Lübbes Computerbücher

Howard Budin

Computer-spiele

mit Commodore 64

BASTEI LÜBBE

BASTEI
LÜBBE

Howard Budin

Computer-spiele

mit Commodore 64

Aus dem Amerikanischen übersetzt
von Bernd Gatter

BASTEI
LÜBBE

BASTEI LÜBBE TASCHENBUCH

Band 63 084

Deutsche Erstveröffentlichung

Titel der Originalausgabe: SPEED WALKER —
FUN TO PROGRAM YOUR COMMODORE 64

Erschienen bei Pinnacle Books, Inc., New York

Einbandgestaltung: Roberto Patelli

Illustrationen: Cris Hammond
Gesamtherstellung: Ebner Ulm

ISBN 3-404-63084-0

Der Preis dieses Bandes versteht sich einschließlich der gesetzlichen Mehrwertsteuer.

Inhalt

Einleitung

Willkommen bei der Programmierung mit Speed Walker! Wenn Sie schon etwas von BASIC verstehen und es anwenden wollen, können wir zusammen einige interessante Programme entwikkeln. Wichtig ist, daß Sie die folgenden BASIC-Kommandos kennen:

REM	FOR/NEXT	END
PRINT	IF/THEN	
INPUT	GOTO	

Ein Programmierer sein bedeutet gleichzeitig ein Detektiv sein — denn es gibt ständig neue Geheimnisse in Programmen aufzudecken. Programmierer verbringen die Hälfte ihres Lebens mit der Suche nach möglichen Fehlerquellen in ihren Programmen. Um Ihnen nun das Lesen, Verändern und Testen Ihrer Programme zu erleichtern, benutzen wir in diesem Buch die strukturierte Programmierung. Das bedeutet, daß wir zunächst die Hauptteile des Programms planen und dann erst in die Herstellung der Einzelteile einsteigen. Wir benutzen häufig REM-Anweisungen, um die Bedeutung von Variablen, die wir verwenden, und den Inhalt von einzelnen Programmteilen genau zu dokumentieren. Wir rükken auch einzelne Programmbausteine ein, um sie von anderen abzuheben beziehungsweise ihre Abhängigkeit von anderen Befehlen aufzuzeigen.

Wenn wir uns auf dieses Spiel einlassen, werden wir mysteriöse Techniken kennenlernen: die Voraussage der Zukunft, die Verschlüsselung und Dechiffrierung von geheimen Nachrichten, die Bewegung von Objekten kreuz und quer über den ganzen Schirm. In jedem Kapitel bauen wir neue Konzepte auf, erweitert sich unser Befehlsvorrat, und die Kombination schon bekannter und neuer Techniken führt dann zu einem weiteren Spiel. Jeder neuer Gedanke wird ausführlich behandelt. Darüber hinaus bringen wir Beispiele, die Sie in Ihren Computer eingeben und ausprobieren können. Im letzten Kapitel haben wir dann ein weitaus komplizierteres Programm konstruiert als das, womit wir im ersten Kapitel begonnen haben. Seine Handhabung dürfte Ihnen keine Schwierigkeiten bereiten, da wir uns Zug um Zug mit allen Bausteinen dieses Programms anfreunden werden.
Am Ende eines jeden Kapitels bieten wir Ihnen noch verschiedene Variationsmöglichkeiten an, die das gerade aufgebaute Programm vielleicht noch interessanter machen. Programme sind gewissermaßen nie fertig — Sie können immer noch die eine oder andere Ergänzung oder Verbesserung hinzufügen. Und das ist das Schöne an der Programmierung: Sie können Ihre eigenen Ideen einbringen. Obwohl wir in allen Kapiteln »schlüsselfertige« Programme vorstellen, die Sie eingeben und sofort durchführen können, ist das Ende dieser Programme wirklich noch offen. Wir wollen sogar, daß Sie sie verändern und verbessern, daß Sie neue Elemente ein-

fügen und somit Ihre eigenen Programme schreiben. Aber vor allem wollen wir, daß Sie Spaß an der Programmierung und an diesem Buch finden.

1
Wie steht's mit Ihrer Zukunft?

Stellen Sie sich einmal vor, Sie sitzen in einem chinesischen Restaurant und öffnen eins der chinesischen Glückskekse. Haben Sie sich je gefragt, wie gerade diese spezielle Zukunftsvoraussage in Ihr Plätzchen kommt? Ob es wohl jemanden gibt, der wollte, daß Sie genau dieses Plätzchen bekommen? Vermutlich nicht. Irgend jemand muß sich wohl hingesetzt und eine Menge solcher Vorhersagen niedergeschrieben haben, ein anderer hat sie dann alle in die kleinen Kuchen gesteckt. In jedem Fall handelt es sich hier um eine »Zufallsarbeit«. Mit anderen Worten: Sie hätten jede beliebige Voraussage, die geschrieben wurde, erhalten können.

Computer können eine ganze Menge, unter anderem auch die Zukunft vorhersagen, und das ohne große Zaubertricks.

Sie wissen schon, daß irgend jemand sie programmieren — ihnen genau sagen muß, welche Schritte sie durchführen sollen. Wir werden nun ein »Blick-in-die-Zukunft-Programm« schreiben. Wir machen das in der gleichen Schrittfolge, als ob wir wahrsagende Lose für Glückssplätzchen erstellen würden:

1. Eine Menge Vorhersagen festlegen und speichern,
2. per Zufall eine der Vorhersagen herausfischen,
3. die Zukunft auf dem Bildschirm anzeigen.

