht $\frac{3}{4}$ Pf. pro Partie (die Simulation ergab einen Verlust von 0.6 pro Partie für Armin). Der Anteil der Einsen in Armins Strategie ist also zu hoch.

Angenommen, Armin wählt in 50% der Fälle einen Finger. Dann kann Bettina im die reine Strategie "immer zwei Finger zeigen" entgegensetzen, wie das Baumdiagramm zeigt:



Bei dieser Spielweise gewinnt wegen $3\cdot\frac{1}{2} - 4\cdot\frac{1}{2} = -\frac{1}{2}$ also Bettina auf lange Sicht $\frac{1}{2}$ Pf. pro Partie. (Vergleichen Sie bitte den Wert der Simulation!).

Daraus folgt: wenn es für Armin eine optimale Mischung von 1 zu 2 gibt, so muß der Anteil der 1 zwischen 0.5 und 0.75 liegen. Wie finden wir ihn? Sei p_A die Wahrscheinlichkeit dafür, daß A einen Finger zeigt, und sei p_B die, daß B einen Finger zeigt. Dann sagt das Baumdiagramm



daß $w = (-5p_A + 3)\cdot p_B + (7p_A - 4)(1 - p_B)$ die mittlere Auszahlung pro

Partie auf lange Sicht ist. Armin will sich - unabhängig von der Spielweise Bettinas - eine mittlere Auszahlung sichern; diese muß also für beliebige Werte von p_B gelten, z.B. für $p_B = 1$ und für $p_B = 0$. Daraus folgt: $w = -5p_A + 3$ und $w = 7p_A - 4$ oder $-5p_A + 3 = 7p_A - 4$, was $p_A = \frac{7}{12} \approx 58\%$ ergibt.
Das heißt: wählt Armin in $\frac{7}{12}$ der Fälle 1 Finger, so wird er auf lange Sicht eine gewisse Auszahlung erreichen - unabhängig davon, welche Strategie Bettina anwendet. Ist nun diese Auszahlung positiv oder negativ, d.h. gewinnt oder verliert Armin auf lange Sicht? Die Frage ist leicht beantwortet; wir brauchen nur die gefundene Wahrscheinlichkeit in die Formel für w auf der letzten Seite unten einzusetzen:

$$\begin{aligned} w &= (-5\cdot\tfrac{7}{12} + 3)\cdot p_B + (7\cdot\tfrac{7}{12} - 4)\cdot(1 - p_B) \\ &= -\tfrac{1}{12}p_B + \tfrac{1}{12}(1 - p_B) \\ &= \tfrac{1}{12} \,. \end{aligned}$$

Die mittlere Auszahlung (der garantierte Mindestgewinn auf lange Sicht) an Armin ist also positiv: er wird bei der Strategie "zeige in $\frac{7}{12}$ der Fälle 1 Finger, in $\frac{5}{12}$ der Fälle 2 Finger" auf lange Sicht $\frac{1}{12}$ Pf. pro Partie gewinnen. (Das Spiel ist also doch nicht fair!)
Auf die gleiche Weise läßt sich Bettinas optimale Strategie berechnen: diese muß mit Wahrscheinlichkeit $\frac{7}{12}$ 1 Finger und mit $\frac{5}{12}$ Wahrscheinlichkeit 2 Finger zeigen und erleidet damit einen mittleren Verlust von $\frac{1}{12}$ Pf. pro Partie. Damit ist klar, wie wir programmieren müssen!

```
100 : PRINT "■
110 : PRINT "          PFENNIGKNOBELN
111 : PRINT "          --------------
112 :
120 REM SPIEL MIT GEMISCHTER STRATEGIE
121 :
200 REM === SPIELREGELN UND ANFANGSWERTE ==============================
201 :
210 : PRINT
220 : PRINT "SPIELREGELN:
230 : PRINT "SIE ZEIGEN EINEN ODER ZWEI FINGER; ICH, DER COMPUTER AUCH.
231 : PRINT "IST DIE ANZAHL DER INSGESAMT AUSGESTRECKTEN FINGER GERADE,
232 : PRINT "ZAHLE ICH AN SIE SOVIELE PFENNIGE; IST SIE UNGERADE, ZAH-
233 : PRINT "LEN SIE SOVIELE PFENNIGE AN MICH, WIE DIESE ANZAHL ANGIBT.
239 :
240 : LET KA = 100 : LET KB = KA : REM ANFANGSKAPITAL DER SPIELER
245 :
250 : PRINT "■JEDER VON UNS STARTET MIT EINEM KAPITAL VON";SA;
255 : PRINT "PFENNIGEN.
260 : PRINT "■■WIEVIELE DURCHGAENGE WOLLEN WIR SPIELEN";
265 : INPUT N
269 :
270 : LET G = INT(KA + N/5 + 30/N) : REM GEWINNGRENZE
271 :
280 : PRINT "■UM ZU GEWINNEN, BENOETIGEN SIE";G;" PFENNIGE.
281 :
290 : PRINT "■■SOLL ICH STARK SPIELEN ? (J/N)
291 :
300 : GET A$ : IF A$ = "" THEN 300
310 : IF A$ = "J" THEN LET PA = 7/12 : GOTO 330
320 :                  LET PA = 5/12
321 :
330 : PRINT "■■"TAB(26)"ICH";TAB(34)"SIE
340 : PRINT TAB(25) "──────────────
341 :
350 : LET ZZ = RND(-TI) : REM RANDOMISIERUNG
390 :
400 REM === SPIEL ====================================================
401 :
410 : FOR I = 1 TO N
411 :
420 :     PRINT "■";I". DURCHGANG:
421 :
430 :   REM --- COMPUTERZUG
431 :
440 :     LET Z = RND(1)
450 :     IF Z < PA THEN LET F = 1 : GOTO 480
460 :                    LET F = 2
470 :
480 :   REM --- SPIELERZUG
481 :
490 :     INPUT "IHRE WAHL(1/2)    ";B$
495 :     IF B$ <> "1" AND B$ <> "2" THEN 490
500 :     LET B = VAL(B$)
501 :
```

```
510 :   REM --- AUSWERTUNG
511 :
520 :     PRINT "MEINE WAHL WAR:    "; F
521 :
530 :     IF F <> B THEN LET KA = KA + 3 : LET KB = KB - 3 : GOTO 590
540 :     IF F = 1  THEN LET KA = KA - 2 : LET KB = KB + 2 : GOTO 580
550 :                     LET KA = KA - 4 : LET KB = KB + 4
560 :
570 :     PRINT "■AUSZAHLUNG AN SIE:  4";: GOTO 600
580 :     PRINT "■AUSZAHLUNG AN SIE:  2";: GOTO 600
590 :     PRINT "■AUSZAHLUNG AN MICH: 3";
591 :
600 :     PRINT TAB(26) KA; TAB(34) KB
601 :
610 :   REM --- SPIEL ZU ENDE?
611 :
620 :     IF KB < G THEN 640 : REM DAS SPIEL GEHT WEITER
621 :
630 :     PRINT "■■SIE HABEN GEWONNEN!■" : GOTO 690
631 :
640 : NEXT I
641 :
690 : END
```

*

Die Anwendung gemischter Strategien verhindert Situationen, wie sie E. A. Poe plastisch beschreibt:

"Ich kannte einen Schuljungen von etwa acht Jahren, der durch seine Erfolge im Raten beim Spiel 'Gerade - Ungerade' allgemeine Bewunderung erregte. Das Spiel ist ganz einfach, man nimmt dazu nur ein paar Murmeln. Einer der Spieler hält in seiner Hand eine gewisse Anzahl dieser Kugeln und fragt den Gegenspieler, ob diese Zahl gerade oder ungerade sei. Wer richtig geraten hat, gewinnt eine Murmel; wer falsch geraten hat, verliert eine. Der Junge nun, den ich meine, gewann der ganzen Schule die Murmeln ab. Natürlich befolgte er beim Raten eine Strategie; sie bestand einfach darin, daß er die Gewitztheit seiner Gegenspieler beobachtete und abschätzte.
Ist zum Beispiel ein heilloser Einfaltspinsel sein Gegner, und der hebt die geschlossene Hand und fragt: 'Gerade oder ungerade?', und unser Schuljunge antwortet 'ungerade' und verliert, so wird er beim zweiten Mal gewinnen, denn da sagt er sich: 'Beim ersten Mal hatte der Dummkopf eine gerade Anzahl; seine Schlauheit reicht eben dazu aus, beim zweiten Mal eine ungerade Anzahl wählen zu lassen: darum will ich 'ungerade' raten.' Er tut es und gewinnt.
Bei einem Bürschchen, das einen Grad schlauer ist, würde er die folgende Überlegung angestellt haben: 'Dieser Tölpel weiß, daß ich beim ersten Mal 'ungerade' geraten habe, und so wird er sich im ersten Impuls einen einfachen Wechsel von gerade auf ungerade vornehmen, wie es der andere Einfaltspinsel tat. Aber dann wird ihm ein zweiter Gedanke sagen, daß dies eine zu einfache Variation wäre; er wird sich also schließlich

entscheiden, wie zuvor eine gerade Anzahl zu wählen. Also werde ich jetzt 'gerade' raten.' Er tut es und gewinnt.
Was geht nun eigentlich im Kopf des Schuljungen vor, den seine Kameraden einen Glückspilz nannten; worin besteht seine Denkarbeit? Sie besteht ganz einfach darin, daß er sich mit dem Denken seines Gegners identifiziert."

E. A. Poe
Der entwendete Brief

*

Aufgaben

1. Diesmal spielen Armin und Bettina wie folgt: jeder zeigt zwei oder drei Finger; ist deren Anzahl gerade, zahlt Armin das Produkt, andernfalls zahlt Bettina die Summe an den jeweils anderen. Probieren Sie am Computer aus, für wen das Spiel vorteilhaft ist und ermitteln Sie die optimalen Strategien.

2. Spieler A und Spieler B legen je eine Münze verdeckt auf den Tisch und decken sie anschließend auf. Zeigen beide Münzen KOPF, zahlt A an B 1,5 Geldeinheiten; zeigen beide ZAHL, zahlt A an B 0,5 Geldeinheiten. Erscheint einmal KOPF, einmal ZAHL, so zahlt B an A jeweils eine Geldeinheit. Der Computer soll für A spielen und sich dabei möglichst gut aus der Affaire ziehen.

3. Morra ist die italienische Form des Knobelns und besteht in folgendem: Jeder der beiden Spieler zeigt einen, zwei oder drei Finger und nennt zugleich seine Vermutung über die Summe aus der eigenen Fingeranzahl und der des Gegenspielers.
Zeigen Sie durch Simulation, daß man auf lange Sicht nicht verlieren wird, wenn man immer 'vier' rät und einen Finger in fünf, zwei Finger in vier und drei Finger in drei von zwölf Fällen zeigt.

4. Spieler A hat eine beidseitig begilderte Spielkarte: ihre Vorderseite ist ein rotes As, die Rückseite eine schwarze Fünf. Spieler B besitzt eine Karte mit schwarzer Zwei (Vorderseite) und roter Vier (Rückseite). Jeder wählt eine Seite seiner Karte, beide werden gleichzeitig gezeigt. Stimmen die Farben überein, gewinnt A, andernfalls B; und zwar gewinnt jeder vom anderen soviele Groschen, wie die Kartenseite des Gegners wert ist.
Frage: ist einer der Spieler begünstigt - und wenn ja, welcher? Ein Computerprogramm, welches das Spiel simuliert, kann Ihnen bei der Beantwortung der Frage helfen.

5. Schreiben Sie für das Spiel 'Gerade - ungerade' ein Computerprogramm und versuchen Sie etwas ähnliches wie die Strategie von Poe's Schuljungen!

*

Beispiel 2: Rate die Karte

Es werden 11 Spielkarten mit Werten vom As bis zum Buben benötigt, wobei der Bube den Wert 11 hat. Nach dem Mischen wird eine Karte zufällig gezogen und verdeckt auf den Tisch gelegt; ihr Wert ist den Spielern also unbekannt. Die restlichen 10 Karten werden so auf die Spieler verteilt, daß jeder fünf Karten hat. Ziel des Spieles ist es, die verdeckte Karte zu erraten. Zu diesem Zweck werden Fragen der Art "Hast du die und die Karte?" gestellt. Der gefragte Spieler muß wahrheitsgemäß antworten. Nach einer Karte darf nur einmal gefragt werden. Anstatt zu fragen kann ein Spieler dem Spiel jederzeit dadurch ein Ende bereiten, daß er eine 'Ansage' wagt; sie besteht darin, daß er die verdeckte Karte nennt. Diese wird umgedreht; hat der Ansagende richtig geraten, ist er Sieger, andernfalls Verlierer der Partie.

Martin Gardner kommentiert: "Um gut zu spielen, muß jeder Spieler versuchen, soviel Information wie nur möglich zu bekommen und so wenig wie nur möglich zu enthüllen bis er glaubt, genug zu wissen, um eine Ansage wagen zu können. Der Reiz des Spiels liegt in der Möglichkeit, zu b l u f f e n, d.h. man fragt nach einer Karte, die man selbst besitzt. Würde man niemals bluffen, wüßte der Gegner jedesmal, wenn er nach einer Karte gefragt wird, die sich nicht auf seiner Hand befindet, sofort, daß diese Karte die verdeckte sein muß: er würde eine Ansage machen und gewinnen. Das Bluffen ist daher ein wesentlicher Teil der Strategie, sowohl zur Verteidigung als auch zum Zweck, den Gegner zu einer falschen Ansage zu verleiten.

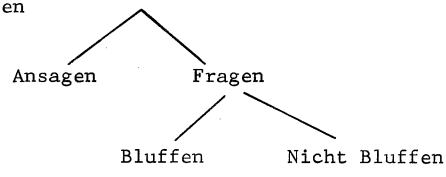
Fragt Spieler A nach einer Karte, sagen wir nach dem Buben, und lautet die Antwort 'ja', dann wissen beide Spieler, daß B diese Karte hat; sie wird daher aus dem Spiel genommen. Hat B den Buben nicht, antwortet er mit 'nein'. Wenn er glaubt, daß A nicht blufft, sagt er den Buben an und gewinnt damit das Spiel, wenn sich seine Vermutung als richtig erweist. Sagt er nicht an, und die verdeckte Karte i s t der Bube, so wird A sie in der nächsten Runde ansagen. Wenn also A in der nächsten Runde den Buben n i c h t ansagt, so bedeutet das nichts anderes, als daß er ihn auf der Hand hat, also vorher geblufft hatte. Da dies nun beiden Spielern bekannt ist, wird auch diese Karte aus dem Spiel genommen. Auf diese Weise verringert sich das Blatt beider Spieler in jeder Runde."

Das Programm, welches wir schreiben wollen, soll einen Dialog wie folgt ermöglichen:

Rate die Karte

```
Sie bekommen die Karten  5  7  2  1  4 .
Möchten Sie anfangen? (j/n) j
Wollen Sie ansagen? (j/n) n
Welche Karte? 3
... die habe ich.
Haben Sie die 11? (j/n) n
Wollen Sie ansagen? (j/n) n
Welche Karte? 6
... die habe ich.
Haben Sie die 2? (j/n) j
Wollen Sie ansagen? n
Welche Karte? 8
... die habe ich.
Haben Sie die 10? n
Wollen Sie ansagen? n
Welche Karte? 9
... die habe ich nicht.
Ich glaube, die verdeckte Karte ist die 1.

Die verdeckte Karte ist die 1.
Gratuliere, Sie haben gewonnen!

Noch eine Partie?
```

Die vom Computer angewandte Strategie läßt sich wie folgt beschreiben:
Er hat die Möglichkeiten



(1) Unter folgenden Bedingungen sagt der Computer an:

a) Die verdeckte Karte ist ihm bekannt
(dies ist der Fall, wenn der Gegner keine Karten mehr hat, oder wenn der Computer - ohne zu bluffen - nach einer Karte gefragt hat, als Antwort 'nein' erhielt und der Gegner die Karte nicht ansagt; im letzteren Fall sagt der Computer diese Karte dann auch an);

b) Der Computer hat keine, der Spieler hat eine oder mehrere Karten (denn in der nächsten Runde wird der Spieler ansagen, weil er die verdeckte Karte kennt);

c) Der Computer hat eine Frage nach einer Karte mit 'nein' beantwortet. In diesem Fall macht er die Entscheidung, ob er ansagt oder nicht, vom Drehen eines Glücksrads abhängig; seine Sektoren sind in einer Wahrscheinlichkeitstabelle p( ) festgehalten.
(Wie die einzelnen Wahrscheinlichkeiten berechnet worden sind, können wir hier leider nicht auseinandersetzen.)
Welche Karte der Computer in den Fällen b) und c) ansagt, bestimmt er durch Zufall.

(2) Wenn der Computer nicht ansagt, so muß (darf) er nach einer Karte des Spielers fragen.
Die Entscheidung, ob er blufft oder nicht, macht er wieder vom Drehen seines Glücksrads abhängig.

```
1000 : PRINT "□
1010 : PRINT "                RATE DIE KARTE
1011 : PRINT "                -------------
1012 :
1020 REM SPIEL NACH RUFUS ISAAKSOHN
1021 :
1200 REM === ANLEITUNG ================================================
1201 :
1210 : PRINT
1220 : PRINT "MOECHTEN SIE DIE SPIELREGELN KENNENLERNEN ?(J/N) ";
1230 : GET A$ : IF A$ = "" THEN 1230
1235 : PRINT A$
1240 : IF A$ <> "J" THEN 1300
1250 :
1260 : REM HIER MUESSEN DIE SPIELREGELN EINGEFUEGT WERDEN
1290 :
1300 REM === INITIALISIERUNG ============================================
1301 :
1310 : LET SP = 0 : REM ANZAHL DER GESPIELTEN PARTIEN
1320 : LET VE = 0 : REM ANZAHL DER VOM COMPUTER VERLORENEN PARTIEN
1340 :
1350 : DIM P(7,7) : REM TABELLE DER WAHRSCHEINLICHKEITEN
1360 : DIM M(11)  : REM TABELLE ZUM KARTENMISCHEN
1370 : DIM C(5)   : REM TABELLE DER KARTEN DES COMPUTERS
1380 : DIM U(6)   : REM TABELLE DER UNBEKANNTEN KARTEN
1390 : DIM A(11)  : REM TABELLE DER AUFGEDECKTEN KARTEN
1399 :
1400 REM === WAHRSCHEINLICHKEITEN ======================================
1401 :
1410 : LET P(0,0) = 0
1420 : FOR I = 1 TO 5
1430 :   FOR J = 1 TO 5
1440 :     LET P(I,J) = (1+J*P(J,I-1)*(1-P(J-1,I)))/(1+(J+2)*P(J,I-1))
1450 :     LET P(J,I) = (1+I*P(I,J-1)*(1-P(I-1,J)))/(1+(I+1)*P(I,J-1))
1460 :   NEXT J
1470 : NEXT I
1490 :
2000 REM === PARTIE ====================================================
2001 :
2005 : REM --- ANFANGSWERTE
2006 :
2010 :   LET SP = SP + 1       : REM ANZAHL DER PARTIEN
2020 :   LET E$ = "WEITER"     : REM FLAGGE, DIE SPIELABBRUCH BESTIMMT
2030 :   LET M$ = "UNBEKANNT" : REM VERDECKTE KARTE (NOCH) UNBEKANNT
2035 :
2040 :   FOR I = 1 TO 11 : LET A(I) = 0 : NEXT I
2050 :
2060 :   LET CK = 5  : REM ANZAHL DER KARTEN DES COMPUTERS
2070 :   LET UK = 6  : REM ANZAHL DER UNBEKANNTEN KARTEN
2080 :   LET AK = 0  : REM ANZAHL DER AUFGEDECKTEN KARTEN
2090 :
2100 : REM --- MISCHEN
2101 :
2110 :   FOR I = 1 TO 11 : LET M(I) = I : NEXT I
2115 :
2120 :   FOR I = 1 TO 11
2130 :     LET R = INT(RND(1)*(12-I)) + I
2140 :     LET V = M(R) : M(R) = M(I) : M(I) = V
2150 :   NEXT I
2190 :
```

```
2200 : REM --- AUSTEILEN
2201 :
2210 :   FOR I = 1 TO 5  : LET C(I) = M(I)   : NEXT I
2220 :
2230 :   FOR I = 6 TO 11 : LET U(I-5) = M(I) : NEXT I
2240 :
2250 :   LET VK = U(6)   : REM VERDECKTE KARTE
2260 :
2270 :   PRINT "■■■SIE HABEN DIE KARTEN: ";
2280 :   FOR I = 1 TO 5 : PRINT U(I);"  ";: NEXT I
2290 :
2300 : REM --- BESTIMMUNG DES ANZIEHENDEN
2301 :
2310 :   LET DR$ = "COMPUTER"
2320 :
2330 :   PRINT
2340 :   PRINT "■WOLLEN SIE ANFANGEN ?(J,N) ";
2350 :   GET A$ : IF A$ = "" THEN 2350
2360 :   PRINT A$
2370 :   IF A$ = "J" THEN LET DR$ = "SPIELER"
2390 :
2400 : REM --- FRAGE UND ANTWORT
2401 :
2410 :   IF DR$ = "SPIELER" THEN 2440 : REM SPIELER BEGINNT
2420 :   GOSUB 3000                   : REM FRAGEN DES COMPUTERS
2430 :   IF E$ = "ENDE" THEN 2500     : REM ZUR ENTHUELLUNG
2440 :   GOSUB 4000                   : REM FRAGEN DES SPIELERS
2450 :   IF E$ <> "ENDE" THEN 2420    : REM PARTIE GEHT WEITER
2490 :
2500 : REM --- ENTHUELLUNG
2501 :
2510 :   PRINT "■■DIE VERDECKTE KARTE IST DIE"; VK; ". ";
2520 :
2530 :   IF GW$ = "COMPUTER" THEN PRINT "ICH HABE GEWONNEN.": GOTO 2600
2540 :   PRINT "GRATULIERE! SIE HABEN GEWONNEN."
2550 :   LET VE = VE + 1
2560 :
2600 : REM --- WIEDERHOLUNG?
2601 :
2610 :   PRINT "■■NOCH EINE PARTIE ? (J/N)";
2620 :   GET A$ : IF A$ = "" THEN 2620
2630 :   PRINT A$
2640 :   IF A$ = "J" THEN 2000
2690 :
2700 : REM --- VERABSCHIEDUNG
2701 :
2710 :   PRINT "■■WIR SPIELTEN"; SP;" PARTIEN,
2720 :   PRINT "DAVON HABEN SIE";VE;" GEWONNEN ";
2730 :   PRINT "UND"; SP - VE;" VERLOREN.
2740 :
2750 :   IF VE < SP/2 THEN PRINT "■■ES WAR NETT MIT IHNEN! ";
2760 :   PRINT "TSCHUESS!"
2770 :
2780 :   END
2781 :
2790 REM === ENDE DER PARTIE ========================================
2798 :
2799 :
```

```
3000 REM +++ UNTERPROGRAMM 'FRAGEN DES COMPUTERS' ++++++++++++++++++++
3001 :
3100 : REM --- ENTSCHEIDUNG, OB ANSAGEN ODER FRAGEN
3101 :
3110 :   IF M$ = "BEKANNT" THEN 3200      : REM VERDECKTE KARTE BEKANNT
3120 :   IF CK = 0 OR UK = 1 OR CK + UK = 3 THEN 3300    : REM ANSAGEN
3130 :   IF RND(1) < 1/(1 + UK * P(UK-1,CK-1)) THEN 3400 : REM BLUFFEN
3140 :   GOTO 3600                                      : REM NICHT BLUFFEN
3150 :
3200 : REM --- COMPUTER SAGT KARTE AN, NACH DER ER VORHER FRAGTE
3201 :
3210 :   LET T = A(L) : REM LETZTE AUFGEDECKTE KARTE IST DER TIP
3220 :   GOSUB 8000   : REM COMPUTER SAGT AN
3225 :
3230 :   RETURN
3290 :
3300 : REM --- COMPUTER SAGT ZUFAELLIG GEWAEHLTE KARTE AN
3301 :
3310 :   LET R = INT(6*RND(1))+1 : IF U(I) = 0 THEN 3310
3320 :   LET T = R : REM ZUFAELLIGE STELLE WIRD ZUM TIP
3330 :   GOSUB 8000  : REM COMPUTER SAGT AN
3340 :
3380 :   RETURN
3390 :
3400 : REM --- COMPUTER BLUFFT
3401 :
3410 :   LET R = INT(5*RND(1))+1 : IF C(R) = 0 THEN 3410
3420 :
3470 :   PRINT "HABEN SIE DIE";C(R);"?(J/N) ";
3480 :   GET A$ : IF A$ = "" THEN 3480
3490 :   PRINT A$
3499 :
3500 :   LET AK = AK + 1   : REM EINE AUFGEDECKTE KARTE MEHR
3510 :   LET A(AK) = C(R)  : REM EINTRAGUNG DER AUFGEDECKTEN KARTE
3520 :   LET C(R) = 0      : REM LOESCHEN DER AUFGEDECKTEN KARTE
3530 :   LET L = AK        : REM LETZTER ZUG
3540 :   LET CK = CK - 1   : REM COMPUTER HAT EINE KARTE WENIGER
3545 :
3550 :   RETURN
3590 :
3600 : REM --- COMPUTER BLUFFT NICHT
3601 :
3610 :   LET R = INT(6*RND(1))+1 : IF U(R) = 0 THEN 3610
3620 :
3670 :   PRINT "HABEN SIE DIE"; U(R);"?(J,N) ";
3680 :   GET A$ : IF A$ = "" THEN 3680
3690 :   PRINT A$
3692 :   IF A$ <>"J" AND A$ <>"N" THEN PRINT "'J' ODER 'N'!":GOTO 3670
3699 :
3700 :   LET AK = AK + 1   : REM EINE AUFGEDECKTE KARTE MEHR
3710 :   LET A(AK) = U(R)  : REM EINTRAGEN DER AUFGEDECKTEN KARTE
3720 :   LET U(R) = 0      : REM LOESCHEN DER AUFGEDECKTEN KARTE
3730 :   LET L = AK        : REM LETZTER ZUG
3740 :   LET UK = UK - 1   : REM EINE UNBEKANNTE KARTE WENIGER
3750 :
3760 :   IF A$ = "N" THEN M$ = "BEKANNT" : REM VERDECKTE KARTE BEKANNT
3770 :
3790 :   RETURN
3791 :
3900 REM ENDE DER FRAGEN DES COMPUTERS ++++++++++++++++++++++++++++++
3990 :
```

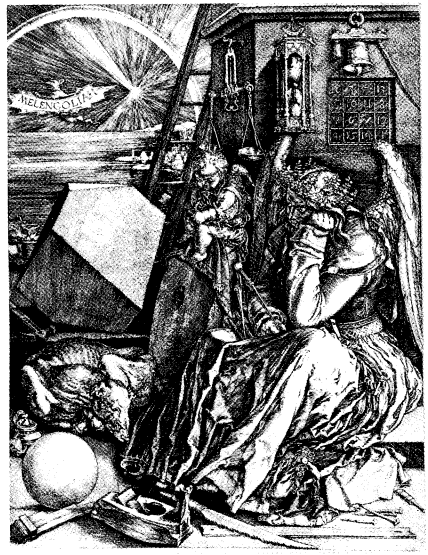
```
4000 REM +++ UNTERPROGRAMM 'FRAGEN DES SPIELERS ++++++++++++++++++++++
4001 :
4100 : REM --- AUFFORDERUNG ZUR FRAGE
4101 :
4120 :   PRINT "WOLLEN SIE DIE VERDECKTE KARTE ANSAGEN ?(J/N) ";
4130 :   GET A$ : IF A$ = "" THEN 4130
4140 :   PRINT A$
4150 :   IF A$ = "J" THEN 7000 : REM SPIELER SAGT AN
4190 :
4200 :   INPUT "WELCHE KARTE "; K
4210 :   IF K < 1 OR K > 11 THEN 4200
4220 :
4230 : REM --- TEST, OB NACH DER KARTE SCHON GEFRAGT WURDE
4231 :
4240 :   LET J = 0
4250 :   FOR I = 1 TO AK
4260 :     IF A(I) = K THEN LET J = I
4270 :   NEXT I
4280 :   IF J > 0 THEN PRINT "NACH DER WURDE SCHON GEFRAGT!": GOTO 4200
4290 :
4300 : REM --- TEST, OB KARTE IN DER HAND DES COMPUTERS
4301 :
4310 :   LET J = 0
4320 :   FOR I = 1 TO 5
4330 :     IF C(I) = K THEN LET J = I
4340 :   NEXT I
4350 :   IF J = 0 THEN 4490 : REM COMPUTER HAT DIE KARTE NICHT
4360 :
4400 :   PRINT "                    ... DIE HABE ICH.
4410 :
4420 :   LET AK = AK + 1   : REM EINE AUFGEDECKTE KARTE MEHR
4430 :   LET A(AK) = C(J) : REM EINTRAGEN DER AUFGEDECKTEN KARTE
4440 :   LET C(J) = 0      : REM LOESCHEN DER AUFGEDECKTEN KARTE
4450 :   LET CK = CK - 1   : REM COMPUTER HAT EINE KARTE WENIGER
4460 :
4480 :   RETURN
4481 :
4490 :   PRINT "                    ... DIE HABE ICH NICHT!
4491 :
4500 : REM --- REAKTION AUF DIE SPIELERFRAGE
4501 :
4530 :   IF CK = 0 OR UK = 1 THEN 4800 : REM COMPUTER SAGT AN
4535 :
4540 :   LET Z = (CK+1) * P(CK, UK-2) - CK * P(CK-1,UK-1)
4545 :   LET N = 1 + (CK+1) * P(CK, UK-2)
4550 :   IF RND(1) < Z/N THEN 4700      : REM COMPUTER SAGT AN
4560 :
4570 :   IF UK = 2 THEN 4800 : REM COMPUTER SAGT ZUFAELLIGE KARTE AN
4580 :
4600 :   LET J = 0
4610 :   FOR I = 1 TO 6
4620 :     IF U(I) = K THEN LET J = I
4630 :   NEXT I
4640 :
4650 :   LET AK = AK + 1   : REM EINE AUFGEDECKTE KARTE MEHR
4660 :   LET A(AK) = U(J) : REM EINTRAGEN DER AUFGEDECKTEN KARTE
4670 :   LET U(J) = 0      : REM LOESCHEN DER AUFGEDECKTEN KARTE
4680 :   LET UK = UK - 1   : REM EINE UNBEKANNTE KARTE WENIGER
4685 :
4690 :   RETURN
4699 :
```

```
4700 : REM --- COMPUTER SAGT LETZTEN SPIELERZUG AN
4701 :
4710 :   LET T = K   : REM LETZTE VOM SPIELER GENANNTE KARTE WIRD TIP
4720 :   GOSUB 8000 : REM COMPUTER SAGT AN
4730 :
4790 :   RETURN
4799 :
4800 : REM --- COMPUTER SAGT LETZTEN SPIELERZUG AN
4801 :
4810 :   LET R = INT(6*RND(1))+1 : IF U(R) = 0 THEN 4810
4820 :   LET T = R : REM ZUFAELLIGE NUMMER WIRD ZUM TIP
4830 :   GOSUB 8000  : REM COMPUTER SAGT AN
4840 :
4890 :   RETURN
4899 :
7000 REM +++ UNTERPROGRAMM 'SPIELER SAGT AN' ++++++++++++++++++++++++++
7001 :
7100 : PRINT "WELCHE KARTE SAGEN SIE AN ";
7110 : INPUT A
7115 :
7120 : LET GW$ = "COMPUTER"                 : REM COMPUTER GEWINNT
7130 : IF A = VK THEN LET GW$ = "SPIELER" : REM SPIELER GEWINNT
7140 : LET E$ = "ENDE"
7150 :
7900 : RETURN
7901 :
7990 REM ENDE DES UNTERPROGRAMMS 'SPIELER SAGT AN' ++++++++++++++++++++
7999 :
8000 REM +++ UNTERPROGRAMM 'COMPUTER SAGT AN' ++++++++++++++++++++++++++
8001 :
8100 : PRINT "ICH GLAUBE, DIE VERDECKTE KARTE IST DIE"; T
8110 :
8120 : LET GW$ = "SPIELER"              : REM SPIELER GEWINNT
8130 : IF T = VK THEN GW$ = "COMPUTER" : REM COMPUTER GEWINNT
8140 : LET E$ = "ENDE"
8150 :
8900 : RETURN
8901 :
8990 REM ENDE DES UNTERPROGRAMMS 'COMPUTER SAGT AN' ++++++++++++++++++
8999 :
```

Aufgaben

6. In das Programm 'Rate die Karte' muß noch eine Sicherung dagegen eingebaut werden, daß der Spieler auf die Frage nach einer eigenen Karte nicht wahrheitsgemäß antwortet. Tun Sie dies!

7. Spieler A wirft eine Münze, verbirgt den Ausfall aber vor Spieler B. Ist KOPF erschienen, so kann er dies melden und von B die Zahlung eines Groschens verlangen. Ist ZAHL erschienen, so kann A bluffen, KOPF melden und einen Groschen kassieren, oder ZAHL melden - und muß dann an B einen Groschen zahlen. Spieler B kann dem A glauben und den Groschen zahlen oder Bestätigung verlangen. Im letzteren Fall muß A die Münze aufdecken: zeigt sie KOPF, zahlt B zwei Groschen, sonst zahlt A zwei Groschen an B. Schreiben Sie ein Programm (Computer = A).



8
Geduldspiele

Man könnte sie auch 'Robinson-Crusoe-Spiele' nennen: es handelt sich um Denkspiele, Gedankenakrobatik für den, der gern mit sich allein spielt; im Englischen nennt man dergleichen 'Puzzles'.
Wir beginnen mit einem Gegenstand, der wie kaum ein anderer ähnlicher Art die Aufmerksamkeit und das Studium nachdenklich veranlagter Menschen herausgefordert hat: die magischen Quadrate.
Ihre Entstehung ist in tiefes Dunkel gehüllt. Vermutlich wurden sie im alten China entdeckt, jene harmonischen Anordnungen natürlicher Zahlen in einem quadratischen Schema derart, daß die Summe jeder Zeile, jeder Spalte und der beiden Diagonalen gleich einer Konstanten ist. Das einfachste und älteste magische Quadrat sieht so aus:

4 9 2
3 5 7
8 1 6

Einer alten Legende zufolge erschien es dem sagenhaften Kaiser Yu der Shang-Dynastie (um 2000 v.Chr.) beim Gang am Flusse Lo (einem Nebenfluß des gelben Flusses) - es heißt daher 'Lo-shu' (Lo-Dokument). Für den Chinesen stellen die geraden Zahlen YIN, das weibliche, die ungeraden Zahlen dagegen YANG, das männliche Prinzip dar. Das Lo-shu verkörpert

die Ausgewogenheit dieser Kräfte, es ist ein Symbol der kosmischen Harmonie.

In der Astrologie des 16. und 17. Jahrhunderts spielten die magischen Quadrate eine wichtige Rolle: man brachte sie in feste Beziehung zu den sieben 'Planeten' der vorkopernikanischen Weltanschauung Mond, Merkur, Venus, Sonne, Mars, Jupiter und Saturn. Letzterem, dem entferntesten der sieben, wurde das kleinstmögliche magische Quadrat in der Gestalt

2	9	4
7	5	3
6	1	8

zugeordnet. Über dies 'Saturnsiegel' schreibt Theophrast von Hohenheim, genannt Paracelsus, folgendes:

"Dies Siegel soll von feinem, puren, lauteren Villacher Blei gemacht werden, auf der einen Seite sein Quadrat mit drei Reihen, und in einer jeglichen Linie soll 15 stehen; auf der anderen Seite soll das Bildnis des Planeten stehen, nämlich ein alter Mann mit langem Bart, der mit einer Grabschaufel in der Erde gräbt, auf seinem Haupt ein Stern und der Name Saturnus.

Fürs erste ist das Siegel gut, wenn eine Frau schwanger ist, und sie es bei sich trägt, kann es ihr in der Geburt nit mißlingen. Zum anderen: wo man dies Siegel hinlegt, das mehrt sich und nimmt zu. Und wenn ein Reitersmann dies Siegel in dem linken Stiefel bei sich trägt, selbigem kann sein Pferd keinen Schaden zufügen.

Wenn aber das Siegel gemacht, wenn Saturnus im Abnehmen ist, an einem Samstag und in seiner Stund, so verhindert es alles gute Vorhaben, und zu was es gelegt wird, das nimmt ab von Tag zu Tag und mindert sich und zergeht. Und wenn es in ein Feldlager unter einen Haufen Kriegsvolk eingegraben wird, werden dieselbigen da nicht viel Glück mehr haben, sondern bald aufbrechen und davonziehen." (!)

*

Die Konstruktion magischer Quadrate ist eines der ältesten mathematischen Probleme überhaupt. Wir wollen es dem Computer anvertrauen.

Beispiel 1: Magische Quadrate

Es soll ein Verfahren zur Konstruktion magischer Quadrate entwickelt und programmiert werden.

Wir beginnen mit einer ungeraden Anzahl von Zeilen (bzw. Spalten). Die Aufgabe besteht also darin, zu gegebenem ungeraden n die Zahlen $1,2,3,\dots, n^2$ so in ein quadratisches Schema einzuordnen, daß die Summe der Zeilen-, der Spalten- und der Diagonalelemente gleich einer Konstanten ist. Diese sogenannte magische Konstante berechnet sich zu

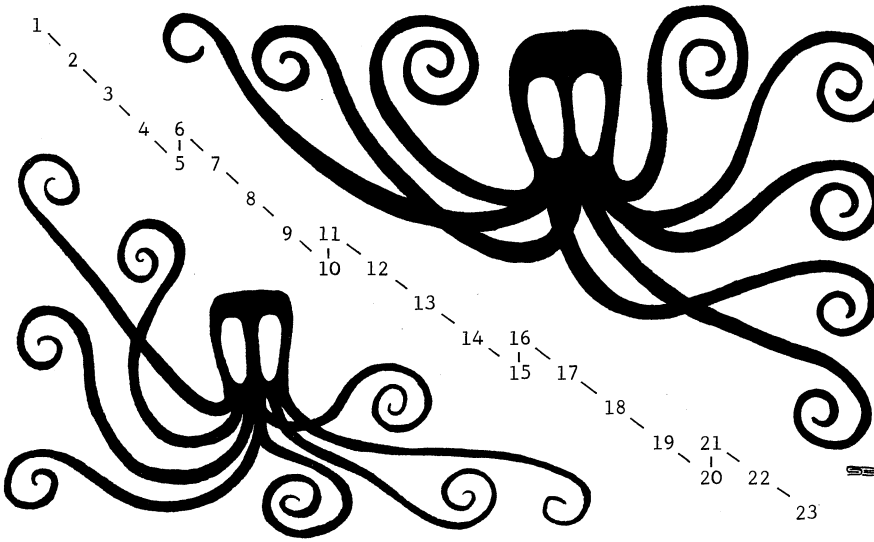
$$m = \frac{1}{n}\cdot(1+2+\dots+n^2) = \frac{1}{n}\cdot\frac{n^2(n^2+1)}{2} = \frac{n(n^2+1)}{2}\ .$$

Im Fall $n = 3$ haben wir $m = 15$, im Fall $n = 5$ dagegen $m = 65$.-

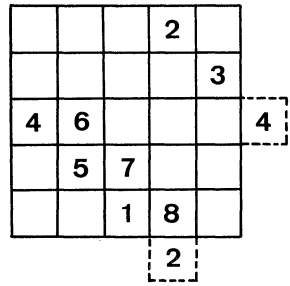
Der nächstliegende Weg zum Algorithmus ist, daß wir ein bekanntes magisches Quadrat analysieren und sein Bildungsgesetz ergründen. Nehmen wir dazu das Lo-shu! Sein Bildungsgesetz erschließt sich uns leichter, wenn wir es zum magischen Parkett erweitern:

4	9	2	4	9	2	4	9	2	4	9	2	4	9	2	4	9	2
3	5	7	3	5	7	3	5	7	3	5	7	3	5	7	3	5	7
8	1	6	8	1	6	8	1	6	8	1	6	8	1	6	8	1	6
4	9	2	4	9	2	4	9	2	4	9	2	4	9	2	4	9	2
3	5	7	3	5	7	3	5	7	3	5	7	3	5	7	3	5	7
8	1	6	8	1	6	8	1	6	8	1	6	8	1	6	8	1	6
4	9	2	4	9	2	4	9	2	4	9	2	4	9	2	4	9	2
3	5	7	3	5	7	3	5	7	3	5	7	3	5	7	3	5	7
8	1	6	8	1	6	8	1	6	8	1	6	8	1	6	8	1	6
4	9	2	4	9	2	4	9	2	4	9	2	4	9	2	4	9	2
3	5	7	3	5	7	3	5	7	3	5	7	3	5	7	3	5	7
8	1	6	8	1	6	8	1	6	8	1	6	8	1	6	8	1	6
4	9	2	4	9	2	4	9	2	4	9	2	4	9	2	4	9	2
3	5	7	3	5	7	3	5	7	3	5	7	3	5	7	3	5	7
8	1	6	8	1	6	8	1	6	8	1	6	8	1	6	8	1	6

Das Auffälligste an diesem Parkett ist der eingezeichnete Pfeilweg. Wir wollen einen analogen Pfeilweg konstruieren, um zu einem magischen Quadrat der Ordnung 5 zu gelangen!



Wie wird daraus nun ein Quadrat? Wir beginnen wie beim Lo-shu mit der 1 im mittleren Feld der untersten Zeile (Feld auf Zeile 5, Spalte 3).



Damit nun die 2 nicht außerhalb des Quadrats zu liegen kommt, stellen wir uns vor, daß dessen unterer und oberer Rand zusammengeklebt sind derart, daß eine Röhre entsteht: dann kommt die 2 in das Feld auf Zeile 1, Spalte 4.

Die 3 kommt, wie auf dem Pfeilweg, nach Zeile 2, Spalte 5. Damit die 4 nicht außerhalb liegt, stellen wir uns vor, daß auch der linke und der rechte Rand zusammengeklebt sind (es entsteht dann eine Art Fahrradschlauch; der Mathematiker nennt dies Gebilde 'Torus'). Nun kommt die 4 in das Feld auf Zeile 3, Spalte 1 - und so geht es weiter. Nach jeweils n plazierten Elementen stoßen wir auf ein schon besetztes Feld: hier weichen wir - siehe Pfeilweg - nach oben aus (Zeilennummer - 1, Spalte bleibt). Das nach diesen Regeln konstruierte Quadrat sieht so aus:

11	18	25	2	9
10	12	19	21	3
4	6	13	20	22
23	5	7	14	16
17	24	1	8	15

Beachten Sie bitte: das Element $\frac{1}{2}\cdot(n^2+1) = 13$ steht in der Mitte. Die Summe je zweier zu diesem Mittelelement symmetrischer Elemente ist $n^2+1 = 26$. -

Dies Konstruktionsverfahren wurde von S. de la Loubère, vormals französischer Gesandter in Siam, im Jahre 1687 von dort mitgebracht; es heißt daher auch die Siamesische Methode. Allgemein lautet sie:

SIAMESISCHE METHODE ZUR KONSTRUKTION MAGISCHER QUADRATE UNGERADER ORDNUNG

1. Schreibe die 1 in das mittlere Feld der n. Zeile
2. Steht eine Zahl auf Feld (i,j), so kommt ihr Nachfolger in Feld (i+1, j+1), sofern dies Feld nicht schon besetzt ist, andernfalls in Feld (i-1, j).
3. Überschreitet ein Index die Ordnung n, so wird er durch 1 ersetzt; unterschreitet er 1, so wird er durch n ersetzt.