In diesem Buch werden wir die Programme strukturieren, damit sie leicht zu lesen und zu verändern sind. Wir tun das, indem wir das Programm in ähnliche Schritte unterteilen wie die drei oben genannten. Diese Schritte werden UNTERROUTINEN genannt. Der STEUERUNGSTEIL eines Programms zeigt dem Computer, wo diese Unterroutinen zu finden sind. Der Steuerungsteil liest sich, als ob er ein Inhaltsverzeichnis des Programms wäre.
Wir benutzen auch eine größere Anzahl von REMarks (Bemerkungszeilen), die uns mitteilen, was jeder Programmbaustein im einzelnen für eine Aufgabe hat: welchen Inhalt die von uns benutzten Variablen haben und welche Schritte in-

nerhalb einer Unterroutine durchgeführt werden. Und so sieht der Steuerungsteil für unser Weissagungsprogramm aus:

```
 10 REM ——WIE SIEHT IHRE ZUKUNFT
    AUS?——————
 20 REM F$(4)  : BEREICH FÜR VIER VOR-
    HERSAGEN
 30 REM J      : ZÄHLER
 40 REM N      : ZUFALLSZAHL
 50 REM ——————————————
100 GOSUB 1000:REM——SPEICHERUNG
    DER VORHERSAGEN
200 GOSUB 2000:REM——AUSWAHL EI-
    NER VORHERSAGE
300 GOSUB 3000:REM——AUSGABE DER
    VORHERSAGE
400 END
```

Die erste REMarkzeile enthält den Namen des Programms. Die nächsten drei Zeilen beschreiben alle Variablen, die in diesem Programm gebraucht werden. Die Zeilen 100, 200 und 300 zeigen uns und dem Computer, wo sich die Unterroutinen befinden. GOSUB 1000 weist den Computer an, zur Zeile 1000 zu springen und ab da alle Befehle auszuführen bis zum Kommando RETURN, das den Rücksprung zu der Programmstelle bedeutet, die mit GOSUB verlassen wurde.

Die Vorhersagen in einem Bereich speichern

Nun beginnen wir mit dem Schreiben der Unterroutinen. Die auf Zeile 1000 soll so viele Vorhersagen, wie wir wollen, mittels DATA-Anweisungen speichern. Und so sieht der Beginn der Unterroutine aus:

```
1000 REM-----SPEICHERUNG DER VORHERSAGEN------
1010 DATA "Sie werden sehr reich sein"
1020 DATA "Sie werden überaus glücklich sein"
1030 DATA "Sie werden ein Computer-Programmierer"
1040 DATA "Sie werden auf eine lange Reise gehen"
```

Das sind natürlich nur Beispiele. Sie sollten Ihre eigenen Wunschträume eingeben. Achten Sie

nur darauf, daß jede Zeile mit dem Wort DATA beginnt und die Weissagungstexte in Anführungsstriche gesetzt werden müssen. DATA-Anweisungen können überall im Programm vorkommen. Wenn der Computer einen READ-Befehl findet, sucht er automatisch nach den zu lesenden Daten. Er beginnt die Suche am Programmanfang, und zwar so lange, bis er welche gefunden hat. Beim nächsten READ-Kommando, das er erhält, setzt er die Suche hinter den zuletzt gefundenen Daten fort. Da wir in unserem Beispiel vier Datendefinitionen haben, benötigen wir auch vier READ-Befehle:

```
1050 DIM F$(4)
1060 FOR J = 1 TO 4
1070       READ F$(J)
1080 NEXT J
1090 RETURN
```

Die Zeile 1050 markiert für den Computer, daß es sich bei F$ um mehr als eine Variable handelt — die Zahl in Klammern gibt die Anzahl der Datenelemente oder -zellen an. Die Gruppe der vier Vorhersagen wird Bereich oder Tabelle genannt, das sind mehrere Dateneinheiten, für die der gleiche Variablenname verwendet wird.
Die Schleife in den Zeilen 1060 und 1080 befiehlt dem Computer, vier Dateneinheiten (DATA) zu lesen und in F$ zu speichern (READ in 1070): die erste Zeichenkette (Vorhersage) wird mit F$(1), die zweite mit F$(2) und so weiter bezeichnet.
Die Zeile 1090 bringt den Computer dazu, in den

Steuerungsteil, den man auch als HAUPTPROGRAMM bezeichnet, zurückzukehren und die Arbeit an der Stelle fortzusetzen, wo er aufgehört hat. Er wird nun die Zeile 200 durchführen, die ihn anweist, in die Zeile 2000 zu verzweigen und eine der Weissagungen auszuwählen.

Eine Vorhersage per Zufall auswählen

Nachdem wir nun vier Zukunftsvisionen als F$(1), F$(2), F$(3) und F$(4) gespeichert haben, müssen wir eine nach dem Zufallsprinzip auswählen. Der Commodore 64-Computer ist mit einer eingebauten Funktion ausgerüstet, die jedesmal, wenn Sie sie benutzen, eine andere Zahl auswählen kann. Wenn wir diese Funktion an-

weisen, immer eine der Zahlen 1, 2, 3 oder 4 zu nehmen, dann können wir diese Zahl zur Auswahl einer unserer vier F$ genannten Datenzellen benutzen. Da diese Funktion, die übrigens RANDOM-Funktion heißt, so häufig in diesem Buch vorkommt, wollen wir am besten erst einmal etwas Zeit darauf verwenden, sie kennenzulernen und zu verstehen, wie sie arbeitet.
Bauen wir zunächst einmal ein kleines Testprogramm auf. Wenn Sie schon mit der Eingabe des vorausschauenden Programms begonnen haben, dann sichern Sie bitte die Eingabe mit SAVE und geben das NEW-Kommando ein. Anschließend machen Sie bitte folgende Eingabe:

```
10 FOR X = 1 TO 10
20       PRINT RND(1)
30 NEXT X
```

Das ist das gesamte Programm. Führen Sie es mit RUN mehrmals durch, und schauen Sie sich die ausgegebenen Zahlen genau an. Es sollten alles Zahlen mit Dezimalstellen zwischen 0 und 1 sein. Wir benötigen aber Zahlen, die größer sind als 1, also ändern wir die Zeile 20 so:

```
20 PRINT 4 * RND(1)
```

Spielen Sie auch diese Version mehrmals durch. Nun sollten sich die Zahlen alle zwischen 0 und 3.999... bewegen. Sie werden kaum die 4 erreichen.
Für unsere vier Vorhersagen benötigen wir die Ganzzahlen 1, 2, 3 oder 4 — wir wollen ja gar kei-

ne Dezimalstellen in den Zahlen. BASIC hat eine Ganzzahlenfunktion, die den Dezimalteil abschneidet. Ändern wir also ein zweites Mal die Zeile 20:

```
20 PRINT INT(4 * RND(1))
```

Auch diese Eingabe lassen wir noch einmal durchlaufen. Nun greift der Computer zufällig dezimale Bruchzahlen auf, multipliziert sie mit vier und trennt die Dezimalstellen ab. Dabei läuft aber immer noch etwas falsch. Sie bemerken sicher, daß sich die Zahlen zwischen 0 und 3 bewegen. Wir wollen aber einen Bereich von 1 bis 4. Also wird der Befehl noch mit einer notwendigen Angabe ergänzt:

```
20 PRINT INT(4 * RND(1)) + 1
```

Diese Befehlsversion sollte uns nun eigentlich die Zahlen geben, die wir haben wollen. Mit Hilfe dieser Formel können wir nun die ganze Unterroutine in drei Zeilen schreiben:

```
2000 REM-----EINE ZUFALLSZAHL GE-
     NERIEREN------
2010 N = INT(4 * RND(1)) + 1
2020 RETURN
```

Wir erhalten nun eine Zufallszahl von 1 bis 4 und speichern sie als Variable N. Nun müssen wir nur noch das Orakel mit der Zahl N anzeigen.