Das Programm auf der folgenden Seite ist nach diesem Algorithmus gestrickt.

```
100 : PRINT "□"
110 : PRINT "                    MAGISCHE QUADRATE"
111 : PRINT "                    ----------------"
112 :
120 REM ERZEUGT MAGISCHE QUADRATE UNGERADER ORDNUNG
121 REM NACH DER SIAMESISCHEN METHODE (DE LA LOUBERE)
129 :
200 REM === EINGABE ================================================
201 :
210 : PRINT
220 : INPUT "ORDNUNG (UNGERADE ZAHL) "; N
230 : IF N < 3 OR 2*INT(N/2) = N THEN 220
290 :
300 REM === INITIALISIERUNG ========================================
301 :
310 : DIM Q(N,N)
320 :
330 : LET I = N-1 : LET J = (N-1)/2 : REM VORPOSITION DER EINS
390 :
400 REM === EINTRAGEN DER ZAHLEN ===================================
401 :
410 : FOR U = 0 TO N-1
420 :   FOR V = 0 TO N-1
430 :     IF I < N THEN LET I = I+1 : GOTO 450
440 :                   LET I = 1
450 :     IF J < N THEN LET J = J+1 : GOTO 470
460 :                   LET J = 1
470 :     LET Q(I,J) = U*N+V+1       : REM EINTRAGEN DER ZAHL
480 :   NEXT V
500 :   LET I = I-2 : LET J = J-1    : REM AUSWEICHEN NACH OBEN
520 : NEXT U
590 :
600 REM === AUSGABE ================================================
601 :
610 : PRINT : PRINT
620 : FOR I = 1 TO N
630 :   LET S = 0
640 :   FOR J = 1 TO N
650 :     PRINT TAB(5*J) Q(I,J);
660 :     LET S = S+Q(I,J)
670 :   NEXT J
680 :   PRINT TAB(5*(N+1)) "----"; S
690 :   PRINT
700 : NEXT I
710 :
720 : FOR J = 1 TO N : PRINT TAB(5*J) "-----";: NEXT J
730 :
740 : PRINT
750 : FOR J = 1 TO N
760 :   LET S = 0
770 :   FOR I = 1 TO N : LET S = S+Q(I,J) : NEXT I
780 :   PRINT TAB(5*J) S;
790 : NEXT J
795 :
799 : END
```

Woher wissen wir nun, daß dies Programm magische Quadrate beliebiger ungerader Ordnung erzeugt? Den Mathematiker verlangt es nach einem Beweis. Er sei hier angedeutet.
Schauen wir uns noch einmal den Pfeilweg auf der vorletzten Seite an! Er zerfällt in 5 'Ketten' (1,2,3,4,5), (6,7,8,9,10), (11,12,13,14,15), (16,17,18,19,20), (21,22,23,24,25), die sich allgemein so darstellen lassen:

	Nummer des Kettengliedes v					
Kette u	0	1	2	3	...	n-1
0	1	2	3	4	⋮	n
1	n+1	n+2	n+3	n+4	⋮	2n
2	2n+1	2n+2	2n+3	2n+4	⋮	3n
u	.....				u·n+v+1	
n-1	.....					n^2

Jede Zahl $z \in \{1,2,3,\ldots, n^2\}$ des Quadrats läßt sich demnach in der Form

$$z = u \cdot n + v + 1 \qquad \text{mit} \quad u,v \in \{0,1,2,\ldots,n-1\}$$

eindeutig darstellen; dabei ist u die Nummer der Kette und v die Nummer des Kettenglieds (Rest bei Division von z durch n).
Diese Darstellung haben wir auch im Programm (in den Zeilen 410-520) gewählt.
Das siamesische Verfahren bewirkt nun, daß jedes Feld mit der Kettennummer u und der Gliednummer v folgende Nachbarn besitzt:

u+1, v+1	u+2, v+3
u , v	u+1, v+2

wobei die Nummern modulo n zu nehmen sind. Beispiel (n = 5):

19	21	→	3·5 + 3 + 1	4·5 + 0 + 1	→	3 , 3	4 , 0
13	20		2·5 + 2 + 1	3·5 + 4 + 1		2 , 2	3 , 4

Die Null kommt durch 2 + 3 = 5 = 0 (modulo 5) zustande.

Wir müssen nun zeigen, daß die Summe der i-Zeile, j-ten Spalte und der beiden Diagonalen jeweils gleich m ist. Dies sieht man wie folgt: Beim Summieren der Elemente irgend einer Zeile kommt jede Kettennummer u und jede Gliednummer v genau einmal vor, daher ergibt sich als Summe

$$(0+1+2+\ldots+n-1)\cdot n + (0+1+2+\ldots+n-1) + n\cdot 1$$

$$= \frac{n(n-1)}{2}\cdot n + \frac{n(n-1)}{2} + n = \frac{n^3+n}{2} = m \quad \text{(gemäß der Definition von m)}.$$

Analog findet man die magische Konstante m als Summe irgendeiner Spalte. Das Element in der Mitte ist $(n\cdot n+1)/2$, d.h. der Zentralwert der Folge $1,2,3,\ldots, n^2$ (im Fall $n = 5$ ist dies die 13). Je zwei zum Mittelelement symmetrisch liegende Elemente haben die Summe $(n^2+1)/2$; im Fall $n=5$ ist dies 26. Damit ist die Summe jeder der beiden Diagonalen

$$\frac{1}{2}\cdot(n-1)\cdot(n^2+1) + \frac{n^2+1}{2} = \frac{n^3+n}{2} = m, \quad \text{wie behauptet.}$$

Anzahl der symmetrisch liegenden Elemente

Mittel-element

*

Wir wollen nun ein weiteres Verfahren zur Erzeugung magischer Quadrate kennenlernen, das zugleich einen interessanten Zusammenhang zu den sog. l a t e i n i s c h e n Q u a d r a t e n offenlegt.
Betrachten wir wieder unser Lo-shu! Wie eben festgestellt, läßt sich jedes seiner Elemente in der Gestalt $z = 3\cdot u + v + 1$ mit $u,v \in \{0,1,2\}$ darstellen. In der gleichen Weise können wir das ganze Quadrat zerlegen und erhalten

$$\begin{array}{|ccc|} \hline 4&9&2\\ 3&5&7\\ 8&1&6\\ \hline \end{array} = 3\cdot \begin{array}{|ccc|} \hline 1&2&0\\ 0&1&2\\ 2&0&1\\ \hline \end{array} + \begin{array}{|ccc|} \hline 0&2&1\\ 2&1&0\\ 1&0&2\\ \hline \end{array} + \begin{array}{|ccc|} \hline 1&1&1\\ 1&1&1\\ 1&1&1\\ \hline \end{array}$$

Diese Quadrate zeigen einen äußerst interessanten Aufbau! Das erste rechts vom Gleichheitszeichen enthält drei Zeilen, die durch zyklische Vertauschung auseinander hervorgehen. In jeder Zeile und jeder Spalte

kommt jede der Zahlen 0,1,2 genau einmal vor. Ein Schema mit dieser Eigenschaft (bei ggf. anderen Elementen) heißt l a t e i n i s c h e s Q u a d r a t . Das nächste Quadrat können wir uns durch Drehung um 90° im Gegenuhrzeigersinn aus dem ersten entstanden denken. Legen wir beide übereinander, so erhalten wir

10 22 01
02 11 20
21 00 12

Wir stellen fest, daß jedes der Paare uv mit $u,v \in \{0,1,2\}$ genau einmal vorkommt. Ein solches Quadrat heißt E u l e r - Q u a d r a t; zwei lateinische Quadrate, die durch Übereinanderlegen ein Euler-Quadrat ergeben, heißen zueinander o r t h o g o n a l . Wir stellen fest:

Aus zueinander orthogonalen lateinischen Quadraten läßt sich stets ein magisches Quadrat konstruieren.

Und wie kommt man nun zu orthogonalen lateinischen Quadraten?
Für ungerade Ordnung n geht sicher folgendes:

' MAGISCHE QUADRATE AUS ORTHOGONALEN LATEINISCHEN QUADRATEN
BEI UNGERADER ORDNUNG

1. Die erste Zeile wird beliebig mit den Zahlen 0,1,...,n-1 gefüllt.
2. Die Zeilen werden zyklisch verschoben:
   aus $a_0, a_1, a_2, \ldots a_{n-1}$ wird $a_{n-1}, a_0, a_1, \ldots, a_{n-2}$
   u.s.w. Ergebnis: A.
3. Das Quadrat A wird um 90° gedreht, Ergebnis: B
4. $Q = n \cdot A + B + (1)$

Das Programm auf der folgenden Seite realisiert diesen Algorithmus.
Es lieferte beispielsweise

12 24 1 20 8
9 11 25 3 17
16 10 13 22 4
5 18 7 14 21
23 2 19 6 15

```
100 : PRINT "□"
110 : PRINT "                    MAGISCHES QUADRAT
111 : PRINT "                    ----------------
112 :
120 REM KONSTRUIERT MAGISCHE QUADRATE NACH DER EULER-METHODE
121 REM DIE SEITENLAENGE MUSS EINE UNGERADE ZAHL SEIN
190 :
200 REM === EINGABE ================================================
201 :
210 : PRINT
220 : INPUT "SEITENLAENGE (UNGERADE ZAHL) "; N
225 : IF N < 3 OR 2*INT(N/2) = N THEN 220
230 :
240 : DIM V(N), A(N,N), Q(N,N)
245 :
250 : LET R = RND (-TI) : REM RANDOMISIERUNG
290 :
300 REM === BERECHNUNG =============================================
301 :
310 : REM --- EINSETZEN VON ZUFALLSZIFFERN ZWISCHEN 0 UND N
311 :
320 :   FOR J = 1 TO N
330 :     LET V(J) = INT(N*RND(1))
340 :     FOR K = 1 TO J-1
350 :       IF V(J-K) = V(J) THEN 330
360 :     NEXT K
370 :   LET A(1,J) = V(J)
380 :  NEXT J
390 :
400 : REM --- ZYKLISCHES RECHTSSCHIEBEN
401 :
410 :   FOR I = 2 TO N
420 :     FOR J = 1 TO N
430 :       LET L = J-1-N*INT((J-2)/N)
440 :       LET A(I,J) = A(I-1,L)
450 :     NEXT J
460 :   NEXT I
490 :
500 : REM --- DREHEN UND MULTIPLIZIEREN
501 :
510 :  FOR I = 1 TO N
520 :    FOR J = 1 TO N
530 :      LET Q(I,J) = N*A(I,J)+A(J,N-I+1)+1
540 :    NEXT J
550 :  NEXT I
590 :
700 REM === AUSGABE ================================================
701 :
710 : PRINT
720 : FOR I = 1 TO N
725 :   LET S = 0
730 :   FOR J = 1 TO N
740 :     PRINT TAB(5*J) Q(I,J);
745 :     LET S = S + Q(I,J)
750 :   NEXT J
755 :   PRINT TAB(5*(N+1)) "---";S
760 :   PRINT
770 : NEXT I
775 :
```

```
780 : FOR J = 1 TO N : PRINT TAB(5*J) "-----";: NEXT J
785 :
786 : PRINT
790 : FOR J = 1 TO N
800 :   LET S = 0
810 :   FOR I = 1 TO N
820 :     LET S = S + Q(I,J)
830 :   NEXT I
840 :   PRINT TAB(5*J) S;
850 : NEXT J
860 :
870 : PRINT : PRINT
880 :
890 : END
```

*

Die Konstruktion magischer Quadrate g e r a d e r Ordnung ist wesentlich schwieriger als die ungerader Ordnung. Es gibt dazu ausgeklügelte mathematische Verfahren; wir wollen hier einen anderen Weg einschlagen.

```
100 : PRINT "□
110 : PRINT "              MAGISCHES QUADRAT
111 : PRINT "              ----------------
112 :
120 REM KONSTRUIERT MAGISCHE QUADRATE GERADER ORDNUNG
129 :
200 REM === EINGABE UND INITIALISIERUNG ==============================
201 :
210 : PRINT
220 : INPUT "SEITENLAENGE (GERADE) "; N
230 : IF N < 4 OR N > 20 OR 2*INT(N/2) <> N THEN 220
235 :
240 : LET H = N/2 : REM HALBE SEITENLAENGE
245 :
250 : DIM Q(N,N), C(N,H)
260 :
270 : FOR I = 1 TO N : FOR J = 1 TO H : LET C(I,J) = 0 : NEXT J,I
290 :
300 REM === ERZEUGUNG ================================================
301 :
305 : REM --- EINSETZEN DER ZAHLEN, MAGISCHE SUMME DER ZEILEN
306 :
310 :   LET X = 0
320 :
330 :   FOR I = 1 TO N
340 :     FOR J = 1 TO H
350 :       LET X = X+1 : LET Q(I,J) = X
360 :     NEXT J
370 :     FOR J = H+1 TO N
380 :       LET X = X+1 : LET Q(N-I+1,J) = X
390 :     NEXT J
395 :   NEXT I
399 :
```

```
400 : REM --- MAGISCHE SUMME DER DIAGONALEN
401 :
405 :   LET J = H
410 :   FOR I = H TO 1 STEP -1
420 :     LET J = J+1
430 :     LET V = Q(I,J): LET Q(I,J) = Q(N-I+1,J): LET Q(N-I+1,J) = V
440 :   NEXT I
450 :
500 :   LET I = H-1
510 :   FOR J = 1 TO H
520 :     LET I = I+1
530 :     IF I > H THEN LET I = 1
540 :     LET V = Q(I,J): LET  Q(I,J) = Q(N-I+1,J): LET Q(N-I+1,J) = V
550 :   NEXT J
559 :
560 : REM --- MAGISCHE SUMME DER SPALTEN
561 :
565 :   LET T = H
570 :   IF H = 2*INT(H/2) THEN 620 : REM N IST DURCH 4 TEILBAR
580 :
590 :   LET I = H-1 : LET T = T-1
595 :   FOR J = 1 TO H
600 :     LET I = I+1
605 :     IF I > H THEN LET I = 1
610 :     LET C(I,J) = 1
615 :   NEXT J
619 :
620 :   LET T = T/2
630 :   FOR K = 1 TO T
640 :     LET I = K
650 :     FOR J = 1 TO H
660 :       LET I = I+1
670 :       IF I > H THEN LET I = 1
680 :       LET C(I,J) = 1
685 :       LET C(N-I+1, J) = 1
690 :     NEXT J
695 :   NEXT K
699 :
700 :   FOR I = 1 TO N
710 :     FOR J = 1 TO H
720 :       IF C(I,J) <> 1 THEN 730
725 :       LET V = Q(I,J) : Q(I,J) = Q(I,N-J+1) : Q(I,N-J+1) = V
730 :     NEXT J
740 :   NEXT I
790 :
800 REM === AUSGABE ================================================
801 :
805 : REM --- ZEILEN
806 :
809 :   PRINT
810 :   FOR I = 1 TO N
815 :     LET S = 0
820 :     FOR J = 1 TO N
825 :       PRINT TAB(5*J) Q(I,J);
830 :       LET S = S + Q(I,J)
835 :     NEXT J
840 :     PRINT TAB(5*(N+1)) "---"; S
845 :     PRINT
847 :   NEXT I
848 :
849 :   PRINT "□";
850 :   FOR J = 1 TO N : PRINT TAB(5*J) "-------";: NEXT J
855 :
```

```
860 : REM --- SPALTENSUMMEN
861 :
864 :   PRINT
865 :   FOR J = 1 TO N
870 :     LET S = 0
875 :     FOR I = 1 TO N : LET S = S + Q(I,J) : NEXT I
880 :     PRINT TAB(5*J) S;
885 :   NEXT J
890 :
900 : REM --- DIAGONALENSUMMEN
901 :
905 :   LET S1 = 0 : LET S2 = 0
910 :   FOR I = 1 TO N
920 :     LET S1 = S1 + Q(I,I)
930 :     LET S2 = S2 + Q(I, N-I+1)
940 :   NEXT I
950 :
960 :   PRINT "***";S1; S2
970 :
990 :   END
```

Erläuterung des Programms: Zuerst wird das Quadrat Q mit den Zahlen, 1,2,3,..., n^2 gefüllt, und zwar derart, daß die Zeilensummen gleich der magischen Konstanten sind (111 bei n = 6); dies geschieht in den Zeilen 310 - 395 mit nebenstehendem Ergebnis.

1	2	3	34	35	36
7	8	9	28	29	30
13	14	15	22	23	24
19	20	21	16	17	18
25	26	27	10	11	12
31	32	33	4	5	6

Nun sollen die Diagonalen magisch gemacht werden, doch so, daß die Zeilensummen nicht mehr verändert werden (405 - 550). Dabei werden der rechte Teil der Haupt- und der Nebendiagonale ausgetauscht und auf der linken Seite ein Ausgleich dafür geschaffen (wie nebenstehend gezeigt).

1	32	3	34	35	6
7	8	27	28	11	30
19	14	15	16	23	24
13	20	21	22	17	18
25	26	9	10	29	12
31	2	33	4	5	36

Am schwierigsten ist die Behandlung der Spalten (Zeilen 565 - 740). Zunächst wird die Matrix C an den Stellen mit einer 1 besetzt, die im magischen Quadrat vertauscht werden sollen; hierfür ist nur die linke Hälfte erforderlich, da die Vertauschungen symmetrisch zur Mittellinie vorgenommen werden. Der interessierte Leser wird vielleicht das Prinzip, das hinter diesem Vorgehen steckt, durch eigenes Nachdenken und Probieren entdecken.

Aufgaben

1. Zeigen Sie, daß das berühmte Dürer-Quadrat

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

sich nicht nach der Eulerschen Methode konstruieren läßt.

2. Von Cornelius Agrippa (Agrippa von Nettesheim, 1486 - 1535) stammt eine Methode zur Konstruktion magischer Quadrate ungerader Ordnung, die im Fall $n = 5$ folgendes liefert:

11	24	7	20	3
4	12	25	8	16
17	5	13	21	9
10	18	1	14	22
23	6	19	2	15

Erweitern Sie das Quadrat zum magischen Parkett, entdecken Sie das Bildungsgesetz und schreiben Sie ein Programm!

3. Zeigen Sie, daß für magische Quadrate M, M_1, M_2 folgende Rechenregeln gelten:
Mit M_1 und M_2 ist auch

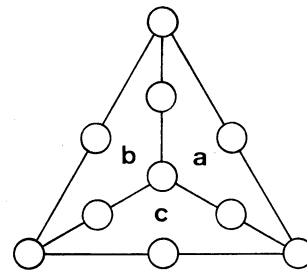
$$M = a + M_1$$

$$M = a \cdot M_1$$

$$M = M_1 + M_2$$

wieder magisch. (Dabei ist Addition und Multiplikation mit einer Zahl komponentenweise zu verstehen.)

4. In die 10 Felder sind die Zahlen 1,2,..., 10 so zu verteilen, daß sich für die drei Teildreiecke a,b, und c jeweils die gleiche Summe aus den je 6 umliegenden Zahlen ergibt. Programm gesucht!

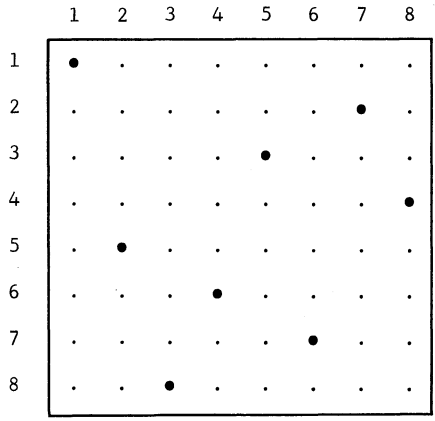


Berühmt und lehrreich ist folgendes klassische Problem:

Beispiel 2 : n - Damen - Problem

Man stelle auf ein n x n - Schachbrett n Damen (Königinnen) so auf, daß sie sich gegenseitig nicht schlagen können, d.h. so, daß keine zwei Damen in der gleichen Zeile, Spalte oder Diagonale stehen.

Auch der große Carl Friedrich G a u ß hat sich mit dieser Aufgabe für den Fall n = 8 beschäftigt; und natürlich interessierte er sich nicht nur für eine, sondern für a l l e Lösungen, mehr noch: er suchte einen Algorithmus zur Erzeugung aller Lösungen. Der Gauß'sche Algorithmus läßt sich wie folgt beschreiben: "Man setze von links nach rechts fortschreitend in jede Spalte des Schachbretts eine Dame, und zwar jedesmal an die tiefstmögliche Stelle. Wenn nun der Augenblick kommt, wo man keine Dame in einer Spalte aufstellen kann, so erhöhe man den Platz der Dame in der v o r h e r g e h e n d e n Spalte solange, bis man fortfahren kann."
Um diesen Algorithmus durch ein Programm ausführen zu lassen, müssen wir zunächst eine geeignete Bezeichnung für eine Damen-Konstellation finden.



Obige Lösung kennzeichnen wir durch die Tabelle

Spalte i	1	2	3	4	5	6	7	8
Zeile d(i)	1	5	8	6	3	7	2	4

oder kürzer durch die Permutation 1 5 8 6 3 7 2 4 . In dieser Gestalt soll der Computer die Lösungen auf dem Bildschirm ausgeben. Das Gauß'sche Verfahren lautet nun:

N - DAMEN - PROBLEM

```
spalte := 1
zeile(spalte) := 0
SOLANGE  spalte > 0  WIEDERHOLE
  WIEDERHOLE
    zeile(spalte) := zeile(spalte) + 1
  BIS  nicht bedroht  ODER  zeile(spalte) > n  ENDE-WIEDERHOLE
  WENN  zeile(spalte) > n  DANN
    spalte := spalte - 1   (* Rückzug *)
  WENNABER  spalte = n  DANN
    ausgabe                (* Lösung gefunden *)
    spalte := spalte - 1
  SONST
    spalte := spalte + 1   (* neue Dame *)
    zeile(spalte) := 0
  ENDE-WENN
ENDE-WIEDERHOLE
```

Statt "d(i)" haben wir hier "zeile(spalte)" geschrieben. Wäre uns die Programmiersprache PASCAL vertraut, könnten wir diesen Algorithmus unmittelbar in ein Programm übersetzen; in BASIC wird die klare Struktur wieder verwischt.

Vor dem Aufschreiben des Programms müssen wir noch präzisieren, was "bedroht" heißt. Die Damen stehen in der gleichen Zeile, wenn $d(i) = d(k)$ ist; dabei ist i und k wieder als Spaltenindex gewählt. Die Damen stehen in der gleichen Diagonale, wenn ihre Verbindungsgerade die Steigung ± 1 hat; diese Steigung ist aber durch den Term

$$\frac{d(i) - d(k)}{i - k}$$

gegeben; wegen $i > k$ kann man als Kriterium für Bedrohtheit auch schreiben: $|d(i) - d(k)| = i - k$.
Wir haben die Prüfung auf Bedrohtheit in ein kleines Unterprogramm gelegt:

```
100 : PRINT "□
110 : PRINT "            N-DAMEN-PROBLEM
111 : PRINT "            ---------------
112 :
120 REM AUF EINEM N X N - SCHACHBRETT SIND
121 REM N SICH NICHT BEDROHENDE DAMEN ZU PLAZIEREN
129 :
200 REM === EINGABE UND INITIALISIERUNG ==============================
201 :
210 : PRINT : INPUT "WIEVIELE DAMEN "; N
220 : DIM D(N)
230 : FOR I = 1 TO N : LET D(I) = 0 : NEXT I
290 :
300 REM === SUCHE ====================================================
301 :
310 : LET I = 1
320 : LET D(I) = 0
330 :
340 : LET D(I) = D(I) + 1                          : REM DAME RUECKT VOR
350 : GOSUB 500                                    : REM BEDROHUNGSPRUEFUNG
360 : IF B$ = "BEDROHT" AND D(I) <= N THEN 340     : REM WIEDERHOLUNG
370 :
380 : IF D(I) > N THEN 440                         : REM RUECKZUG
390 :
400 : IF I < N THEN LET I = I+1 : LET D(I) = 0 : GOTO 340
410 :
415 : PRINT
420 : FOR J = 1 TO N : PRINT D(J);: NEXT J         : REM LOESUNG GEFUNDEN
430 :
440 : LET I = I-1 : IF I > 0 THEN 340              : REM WEITERSUCHEN
450 :
490 : END
499 :
500 REM +++ UNTERPROGRAMM 'BEDROHUNGSPRUEFUNG' ++++++++++++++++++++++
501 :
510 : LET B$ = "NICHT BEDROHT"
520 : IF I = 1 THEN RETURN : REM SONDERFALL
530 : FOR K = 1 TO I-1
540 :   LET A = ABS(D(I)-D(K))
550 :   IF A = 0 OR A = I-K THEN LET B$ = "BEDROHT" : RETURN
560 : NEXT K
570 :
590 : RETURN
599 :
```

Das Programm liefert im Fall n = 5 die Lösungen

```
1  3  5  2  4
1  4  2  5  3
2  4  1  3  5
2  5  3  1  4
3  1  4  2  5
3  5  2  4  1
4  1  3  5  2
4  2  5  3  1
5  2  4  1  3
5  3  1  4  2  .
```

Bei 8 Damen gibt es insgesamt 92 Lösungen, von denen die ersten so lauten:

```
1  5  8  6  3  7  2  4
1  6  8  3  7  4  2  5
1  7  4  6  8  2  5  3
1  7  5  8  2  4  6  3
2  4  6  8  3  1  7  5
2  5  7  1  3  8  6  4
2  5  7  4  1  8  6  3
2  6  1  7  4  8  3  5
2  6  8  3  1  4  7  5
2  7  3  6  8  5  1  4
2  7  5  8  1  4  6  3
2  8  6  1  3  5  7  4
3  1  7  5  8  2  4  6
3  5  2  8  1  7  4  6
3  5  2  8  6  4  7  1
3  5  7  1  4  2  8  6
3  5  8  4  1  7  2  6
3  6  2  5  8  1  7  4
3  6  2  7  1  4  8  5
3  6  2  7  5  1  8  4
3  6  4  1  8  5  7  2
3  6  4  2  8  5  7  1
3  6  8  1  4  7  5  2
3  6  8  1  5  7  2  4
3  6  8  2  4  1  7  5
3  7  2  8  5  1  4  6
3  7  2  8  6  4  1  5
3  8  4  7  1  6  2  5
4  1  5  8  2  7  3  6
```

Schön wäre es, wenn wir den Suchvorgang auf dem Bildschirm verfolgen könnten. Das Programm auf der nächsten Seite macht's möglich.

```
100 : PRINT "□
110 : PRINT "            N-DAMEN-PROBLEM
111 : PRINT "            ---------------
112 :
120 REM DER SUCHVORGANG KANN AM BILDSCHIRM MITERLEBT WERDEN
121 :
200 REM === EINGABE UND INITIALISIERUNG ==============================
201 :
210 : PRINT : INPUT "WIEVIELE DAMEN "; N
215 : IF N < 2 OR N > 10 THEN 210
220 :
230 : DIM D(N)
235 : FOR I = 1 TO N : LET D(I) = 0 : NEXT I
240 :
250 : DEF FN D(X) = 32768-1 + 2*X + 80*D(X)
255 :
260 : LET D = 81 : REM CODEZAHL DES DAMEZEICHENS
290 :
300 REM === SPIELFELD ================================================
301 :
310 : PRINT "□
320 : FOR W = 1 TO N : PRINT " —";: NEXT W : PRINT
330 : FOR I = 1 TO N
340 :   FOR J = 1 TO N : PRINT "|  ";: NEXT J : PRINT "|"
350 :   FOR W = 1 TO N : PRINT " —";: NEXT W : PRINT
360 : NEXT I
370 : PRINT
390 :
400 REM === SUCHE ====================================================
401 :
410 : REM --- ANFANGSPOSITION
411 :
415 :   LET I = 1
420 :   LET D(I) = 0
430 :
450 : REM --- DAME RUECKT VOR
451 :
460 :   LET D(I) = D(I)+1                      : REM DAME ZIEHT
470 :   POKE FND(I),D                          : REM ZEICHEN SETZEN
480 :   IF D(I) > 1 THEN POKE FND(I)-80,32     : REM ZEICHEN LOESCHEN
490 :
500 : REM --- BEDROHUNGSPRUEFUNG
501 :
510 :   LET B$ = "NICHT BEDROHT"
520 :   IF I = 1 THEN 600 : REM SONDERFALL
530 :   FOR K = 1 TO I-1
540 :     LET A = ABS(D(I)-D(K))
550 :     IF A = 0 OR A = I-K THEN LET B$ = "BEDROHT"
560 :   NEXT K
570 :
580 :  IF B$ = "BEDROHT" AND D(I) <= N THEN 450 : REM DAME RUECKT VOR
590 :
600 : REM --- DAME AUF DEM RAND ?
601 :
610 :   IF D(I) > N THEN 800 : REM RUECKZUG
630 :
650 : REM --- DAME AUF DIE NAECHSTE SPALTE SETZEN
651 :
660 :   IF I < N THEN LET I = I+1 : GOTO 420
670 :
```

```
700 : REM --- LOESUNG AUSGEBEN
701 :
710 :   PRINT "§": FOR K = 1 TO 2*N : PRINT : NEXT K
720 :   PRINT : FOR J = 1 TO N : PRINT D(J);: NEXT J
740 :
750 : REM --- WEITERSUCHEN ?
751 :
760 :   PRINT : PRINT "ZUM WEITERSUCHEN TASTE DRUECKEN!
770 :   GET T$ : IF T$ = "" THEN 770
790 :
800 : REM --- RUECKZUG
801 :
810 :   POKE FND(I),32     : REM DAME LOESCHEN
820 :   LET I = I-1        : REM AUF SPALTE DAVOR GEHEN
830 :   IF I > 0 THEN 450 : REM DAME RUECKT WIEDER VOR
850 :
900 : REM --- FERTIG
901 :
910 :   PRINT : PRINT "ALLE LOESUNGEN SIND GEFUNDEN.
920 :
990 :   END
```

*

Aufgaben

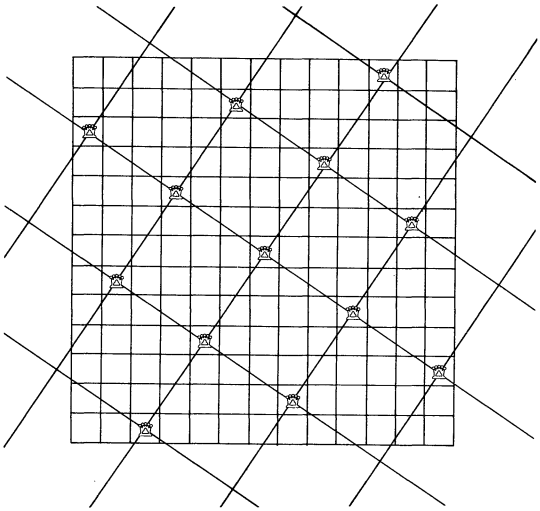
5. Das vorstehende Programm zur graphischen Darstellung der Lösung von n-Damen-Problemen läßt zu, daß die Damen über den unteren Brettrand kurz hinaustreten. Beheben Sie diesen kleinen Übelstand!

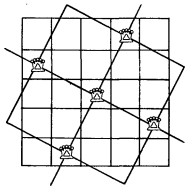
6. Durch Drehung und Spiegelung des Schachbretts kann man aus wenigen Grundlösungen des n-Damen-Problems die übrigen erzeugen. Zeigen Sie, daß das 8-Damen-Problem nur die 12 Grundlösungen

   15863724 16837425 24683175 25713864 25741863
   26174835 26831475 27368514 27581463 35281746
   35841726 36258174

   besitzt. Ändern Sie das Programm so ab, daß es nur die Grundlösungen auswirft.

7. Von Pierre de Fermat (1601 - 1655), Humanist und Mathematiker, stammt der berühmte Zwei-Quadrate-Satz: "Jede Primzahl der Form $p = 4n + 1$ ist auf genau eine Weise als Summe zweier Quadrate darstellbar." Beispiele: $5 = 1+4$, $13 = 4+9$, $17 = 1+16$, $29 = 4+25$, ... Bewiesen wurde der Satz zuerst von Leonhard Euler (1707 - 1783), aber kürzlich hat C. Larson einen neuen Beweis gefunden, der ihn in überraschender Weise mit dem n-Damenproblem in Zusammenhang bringt. Larson zeigt, daß Lösungen des n-Damenproblems folgender Art:





nur auftreten können, wenn die Seitenlänge eine Primzahl p der Form $4n+1$ ist. Jedes der kleineren Quadrate (mit den Damen als Ecken) hat die Seitenlänge $\sqrt{p}$; sie ist Hypotenuse eines rechtwinkligen Dreiecks mit den Seiten a und b (verallgemeinerter Rösselsprung).

Daher gilt $p = a^2 + b^2$, d.h. p ist auf genau eine Weise als Summe zweier Quadrate darstellbar.
Vermutlich hätte Vater Gauß, der sich ja mit beiden Problemen intensiv beschäftigt hat, auf die Mitteilung von Larsons Beweis darauf verwiesen, daß dieser sich schon unter seinen unveröffentlichten Papieren befindet.
Schreiben Sie ein Programm, welches die ersten 100 Primzahlen der Form $p = 4n+1$ ermittelt, geben Sie die Quadratzerlegung an und finden Sie die zugehörige Lösung des Damen-Problems!

8. 'Superdamen' können wie Springer und Dame zugleich ziehen. Schreiben Sie ein Programm zur Plazierung von sich nicht bedrohenden Superdamen auf einem $n \times n$-Brett.

9. Auf einem $n \times n$-Brett sollen n Spielsteine so plaziert werden, daß in jeder Zeile, in jeder Spalte und in jeder der beiden Diagonalen genau 1 Stein steht. Dabei sollen nur solche Lösungen als verschieden angesehen werden, die nicht durch Drehung oder Spiegelung des Bretts auseinander hervorgehen. Im Fall $n = 4$ gibt es genau eine Lösung: 1342, für $n = 5$ gibt es viere : 13524, 14532, 21354, 25314.

Der Springer gilt als "Komödiant des Schachspiels"; keine andere Figur gibt Anlaß zu so vielen unterhaltsamen kombinatorischen Problemen wie er. Das älteste davon ist die sogenannte Springertour.
Das Schachspiel benutzt bekanntlich 32 Figuren; sie reichen gerade aus, um eine Hälfte des Schachbretts zu füllen. Diesen Umstand nahmen Schachautoren des Mittelalters zum Anlaß, um folgende Aufgabe zu stellen: Man denke sich die eine Hälfte des Bretts mit Figuren besetzt und versuche nun, mit dem Springer alle in ununterbrochener Reihenfolge zu schlagen. Hieraus hat sich nun die weitere Aufgabe entwickelt, nicht nur das halbe, sondern das ganze Schachbrett zu durchlaufen:

Beispiel 3 : Springertour

Auf einem n x n - Schachbrett soll eine vollständige Springertour konstruiert werden, d.i. die Zugfolge eines (Schach-)Springers, bei der jedes Feld genau einmal betreten wird.

Beispiele:

1	12	23	18	7
22	17	8	13	24
11	2	25	6	19
16	21	4	9	14
3	10	15	20	5

28	23	6	15	34	21
7	16	27	22	5	14
24	29	8	35	20	33
9	36	17	26	13	4
30	25	2	11	32	19
1	10	31	18	3	12

Die obige Springertour im Fall n = 6 ist offenbar g e s c h l o s s e n , d.h. vom Endfeld (Sprung 36) gelangt man in einem Springerzug zum Anfangsfeld (Sprung 1).

*

Wir wollen nun ein Programm zur Durchführung von Springertouren entwerfen; genauer: der Leser soll dazu angeleitet werden. Das Schachbrett stellen wir als zweidimensionale Tabelle b(,) dar; b(i,j) kennzeichnet den Inhalt des Feldes in Zeile i und Spalte j.

	1	2	3	4	5	6	7	8
1	.	.	.	.	.	.	.	.
2	.	.	6	.	7	.	.	.
3	.	5	.	.	.	8	.	.
4	.	.	.	•	.	.	.	.
5	.	4	.	.	.	1	.	.
6	.	.	3	.	2	.	.	.
7	.	.	.	.	.	.	.	.
8	.	.	.	.	.	.	.	.

Die Koordinaten der 8 möglichen Springerzüge speichern wir in zwei Tabellen si( ), sj( ); und zwar ist si(1) = 1, sj(1) = 2 , d.h. der Springer bewegt sich ein Feld nach unten und zwei Felder nach rechts. Weiter ist si(2) = 2, sj(2) = 1 (Sprung Nr. 2 oben),
si(3) = 2, sj(3) = -1 (Sprung Nr. 3 oben) und so fort.
Wir lassen nun den Springer in Punkt (1,1) starten: er springt auf Punkt (2,3), dann auf (3,5) u.s.w. Bei jedem Punkt, auf den er springen möchte, prüft er, ob der für ihn zulässig ist, d.h. ob er nicht außerhalb des Brettes liegt, und ob er nicht schon früher einmal besucht wurde. Ist der Punkt nicht zulässig, versucht der Springer den nächsten Zug unter den möglichen 8 Zügen. Ist keiner der 8 Züge möglich, sitzt er in der Falle und kann nicht mehr ziehen. Was dann zu machen ist, besprechen wir gleich; zuerst das Programm:

```
100 : PRINT "
110 : PRINT "              SPRINGERTOUR
111 : PRINT "              ------------
112 :
120 REM DER SPRINGER BEWEGT SICH AUF EINEM N X N - SCHACHBRETT
121 REM BIS ER IN EINE SACKGASSE GERAET
129 :
200 REM === EINGABE UND INITIALISIERUNG ==============================
201 :
210 : REM --- SCHACHBRETT UND ANFANGSPOSITION
211 :
215 :   PRINT : INPUT "SEITENLAENGE DES BRETTS "; N
219 :
220 :   DIM B$(N,N)
225 :   FOR I = 1 TO N
230 :     FOR J = 1 TO N : LET B$(I,J) = "." : NEXT J
235 :   NEXT I
240 :
245 :   PRINT : INPUT "ANFANGSKOORDINATEN "; I0, J0
250 :
260 : REM --- SPRUNGKOORDINATEN
261 :
270 :   FOR K = 1 TO 8 : READ SI(K), SJ(K) : NEXT K
280 DATA 1,2,2,1,2,-1,1,-2,-1,-2,-2,-1,-2,1,-1,2
290 :
300 REM === TOUR ====================================================
301 :
310 : REM --- ANFANGSWERTE
311 :
320 :   LET I = I0 : LET J = J0
330 :   LET Z = 1  : REM NUMMER DES JEWEILIGEN SPRUNGS
340 :
350 : REM --- SPRUNG NUMERO Z
351 :
360 :   LET B$(I,J) = STR$(Z)
390 :
400 : REM --- NAECHSTER MOEGLICHER ZUG
401 :
410 :   LET Z = Z+1
415 :
420 :   FOR K = 1 TO 8
430 :     LET I1 = I + SI(K) : LET J1 = J + SJ(K)
440 :     IF I1 < 1 OR I1 > N OR J1 < 1 OR J1 > N THEN 460
450 :     IF B$(I1,J1) = "." THEN LET I = I1 : LET J = J1 : GOTO 350
460 :   NEXT K
490 :
500 REM === AUSGABE ================================================
501 :
510 : PRINT
520 : FOR R = 1 TO N
530 :   FOR S = 1 TO N
540 :     PRINT TAB(4*S);
550 :     IF B$(R,S) <> "." THEN PRINT "■" B$(R,S);: GOTO 570
560 :     PRINT B$(R,S);
570 :   NEXT S
580 :   PRINT
590 : NEXT R
598 :
599 : END
```

Das Programm liefert folgende Tour:

1	.	.	.	.	.	.	.
.	.	2	.	.	.	.	.
.	32	.	.	3	.	.	.
34	.	.	27	.	.	4	.
31	10	33	.	.	26	23	.
18	35	28	11	24	21	14	5
9	30	19	16	7	12	25	22
36	17	8	29	20	15	6	13

Nach dem 36. Zug sitzt der Springer in der linken unteren Ecke fest. Das Absuchen nach möglichen Zügen im Uhrzeigersinn ließ ihn sich fast nur in der unteren Bretthälfte aufhalten, aus der er schließlich nicht mehr herauskam. Können wir diesen schädlichen Trend vermeiden? Lassen wir - um jede Vorliebe für eine bestimmte Richtung auszuschalten - den Springer seine Züge einmal z u f ä l l i g bestimmen! Jetzt läuft jede Tour anders ab; hier zwei Stichproben:

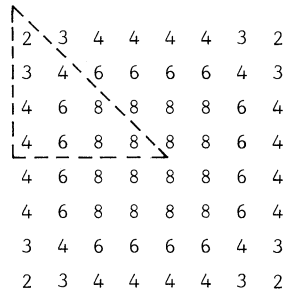
1	.	3	.	13	.	25	.
4	.	.	.	.	21	14	23
.	2	.	12	7	24	.	26
.	5	.	.	20	15	22	.
.	.	19	6	11	8	27	16
32	.	.	35	18	41	10	39
.	36	33	30	9	38	17	28
.	31	.	37	34	29	40	.

1	4	19	24	17	.	.	.
48	23	2	5	20	25	16	.
3	8	21	18	37	.	.	26
22	47	6	.	.	15	38	35
7	.	9	46	39	36	27	14
10	45	42	.	12	31	34	29
43	.	11	40	.	28	13	32
.	41	44	.	.	33	30	.

In beiden Fällen kam der Springer entschieden weiter als bei dem deterministischen Verfahren - doch schließlich verrannte er sich ebenfalls. Wir erkennen auch klar seinen Fehler: unzugängliche Rand- und Eckfelder ließ er unbedachtsamerweise frei: sie zu besuchen hatte er später keine Möglichkeit mehr.

Aus dieser Beobachtung stammt eine heuristische Regel zur Erzeugung von Springertouren, die nach dem Mathematiker J.C. Warnsdorff (1823) benannt

ist. Sie lautet: "Bei jedem Springerzuge wähle man unter den verschiedenen Feldern, die durch diesen Zug zu erreichen sind, dasjenige, von dem am wenigsten Springerzüge nach unbesetzten Feldern hin noch möglich sind." Denn hier ist die Gefahr, das betreffende Feld nicht wieder zu erreichen und es somit ganz auszulassen, verhältnismäßig am größten.
Die praktische Brauchbarkeit dieser Regel ist so groß, daß sie selbst bei ganz willkürlich angefangenen und schon ziemlich weit ohne Beachtung der Regel fortgesetzten Springerzügen noch zum Ziele führt.
Um diese Regel zu programmieren, müssen wir jedem Feld des Schachbretts einen 'Platzwert' zuordnen: er gibt an, wieviel Felder von ihm aus durch einen Springerzug erreichbar sind. Die Platzwerte eines 8 x 8 - Bretts sind:



Beachten Sie die hohe Symmetrie dieser Anordnung! Es genügt, die Platzwerte der 10 Felder in dem angedeuteten Dreieck zu kennen.
Der Springer wählt nun unter allen ihm zugänglichen Feldern dasjenige mit dem kleinsten Platzwert aus. Ist er auf ein Feld gesprungen, so wird der Platzwert aller Felder, die von diesem Feld mit einem Springerzug erreichbar sind, um 1 erniedrigt.
Die Ausführung dieser Idee in einem Programm sei dem Leser anempfohlen!

*

Aufgaben

10. Beweisen Sie:
    a) Das kleinste (quadratische) Schachbrett, auf dem eine vollständige Springertour möglich ist, hat die Seitenlänge 5 .
    b) Das kleinste (quadratische) Schachbrett, auf dem eine geschlossene vollständige Springertour möglich ist, hat die Seitenlänge 6.

11. Schreiben Sie ein Programm zur Erzeugung von Springertouren nach der Warnsdorffschen Regel. Lassen Sie den Springer dann - sagen wir - 20 Züge zufällig machen; erst dann soll das Programm die Warnsdorffsche Regel anwenden.

12. Eine schlaue Methode, den Springer auf seiner Tour aus einer Sackgasse zu befreien ist das Umnumerieren: er geht auf ein Feld, das er früher schon einmal betreten hatte (und von dem aus er weiterziehen kann) und erklärt es zum Endfeld; das alte Endfeld (auf dem er festsaß) erklärt zum Zwischenfeld. Die Felder müssen dazu geeignet umnumeriert werden.
    Führen Sie diese clevere Idee Eulers durch!

13. Eine andere Möglichkeit zur Konstruktion von Springertouren ist - ähnlich wie beim Algorithmus zur Lösung des n-Damen-Problems - , Züge, die in eine Sackgasse führten, zurückzunehmen (engl.: backtracking). Versuchen Sie, diese Möglichkeit zu programmieren.

14. Springertouren braucht man nicht unbedingt auf quadratischen Schachbrettern auszuführen; man kann auch Rechtecke oder andere geometrischen Formen nehmen.

15. Euler hat es geschafft, eine Springertour zu konstruieren, die zugleich ein (halb-)magisches Quadrat darstellt (die Diagonalen tuns nicht). Versuchen Sie etwas ähnliches.

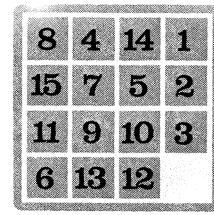
1	48	31	50	33	16	63	18
30	51	46		62	19		35
47		49	32	15	34	17	64
52	29		45	20	61	36	
	44	25	56		40	21	60
28	53		41	24	57		37
43		55	26	39		59	22
54	27	42		58	23	38	

Um die Jahrhundertwende hielt ein Geduldspiel die gebildete Welt in Atem und sorgte für Aufregung, gegen die der Rummel, der gegenwärtig um den sogenannten Zauberwürfel (Rubiks Würfel) veranstaltet wird, gar nichts ist. Es handelt sich um das - auch heute noch bekannte und beliebte - Fünfzehner-Spiel, dessen Erfindung dem geistvollen Sam Loyd zugeschrieben wird (Proben seiner Phantasie haben wir in diesem Buch schon mehrfach kennengelernt). Schon bald nach seiner Entdeckung (1878) verbreitete das Spiel sich über die ganze zivilisierte Erde und wurde in jenen ersten Jahren überall mit solchem Eifer gespielt, wie wohl kaum ein anderes Geduldspiel je zuvor. So wird beispielsweise von Hamburg erzählt, daß man dort die kleinen Kästen mit den Holzklötzchen darin selbst in den Pferdebahnen erblicken und unruhige Hände darin schieben sehen konnte, daß die Prinzipale in den Handelskontoren über das Puzzlefieber ihrer Angestellten in Verzweiflung gerieten und durch Anschläge das Spielen während der Bürozeit aufs strengste verbieten mußten, daß große Turniere veranstaltet wurden u.s.w. Selbst im Sitzungssaal des Deutschen Reichstags konnte man damals auf den Bänken an der Wand Abgeordnete aller Parteien sehen, die den Reden keine Aufmerksamkeit schenkten, dafür aber umso eifriger 'boß-puzzleten' (das Spielzeug nannte sich 'Boß-Puzzle').