Die Ausgabe der Vorhersage

Eigentlich sollten Sie keine Probleme mit dem Verstehen der Funktionsweise dieser Unterroutine haben:

```
3000 REM-----DIE VORHERSAGE AUS-
     GEBEN--------
3010 PRINT CHR$(147)
3020 PRINT "Und hier ist Ihre Zukunft: "
3030 PRINT:PRINT
3040 PRINT F$(N)
3050 RETURN
```

Mit Zeile 3010 löschen Sie den gesamten Bildschirm. Hierbei wird der Cursor auf die erste Stelle der ersten Bildschirmzeile, also links oben, positioniert. Alles, was wir als nächstes ausgeben, wird nun oben auf dem Bildschirm angezeigt. Sie können den Bildschirm auch einfach durch Drük-

ken der Taste CLR-HOME rechts oben auf der Tastatur und gleichzeitiges Drücken der SHIFT-Taste löschen. In einem Programm wird dieser Vorgang simuliert durch den Befehl PRINT »(CLR-HOME)«, wobei das (CLR-HOME) für das Drücken der oben bezeichneten Tasten steht. Wenn Sie im Anhang des Commodore-64-Benutzerhandbuchs in der Liste der Zeichen nachschlagen, werden Sie feststellen, daß das Zeichen 147 identisch ist mit der CLR-HOME-Funktion. In diesem Buch benutzen wir also den Befehl PRINT CHR$(147).

Das Programm zusammenfügen

Alle Unterroutinen sind geschrieben, und das Programm ist bereit zur Durchführung. Auch wenn Sie das ganze Programm an den verschiedenen Stellen in diesem Kapitel nachsehen können, halten wir es für nützlich, es hier noch einmal in seiner Gesamtheit darzustellen:

```
 10 REM ——WIE SIEHT IHRE ZUKUNFT
    AUS?—————
 20 REM F$(4)  : BREICH FÜR VIER VOR-
    HERSAGEN
 30 REM J      : ZÄHLER
 40 REM N      : ZUFALLSZAHL
 50 REM ——————————————
100 GOSUB 1000:REM——SPEICHERUNG
    DER VORHERSAGEN
200 GOSUB 2000:REM——AUSWAHL EI-
    NER VORHERSAGE
```

```
 300 GOSUB 3000:REM——AUSGABE DER
     VORHERSAGE
 400 END
1000 REM————SPEICHERUNG DER VOR-
     HERSAGEN————————
1010 DATA "Sie werden sehr reich sein"
1020 DATA "Sie werden überaus glücklich
     sein"
1030 DATA "Sie werden ein Computer-Pro-
     grammierer"
1040 DATA "Sie werden auf eine lange
     Reise gehen"
1050 DIM F$(4)
1060 FOR J = 1 TO 4
1070      READ F$(J)
1080 NEXT J
1090 RETURN
2000 REM————EINE ZUFALLSZAHL
     AUSWÄHLEN————————
2010 N = INT(4 * RND(1)) + 1
2020 RETURN
3000 REM————DIE VORHERSAGE AUS-
     GEBEN————————
3010 PRINT CHR$(147)
3020 PRINT "Und hier ist Ihre Zukunft"
3030 PRINT:PRINT
3040 PRINT F$(N)
3050 RETURN
```

Variationen

Es gibt viele Dinge, die man nach der Erarbeitung eines Programms noch zu dessen Verschönerung und Abrundung tun kann. Am Ende eines jeden Kapitels wollen wir Ihnen einige Variationen, die Sie ausprobieren können, vorstellen und Tips für deren Realisierung geben. Hier zunächst einige Variationen für unser Weissagungsprogramm:

1. Vier Voraussagen mögen vielen von Ihnen nicht genug erscheinen. Sie können eigentlich so viele, wie Sie wollen, erfinden und speichern. Natürlich müssen Sie dann einige Teile im Programm anpassen beziehungsweise hinzufügen:

a) Ergänzen Sie die DATA-Zeilen in der ersten Unterroutine.

b) Ändern Sie die Zahl in den Zeilen 1050 und 1060.

c) Ändern Sie auch die 4 in der Random-Funktionszeile 2010.

2. Geben Sie zwei oder mehr Vorhersagen aus: Nehmen Sie an, zwei Menschen wollen ihr Schicksal zur gleichen Zeit erfahren. Sie müssen also zwei Prophezeiungen gleichzeitig auswählen — nennen wir Sie N1 und N2:

```
2010 N1 = INT(4 * RND(1)) + 1
2015 N2 = INT(4 * RND(1)) + 1
```

In der letzten Unterroutine würden Sie in diesem Fall sowohl den Inhalt von F$(N1) als auch von F$(2) auf dem Bildschirm anzeigen. Sie können natürlich auch durch freizügige Anwendung des PRINT-Befehls den Bildschirm effektvoller gestalten (zum Beispiel durch einen Rahmen der aus Sternchen besteht). Das sieht dann ungefähr so aus:

Ihr persönliches Horoskop....

Hier sind der Phantasie keine Grenzen — außer denen des Bildschirms — gesetzt.
Auf diese Weise können Sie nun jede beliebige Anzahl an Weissagungen speichern und ausgeben.

2
Sprechen Sie eine Geheimsprache?

Haben Sie jemals einen Geheimcode benutzt, um einem Freund etwas mitzuteilen? Bei allen Codes müssen ganz bestimmte Regeln befolgt werden — wenn Sie diese kennen, können Sie jeden Code problemlos dechiffrieren. Computer können sehr leicht verschlüsseln und entschlüsseln, wenn Sie die Regeln programmieren. In diesem Kapitel werden wir ein Spielprogramm für zwei Personen schreiben. Die erste gibt eine Nachricht ein. Der Commodore 64 nimmt die Nachricht auf, setzt sie in einen Code um und zeigt sie verschlüsselt dem zweiten Spieler. Dieser muß dann versuchen, die Nachricht zu entziffern.
Es gibt eine große Anzahl unterschiedliche Codierungen, die wir benutzen können. In diesem Fall werden wir den Umkehrungscode verwenden: der Computer erhält die Nachricht und gibt sie rückwärts wieder aus. Der zweite Spieler muß also beim Entziffern von rechts nach links lesen:

?NESEL SAD EIS NENNÖK

Wenn Sie es können, dann sind Sie in der glücklichen Lage, alles in diesem Programm entschlüsseln zu können:

```
 10 REM--------GEHEIMSPRACHE
 20 REM M$  : DIE NACHRICHT
 30 REM L$  : EIN BUCHSTABE IN DER
    NACHRICHT
 40 REM C$  : DER SCHLÜSSEL
 50 REM A$  : DIE ANTWORT
 60 REM  J  : EIN ZÄHLER
 70 REM----------
100 GOSUB 1000:REM-----DIE NACH-
    RICHT EMPFANGEN
200 GOSUB 2000:REM-----DIE NACH-
    RICHT VERSCHLÜSSELN
300 GOSUB 3000:REM-----QUIZ
400 END
```

Das Hauptprogramm (Steuerungsteil) zeigt uns fünf benötigte Variablennamen und drei Abteilungen (SECTIONS) des Programms in Form von Unterroutinen.

Die Nachricht empfangen

In der Unterroutine auf Zeile 1000 bitten wir darum, daß der zweite Spieler während der Eingabe der Nachricht durch den ersten Spieler nicht zusieht. Die Nachricht wird als Variable M$ gespeichert.

```
1000 REM————————DIE NACHRICHT
     EMPFANGEN
1010 PRINT CHR$(147)
1020 PRINT "Sagen Sie Ihrem Mitspieler,
     er soll"
1030 PRINT "wegsehen, wenn Sie die
     Nachricht"
1035 PRINT "eingeben"
1040 PRINT:PRINT
1050 PRINT "Geben Sie nun Ihre Nachricht
     ein"
1055 PRINT "Dann drücken Sie die RE-
     TURN-Taste"
1060 INPUT M$
1070 RETURN
```

Die Nachricht umkehren

Zum Programm gehört eine Zeichenkette, M$ genannt — in ihr kann jedes Zeichen, jeder beliebige Buchstabe oder jede willkürlich eingegebene Ziffer, jedes Sonderzeichen wie beispielweise das $-Zeichen, Ausrufezeichen, Komma und so weiter enthalten sein. Wir müssen nun die Reihenfolge all dieser Zeichen umkehren und die geänderte Zeichenkette in einer neuen Variablen, die wir C$ nennen werden, abspeichern. Diese Prozedur läuft folgendermaßen ab:

1. Nimm das letzte Zeichen aus M$ und speichere es als erstes Zeichen in C$.
2. Nimm das vorletzte Zeichen aus M$ als zweites Zeichen in C$.
3. Nimm das vorvorletzte Zeichen aus M$ als drittes Zeichen in C$.
4. Führe diese Arbeit aus bis zum letzten Zeichen von M$, das das erste Zeichen in C$ werden soll.

Nehmen wir an, daß in M$ das Wort »HALLO« als Nachricht enthalten ist. Der letzte Buchstabe in M$ ist also »O«, und dieser wird der erste Buchstabe in C$. Dann wird das letzte »L« in M$ der zweite Buchstabe in C$. Wenn die Codierung erfolgreich abgeschlossen ist, enthält C$ das Wort

"OLLAH"

Der Commodore 64 besitzt eine integrierte Funktion, die jedes gewünschte Zeichen aus einer Zeichenkette auswählen kann. Wir probieren diese spezielle Funktion einmal aus, um ihre Arbeitsweise zu begreifen. Zuerst geben wir ein:

```
M$ = "PFERD"
```

Dann erfolgt die Anweisung:

```
PRINT MID$(M$,4,1)
```

(Vergessen Sie dabei natürlich bitte nicht, die RETURN-Taste zu drücken.)
Der Commodore 64 sollte mit dem Buchstaben »R« antworten, da R der vierte Buchstabe in dem in M$ gespeicherten Wort ist. Nun versuchen Sie es mit dem folgenden Kommando:

```
PRINT MID$(M$,3,2)
```

Der Commodore wird Ihnen daraufhin zwei Buchstaben aus M$ zeigen, beginnend mit dem dritten Buchstaben des Wortes. Und zwar wer-

den die Buchstaben ER angezeigt, wenn die RETURN-Taste gedrückt wurde. Bei der Benutzung der MID$-Funktion werden innerhalb der Klammern drei wesentliche Informationen festgelegt:
Der Name der Zeichenkette.
Die Zeichenposition, mit der begonnen werden soll.
Wie viele Zeichen verarbeitet werden sollen.
Anstelle der direkten Zahlenangaben können auch Variablen genommen werden. Geben Sie beispielsweise dem Commodore ein:

```
J = 5
```

und anschließend

```
PRINT MID$(M$,J,1)
```

Mit anderen Worten: Sie wollen 1 Zeichen aus M$ sehen, beginnend bei dem Zeichen mit der Nummer J — oder 5. In unserem Beispiel ist das der Buchstabe »D«. Was passiert, wenn J den Wert 4 enthält? Dann entspricht MID$(M$,J,1) dem Buchstaben »R«.Durch die Veränderung des Wertes von J können wir also alle Zeichen in M$ betrachten.
Aber woher kennen wir die Anzahl der Zeichen in M$?
Dies teilt uns eine andere Commodore-Funktion mit. Versuchen Sie einmal:

```
PRINT LEN(M$)
```

LEN steht für die Länge irgendeiner Zeichenkette, deren Namen man innerhalb der Klammern angibt. Verändern wir einmal M$ wie folgt:

```
M$ = "HALLO, SIE DA!"
PRINT LEN(M$)
```