Beispiel 4 : Fünfzehnerspiel

Fünfzehn, mit den Zahlen 1 bis 15 numerierte Steine werden in willkürlicher Reihenfolge in den Holzkasten gelegt. Nun soll lediglich durch Verschieben der Steine untereinander, wie dies ja infolge des einen leer gebliebenen Platzes möglich ist, die natürliche Reihenfolge herbeigeführt werden.

Es gibt 20.922.789.888.000 verschiedene Ausgangsstellungen; davon läßt sich genau die Hälfte lösen, die andere nicht.

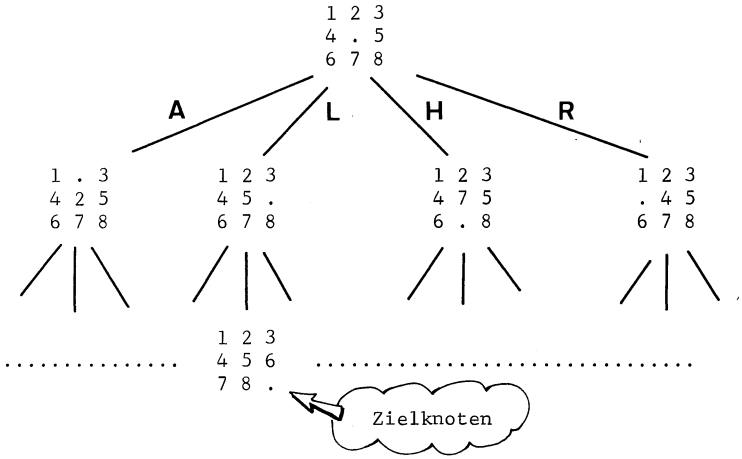


Sam Loyd hatte für die Lösung der Stellung

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

100 Dollar ausgelobt; sein Geld war sicher, da es aus dieser Stellung

keine Zugfolge zur Grundstellung gibt. Mit Hilfe der Theorie der Permutationen läßt sich ein einfaches Kriterium dafür herleiten, daß eine gegebene Ausgangsstellung eine Lösung zuläßt.-
Wir wollen im folgenden ein Programm entwickeln, das aus einer beliebigen Ausgangsposition die Zielposition herstellt - sofern dies möglich ist, und eine entsprechende Mitteilung macht. Aus Gründen der Einfachheit beschränken wir uns auf das Achter-Spiel (8 Steine auf einem 3 x 3 - Feld). Anders als z.B. bei der Konstruktion magischer Quadrate oder beim Rösselsprung werden wir hier ein reines Suchverfahren entwickeln. Dazu müssen wir alle Möglichkeiten betrachten, wie man aus einer gegebenen Konstellation weitere Konstellationen gewinnen kann, zum Beispiel:



Dabei bedeutet

A ..... ab (Stein wird abwärts ins Leerfeld geschoben)
L ..... links (Stein nach links ins Leerfeld)
H ..... hoch (Stein wird hochgeschoben)
R ..... rechts (Stein wird nach rechts geschoben).

Das entstandene Gebilde nennt sich Baum (allgemeiner: Graph), die einzelnen Konstellationen heißen Knoten des Baums. Unsere Aufgabe besteht nun darin, den Baum nach der Zielkonstellation zu durchsuchen.

Wie verfahren wir nun bei dieser sogenannten B a u m s u c h e ? Jedes Suchverfahren hat folgende Komponenten:

(1) Zwei Listen, nämlich die 'Liste der bereits behandelten Knoten' (das sind diejenigen Knoten, deren sämtliche Nachfolger-Knoten bereits untersucht sind) und die 'Liste der unbehandelten Knoten' (das sind diejenigen Knoten, bei denen noch nicht alle Nachfolger untersucht sind),

(2) ein Verfahren zum Erzeugen aller Nachfolger eines Knotens (Nachfolger eines Knotens ist ein Knoten, der durch die Anwendung einer der Operationen A, H, L oder R aus dem gegebenen Knoten hervorgeht),

(3) ein Verfahren, mittels dem geprüft werden kann, ob ein erzeugter Knoten der Zielknoten ist,

(4) ein Verfahren, welches unter den unbehandelten Knoten denjenigen auswählt, dessen Nachfolger als nächste untersucht werden sollen.

BAUMSUCHE

```
Lege den Anfangsknoten auf die Liste unbehandelter Knoten
WIEDERHOLE
  A := oberstes Element in der Liste unbehandelter Knoten
  Lege A auf die Liste behandelter Knoten
  Bestimme die Nachfolger von A
  WENN ein Nachfolger existiert DANN
    FÜR jeden Nachfolger von A WIEDERHOLE
      prüfe, ob Zielknoten erreicht ist
      WENN Zielknoten erreicht DANN
        z := Nummer des Zielknotens
        Z$ := "Ziel erreicht"
        BEENDE WIEDERHOLUNG
      SONST
        berechne den Ordnungswert des Knotens
        füge den Knoten gemäß seinem Ordnungswert in die
        Liste unbehandelter Knoten ein
      ENDE-WENN
    ENDE-WIEDERHOLE
  ENDE-WENN
BIS Liste der unbehandelten Knoten leer ENDE-WIEDERHOLE
WENN Z$ = "Ziel erreicht" DANN
  bestimme Lösungspfad durch Rückverfolgen von z aus
  Ausgabe der Lösung
SONST
  Ausgabe "Es existiert keine Lösung"
ENDE-WENN
```

Wie stellen wir nun die Knoten des Suchbaumes im Programm dar? Um sie gut speichern und mit ihnen hantieren zu können, werden die Knoten als Zeichenketten dargestellt, d.h. aus (beispielsweise) 1.2
463
758

wird 1.2463758. In einer solchen Zeichenkette werden aber noch zwei weitere Informationen untergebracht: erstens ein Zeiger auf den Vorgängerknoten (damit man ihn beim Rückwärtsmarschieren wiederfindet) und zweitens die Zugrichtung, die vom Vorgängerknoten zum betrachteten Knoten führte. Zusätzlich ist jedem Knoten K$(NR) ein Ordnungswert O(NR) beigegeben, welcher bestimmt, ob unser Knoten als nächster entwickelt wird oder nicht ('entwickeln' eines Knotens heißt: seine sämtlichen Nachfolger untersuchen). Ein Beispiel: die Zeichenkette

K$(10) = 007 L 1.2463758 mit O(10) = 5

sagt aus, daß die Konstellation 1.2
463
758 den Vorgänger Nr. 7 hatte und durch den Zug L aus diesem hervorging; ihr Ordnungswert ist 5.
Die bereits behandelten Knoten werden mit einem Ordnungswert > 90 000 ausgestattet: damit ist leicht zu verhindern, daß sie erneut untersucht werden. Von den vier möglichen Nachfolgern eines Knotens scheidet sofort einer aus, nämlich der, welcher mit dem Vorgänger des betrachteten Knotens übereinstimmt: dies wird in den Zeilen 5100 bis 5190 des Programms auf der folgenden Seite geprüft.
Das Programm ist so geschrieben, daß es sich (hoffentlich) selbst dokumentiert: dadurch ist es natürlich sehr langsam geworden. Um es schneller zu machen, kann der Leser die vielen Kommentare herausnehmen. Auch ist es - um dem Leser noch etwas zu überlassen - nicht vollständig: es fehlt das Unterprogramm, welches - vom Zielknoten ausgehend - die Zugfolge rekonstruiert, welche vom Ausgangsknoten zum Zielknoten führte (Zeile 7500 ff). Die Suchstrategie ist in Zeile 3710 beschlossen: um sie zu erforschen, möge der Leser mit dem Programm experimentieren. Anregungen dazu enthalten die Aufgaben.

```
1000 : PRINT "□
1010 : PRINT "          FUENFZEHNERSPIEL
1011 : PRINT "          ----------------
1012 :
1020 REM SUCHPROGRAMM, DAS AUS EINER GEGEBENEN ANFANGSSTELLUNG
1021 REM DIE GRUNDSTELLUNG WIEDERHERSTELLT (SOFERN MOEGLICH)
1022 :
2000 REM === INITIALISIERUNG UND EINGABE ============================
2001 :
2100 : REM --- TABELLEN
2101 :
2110 :   DIM K$(100) : REM LISTE DER KNOTEN
2120 :   DIM O(100)  : REM TABELLE DER ORDNUNGSWERTE
2140 :   DIM NF$(4)  : REM TABELLE DER NACHFOLGER EINES KNOTENS
2150 :   DIM E$(3,3) : REM SPIELFELD
2160 :   DIM F$(3,3) : REM HILFSSPIELFELD
2190 :
2300 : REM --- INKREMENTE FUER DIE WANDERUNG DES LEERFELDS (PUNKTS)
2301 :
2310 :   LET X(1) = -1 : LET Y(1) =  0
2320 :   LET X(2) =  0 : LET Y(2) =  1
2330 :   LET X(3) =  1 : LET Y(3) =  0
2340 :   LET X(4) =  0 : LET Y(4) = -1
2390 :
2400 : REM --- ZEICHENKETTEN FUER DIE ZUGRICHTUNG
2401 :
2410 :   LET R$ = "ALHRB" : REM AB, LINKS, HOCH, RECHTS, BEGINN
2420 :   LET I$ = "HRAL"  : REM INVERSE RICHTUNG
2430 :   LET ZR$ = "B"    : REM ZUGRICHTUNG (ANFANGSWERT 'BEGINN')
2490 :
2500 : REM --- EINGABE DER ANFANGSSTELLUNG
2501 :
2510 :   PRINT
2520 :   PRINT "GEBEN SIE DIE ANFANGSSTELLUNG ";
2530 :   PRINT "IN DER FORM '1234.5678' EIN ";
2540 :   INPUT A$
2560 :   IF LEN(A$) <> 9 THEN PRINT "EINGABEFEHLER!": GOTO 2510
2590 :
2600 : REM --- INITIALISIERUNG DER VARIABLEN
2601 :
2610 :   LET NR = 1    : REM LAUFENDE NUMMER IN DER KNOTENLISTE
2620 :   LET NN = 1    : REM NUMMER DES KNOTENS, DER ENTWICKELT WIRD
2630 :   LET O(1) = 0 : REM ORDNUNGSWERT DES ERSTEN KNOTENS
2640 :   LET BK = 0    : REM ANZAHL DER BEHANDELTEN KNOTEN
2690 :
2700 : REM --- INITIALISIERUNG DER LISTE UNBEHANDELTER KNOTEN
2701 :
2710 :   GOSUB 6000 : REM KNOTENNUMMER ALS ZEICHENKETTE
2720 :
2730 :   LET K$(1) = NN$+"B"+A$ : REM KENNZEICHUNG DES STARTKNOTENS
2740 :
2750 :   LET K$ = K$(1) : GOSUB 6200 : REM ZIEL SCHON ERREICHT?
2760 :   IF ZF$ = "ZIEL ERREICHT" THEN PRINT "WAS SOLL DAS ?": END
2790 :
2800 : REM --- INITIALISIERUNG DER ENDEFLAGGE
2801 :
2810 :   LET EF$ = "NICHT FERTIG"
2999 :
```

```
3000 REM === SUCHE ================================================
3001 :
3200 : REM --- BESTIMME OBERSTES ELEMENT DER UNBEHANDELTEN-LISTE
3201 :
3210 :   GOSUB 4200 : REM BESTIMMUNG DES WERTKLEINSTEN KNOTENS
3220 :
3230 :   IF O(NN) >= 90000 THEN LET EF$ = "FERTIG" : GOTO 3900
3240 :
3300 : REM --- LEGE KNOTEN AUF DIE LISTE DER BEHANDELTEN KNOTEN
3301 :
3305 :   PRINT
3310 :   PRINT "KNOTEN NUMERO"; NN;"NAMENS "; K$(NN);
3320 :   PRINT " MIT ORDNUNGSWERT "; O(NN);" WIRD ENTWICKELT
3321 :
3330 :   PRINT "NACHFOLGERKNOTEN: "
3331 :
3340 :   LET BK = BK+1              : REM ANZAHL DER BEHANDELTEN KNOTEN
3350 :   LET O(NN) = O(NN) + 90000 : REM KENNZEICHUNG ALS BEHANDELT
3390 :
3400 : REM --- BESTIMME DIE NACHFOLGER
3401 :
3410 :   LET E$  = MID$(K$(NN),5,9) : REM ZAHLFELD HERAUSHOLEN
3420 :   LET ZR$ = MID$(K$(NN),4,1) : REM ZUGRICHTUNG
3430 :
3440 :   GOSUB 4600                 : REM ENTPACKE ZEICHENKETTE
3450 :   GOSUB 5000                 : REM ENTWICKLE KNOTEN
3460 :
3500 : REM --- EXISTIERT NACHFOLGER?
3501 :
3510 :   IF NF > 0 THEN 3600 : REM NACHFOLGER EXISTIERT
3520 :
3540 :   GOTO 3900            : REM SUCHE BEENDET?
3550 :
3590 :
3600 : REM --- SCHLEIFE UEBER ALLE LEGALEN NACHFOLGERKNOTEN
3601 :
3610 :   FOR R = 1 TO NF : REM SCHLEIFENANFANG ------------------------
3611 :
3615 :       LET K$ = NF$(R) : GOSUB 6200 : REM VERGLEICH MIT ZIEL
3619 :
3620 :     REM --- VERGLEICH MIT ZIELKNOTEN
3621 :
3630 :       IF ZF$ <> "ZIEL ERREICHT" THEN 3680
3631 :
3640 :     REM --- ZIELKNOTEN ERREICHT
3641 :
3670 :       LET EF$ = "FERTIG" : GOTO 3900
3671 :
3680 :     REM --- ZIELKNOTEN NICHT ERREICHT
3681 :
3685 :       LET E$ = MID$(NF$(R),5,9)
3690 :       GOSUB 4600 : REM ENTPACKE ZEICHENKETTE
3691 :
3700 :     REM --- FUEGE KNOTEN IN LISTE EIN
3701 :
3710 :       LET RG = O(NN)+1 - 90000 : REM SUCHSTRATEGIE
3720 :
3750 :       LET NR = NR+1       : REM LAUFENDE KNOTENNUMMER ERHOEHEN
3760 :       LET K$(NR) = NF$(R): REM KNOTEN AUF LISTE LEGEN
3770 :       LET O(NR) = RG      : REM NEUE ORDNUNGSNUMMER
3780 :
3790 :   NEXT R : REM SCHLEIFENENDE ------------------------------
3791 :
```

```
3900 :   IF EF$ <> "FERTIG" THEN 3200
3901 :
3910 :   GOTO 7000 : REM ZUR AUSGABE
3999 :
4000 REM +++ UNTERPROGRAMM 'LIEGT NEUER KNOTEN VOR?' ++++++++++++++++++
4001 :
4010 : REM EINGABE: BRETT F$( , ), LISTE K$( )
4020 : REM ES WIRD GEPRUEFT, OB F$ SCHON MAL ENTWICKELT WURDE
4030 : REM AUSGABE: DF$ = "DOPPELT" ODER DF$ = "NICHT DOPPELT"
4050 :
4100 : LET DF$ = "NICHT DOPPELT"
4110 :
4120 : FOR I = 1 TO NR : REM DURCHLAUFEN ALLER ENTWICKELTEN KNOTEN
4130 :   IF O(I) < 90000 THEN 4140
4135 :   IF F$ = MID$(K$(I),5,9) THEN LET DF$ = "DOPPELT": RETURN
4140 : NEXT I
4150 :
4160 : IF DF$ = "DOPPELT" THEN PRINT "KNOTEN DOPPELT"
4170 :
4180 : RETURN
4181 :
4190 REM ENDE DES UNTERPROGRAMMS 'LIEGT NEUER KNOTEN VOR?' ++++++++++++
4199 :
4200 REM +++ UNTERPROGRAMM 'BESTIMMUNG DES WERTKLEINSTEN KNOTENS' ++++
4201 :
4210 : REM EINGABE: LISTE K$( ) DER KNOTEN UND IHRE ANZAHL NR
4220 : REM ES WIRD DER KNOTEN MIT KLEINSTEM ORDNUNGSWERT ERMITTELT
4230 : REM AUSGABE: NN UND T, WOBEI T = O(NN) DER KLEINSTE WERT IST
4250 :
4260 : LET T = 99999
4270 : LET N = 1
4280 :
4300 : FOR I = 1 TO NR
4310 :   IF T <= O(I) THEN 4350
4320 :   LET T = O(I)
4340 :   LET N = I
4350 : NEXT I
4351 :
4360 : LET NN = N : REM BESTIMMUNG DES ZU ENTWICKELNDEN KNOTENS
4361 :
4380 : RETURN
4381 :
4390 REM ENDE DER BESTIMMUNG DES WERTKLEINSTEN KNOTENS ++++++++++++++++
4399 :
4400 REM +++ UNTERPROGRAMM 'PACKE ZAHLENFELD IN ZEICHENKETTE' ++++++++
4401 :
4410 : REM EINGABE: BRETT F$( , )
4420 : REM UMWANDLUNG IN EINE ZEICHENKETTE
4430 : REM AUSGABE: F$ = ZEICHENKETTE AUS 9 ZEICHEN
4440 :
4450 : LET F$ = ""
4460 :
4470 : FOR P = 1 TO 3
4480 :   FOR Q = 1 TO 3 : LET F$ = F$ + F$(P,Q) : NEXT Q
4490 : NEXT P
4500 :
4510 : RETURN
4520 :
4590 REM ENDE UNTERPROGRAMM 'ZAHLENFELD IN ZEICHENKETTE' ++++++++++++++
4599 :
```

```
4600 REM +++ UNTERPROGRAMM 'ENTPACKE ZEICHENKETTE ZU SPIELFELD' ++++++
4601 :
4610 : REM EINGABE: ZEICHENKETTE E$
4620 : REM E$ WIRD IN ZAHLENFELD E$( , ), F$( , ) UMGEWANDELT
4630 : REM AUSGABE: GLEICHE ZAHLENFELDER E$( , ) UND F$( , )
4640 :
4650 : FOR I = 1 TO 3
4660 :   FOR J = 1 TO 3
4670 :     LET U = 3*(I-1)+J
4680 :     LET E$(I,J) = MID$(E$,U,1)
4690 :     LET F$(I,J) = E$(I,J)
4700 :   NEXT J
4710 : NEXT I
4720 :
4780 : RETURN
4781 :
4790 REM ENDE DES UNTERPROGRAMMS 'ENTPACKE' ++++++++++++++++++++++++++
4799 :
5000 REM +++ UNTERPROGRAMM 'ENTWICKLE KNOTEN' ++++++++++++++++++++++++
5001 :
5010 : REM EINGABE: E$( , ) = F$( , ),KNOTENNUMMER NN, ZUGRICHTUNG ZR$
5020 : REM ES WERDEN BIS ZU DREI ZULAESSIGE NACHFOLGERKNOTEN ERZEUGT,
5021 : REM BEREITS ENTWICKELTE KNOTEN ELIMINIERT UND DER VOLLSTAENDIGE
5022 : REM KNOTEN AUS ZUGRICHTUNG, ZEIGER UND BRETT GEBILDET
5030 : REM AUSGABE: LISTE NF$( ) DER NACHFOLGER UND IHRE ANZAHL AN
5050 :
5100 : REM --- BESTIMMUNG DER VERBOTENEN ENTWICKLUNGSRICHTUNG
5101 :
5110 :   FOR I = 1 TO 5
5120 :     IF ZR$ = MID$(R$,I,1) THEN 5150 : REM ZUGRICHTUNG
5130 :   NEXT I
5140 :
5150 :   LET VR$ = MID$(I$,I,1) : REM VERBOTENE ENTWICKLUNGSRICHTUNG
5190 :
5200 : REM --- BESTIMMUNG DER KOORDINATEN X1,Y1 DES LEERFELDS
5201 :
5210 :   FOR Y1 = 1 TO 3
5220 :     FOR X1 = 1 TO 3
5230 :       IF E$(X1,Y1) = "." THEN 5300 : REM STELLE GEFUNDEN
5240 :     NEXT X1
5250 :   NEXT Y1
5260 :
5300 : REM --- ERZEUGUNG VON VIER MOEGLICHEN NACHFOLGERN
5301 :
5310 :   LET AN = 0 : REM ANZAHL DER NACHFOLGER NACH PRUEFUNG
5315 :   LET NF = 0 : REM ANZAHL DER NACHFOLGER VOR DOPPELTPRUEFUNG
5319 :
5320 :   LET S1 = 1       : REM SCHLEIFE UEBER ALLE NACHFOLGER
5321 :
5360 :       IF MID$(R$,S1,1) = VR$ THEN 5900 : REM ZUM NAECHSTEN
5390 :
5400 :     REM --- NEUE STELLE DES LEERFELDS
5401 :
5410 :       LET X2 = X1 + X(S1)
5420 :       LET Y2 = Y1 + Y(S1)
5430 :
5440 :       IF X2 < 1 OR X2 > 3 THEN 5900 : REM NICHT LEGAL
5450 :       IF Y2 < 1 OR Y2 > 3 THEN 5900 : REM NICHT LEGAL
5490 :
```

```
5500 :      REM --- TAUSCHE FELDER X1,Y1 UND X2,Y2 AUS
5501 :
5510 :        FOR I = 1 TO 3
5520 :          FOR J = 1 TO 3 : LET F$(I,J) = E$(I,J) : NEXT J
5530 :        NEXT I
5540 :
5550 :        LET F$(X1,Y1) = F$(X2,Y2)
5560 :        LET F$(X2,Y2) = "."
5590 :
5600 :      REM --- KNOTENNUMMER ALS ZEICHENKETTE
5601 :
5610 :        GOSUB 6000 : REM KNOTENNUMMER WIRD ZEICHENKETTE
5620 :
5700 :      REM --- PRUEFE, OB KNOTEN IN DER TABELLE SCHON VORHANDEN
5701 :
5710 :        GOSUB 4000        : REM LIEGT NEUER KNOTEN VOR ?
5730 :        IF DF$ = "DOPPELT" THEN 5900 : REM ZUM NAECHSTEN
5790 :
5800 :      REM --- BILDE VOLLSTAENDIGE ZEICHENKETTE
5801 :
5810 :        LET AN = AN+1 : REM ANZAHL DER ERZEUGTEN NACHFOLGER
5830 :        GOSUB 4400    : REM VERPACKE ALS ZEICHENKETTE
5850 :        LET NF$(AN) = NN$ + MID$(R$,S1,1) + F$
5860 :        LET NF = NF+1
5861 :
5870 :        PRINT "■" AN; "  "; NF$(AN)
5871 :
5900 :    IF S1 < 4 THEN LET S1 = S1+1 : GOTO 5360 --------------------
5910 :
5980 : RETURN
5981 :
5990 REM ENDE DES UNTERPROGRAMMS 'KNOTEN ENTWICKELN' ++++++++++++++++++
5999 :
6000 REM UNTERPROGRAMM 'KNOTENNUMMER ALS ZEICHENKETTE' +++++++++++++++
6001 :
6010 : REM EINGABE: NUMMER NN
6020 : REM MACHT NN ZU EINER ZEICHENKETTE, SETZT FUEHRENDE NULLEN EIN
6030 : REM AUSGABE: ZEICHENKETTE FUER NN ALS KNOTENNUMMER
6040 :
6050 : LET NN$ = STR$(NN)
6055 : LET NN$ = RIGHT$(NN$, LEN(NN$)-1) : REM VORZEICHEN WEGNEHMEN
6060 : LET L = 3-LEN(NN$)
6070 :
6080 : IF L <  0 THEN PRINT "UEBERLAUF-FEHLER IN 6080!": END
6090 : IF L = 0  THEN RETURN
6100 :
6110 : FOR I = 1 TO L  : LET NN$ = "0" + NN$ : NEXT I
6120 :
6180 : RETURN
6181 :
6190 REM ENDE VON 'KNOTENNUMMER ALS ZEICHENKETTE' ++++++++++++++++++++
6199 :
```

```
6200 REM +++ UNTERPROGRAMM 'PRUEFUNG OB ZIELKNOTEN ERREICHT' +++++++++
6201 :
6210 : REM EINGABE: KNOTEN K$
6220 : REM ZIEHT BRETT AUS ZEICHENKETTE UND VERGLEICHT MIT ZIELBRETT
6230 : REM AUSGABE: ZF$ = "ZIEL ERREICHT" ODER "ZIEL NICHT ERREICHT"
6231 :
6240 : LET ZF$ = "ZIEL NICHT ERREICHT"
6241 :
6250 : LET ZK$ = "12345678."   : REM ZIELKNOTEN
6260 : LET PK$ = RIGHT$(K$,9) : REM ZU PRUEFENDER KNOTEN
6270 :
6280 : IF PK$ = ZK$ THEN LET ZF$ = "ZIEL ERREICHT"
6281 :
6285 : RETURN
6286 :
6290 REM ENDE DER PRUEFUNG, OB ZIELKNOTEN ERREICHT ++++++++++++++++++
6299 :
6900 REM ENDE DER SUCHE ==============================================
6901 :
6999 :
7000 REM === AUSGABE ==================================================
7001 :
7100 :   IF ZF$ <> "ZIEL ERREICHT" THEN 7300
7101 :
7200 : REM --- AUSGABE DES LOESUNGSPFADS
7201 :
7210 :   PRINT : PRINT : PRINT "LOESUNG GEFUNDEN!
7220 :   GOSUB 7500 : REM RUECKVERFOLGEN
7230 :   PRINT "DER LOESUNGSPFAD LAUTET: ****** ";
7240 :   GOTO 7400
7290 :
7300 : REM --- FEHLANZEIGE
7301 :
7310 :   PRINT : PRINT "KEINE LOESUNG GEFUNDEN.
7390 :
7400 : REM --- AUSGABE DER KNOTENANZAHLEN
7401 :
7410 :   PRINT
7420 :   PRINT : PRINT "ANZAHL DER UNBEHANDELTEN KNOTEN: "; NR-BK
7430 :   PRINT "ANZAHL DER BEHANDELTEN KNOTEN: "; BK
7440 :
7470 :   END
7480 :
7490 REM ENDE DER AUSGABE ============================================
7499 :
7500 REM +++ UNTERPROGRAMM 'RUECKVERFOLGEN DES LOESUNGSPFADS' ++++++++
7501 :
7510 : REM EINGABE: AKTUELLER KNOTEN K$(NN)
7520 : REM MITTELS DER ZEIGER WIRD DER LOESUNGSPFAD ZURUECKGEGANGEN
7521 : REM UND ES WERDEN DIE RICHTUNGSBUCHSTABEN AUFGENOMMEN
7530 : REM AUSGABE: FOLGE DER LOESUNGSZUEGE
7590 :
7600 : REM *** DIESEN TEIL MOEGE DER LESER EINFUEGEN! ***
7660 :
7880 : RETURN
7881 :
7890 REM ENDE DER AUSGABE DES LOESUNGSPFADS ++++++++++++++++++++++++++
7899 :
```