Wenn Sie diese Befehle eingeben, sehen Sie nach dem zweiten Drücken der RETURN-Taste, daß die Länge von M$ dem Wert 13 entspricht. Leerstellen und Punktuation zählen natürlich mit, da sie auch Bestandteil der Zeichenkette sind. Bevor wir beginnen, unsere Unterroutine zu schreiben, probieren wir diese beiden neuen Funktionen in einem kurzen Testprogramm aus, so wie wir es auch in Kapitel 1 mit der Random-Funktion getan haben. Die beste Art und Weise, eine neue Computerlogik zu begreifen, ist, sie im einfachsten Programm, das man sich vorstellen kann, anzuwenden.
Wenn Sie schon einen Teil unseres Programms eingegeben haben, sichern Sie es bitte und geben die folgenden Testanweisungen ein:

```
10 M$ = "Hilfe"
20 FOR J = 1 TO 5
30      PRINT MID$(M$,J,1)
40 NEXT J
```

Führen Sie dieses Programm mit RUN durch — die Ausgabe sollte aus den fünf Buchstaben des Wortes HILFE bestehen, die einzeln pro Bildschirmzeile angezeigt werden. Bei der ersten

Schleifendurchführung hatte J den Inhalt 1, also wurde der erste Buchstabe ausgegeben. Beim zweiten Mal wurde der zweite Buchstabe gezeigt und so weiter. Nun ändern Sie bitte:

```
10 FOR J = 5 TO 1 STEP —1
```

Führen Sie das Programm erneut durch. Dieses Mal werden die Buchstaben in umgekehrter Reihenfolge angezeigt. Bei der ersten Schleife trägt J den Wert 5, also wird der fünfte Buchstabe sichtbar. Da J bei jeder Schleife um 1 vermindert wird, entspricht die Variable beim zweiten Mal dem Wert 4, und der vierte Buchstabe wird ausgegeben. Jetzt modifizieren wir das Testprogramm noch einmal:

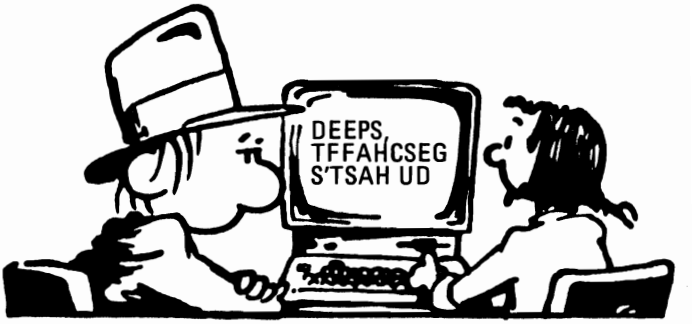
```
10 FOR J = LEN(M$) TO 1 STEP —1
```

Das Programm sollte nun völlig gleich ablaufen. Der Computer hat nur den Wert 5 durch LEN(M$) ersetzt, da insgesamt fünf Zeichen in M$ enthalten sind. Weil das Gerät dazu in der Lage ist, müssen wir also die Länge einer als Variable M$ gespeicherten Zeichenkette nicht kennen. Die Länge kann sogar bei jeder Durchführung ohne weiteres verändert werden. Nun sind wir bereit, die Unterroutine zu schreiben:

```
2000 REM————DIE NACHRICHT UM-
     KEHREN————————————
2010 FOR J = LEN(M$) TO 1 STEP —1
2020       L$ = MID$(M$,J,1)
```

```
2030      C$ = C$ + L$
2040 NEXT J
2050 RETURN
```

Die Routine hat nur wenige Schritte. Die FOR/NEXT-Schleife weist den Computer an, beginnend mit dem letzten Zeichen abwärts zu zählen, bis das erste Zeichen von M$ erreicht ist. Enthält M$ das Wort »KATZE«, wird die Schleife fünfmal durchgeführt. Beim ersten Mal entspricht J der Zahl 5, beim zweiten Mal der Zahl 4, beim dritten Mal der Zahl 3, beim vierten Mal der Zahl 2 und beim fünften Mal der Zahl 1.
L$ steht für ein Zeichen aus M$. Bleiben wir bei dem obengenannten Wort, so enthält L$ bei der ersten Schleifendurchführung den Buchstaben »E«, bei der zweiten »Z«, bei der dritten »T«, bei der vierten »A« und bei der fünften »K«.



Die Zeile 2030 nimmt jeweils den Wert aus L$ und fügt ihn an das Ende von C$ an. Beim Start

befindet sich in C$ noch gar nichts, also ergibt hier L$ selbst die Variable C$. Am Ende der ersten Schleife enthält C$ den Buchstaben »E«. Am Schluß des zweiten Durchgangs enthält L$ den Buchstaben »Z«, der dem Inhalt von C$ (»E«) zugeordnet wird. Danach beinhaltet C$ die Zeichenkette »EZ«. Beim dritten Durchgang wird aus L$ das »T« beigefügt, C$ besteht danach aus »EZT«. Beim vierten Durchlauf erfolgt das gleiche mit dem Buchstaben »A«, jetzt entspricht C$ dem Wert »EZTA«. Zu guter Letzt kommt das »K« hinzu, die Variable C$ enthält nun die umgedrehte Nachricht: »EZTAK«.

Quiz mit dem zweiten Spieler

Nun haben wir die Originalnachricht in M$ und die verschlüsselte Nachricht in C$ gespeichert —

der Rest ist einfach! Wir löschen den Bildschirm und bitten den zweiten Spieler, sich den Inhalt von C$, den wir auf dem Bildschirm ausgeben, anzusehen. Wenn er kann, soll er das richtige Wort dann eingeben. Wir nennen die Antwort des Spielers A$. Wenn A$ dem Inhalt von M$ gleicht, dann hat der Spieler die Verschlüsselung erraten.