Aufgaben

16. Fügen Sie ins vorstehende Programm die fehlende Prozedur zur Ermittlung des Lösungspfads durch Rückverfolgen ein!

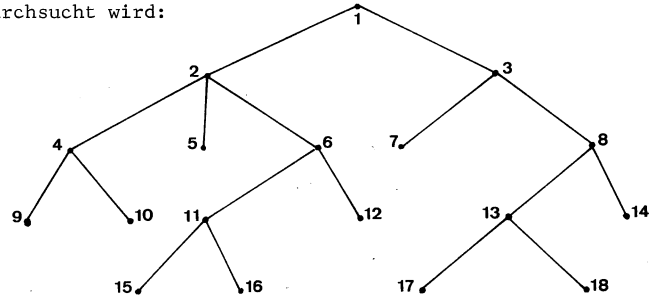
17. Erweitern Sie das Programm auf ein 4 x 4 - Brett.

18. Zeigen Sie, daß der Fall

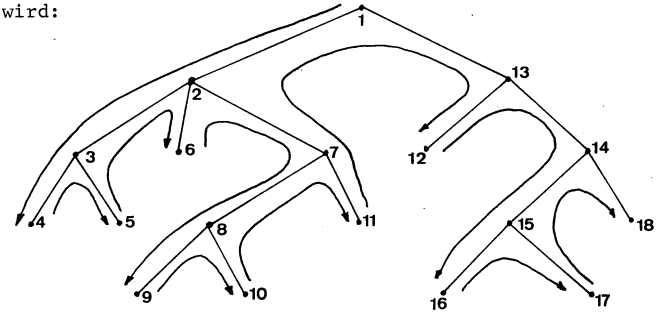
1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	.

unlösbar ist.

19. In Zeile 3710 des Programms 'Fünfzehnerspiel' wird die Suchstrategie festgelegt. Bestätigen Sie an Beispielen, daß der Baum in folgender Weise durchsucht wird:



Ändern Sie die genannte Zeile so ab, daß folgende Suchstrategie angewandt wird:



Seit mehr als drei Jahrhunderten finden sich in Kalendern, Kinderbüchern und Zeitschriften Aufgaben, bei denen vorausgesetzt ist, daß eine gewisse Anzahl von Gefäßen - meistens drei - bestimmter Fassungskraft vorhanden sind. Es wird dann verlangt, daß durch wiederholtes Umfüllen schließlich eine vorgeschriebene Anzahl von Litern in das eine oder das andere Gefäß gebracht wird. Das größte ist meistens gefüllt; so auch in den (von uns bereits erwähnten) Problèmes des Bachet de Méziriac, wo die Aufgabe folgendermaßen lautet:

Beispiel 5 : Umfüllaufgabe

Zwei Freunde haben beschlossen, sich ein Quantum Wein zu teilen, den sie in einem acht Liter fassenden Krug besitzen. Wie können sie den Wein in zwei genau gleiche Teile zu je vier Liter teilen, wenn sie noch zwei kleinere Krüge besitzen, die leer sind und von denen der eine drei Liter, der andere fünf Liter fassen kann? Die Krüge besitzen natürlich keine Inhaltsmarkierung.

Bachet gibt zwei Lösungen an, nämlich

$$800 \to 350 \to 323 \to 620 \to 602 \to 152 \to 143 \to 440$$

und

$$800 \to 503 \to 530 \to 233 \to 251 \to 701 \to 710 \to 412 \to 440 \ .$$

Dabei bedeutet (beispielsweise) 251, daß im ersten Krug 2, im zweiten Krug 5 und im dritten Krug 1 Liter enthalten ist.
Da bei jedem Umfüllen entweder eins der Gefäße in ein anderes entleert oder durch ein anderes gefüllt wird, so muß, eines der Gefäße entweder ganz v o l l oder ganz l e e r sein. Man kann nun einen ganzen Zyklus von Umfüllungen bilden, der mit dem gleichen Füllungszustande wieder schließt, mit dem man begonnen hat, und der unter gewissen Bedingungen sämtliche Zahlen von der Null an bis zur höchsten Zahl enthält. In diesem Zyklus würde ja die Lösung jeder Aufgabe enthalten sein, bei welcher verlangt wird, die vorhandene Literzahl in irgend zwei ganzzahlige Summanden zu zerlegen. Die Regel, nach welcher sich dieser Zyklus ganz von selbst ergibt, lautet: "Man fülle aus dem größten Gefäß in das kleinste, aus diesem in das mittlere, wiederhole diesen Vorgang so oft, bis das mittlere gefüllt ist, entleere dieses in das erste und beginne wieder von vorn, bis der ursprüngliche Zustand wiederkehrt."

```
100 : PRINT "□
110 : PRINT "                UMFUELLAUFGABE
111 : PRINT "                --------------
112 :
120 REM ES WIRD DAS KLASSISCHE DREI-KRUEGE-PROBLEM GELOEST
129 :
130 REM -- VARIABLEN:
131 :
140 REM   G, M, K ..... INHALT DES GROSSEN, MITTLEREN, KLEINEN KRUGS
150 REM   KG, KM, KK ... KAPAZITAETEN DER DREI KRUEGE
160 REM   AG, AM, AK ... ANFANGSFUELLUNGEN DER DREI KRUEGE
170 REM   EG, EM, EK ... ENDFUELLUNGEN DER DREI KRUEGE
180 REM   N ............ ANZAHL DER UMFUELLUNGEN
190 :
200 REM === EINGABE UND INITIALISIERUNG ==============================
201 :
210 : PRINT
220 : PRINT "GEBEN SIE DIE KAPAZITAETEN DER DREI KRUEGE EIN!
230 : INPUT "KAPAZITAET DES GROSSEN KRUGS   "; KG
235 : IF KG <= 0 THEN 230
240 : INPUT "KAPAZITAET DES MITTLEREN KRUGS "; KM
245 : IF KM <= 0 OR KM > KG THEN 240
250 : INPUT "KAPAZITAET DES KLEINEN KRUGS   "; KK
255 : IF KK <= 0 OR KK > KM THEN 250
260 :
270 : PRINT
275 : PRINT "GEBEN SIE DIE ANFANGSFUELLUNGEN DER DREI KRUEGE EIN!
280 : INPUT "ANFANGSFUELLUNG DES GROSSEN KRUGS   "; AG
285 : IF AG < 0 OR AG > KG THEN 280
290 : INPUT "ANFANGSFUELLUNG DES MITTLEREN KRUGS "; AM
300 : IF AM < 0 OR AM > KM THEN 290
305 : INPUT "ANFANGSFUELLUNG DES KLEINEN KRUGS   "; AK
310 : IF AK < 0 OR AK > KK THEN 305
315 :
320 : PRINT
325 : PRINT "GEBEN SIE DIE GEWUENSCHTEN ENDFUELLUNGEN EIN!
330 : INPUT "ENDFUELLUNG DES GROSSEN KRUGS   "; EG
335 : IF EG < 0 OR EG > KG THEN 330
340 : INPUT "ENDFUELLUNG DES MITTLEREN KRUGS "; EM
345 : IF EM < 0 OR EM > KM THEN 340
350 : INPUT "ENDFUELLUNG DES KLEINEN KRUGS   "; EK
355 : IF EK < 0 OR EK > KK THEN 350
357 : IF EG + EM + EK > AG + AM + AK THEN 320
360 :
370 : LET G = AG : LET M = AM : LET K = AK
380 : PRINT "■■■■" G;M;K
385 :
390 : LET N = 0 : REM ANZAHL DER UMFUELLUNGEN ZU BEGINN
395 :
400 REM === UNFUELLVORGANG (MIT AUSGABE) ============================
401 :
405 : REM --- AUSWAHL DES ZU LEERENDEN KRUGS
406 :
410 :    IF G = KG THEN 500 : REM UMGIESSEN GROSS  => KLEIN
420 :    IF M = KM THEN 530 : REM UMGIESSEN MITTEL => GROSS
430 :    IF K = KK THEN 560 : REM UMGIESSEN KLEIN  => MITTEL
440 :    IF G = 0  THEN 530 : REM UMGIESSEN MITTEL => GROSS
450 :    IF M = 0  THEN 560 : REM UMGIESSEN KLEIN  => MITTEL
460 :    IF K = 0  THEN 500 : REM UMGIESSEN GROSS  => KLEIN
470 :
```

```
500 : REM --- UMGIESSEN VOM GROSSEN IN DEN KLEINEN KRUG
501 :
505 :   LET A = G : LET B = K : LET KB = KK : REM PARAMETERUEBERGABE
510 :   GOSUB 700                           : REM UMGIESSEN
515 :   LET G = A : LET K = B                : REM PARAMETERUEBERGABE
520 :   PRINT G;M;K                          : REM AUSGABE
525 :   GOTO 600                             : REM ENDEPRUEFUNG
529 :
530 : REM --- UMGIESSEN VOM MITTLEREN IN DEN GROSSEN KRUG
531 :
535 :   LET A = M : LET B = G : LET KB = KG : REM PARAMETERUEBERGABE
540 :   GOSUB 700                           : REM UMGIESSEN
545 :   LET M = A : LET G = B                : REM PARAMETERUEBERGABE
550 :   PRINT G;M;K                          : REM AUSGABE
555 :   GOTO 600                             : REM ENDEPRUEFUNG
559 :
560 : REM --- UMGIESSEN VOM KLEINEN IN DEN MITTLEREN KRUG
561 :
565 :   LET A = K : LET B = M : LET KB = KM : REM PARAMETERUEBERGABE
570 :   GOSUB 700                           : REM UMGIESSEN
575 :   LET K = A : LET M = B                : REM PARAMETERUEBERGABE
580 :   PRINT G;M;K                          : REM AUSGABE
590 :
600 : REM --- ENDEPRUEFUNG
601 :
610 :   IF NOT (G = EG AND M = EM AND K = EK) THEN 400
620 :
630 :   PRINT
640 :   PRINT "INSGESAMT "; N;" UMFUELLUNGEN.
650 :
690 :   END
699 :
700 REM +++ UNTERPROGRAMM 'UMGIESSEN' +++++++++++++++++++++++++++++++++
701 :
710 : REM DER INHALT DES KRUGS A WIRD IN DEN KRUG B GEGOSSEN
711 : REM KB IST DIE KAPAZITAET DES KRUGES B
715 :
720 :   IF A+B < KB THEN LET B = A+B : LET A = 0 : GOTO 750
730 :   LET A = A - (KB-B) : LET B = KB
740 :
750 :   LET N = N+1 : REM WEITERZAEHLEN
760 :
780 :   RETURN
785 :
790 REM ENDE DES UMGIESSENS ++++++++++++++++++++++++++++++++++++++++++++
799 :
```

*

Aufgaben

20. Programmieren Sie die zweite Lösung des Drei-Krüge-Problems!

21. Schreiben Sie das Programm der Umfüllaufgabe für vier und mehr Krüge um.

22. Schreiben Sie ein Programm nach dem Muster des Fünfzehnerspiels, das den Weg zur vorgegebenen Endfüllung durch ein Suchverfahren ermittelt.

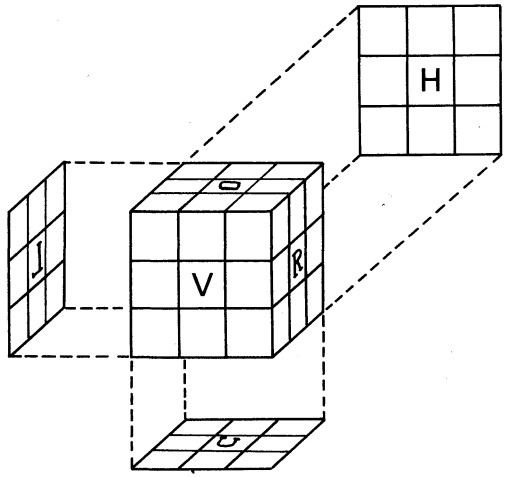
23. Bestätigen Sie an Beispielen und versuchen Sie zu beweisen: Sind die Kapazitäten der drei Krüge teilerfremde ganze Zahlen, so läßt sich durch Umfüllen jede mögliche Literzahl zwischen 1 und der Kapazität des großen Krugs erreichen.

Der Zauberwürfel - Krönung unseres Kapitels über Geduldspiele - hat die Welt des Spiels, der Mathematik und der Computer im Sturm erobert. Seine Wirkung ist mit der des Fünfzehnerspiels vergleichbar, das - wie bereits berichtet - Ende des vorigen Jahrhunderts eine wahre Massenpsychose auslöste. Beide Spiele sind vom gleichen Geist: ging es dort darum, 15 numerierte Scheibchen in einem 4 x 4 - Quadrat in die richtige Reihenfolge zu bringen, so hat man jetzt die farbigen Teilstücke eines 3 x 3 x 3 - Würfels so anzuordnen, daß jede Würfelseite nur eine Farbe zeigt. Bei beiden Geduldspielen wird vom Spieler verlangt, immer wieder mühsam Erreichtes scheinbar zu opfern, damit er ans Ziel gelangt: es gibt keinen Lösungsweg, bei dem man nicht zwischenzeitlich die bis dahin erreichte Ordnung aufgeben muß.

Beispiel 6 : Zauberwürfel

Ein Programm zur Simulation des Ungarischen Würfels ist zu entwerfen.

Das im folgenden vorgestellte Programm weist gegenüber dem realen Würfel den Vorteil auf, daß die Würfel-Grundkonstellation leicht wiederherzustellen ist (nämlich durch Programmabbruch und erneutes Starten). Es ist - im Gegensatz zum Fünfzehnerspiel - kein Suchprogramm, sondern es soll den Leser beim Erforschen des Würfels unterstützen. Dieser wird in dreidimensionaler Darstellung



auf den Bildschirm gebracht; durch Nennung eines der Buchstaben

V(orne), H(inten), R(echts), L(inks), U(nten), O(ben)

wird die zugehörige Scheibe um 90° im Uhrzeigersinn gedreht. Mit diesen Operationen lassen sich alle Anordnungen des Würfels gewinnen.

```
1000 : PRINT "■
1010 : PRINT "              ZAUBERWUERFEL
1020 : PRINT "              ------------
1030 :
1040 REM SIMULATION VON RUBIKS WUEFEL
1050 :
1060 :
1070 REM *** HAUPTPROGRAMM ******************************************
1080 :
1090 : GOSUB 2000 : REM ANLEITUNG
1100 :
1110 : GOSUB 3000 : REM INITIALISIERUNG
1120 :
1130 : GOSUB 4000 : REM AUSGABE DES WUERFELS
1140 :
1150 : GOSUB 5000 : REM ZUGEINGABE
1160 :
1170 : GOSUB 6000 : REM ZUGBERECHNUNG
1180 :
1190 : GOSUB 7000 : REM ZUGAUSFUEHRUNG
1200 :
1210 : GOTO  1150 : REM NAECHSTER ZUG
1220 :
1230 :
1240 REM ENDE DES HAUPTPROGRAMMS ************************************
1250 :
1260 :
2000 REM ### PROZEDUR 'ANLEITUNG' ###################################
2010 :
2020 : PRINT"■■■■ANLEITUNG ERWUENSCHT? (J/N)
2025 :
2030 : GET A$ : IF A$ =  "N" THEN RETURN
2040 :           IF A$ <> "J" THEN GOTO 2030
2050 :
2060 : PRINT "■";
2070 : PRINT "DIESES PROGRAMM IST EINE SIMULATION DES
2080 : PRINT "RUBIK-WUERFELS AUS UNGARN FUER EINEN
2090 : PRINT "COMMODORE-TISCHCOMPUTER MIT MINDESTENS
2100 : PRINT "16 K-BYTE RAM.
2165 : PRINT
2170 : PRINT "DIE ZUGEINGABE IST DEMENTSPRECHEND
2180 : PRINT "SIMPEL: ES IST NUR EINE ART DER DREHUNG
2190 : PRINT "MOEGLICH; UND ZWAR KOENNEN ALLE
2200 : PRINT "FLAECHEN DURCH EINFACHE EINGABE DES
2210 : PRINT "ANFANGSBUCHSTABEN DER JEWEILIGEN FLAECHE
2220 : PRINT "■IM UHRZEIGERSINN GEDREHT WERDEN.
2225 : PRINT
2230 : PRINT "ZUM PROGRAMMABBRUCH DRUECKEN SIE '0'.
2235 : PRINT
2240 : PRINT "LEERTASTE DRUECKEN.
2250 :
2260 : GET A$ : IF A$ <> " " THEN 2260
2950 :
2960 : RETURN
2970 :
2980 REM ENDE DER ANLEITUNG ##########################################
2990 :
2995 :
```

```
3000 REM ### PROZEDUR 'INITIALISIERUNG' ################################
3010 :
3020 : DIM W(6,3,3) : REM WURFELFELDER
3030 :
3040 :  FOR I = 1 TO 6
3050 :    FOR J = 1 TO 3
3060 :      FOR K = 1 TO 3
3070 :        LET W(I,J,K) = I
3080 :      NEXT K
3090 :    NEXT J
3100 :  NEXT I
3110 :
3120 : REM       FLAECHENCODES:
3125 : REM       --------------
3130 : REM       VORNE........1
3140 : REM       RECHTS.......2
3150 : REM       HINTEN.......3
3160 : REM       LINKS........4
3170 : REM       OBEN.........5
3180 : REM       UNTEN........6
3190 :
3950 :
3960 : RETURN
3970 :
3980 REM ENDE DER INITIALISIERUNG ######################################
3990 :
4000 REM ### PROZEDUR 'AUSGABE DES WUERFELS' ##########################
4010 :
4020 : PRINT "□";
4030 :
4100 : PRINT"                  ┌──────┐
4110 : PRINT"                  │      │
4120 : PRINT"                  │      │
4130 : PRINT"                  │      │
4140 : PRINT"                  │      │
4150 : PRINT"                  │      │
4160 : PRINT"                  └──────┘
4165 : PRINT"           ──────
4170 : PRINT"        ┌─┐   ┌──────┐
4180 : PRINT"      ┌─┤ │ ┌─┘    ┌─┤
4190 : PRINT"    ┌─┘ ├─┘ └──────┘ │
4200 : PRINT"    │   │┌──────┐    │
4210 : PRINT"    │   ┌┘      │    │
4220 : PRINT"    │   ┌┘      │    │
4230 : PRINT"    │   ┌┘      │    │
4240 : PRINT"    │ ┌─┘       │  ┌─┘
4250 : PRINT"    ├─┘ │       │ ┌┘
4260 : PRINT"    └─┐ └───────┘ ┌┘
4270 : PRINT"      └──────────┘
4280 : PRINT"           ┌─┘   ┌─┘
4290 : PRINT"         ┌─┘   ┌─┘
4300 : PRINT"       ┌─┘   ┌─┘
4310 : PRINT"       └─────┘
4320 :
4330 : GOSUB 7000 : REM ZUR BESCHRIFTUNG
4340 :
4350 : RETURN
4360 :
4370 REM ENDE DER AUSGABE DES WUERFELS ##################################
4380 :
4390 :
```

```
5000 REM ### PROZEDUR 'ZUGEINGABE' ####################################
5010 :
5020 : PRINT "■■■■■■■■■■■" TAB(20) "IHR ZUG
5030 : PRINT TAB(20) "(V,R,H,L,O,U):  ■■";
5050 :
5060 : REM --- TASTATURABFRAGE
5070 :
5080 :   GET A$ : IF A$ = "" THEN 5080
5081 :
5100 :   IF A$="V" THEN LET SE=1 : GOTO 5200
5110 :   IF A$="R" THEN LET SE=2 : GOTO 5200
5120 :   IF A$="H" THEN LET SE=3 : GOTO 5200
5130 :   IF A$="L" THEN LET SE=4 : GOTO 5200
5140 :   IF A$="O" THEN LET SE=5 : GOTO 5200
5150 :   IF A$="U" THEN LET SE=6 : GOTO 5200
5160 :   IF A$="Q" THEN PRINT "ENDE." : PRINT "■" : END : REM AUSSTIEG
5170 :
5180 :   GOTO 5080
5190 :
5200 :   PRINT A$
5210 :
5390 :   RETURN
5400 :
5410 REM ### ENDE DER ZUGEINGABE ######################################
5980 :
5990 :
6000 REM ### PROZEDUR 'ZUGBERECHNUNG' ##################################
6010 :
6020 : REM --- FLAECHE DREHEN
6030 :
6031 :   LET Z          = W(SE,3,1)
6032 :   LET W(SE,3,1) = W(SE,1,1)
6033 :   LET W(SE,1,1) = W(SE,1,3)
6034 :   LET W(SE,1,3) = W(SE,3,3)
6035 :   LET W(SE,3,3) = Z
6036 :
6037 :   LET Z          = W(SE,3,2)
6038 :   LET W(SE,3,2) = W(SE,2,1)
6039 :   LET W(SE,2,1) = W(SE,1,2)
6040 :   LET W(SE,1,2) = W(SE,2,3)
6041 :   LET W(SE,2,3) = Z
6045 :
6050 : REM --- KANTE DREHEN
6055 :
6060 :   ON SE GOSUB 6100 , 6200 , 6300 , 6400 , 6500 , 6600
6070 :
6080 :   RETURN
6090 :
6100 : REM --- VORNE (1)
6105 :
6110 :   LET Z         = W(5,1,1)
6111 :   LET W(5,1,1) = W(4,1,3)
6112 :   LET W(4,1,3) = W(6,1,1)
6113 :   LET W(6,1,1) = W(2,3,1)
6114 :   LET W(2,3,1) = Z
6115 :
6116 :   LET Z         = W(5,1,2)
6117 :   LET W(5,1,2) = W(4,2,3)
6118 :   LET W(4,2,3) = W(6,1,2)
6119 :   LET W(6,1,2) = W(2,2,1)
6120 :   LET W(2,2,1) = Z
6121 :
```

```
6122 :   LET Z         = W(5,1,3)
6123 :   LET W(5,1,3) = W(4,3,3)
6124 :   LET W(4,3,3) = W(6,1,3)
6125 :   LET W(6,1,3) = W(2,1,1)
6126 :   LET W(2,1,1) = Z
6127 :
6128 :   RETURN
6129 :
6200 : REM --- RECHTS (2)
6205 :
6210 :   LET Z         = W(5,1,3)
6211 :   LET W(5,1,3) = W(1,1,3)
6212 :   LET W(1,1,3) = W(6,3,1)
6213 :   LET W(6,3,1) = W(3,3,1)
6214 :   LET W(3,3,1) = Z
6215 :
6216 :   LET Z         = W(5,2,3)
6217 :   LET W(5,2,3) = W(1,2,3)
6218 :   LET W(1,2,3) = W(6,2,1)
6219 :   LET W(6,2,1) = W(3,2,1)
6220 :   LET W(3,2,1) = Z
6221 :
6222 :   LET Z         = W(5,3,3)
6223 :   LET W(5,3,3) = W(1,3,3)
6224 :   LET W(1,3,3) = W(6,1,1)
6225 :   LET W(6,1,1) = W(3,1,1)
6226 :   LET W(3,1,1) = Z
6227 :
6228 :   RETURN
6229 :
6300 : REM --- HINTEN (3)
6305 :
6310 :   LET Z         = W(5,3,3)
6311 :   LET W(5,3,3) = W(2,1,3)
6312 :   LET W(2,1,3) = W(6,3,3)
6313 :   LET W(6,3,3) = W(4,3,1)
6314 :   LET W(4,3,1) = Z
6315 :
6316 :   LET Z         = W(5,3,2)
6317 :   LET W(5,3,2) = W(2,2,3)
6318 :   LET W(2,2,3) = W(6,3,2)
6319 :   LET W(6,3,2) = W(4,2,1)
6320 :   LET W(4,2,1) = Z
6321 :
6322 :   LET Z         = W(5,3,1)
6323 :   LET W(5,3,1) = W(2,3,3)
6324 :   LET W(2,3,3) = W(6,3,1)
6325 :   LET W(6,3,1) = W(4,1,1)
6326 :   LET W(4,1,1) = Z
6327 :
6328 :   RETURN
6329 :
6400 : REM --- LINKS (4)
6405 :
6410 :   LET Z         = W(5,3,1)
6411 :   LET W(5,3,1) = W(1,3,1)
6412 :   LET W(1,3,1) = W(6,1,3)
6413 :   LET W(6,1,3) = W(3,1,3)
6414 :   LET W(3,1,3) = Z
6415 :
```

```
6416 :    LET Z         = W(5,2,1)
6417 :    LET W(5,2,1) = W(1,2,1)
6418 :    LET W(1,2,1) = W(6,2,3)
6419 :    LET W(6,2,3) = W(3,2,3)
6420 :    LET W(3,2,3) = Z
6421 :
6422 :    LET Z         = W(5,1,1)
6423 :    LET W(5,1,1) = W(1,1,1)
6424 :    LET W(1,1,1) = W(6,3,3)
6425 :    LET W(6,3,3) = W(3,3,3)
6426 :    LET W(3,3,3) = Z
6427 :
6428 :    RETURN
6429 :
6500 : REM --- OBEN (5)
6505 :
6510 :    LET Z         = W(1,3,3)
6511 :    LET W(1,3,3) = W(2,3,3)
6512 :    LET W(2,3,3) = W(3,3,3)
6513 :    LET W(3,3,3) = W(4,3,3)
6514 :    LET W(4,3,3) = Z
6515 :
6516 :    LET Z         = W(1,3,2)
6517 :    LET W(1,3,2) = W(2,3,2)
6518 :    LET W(2,3,2) = W(3,3,2)
6519 :    LET W(3,3,2) = W(4,3,2)
6520 :    LET W(4,3,2) = Z
6521 :
6522 :    LET Z         = W(1,3,1)
6523 :    LET W(1,3,1) = W(2,3,1)
6524 :    LET W(2,3,1) = W(3,3,1)
6525 :    LET W(3,3,1) = W(4,3,1)
6526 :    LET W(4,3,1) = Z
6527 :
6528 :    RETURN
6529 :
6600 : REM --- UNTEN (6)
6605 :
6610 :    LET Z         = W(1,1,1)
6611 :    LET W(1,1,1) = W(4,1,1)
6612 :    LET W(4,1,1) = W(3,1,1)
6613 :    LET W(3,1,1) = W(2,1,1)
6614 :    LET W(2,1,1) = Z
6615 :
6616 :    LET Z         = W(1,1,2)
6617 :    LET W(1,1,2) = W(4,1,2)
6618 :    LET W(4,1,2) = W(3,1,2)
6619 :    LET W(3,1,2) = W(2,1,2)
6620 :    LET W(2,1,2) = Z
6621 :
6622 :    LET Z         = W(1,1,3)
6623 :    LET W(1,1,3) = W(4,1,3)
6624 :    LET W(4,1,3) = W(3,1,3)
6625 :    LET W(3,1,3) = W(2,1,3)
6626 :    LET W(2,1,3) = Z
6627 :
6628 :    RETURN
6629 :
6630 :
6900 REM ### ENDE DER ZUGBERECHNUNG ####################################
6901 :
6999 :
```

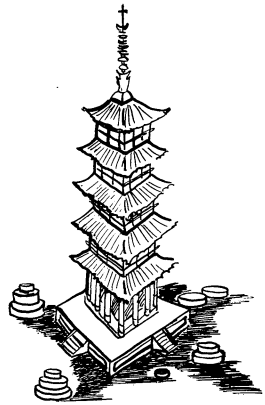
```
7000 REM ### PROZEDUR 'ZUGAUSFUEHRUNG' ###################################
7010 :
7020 : REM --- FLAECHE HINTEN (3)
7030 :
7040 :   PRINT "▒▒" TAB(17);
7050 :   FOR I = 3 TO 1 STEP -1
7070 :     FOR J = 3 TO 1 STEP -1
7080 :       PRINT RIGHT$(STR$(W(3,I,J)),1) "▒";
7090 :     NEXT J
7095 :     PRINT "▒▒▒▒▒▒▒▒";
7100 :   NEXT I
7110 :
7120 : REM --- FLAECHE RECHTS (4)
7130 :
7140 :   PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";
7150 :   FOR I = 3 TO 1 STEP -1
7160 :     FOR J = 3 TO 1 STEP -1
7170 :       PRINT RIGHT$(STR$(W(4,I,J)),1) "□";
7180 :     NEXT J
7190 :     PRINT "▒▒▒□□□□□□";
7200 :   NEXT I
7210 :
7220 : REM --- FLAECHE OBEN (5)
7230 :
7240 :   PRINT "▒▒▒▒▒▒▒▒▒"
7250 :   FOR I = 3 TO 1 STEP -1
7260 :     PRINT TAB(6+I);
7270 :     FOR J = 1 TO 3
7280 :       PRINT RIGHT$(STR$(W(5,I,J)),1) "▒";
7290 :     NEXT J
7295 :     PRINT
7300 :   NEXT I
7310 :
7320 : REM --- FLAECHE VORNE (1)
7330 :
7340 :   PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒"
7350 :   FOR I = 3 TO 1 STEP -1
7360 :     PRINT TAB(6);
7370 :     FOR J = 1 TO 3
7380 :       PRINT RIGHT$(STR$(W(1,I,J)),1) "▒";
7390 :     NEXT J
7395 :     PRINT "▒"
7400 :   NEXT I
7410 :
7420 : REM --- FLAECHE RECHTS (2)
7430 :
7440 :   PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒" TAB(12);
7450 :   FOR I = 3 TO 1 STEP -1
7460 :     FOR J = 1 TO 3
7470 :       PRINT RIGHT$(STR$(W(2,I,J)),1) "□";
7480 :     NEXT J
7490 :     PRINT "▒▒▒□□□□□□";
7500 :   NEXT I
7510 :
```

```
7520 : REM --- FLAECHE UNTEN (6)
7530 :
7540 :   PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
7550 :   FOR I = 3 TO 1 STEP -1
7560 :     PRINT TAB(6+I);
7570 :     FOR J = 3 TO 1 STEP -1
7580 :       PRINT RIGHT$(STR$(W(6,I,J)),1) "▒";
7590 :     NEXT J
7595 :     PRINT
7600 :   NEXT I
7610 :
7620 :   RETURN
7960 :
7970 REM ENDE DER ZUGAUSFUEHRUNG ########################################
7980 :
7990 :
```

Aufgaben

24. Turmspiel

Auf der Weltausstellung im Jahr 1899 zu Paris hatte der Mathematiker E. Lucas einige seiner mathematischen Spiele ausgestellt; darunter das Turmspiel. Dazu erzählte er eine phantasievoll ausgeschmückte Geschichte über die Entstehung des Spiels im fernen Osten (seither heißt es auch 'Türme von Hanoi').
Acht kreisförmige Scheiben sind der Größe nach auf einem von drei Stiften gestapelt. Ziel des Spiels ist es, den Turm auf einen der beiden anderen Plätze umzubauen. Dabei müssen die Scheiben einzeln von einem Turm (Stapel) auf einen anderen umgelegt werden, ohne daß je eine größere Scheibe über eine kleinere zu liegen kommt.
Schreiben Sie ein Programm, das dies Spiel auf dem Bildschirm ermöglicht, und welches auch auf die Einhaltung der Spielregeln achtet.



25. Hirnverzwirner

Auf einem 3 x 3 - Quadrat sind Nullen und Einsen zufällig verteilt. Ein Spielzug besteht in der Angabe einer der neun Positionen; daraus ergeben sich folgende unterschiedliche Wirkungen:

a) Wird eine der vier Ecken genannt, so ändern sich außer der Ecke selbst die beiden benachbarten Positionen sowie die Mitte des Quadrats; 'ändern' heißt: aus 0 wird 1 und umgekehrt.
b) Wird eine der vier Seitenmitten genannt, so ändern sich außer ihnen die beiden jeweils benachbarten Ecken.

c) Wird die Quadratmitte genannt, so ändern sich außer ihr auch alle vier Seitenmitten.

Schreiben Sie ein Programm, mit dem man dies Spiel auf dem Bildschirm spielen kann.

Ziel ist es, folgendes Quadrat zu erreichen:

1 1 1
1 0 1
1 1 1 .

26. Josephsspiel

Während der Niederschlagung des jüdischen Aufstands durch die römische Besatzung (um 70 n. Chr.) flohen 41 Juden in eine Höhle. Um der Gefangennahme zu entgehen, beschlossen Sie ein Programm zur gegenseitigen Vernichtung: sie wollten sich im Kreis aufstellen; dann sollte jeder neunte niedergemacht werden - bis auf zwei, die sich gegenseitig umzubringen hätten. Der spätere Geschichtsschreiber Flavius Josephus hat sich so aufgestellt, daß er und ein besonders schwächlicher Kamerad übrigblieben, den er leicht überwinden konnte. Wie hat er das gemacht?

Allgemein lautet das Problem: n Personen stehen im Kreis; jeder k-te wird ausgeschieden, wobei sich der Kreis wieder schließt. Gesucht ist die Reihenfolge der Ausgeschiedenen.

In der Folgezeit (besonders im Mittelalter) erlebte das Spiel ungezählte Variationen.

Können Sie so auszählen, daß nur die Frauen ausgeschieden werden? Oder nur die Männer? Für Ihre Antwort interessiert sich der 'Mathematiklehrer'.

9

Vermischte Knobeleien

In diesem abschließenden Kapitel werden wir einige Aufgaben unterschiedlichen Schwierigkeitsgrads zum Knobeln stellen, die sich mit Computerhilfe leichter lösen lassen und die damit zugleich reizvolle Programmieraufgaben bilden.

1. In den Räumen eines bekannten Würzburger Verlagshauses steht ein Personalcomputer. Seine Seriennummer besteht aus fünf verschiedenen Ziffern und ist durch neun teilbar; die erste Ziffer ist gerade und sie ist das Produkt der dritten und vierten Ziffer. Die Summe der beiden ersten Ziffern ist fünfzehn, die dritte Ziffer ist die Differenz der beiden ersten.
   Wie lautet die Nummer? (Schreiben Sie ein Suchprogramm!)

2. Im New Yorker Telefonbuch, das weniger als 1000 Seiten umfaßt, stehen genau 999 991 Telefonkunden. Auf jeder Seite steht die gleiche Anzahl von Kunden. Wieviel Seiten hat das Buch?

3. Otto ist ein zuverlässiger Knabe, wenn es um Mathematik geht. Sie können sich darauf verlassen, daß seine Entdeckung Hand und Fuß hat, wenn er behauptet, daß es nur noch eine zweite Lösung für sein Problem gibt. Er verlangt, eine siebenstellige Zahl zu finden, die nach Multiplikation mit einer gewissen ganzen Zahl ihre Spiegelzahl ergibt. Seine Lösung lautet: $1099989 \cdot 9 = 9899901$. Welches ist die andere Ottozahl?

4. Es ist
$$89 = 8^1 + 9^2$$
$$135 = 1^1 + 3^2 + 5^3$$
$$2427 = 2^1 + 4^2 + 2^3 + 7^4$$
Gibt es weitere Zahlen mit dieser Eigenschaft? Lassen Sie den Computer suchen!

5. Mit
$$81 = 9^2 = (8+1)^2$$
$$512 = 8^3 = (5+1+2)^3$$
haben wir Beispiele n-stelliger Zahlen, die gleich der n-ten Potenz ihrer Quersumme sind. Finden Sie Beispiele für $n = 4$ und zeigen Sie, daß für $n = 5$ keine Lösung existiert.

6. Sechs Zahlen zwischen 1 und 10 lösen folgende Gleichungen:
$$A + B + C = D + E + F$$
$$A^2 + B^2 + C^2 = D^2 + E^2 + F^2 \quad ,$$
fünf davon sind paarweise verschieden (zwei sind gleich).

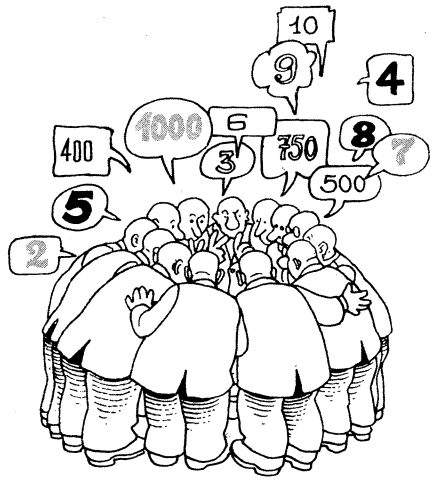
7. Die Zahl 473 684 210 526 315 789 hat die bemerkenswerte Eigenschaft, daß sich ihr Doppeltes ergibt, wenn man ihre Einerziffer 9 ganz links ansetzt. Gibt es weitere Beispiele?

8. Es ist
$$60 \cdot 84 = 35 \cdot (60 + 84),$$
d.h. die Summe von 60 und 84 teilt ihr Produkt. Ist dies das einzige Zahlenpaar mit dieser Eigenschaft? Lassen Sie sich vom Computer helfen!

9. Otto sucht die kleinste Zahl mit der Eigenschaft, daß die Multiplikation mit ihrer Einerziffer die gleiche Wirkung hat, wie wenn man die Einerziffer ganz links an die Zahl anfügt.

10. Noch eine merkwürdige Zahl: vertauscht man die Ziffern von 148 im Kreise, so erhält man die Zahlen 481 und 814. Ihr Abstand voneinander ist jeweils gleich. Weitere Beispiele gesucht!

11. Fünfzehn Personen unterhalten sich über eine Zahl, zwei davon lügen, die übrigen sagen die Wahrheit.
Erster: "Die Zahl ist ein Vielfaches von zwei."
Zweiter: "Die Zahl ist ein Vielfaches von drei."
Dritter: "Die Zahl ist ein Vielfaches von vier."
Vierter: "Die Zahl ist ein Vielfaches von fünf."
Fünfter: "Die Zahl ist ein Vielfaches von sechs."
Sechster: "Die Zahl ist ein Vielfaches von sieben."
Siebter: "Die Zahl ist ein Vielfaches von acht."
Achter: "Die Zahl ist ein Vielfaches von neun."
Neunter: "Die Zahl ist ein Vielfaches von 10."
Zehnter: "Die Zahl ist kleiner als 1000."
Elfter: "Die Zahl ist kleiner als 750."
Zwölfter: "Die Zahl ist kleiner als 550."
Dreizehnter: "Die Zahl ist kleiner als 500."
Vierzehnter: "Die Zahl ist größer als 400."
Fünfzehnter: "Die Zahl ist größer als 450." Wie heißt die Zahl?

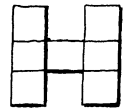


12. Vier Spieler kommen überein, daß jeder, der verliert, den übrigen jeweils den Betrag zahlt, den diese bereits besitzen. Nach vier Partien hat jeder genau einmal verloren. Am Ende zählen die Spieler ihr Kapital, und es stellt sich heraus, daß alle die gleiche Summe, nämlich 160 DM besitzen. Wieviel besaß jeder vor dem Spiel?

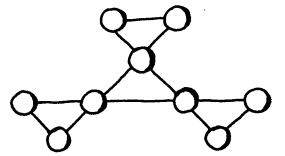
13. Am Neujahrstag 1982 fährt Herr Vogel mit der Bahn von Würzburg nach München (zur CHIP-Redaktion). Ihm gegenüber sitzt Herr Specht. Im Lauf des Gesprächs kommt die Rede aufs Alter. Herr Vogel äußert: "Ich bin so alt, wie die Summe der Ziffern meines Geburtsjahrs angibt." Nach einigem Überlegen gratuliert ihm Herr Specht zum Geburtstag. Wie alt ist Herr Vogel?

14. Sechs Personen gehen zu Tisch. Martina sitzt links von Jan, Gabriela sitzt nicht neben Klaus und nicht rechts von Doris, sondern Alex gegenüber. In welcher Anordnung sitzen sie um den Rundtisch?

15. Die Ziffern 1 bis 7 sind so zu plazieren, daß in jeder Zeile, Spalte und Diagonale die gleiche Summe entsteht.



16. Die Ziffern 1 bis 9 sind so einzuschreiben, daß ihre Summe in jedem der vier Dreiecke immer die gleiche ist.



17. Das Zahlenschema

1	2	3
8	9	4
7	6	5

ist ein sogenannten antimagisches Quadrat: keine zwei Zeilen, Spalten oder Diagonalen haben die gleiche Summe. Lassen Sie den Computer nach weiteren antimagischen Quadraten - auch höherer Ordnung - suchen!

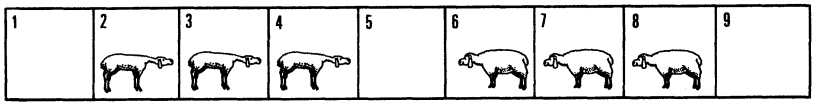
18. Bei Sportwettbewerben wird der Rangplatz einer Mannschaft häufig durch Addition der Plätze der einzelnen Mannschaftsmitglieder ermittelt. Sind etwa die Plätze 1, 3, 6 erreicht worden, so bekommt die Mannschaft die Platzziffer $1+3+6 = 10$. Mit ihr wird noch der erste Mannschaftsplatz erreicht, da die Mitglieder einer anderen Mannschaft bestenfalls die Plätze 2, 4, 5 mit der schlechteren Platzziffer 11 erreichen können (zwei erste Plätze und dergleichen soll es nicht geben, weder für die Mannschaft noch für die einzelnen).
Die 10 ist überdies in dem Sinne maximal, daß die nächsthöhere Platzziffer 11 nicht in jedem Fall den ersten Mannschaftsplatz zur Folge hat, wie soeben aus dem Beispiel ersichtlich.
Die Frage lautet nun: Wie groß ist bei gegebener Mannschaftsstärke $m \in \{1,2,3,\ldots\}$ die maximale Platzziffer p_m für den ersten Mannschaftsplatz? (Man findet $p_1 = 1$, $p_2 = 4$, $p_3 = 10$).