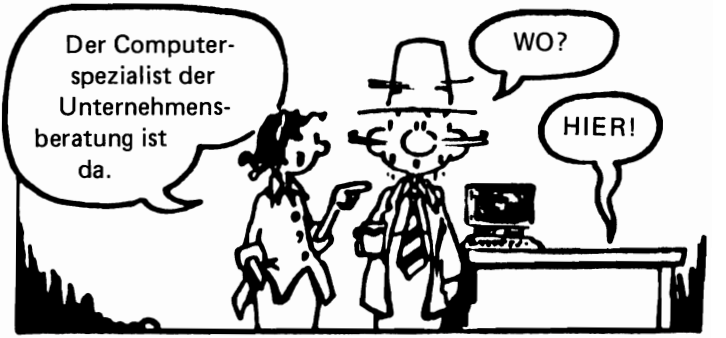
```
3000 REM ————————QUIZ————————
     ———
3010 PRINT CHR$(147)
3020 PRINT "Bitten Sie Ihren Mitspieler,"
3030 PRINT "diese Nachricht zu entziffern"
3040 PRINT:PRINT:PRINT C$
3050 PRINT:PRINT "Geben Sie die Antwort
     ein:"
3050 PRINT:INPUT A$
3070 IF A$ = M$ THEN PRINT "RICHTIG! Das
     war's!"
3080 IF A$ <> M$ THEN PRINT "Schade, die
     Nachricht war: ";M$
3090 RETURN
```

Das ganze Programm


```
 10 REM--------GEHEIMSPRACHE----------
 20 REM M$   : DIE NACHRICHT
 30 REM L$   : EIN BUCHSTABE IN DER
    NACHRICHT
 40 REM C$   : DER SCHLÜSSEL
 50 REM A$   : DIE ANTWORT
 60 REM  J   : EIN ZÄHLER
 70 REM--------------------------
100 GOSUB 1000:REM-----DIE NACH-
    RICHT EMPFANGEN
200 GOSUB 2000:REM-----DIE NACH-
    RICHT VERSCHLÜSSELN
300 GOSUB 3000:REM-----QUIZ
400 END
1000 REM--------DIE NACHRICHT EMP-
    FANGEN----------
1010 PRINT CHR$(147)
1020 PRINT "Sagen Sie Ihrem Mitspieler,
    er soll"
```

```
1030 PRINT "wegsehen, wenn Sie die
     Nachricht"
1035 PRINT "eingeben"
1040 PRINT:PRINT
1050 PRINT "Geben Sie nun die Nachricht
     ein"
1055 PRINT "Dann drücken Sie die RE-
     TURN-Taste"
1060 INPUT M$
1070 RETURN
2000 REM————DIE NACHRICHT UM-
     KEHREN————————————
2010 FOR J = LEN(M$) TO 1 STEP —1
2020       L$ = MID$(M$,J,1)
2030       C$ = C$ + L$
2040 NEXT J
2050 RETURN
3000 REM————————QUIZ————————
     ————
3010 PRINT CHR$(147)
3020 PRINT "Bitten Sie Ihren Mitspieler,"
3030 PRINT "diese Nachricht zu entzif-
     fern"
3040 PRINT:PRINT:PRINT C$
3050 PRINT:PRINT "Geben Sie die Antwort
     ein:"
3060 PRINT:INPUT A$
3070 IF A$ = M$ THEN PRINT "RICHTIG!
     Das war's!"
3080 IF A$ <> M$ THEN PRINT "Schade,
     die Nachricht war: ";M$
3090 RETURN
```

Variationen



Dieses Spiel läßt sich sehr gut abwandeln. Zum einen könnten wir die Codierung der Nachricht verändern. Zum anderen könnten wir eine Zusammenstellung schon formulierter Nachrichten speichern, aus denen der Computer eine auswählt. Das sind zwei Dinge, die wir im nächsten Kapitel besprechen wollen.
Es gibt aber auch einige kleinere Änderungen, die Sie selbständig durchführen können. Beispielsweise sind Sie in der Lage, die Ansprache durch den Computer persönlicher zu gestalten, indem Sie sich mit dem Vornamen anreden lassen.
Wir wollen uns hier jedoch mit einer anderen Variation beschäftigen. Unser Spielkonzept gibt dem zweiten Mitspieler nur eine Chance, die Meldung zu dechiffrieren. Meinen Sie nicht auch, es wäre besser, ihm mehr als einen Ver-

such zuzubilligen? Wenn ja, zeigen wir Ihnen, wie's gemacht wird. Der Beginn der QUIZ-Unterroutine (Zeilen 3000 bis 3050) bleibt bestehen: wir geben C$ auf dem Bildschirm aus und fragen den Mitspieler nach der richtigen Antwort.
Der Rest dieser Unterroutine wird jedoch dann in der Ablauflogik verändert. Anstatt mit einer Eingabe zufrieden zu sein, bauen wir eine Schleife auf, die dreimal durchgeführt wird und jedesmal eine Aussage zuläßt. Bei jeder Durchführung müssen wir prüfen, ob die Antwort A$ der Variablen M$ gleicht. Falls das zutrifft, das heißt, der Spieler hat die Nuß geknackt,muß der Durchlauf beendet werden.

```
3060 FOR J = 1 TO 3
3070      PRINT "VERSUCH NR: ";J
3080      INPUT A$
3090      IF A$ = M$ THEN PRINT "RICH-
          TIG!":GOTO 3130
3100      IF A$ <> M$ THEN PRINT "SCHA-
          DE! ";
3110 NEXT J
3120 PRINT "DIE NACHRICHT WAR: ";M$
3130 RETURN
```

Achten Sie auf das Semikolon (;) in Zeile 3100. Am Ende einer PRINT-Anweisung bewirkt dieses Zeichen, daß der nächste Text in der gleichen Zeile auf dem Bildschirm erscheint. Wenn der Spieler also bei allen Versuchen scheitert, erscheint zweimal die Nachricht:

SCHADE! VERSUCH NR: J
(wobei statt J jeweils die aktuelle Zahl erscheint)

Und beim dritten Mal der Text:

SCHADE! DIE NACHRICHT WAR: M$
(hier folgt der Text aus M$)

3
Die Kunst der Nachrichten-verschlüsselung

Das im letzten Kapitel von uns vorgestellte Spiel funktioniert nur, wenn ein Spieler den Computer mit einer Nachricht füttert. Jedesmal, wenn das Spiel beginnt, muß also eine neue Nachricht eingegeben werden. Das hemmt natürlich den Spielverlauf und kann auf die Teilnehmer ermüdend wirken. Ein weiteres Ärgernis wäre, daß der zweite Spieler einem über die Schulter gucken und die Nachricht vor der Verschlüsselung sehen könnte, was ihm natürlich nachher die Entzifferung erleichtern würde.

Eine bessere Idee ist wohl, eine größere Anzahl von Texten vorab im Programm zu speichern und den Computer einen dieser Texte per Zufall auswählen zu lassen. Kommt Ihnen diese Idee nicht bekannt vor?
Sollte sie eigentlich.
In Kapitel 1 haben wir gelernt, wie man Texte, Nachrichten oder Zeichenketten speichert und sie später nach Zufall einzeln ausgibt, und im zweiten Kapitel lernten wir, wie man diese Texte verschlüsselt.
In diesem Kapitel werden wir zwei Dinge tun: Zuerst die ersten beiden Programme zu einem neuen Programm verbinden, das eine Meldung im Random-Verfahren auswählt und dann chiffriert. Anschließend werden wir das Thema der Verschlüsselung durch Erfinden eines weiteren Codes oder die Benutzung zweier Codes im gleichen Programm noch etwas vertiefen.
Die Steuerung, das Hauptprogramm für unser neues Programm sieht jetzt so aus:

```
10 REM————VERSCHLÜSSELTE NACH-
   RICHTEN————————
20 REM M$(4) :BEREICH FÜR VIER NACH-
   RICHTEN
30 REM J       :ZÄHLER
40 REM N       :ZUFALLSZAHL
50 REM L$      :EIN BUCHSTABE IN DER
   NACHRICHT
60 REM C$      :DIE VERSCHLÜSSELTE
   NACHRICHT
70 REM M$      :DIE ORIGINALNACHRICHT
```

```
 80 REM----------------------
    ---
100 GOSUB 1000:REM-----DIE NACH-
    RICHTEN SPEICHERN
200 GOSUB 2000:REM-----EINE ZUFÄL-
    LIG AUFGREIFEN
300 GOSUB 3000:REM-----SIE VER-
    SCHLÜSSELN
400 GOSUB 4000:REM-----QUIZ
500 END
```

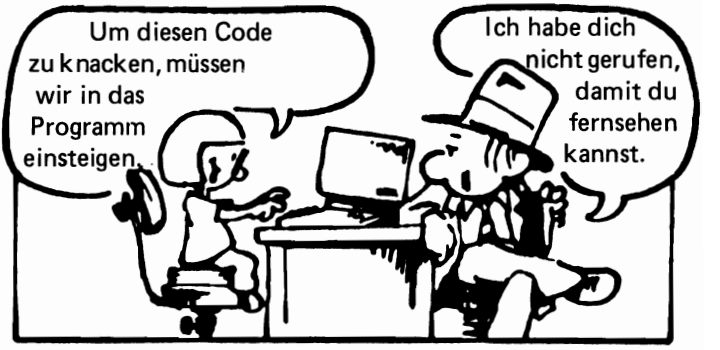
Eigentlich haben Sie sich schon mit jeder dieser Unterroutinen ein wenig vertraut gemacht. Die ersten beiden stammen aus dem Programm in Kapitel 1, und die letzten beiden sollten uns aus Kapitel 2 bekannt sein. Es müssen nur ganz geringfügige Änderungen vorgenommen werden. Schauen wir uns mal die Unterroutine 1000 in Kapitel 1 an, die wir »Speicherung der Vorhersagen« genannt haben. Hier speichern wir nun Nachrichten statt Weissagungen, also nennen wir den Bereich oder die Tabelle M$(4) statt F$(4). Führen Sie bitte diese Änderung durch, und schreiben Sie anstelle der Vorhersagen die von Ihnen gewünschten Nachrichten in die DATA-Zeilen. Der Rest dieser Unterroutine bleibt unverändert. Keine Änderungen sind in der zweiten Unterroutine notwendig. Auch hier greifen wir eine Zufallszahl von 1 bis 4 auf.
Die dritte Unterroutine in unserem neuen Programm, »SIE VERSCHLÜSSELN«, das heißt, eine der Nachrichten chiffrieren, ist die gleiche, die wir in Kapitel 2 mit »DIE NACHRICHT UMKEH-

REN« bezeichnet haben. Dort war es die zweite Unterroutine, und hier ist es die dritte. Wir müssen also die Zeilennummern aus den 2000ern in die 3000er verlegen. Darüber hinaus hatten wir in Kapitel 2 nur eine Nachricht, deshalb nannten wir sie nur M$. Aber jetzt kann unsere Nachricht jede der in der Tabelle M$(4) gespeicherten Meldungen sein. Da wir gerade eine Zufallszahl N gezogen haben, ist die Nachricht, die verschlüsselt werden soll, natürlich in der Variablen M$(N) zu finden. Eine winzige Änderung sorgt für die Anpassung dieser Unterroutine. Fügen Sie bitte am Beginn der Routine die folgende Zeile ein:

```
3005 M$ = M$(N)
```

Diese Anweisung bedeutet, daß welche Nachricht auch immer aufgegriffen wurde, im weiteren Verlauf des Programms nun M$ genannt wird. So kann der Rest der Routine unverändert bleiben.
Die letzte Unterroutine, QUIZ, ist ebenfalls identisch mit der letzten Unterroutine in Kapitel 2. Vergessen Sie aber nicht, die Zeilennumerierung von 3000 auf 4000 zu erhöhen, da wir ja hier vier Unterroutinen haben.

Eine andere Codierung



Mit einer Handvoll Änderungen haben wir zwei einfachere Programme zu einem komplizierteren zusammengefügt. Das neue Programm wird eine Nachricht per Zufallsgenerator auswählen, sie verschlüsseln und einen Mitspieler bitten, den Originaltext wiederherzustellen.
Wir können aber noch einiges mehr tun. Momentan benutzt unser Programm immer dieselbe Codierung. Wenn jemand mehrmals mit diesem Programm gespielt hat, ist er höchstwahrscheinlich in der Lage, den Code zu knakken, und der Spaß an der Sache ist vorbei. Also schreiben wir eine Unterroutine, die die Umstellung in verschiedener Weise ermöglichen soll. Dann können wir beide Codes im Programm benutzen, wobei der Computer auch hier Zufall spielt und den Code auswählt.
Es gibt vermutlich Tausende verschiedener Codierungen, jedenfalls mehr, als man sich vorstel-

len kann. Unser erster Code kehrte einfach die Nachricht um. Ein anderer, oft benutzter Schlüssel besteht aus der Einfügung von Buchstaben zwischen die Buchstaben der Meldungen. Zum Beispiel könnte der Code für »HALLO« so aussehen: HXAXLXLXOX. Eine einfache Änderung der Unterroutine sorgt für eine solche Chiffrierung:

```
3000 REM———————SIE VERSCHLÜS-
     SELN———————————
3005 M$ = M$(N)
3010 FOR J = 1 TO LEN(M$)
3020     L$ = MID$(M$,J,1)
3030     C$ = C$ + L$ + "X"
3040 NEXT J
3050 RETURN
```