19. Von einer Funktion f ist folgendes bekannt:

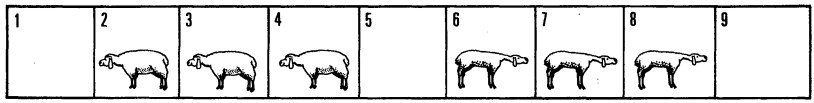
$$f(0, y) = y + 1$$
$$f(x+1,0) = f(x , 1)$$
$$f(x+1,y+1) = f(x , f(x+1,y)) .$$

Was ist f(4, 1981)?

20. Wie können die Schafe aus dieser Position

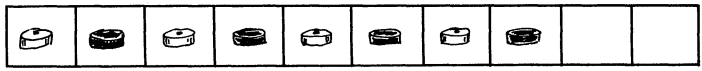


in die folgende gelangen?

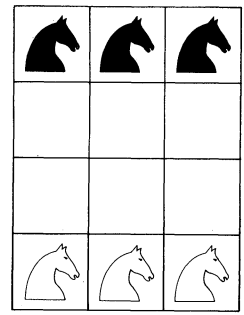


Erlaubt ist das Betreten eines leeren Feldes und das Überspringen eines Schafes auf ein leeres Feld. Nicht erlaubt ist das Rückwärtsgehen oder das Überspringen zweier Schafe.
Schreiben Sie ein Programm für dies Spiel!

21. Die vier weißen und vier schwarzen Damesteine sollen in fünf Zügen so umgelegt werden, daß die vier weißen und die vier schwarzen Steine zusammenliegen. Dabei dürfen in einem Zug immer nur zwei nebeneinanderliegende Steine umgelegt werden, und zwar so, daß diese Steine ihre Lage zueinander vertauschen.



22. Die Positionen der schwarzen und der weißen Springer sollen mit möglichst wenig Zügen vertauscht werden.
Schreiben Sie ein Suchprogramm!
(16 Züge genügen.)



23. Reverend Charles L. Dogson, besser bekannt unter seinem Pseudonym Lewis Carrol ('Alice im Wunderland'), berichtet in seinem Tagebuch, daß er sich vergeblich bemüht habe, drei flächengleiche rechtwinklige Dreiecke mit ganzzahligen Seitenlängen zu finden.
Lassen Sie den Computer helfen!

24. Martina möchte ihre 15 Freunde abwechselnd zum Abendessen einladen und an 35 Tagen jeweils mit drei Freunden zu Abend essen. Sie will die Einladungen so arrangieren, daß zwei der Freunde nicht öfter als an nur einem Abend bei ihr zusammentreffen. Läßt sich diese Absicht verwirklichen? (Fragen Sie den Computer!)

25. In einem Kongreßort treffen mit demselben Zug Delegationen von sieben Staaten ein, jede aus drei Herren bestehend. Sie werden in sieben Autos vom Bahnhof abgeholt. In jedem Auto sollen drei Herren platznehmen, die verschiedenen Staaten angehören. Man will erreichen, daß schon auf der Fahrt ein möglichst enger Kontakt zwischen allen Delegationen entsteht. Würde man die Verteilung der Delegationen nach dem Schema (beispielsweise)

    1,2,3; 2,3,4; 3,4,5; 4,5,6; 5,6,7; 6,7,1; 7,1,2

    vornehmen, so würde die Vertretung des Staates 1 zwar mit denen der Staaten 2,3,6,7 in Kontakt kommen, aber nicht mit denen der Staaten 4 und 5. Wie muß man es machen, damit man auch letzteres erreicht?

26. Sieben Herren bilden einen Spielklub. Jeden Abend (auch sonntags) sollen drei von ihnen sich zum Spiel zusammenfinden. Um die Bande der Freundschaft immer enger zu knüpfen, will man haben, daß möglichst jeder mit jedem allwöchentlich einmal zusammen am Spieltisch sitzt.

27. Jeden Tag in der Woche macht ein Lehrer mit 15 Schulmädchen einen Spaziergang, wobei die Mädchen wohlgeordnet in Dreiergruppen gehen. Kann der Lehrer die Dreiergruppen so arrangieren, daß zwei Mädchen, die einmal zusammen in einer der Dreiergruppen wanderten, an keinem der restlichen Tage der Woche wieder gemeinsam einer Dreiergruppe angehören?

28. Neun gefährliche Häftlinge dürfen ihren Hofgang im Gefängnis nur gefesselt antreten. Sie sind an den sieben Wochentagen in Dreiergruppen aneinandergekettet, und es dürfen innerhalb einer Woche niemals die beiden gleichen Häftlinge aneinandergefesselt sein. Lassen Sie die möglichen Kombinationen vom Computer finden!

Als Bonbon für den Leser, der bis hierher durchgehalten hat, ein hübsches Programm zum Knobeln.
Es wird den Benutzer auffordern, eine Frage an das Orakel zu stellen, das Liebesorakel, versteht sich. Und zwar soll die Frage möglichst allgemein formuliert sein, denn mit konkreten Einzelheiten gibt sich ein zünftiges Orakel bekanntlich nicht ab. Fragen Sie also bitte nicht nach dem Namen Ihrer / Ihres Zukünftigen. Fragen Sie auch nicht, wieviele Kinder das Schicksal (oder der Zufall) Ihnen zugedacht hat. Fragen Sie also ganz allgemein - etwa so: "Werde ich in nächster Zeit Glück in der Liebe haben?", oder: "Was wird mir der Gott der Liebe bescheren?", oder: "Was steht mir in den nächsten zehn Jahren bevor?"
Tippen Sie die Frage ein und drücken Sie die RETURN-Taste: dann erscheint des Orakels Antwort auf dem Bildschirm. Und nun dürfen Sie knobeln, wie das Programm zu so viel Voraussicht kommt. Was ist das Geheimnis der hellsehenden Zahlen?

```
100 : PRINT "␣
110 : PRINT "          ,          LIEBESORAKEL
111 : PRINT "                     ------------
112 :
200 REM === EINGABE ================================================
201 :
205 : PRINT
210 : PRINT "STELLEN SIE IHRE SCHICKSALSFRAGE!
215 : PRINT
220 : INPUT FR$
230 : LET L = LEN(FR$)
240 : IF L < 5 THEN PRINT "ETWAS AUSFUEHRLICHER BITTE!": GOTO 205
250 :
255 : PRINT
260 : PRINT "DAS ORAKEL SPRICHT (BITTE ETWAS GEDULD):
265 : PRINT
270 :
280 : DIM T(3,139)
290 : FOR I = 0 TO 3: FOR J = 0 TO 139: READ T(I,J): NEXT J,I
299 :
```

```
300 REM === AUSWERTUNG DER SCHICKSALSFRAGE ============================
301 :
310 : FOR K = 0 TO 3
320 :   LET Z = INT(L*RND(1)+1)
330 :   LET A = ASC(MID$(FR$,Z,1))
340 :   LET R = A - 9*INT(A/9)+1
350 :   LET Z(K) = R
360 : NEXT K
390 :
400 REM === BEFRAGUNG DER HELLSEHENDEN ZAHLEN ========================
401 :
410 : FOR I = 0 TO 3
420 :   FOR J = Z(I)-1 TO 139 STEP 9
430 :     IF T(I,J) = 0 THEN PRINT " ";: GOTO 460
440 :     PRINT CHR$(64+T(I,J));
450 :   NEXT J
460 : NEXT I
470 :
480 : PRINT: PRINT
485 :
490 : END
499 :
500 REM === ZAHLENTAFELN ============================================
510 :
511 DATA 5,4,5,4,5,4,2,5,5,9,5,9,1,9,1,5,9,9,14,18
512 DATA 14,19,14,19,26,14,14,2,12,7,19,4,7,1,5,8,12,9,12,21
513 DATA 21,18,21,18,15,15,5,21,5,14,15,2,18,12,14,2,5,19,11,19
514 DATA 5,5,4,4,5,8,19,12,19,18,7,5,5,0,5,5,5,5,14,5
515 DATA 19,19,7,14,19,19,0,4,14,0,0,13,4,20,0,24,5,4,11,23
516 DATA 16,5,5,0,4,19,5,20,8,25,19,0,12,1,0,19,12,6,24,0
517 DATA 3,18,6,13,0,21,17,5,4,5,11,22,0,16,5,0,15,4,6,2
520 :
521 DATA 5,19,6,7,22,23,14,1,20,20,3,5,9,5,21,9,2,18,23,8
522 DATA 21,6,18,14,3,5,1,1,9,5,20,8,4,8,14,21,19,3,18,23
523 DATA 1,5,20,20,13,23,11,23,9,5,18,19,5,2,9,19,9,18,14,23
524 DATA 23,21,9,18,1,18,4,7,9,9,5,12,4,12,4,0,14,18,18,18
525 DATA 4,0,23,0,15,9,4,4,23,23,4,9,11,17,19,0,0,9,9,15
526 DATA 18,3,19,23,2,1,18,18,0,4,25,6,9,22,7,4,4,15,0,22
527 DATA 1,18,7,26,0,0,5,11,4,7,4,0,10,23,9,1,18,13,13,0
530 :
531 DATA 4,4,4,4,4,4,4,4,5,9,5,5,5,9,5,9,5,9,3
532 DATA 9,9,9,3,9,3,9,14,8,14,14,14,8,14,8,14,8,0,9,2
533 DATA 7,0,5,0,5,5,23,14,12,5,8,19,26,19,18,7,14,21,23,4
534 DATA 9,17,5,26,13,5,20,9,17,14,11,5,0,13,18,0,19,3,14,1
535 DATA 12,5,20,19,20,19,13,5,19,5,6,14,20,1,5,20,0,3,0,9
536 DATA 21,5,2,14,0,8,13,18,6,11,19,8,0,5,4,6,4,0,19,0
537 DATA 21,18,21,24,15,2,26,3,14,5,24,7,3,18,22,6,0,5,8,14
540 :
541 DATA 2,8,1,2,5,19,2,26,2,5,5,21,5,9,20,5,21,5,7,9
542 DATA 6,18,14,1,20,6,18,12,13,23,1,19,21,15,1,21,21,19,21,21
543 DATA 3,14,5,12,5,5,21,5,19,8,5,18,12,3,3,3,8,3,12,14
544 DATA 5,2,11,11,8,12,8,1,13,14,18,5,5,5,5,5,5,1,0,9
545 DATA 14,14,14,14,14,6,3,2,14,0,0,0,0,0,5,8,3,7,16,9
546 DATA 25,16,8,18,5,24,5,9,13,6,18,14,14,14,1,14,7,0,1,4
547 DATA 23,0,0,4,0,16,22,25,13,11,0,7,11,6,4,0,3,1,1,11
```

*

10
Anhang

10.1 Anmerkungen

Seite 13
"Wer paßt zu wem?" geht auf ein Party-Spiel von E. Flögel in ELCOMP 4 (1981) zurück.

Seite 60
Das Programm "Eidechse" stammt aus der Zeitschrift CREATIVE COMPUTING.

Seite 63
"Ballons" wurde von Klaus Dorwald und Rolf Falkenberg nach einer Vorlage von Knut Hornung geschrieben.

Seite 67
Das Programm "Labyrinth" wurde von Kersten Heger entworfen.

Seite 74
"Zurück von Klondike" steht in [19], S. 103 f. Das Programm stammt von Rolf Falkenberg.

Seiten 79 und 88
Genaueres über die technischen Grundlagen findet man bei Schnell-Hoyer, Sacht und Zaks sowie im CBM-Computer-Handbuch.

Seite 89
Das Foto der Gleiterkanone wurde von Reinhard Hiß aufgenommen. Das Maschinenprogramm stammt von Kai Prisille.

Seite 108
"Knack den Code" wurde von Uwe Mylatz geschrieben.

Seite 115
LAP wurde von L. Pijanowski erfunden; das Spiel wird bei Sackson ([29], S. 64 ff) beschrieben.

Seite 122
Das Programm "Die böse Sechs" stammt von Stefanie Bruning.

Seite 142
Zum Roulette vergleiche man [30].

Seite 195
Das Brückenspiel wird bei Gardner in [9], S. 173 ff. beschrieben. Das Programm dazu stammt von Klaus Dorwald.

Seite 202 ff.
Zum Schachproblem vergleiche man den Artikel "Wie ein Computer Schachspielen lernt" von F. Schwenkel in [31]. Das 6 x 6 - Mikroschach-Programm wurde von Birger Roloff entworfen.

Seite 214
Aufgabe 39 entspricht A. Randolphs "Pferdejagd" (siehe [29, S.80 ff]).

Seite 230 ff.
Das Spiel "Rate die Karte" wird bei Gardner [13], S. 37 ff. beschrieben. Stefan Graeber hat das unleserliche Programm bei Ahl [2] lesbar gestaltet.

Seite 246
Zu dieser Konstruktion magischer Quadrate vergleiche man Hotje [33].

Seite 249
Die magischen Quadrate gerader Ordnung hat Uwe Mylatz konstruiert.

Seite 257
Die Bewegung der Damen auf dem Schachbrett wurde von Kai Prisille sichtbar gemacht.

Seite 248
Über den Satz von Larson berichtet M. Gardner in: Spektrum der Wissenschaft 1981.

Seite 266
Wir folgen der Darstellung von Williams, G.: Tree Searching. In: BYTE, September 1981, S. 72 ff.

Seite 281
Das Programm zum Zauberwürfel stammt von Klaus Dorwald. Andere (doch unleserliche) Programme finden sich in [34].

Seite 297
Das Liebesorakel wurde erfunden von S. Strehl: Fröhliche Wissenschaft, Nürnberg 1941 (Nachdruck: TIPs 10/81, Klett-Verlag).

10.2 Literaturverzeichnis

[1] AHL, D.A.(ed.): Basic Computer Games. Morristown 1978

[2] --- : More Basic Computer Games. Morristown 1979

[3] AHRENS, W.: Mathematische Spiele. Leipzig 1927 (Nachdruck 1979)

[4] CONWAY, J.H.: On Numbers and Games. London 1976

[5] DELFT, P., und J. BOTERMANS: Denkspiele der Welt. München 1977

[6] DRIPKE, A.: CBM-Spiele-Buch I. Vaterstetten 1981

[7] EPSTEIN, R.A.: The Theory of Gambling and Statistical Logic. New York 1977

[8] GARDNER, M.: Mathematische Rätsel und Probleme. Braunschweig 1968

[9] --- : Mathematische Knobeleien. Braunschweig 1973

[10] --- : Logik unterm Galgen. Braunschweig 1971

[11] --- : Mathematischer Karneval. Berlin 1975

[12] --- : Mathematisches Labyrinth. Braunschweig 1979

[13] --- : Mathematische Hexereien. Berlin 1979

[14] GLOISTEHN, H.H.: Mathematische Unterhaltungen und Spiele mit dem programmierbaren Taschenrechner. Braunschweig 1981

[15] GROSSE, W.: Unterhaltende Probleme und Spiele in mathematischer Beleuchtung. Leipzig 1897

[16] KERST, B.: Mathematische Spiele. Berlin 1933 (Nachdruck 1968)

[17] KRAITCHIK, M.: Mathematical Recreations. London 1943

[18] KOWALEWSKI, G.: Alte und neue mathematische Spiele. Leipzig 1930 (Nachdruck 1978)

[19] LOYD, S., und M. GARDNER: Mathematische Rätsel und Spiele. Köln 1978

[20] --- : Noch mehr mathematische Rätsel und Spiele. Köln 1979

[21] MENZEL, K.: Basic in 100 Beispielen. Stuttgart 1981

[22] RÖHRS, H.(Hrsg.): Das Spiel - ein Urphänomen des Lebens. Wiesbaden 1981

[23] SPENCER, D.D.: Game Playing with Basic. Rochelle Park 1977

[24] SPRAGUE, R.: Unterhaltsame Mathematik. Braunschweig 1961

Zu den einzelnen Kapiteln

Kapitel 3

[25] OSBORNE, A., und C.S. DONAHUE: CBM-Computer-Handbuch. München 1981

[26] SACHT, H.J.: Mikroprozessor-Programmierfibel. Würzburg 1980

[27] SCHNELL, G., und K. HOYER: Mikrocomputerfibel. Braunschweig 1981

[28] ZAKS, R.: Programmierung des 6502. Würzburg 1980

Kapitel 4

[29] SACKSON, S.: Spiele anders als andere. München 1981

Kapitel 5

[30] FABER, W.: Roulette. München 1978

Kapitel 6

[31] KETTERLING-SCHWENKEL-WEINER: Schach dem Computer. München 1981

[32] SCHRAGE, G.: Strategiespiele und Gewinnstrategien.
In: Mathematische Semesterberichte XXVIII (1981), 92 - 103

Kapitel 8

[33] HOTJE, H.: Zur Konstruktion von magischen Quadraten.
In: Mitteilungen der Mathematischen Gesellschaft in Hamburg 1978

[34] MORWKA, M., und M. WEBER: Mathematische Überlegungen zum Würfel.
In: Der Mathematikunterricht 27 (1981), Heft 6

10.3 BASIC

Im folgenden wird ein Überblick über die wichtigsten BASIC-Anweisungen und Kommandos gegeben (Version COMMODORE).

ABS ordnet einer Zahl ihren Absolutbetrag zu. Beispiel: PRINT ABS(3) ergibt 3 , PRINT ABS(-3) ergibt ebenfalls 3 .

AND verknüpft zwei logische Ausdrücke dergestalt, daß das Ergebnis genau dann den Wahrheitswert 'wahr' hat, wenn beide Ausdrücke 'wahr' sind.

ASC ordnet einem Zeichen seine ASCII-Nummer zu. Beispielsweise ergibt PRINT ASC("A") die Zahl 65.

CHR$ ist die Umkehrfunktion zu ASC: jeder natürlichen Zahl zwischen 0 und 255 wird das zugehörige ASCII-Zeichen zugeordnet. Beispiel: die Anweisung PRINT CHR$(65) liefert den Buchstaben A.

CLR löscht alle Variablen (d.h. weist numerischen Variablen den Wert 0, Zeichenketten-Variablen die leere Zeichenkette zu), Tabellen-Dimensionierungen, Unterprogramm-Rücksprungadressen.

CONT dient der Fortsetzung des Programmablaufs nach Drücken der Stop-Taste oder nach Ausführung der Anweisungen STOP bzw. END.

DATA / READ / RESTORE sind Anweisungen zur Speicherung und zum Lesen von Daten im Programm.

DEF FN dient zur Definition einer Funktion. Beispiel: DEF FN Y(X) = X*X . Mit (beispielsweise) PRINT FN Y(5) wird die Funktion 'aufgerufen', d.h. es wird 5*5 = 25 gedruckt.

DIM kündigt den Umfang einer Tabelle an (zur Bereitstellung von Speicherplatz).

END beendet die Programmausführung und läßt den Rechner zum direkten Dialog zurückkehren.

FOR ... NEXT (STEP) dient zur Programmierung der Zählschleife (gezählten Wiederholung).

GET liest ein einzelnes Zeichen vom Tastenfeld ein; der Computer wartet nicht auf den Tastendruck (im Gegensatz zu INPUT).

GOSUB ... RETURN dient zum Sprung in ein Unterprogramm sowie zur Rückkehr ins aufrufende Programm.

GOTO ist die (unbedingte) Sprunganweisung.

IF ... THEN ist die Entscheidungsanweisung (bedingte Anweisung).

INPUT dient zur Eingabe von Daten über das Tastenfeld.

INT ordnet einer Dezimalzahl die nächstkleinere ganze Zahl zu. Beispiel:
PRINT INT(-3.14) liefert -4.

LEFT$ holt aus einer Zeichenkette die linksstehenden Zeichen heraus.
Beispiel: PRINT LEFT$("STROHFEUER", 5) liefert STROH.

LEN ordnet einer Zeichenkette die Anzahl ihrer Zeichen zu.

LET ist die Wertzuweisung.

LIST listet ein ganzes Programm oder einzelne Zeilen bzw. Bereiche auf.

MID$ holt aus einer Zeichenkette gewisse Zeichen oder Teilketten heraus.
Beispiel: PRINT MID$("STROHFEUER", 3, 3) liefert ROH.

NEW löscht das gesamte Programm.

NOT ordnet einem logischen Ausdruck seine Negation zu.

ON ... GOTO / ON ... GOSUB ist der berechnete Sprung bzw. Unterprogrammsprung.

OR verknüpft zwei logische Ausdrücke dergestalt, daß das Ergebnis genau dann den Wahrheitswert 'wahr' hat, wenn mindestens einer der beiden Ausdrücke 'wahr' ist.

PEEK dient zum Lesen des Inhalts einer beliebigen Speicherzelle.

POKE dient zum Schreiben von Daten in eine beliebige Speicherzelle.

PRINT ist die Ausgabe-Anweisung (auf den Bildschirm).

REM ermöglicht Programm-Kommentar; wird vom Rechner nicht beachtet.

RIGHT$ holt aus einer Zeichenkette rechtsstehende Zeichen heraus.

RND erzeugt (pseudo-)zufällige Zahlen zwischen 0 und 1.

RUN startet ein Programm.

TAB setzt den Tabulator.

CHIP. Das Mikrocomputer-Fachmagazin. Für alle, die mehr wissen wollen und mehr wissen müssen, wenn es um Mikrocomputer geht. In Hobby und Beruf. Denn **CHIP** bringt umfassend und verständlich alles worauf es ankommt: Hintergrundberichte, Beispiele, umfassende Anleitungen, Software, Nachrichten, Trends und Meinungen. Ausgabe für Ausgabe eine geballte Ladung an Wissenswertem und Wichtigem. Damit Sie Bescheid wissen und mitreden können.

CHIP gibt es monatlich im Zeitschriften- und Buchhandel. Oder einfach bestellen beim **CHIP**-Leserservice, Vogel-Verlag Postfach 67 40, D-8700 Würzburg 1

ISBN 3-8023-0703-8

Spielen ist so alt wie die Menschheit. Einen neuen Grad der Bedeutung erhält das Spielen durch größere Freizeitreserven und vor allem durch den Computer, einen hervorragenden Partner. Das Faszinierende ist, daß man einer Maschine bestimmte Verhaltensweisen beibringen und sie so zu einem ernsthaften Spielgegner machen kann. Dieses Buch ist keine Sammlung von Spielkonserven. Wichtig ist der aktive Umgang mit Computerspielen: Aus der Idee entsteht das Programm – das Programmieren des Computers ist das Spiel!

VOGEL-BUCHVERLAG
WÜRZBURG

Kompetent für Technik · Elektronik
Management · Wirtschaft