Eigentlich müssen zwei Änderungen gemacht werden. In Zeile 3010 zählen wir nun vorwärts anstatt rückwärts, weil wir die Zeichenreihenfolge diesmal nicht umkehren wollen. In Zeile 3030 fügen wir erst einen Buchstaben der eigentlichen Nachricht und dann zusätzlich ein X bei jedem Schleifendurchlauf im Bereich C$ zusammen.
Natürlich können Sie jeden Buchstaben beziehungsweise jedes Zeichen anstelle des X verwenden. Aber ist dieser Code nicht immer noch zu leicht zu entziffern? Was halten Sie von der Idee,wenn wir mit dem Zufallsgenerator (RND) einen beliebigen Buchstaben für X in Zeile 3030 auswählten? Dann könnte HALLO nach der Codierung ungefähr so aussehen: HRAULBLIOW. Das ist dann wirklich schwer zu entziffern!

Um das zu programmieren, müssen Sie zwei weitere BASIC-Funktionen kennenlernen. Jedes vom Computer benutzte Zeichen hat eine sogenannte Codezahl. Der Code wird ASCII-Code genannt und von den meisten Computern benutzt. Die ASCII-Zahl für den Buchstaben A ist 65, für B 66, für C 67 und so weiter bis Z, das die Codezahl 90 hat. Das können Sie leicht selbst überprüfen. Geben Sie beispielsweise:

```
PRINT ASC ("A")
      oder
PRINT ASC ("T")
```

oder zwischen den Anführungsstrichen jeden gewünschten Buchstaben ein.
Die ASC-Funktion ändert den entsprechenden Buchstaben in die ihm zugeordnete Codezahl. Wir aber wollen genau das Gegenteil tun — wenn wir die RND(1)-Funktion verwenden, erhalten wir eine Zufallszahl. Wir müssen diese Zahl in einen Buchstaben verwandeln. Die BASIC-Funktion CHR$ hilft uns dabei. Versuchen Sie es mal mit:

```
PRINT CHR$(65)
```

Der Computer wird nun den Buchstaben A wiedergeben, da A der Buchstabe mit der Nummer 65 im ASCII-Verzeichnis ist. Die CHR$-Funktion ist also das Gegenstück zur ASC-Funktion.
Nun werden wir unsere »SIE VERSCHLÜSSELN«-Unterroutine dahingehend ändern, daß wir bei

jeder Schleifendurchführung eine Random-Ganzzahl von 1 bis 26 (das Alphabet hat 26 Buchstaben) erhalten. Diese Ganzzahl benutzen wir dann, um mit Hilfe der CHR$-Funktion einen Buchstaben zu bestimmen. Auf diese Weise erhalten wir »Zufallsbuchstaben«, die wir dem Code anstelle eines X jedesmal einfügen können:

```
3000 REM———————SIE VERSCHLÜS-
     SELN———————————
3005 M$ = M$(N)
3010 FOR J = 1 TO LEN(M$)
3020     L$ = MID$(M$,J,1)
3030     R = INT(26 * RND(1)) + 65
3040     C$ = C$ + L$ + CHR$(R)
3050 NEXT J
3060 RETURN
```

Die Zeile 3030 weist den Computer an, die Zufallszahl um 65 zu erhöhen. Wir wollen ja eine Zahl zwischen 65 und 90, da dieser Zahlenbereich die Buchstaben von A bis Z im ASCII-Code darstellt. Mit irgendeiner Zahl zwischen 1 und 64 sind wir nicht zufrieden.
Außerdem werden in Zeile 3040 drei Zeichenketten zusammengefügt, und zwar jedesmal, wenn die Schleife durchgeführt wird: das, was schon codiert ist (C$), der nächste Buchstabe aus der Meldung (L$) und der zufällig ermittelte Buchstabe (CHR$(R)).

Der eine Code oder der andere?

Nun kennen wir zwei Wege der Nachrichtenverschlüsselung, wir können die eine oder andere Form für das Programm wählen. Noch besser ist es jedoch, wenn wir BEIDE im gleichen Programm benutzen können und dem Computer jedesmal die Wahl der Codierungsart überlassen. Damit das funktioniert, müssen wir beide Unterroutinen in das Programm einbauen. Da sie nicht auf derselben Zeilennummer starten können, legen wir fest, daß die Umkehrungscode-Routine auf Zeile 3000 beginnt und die Einführungscode-Routine auf 3500. Und so wird's gemacht:

```
3000 REM———UMKEHRUNGSCODE———
     ——————————
```

und

```
3500 REM——————EINFÜGUNGSCODE——
     ——————————
```

Sie kennen die Details jeder Routine bereits und können sie also mit Leben füllen. Das gesamte Programm werden wir Ihnen später in diesem Kapitel vorstellen.
Doch zunächst muß der Computer angewiesen werden, zwischen den beiden Routinen auszuwählen. Dazu müssen wir den Steuerungsteil des Programms verändern. Das Hauptprogramm nach den REMark-Anweisungen sieht dann folgendermaßen aus:

```
100 GOSUB 1000:REM—
    ——NACHRICHTENSPEICHERUNG——
200 GOSUB 2000:REM———EINE ZUFÄL-
    LIG AUFGREIFEN
250 S = INT(2 * RND(1)) + 1
300 ON S GOSUB 3000,3500:REM—
    —VERSCHLÜSSELUNG
400 GOSUB 4000:REM————QUIZ
500 END
```

Wie Sie sehen können, benutzen wir wieder einmal die Random-Funktion. Dieses Mal, um entweder eine 1 oder eine 2 aufzugreifen und sie als »S« zu speichern. Wenn S den Wert 1 annimmt, soll das Programm die Unterroutine 3000 durchführen, wenn S eine 2 beinhaltet, sollte die Unterroutine 3500 abgearbeitet werden.
Die ON————————GOSUB-Anweisung in Zeile 300 bewirkt genau das. Sie bedeutet bei S gleich 1 die Verzweigung auf die erste angeführte Unterroutine und bei S gleich 2 die zur zweiten. Falls es sich um drei verschiedene Codierungs-

Unterroutinen handeln würde, und die letzte würde bei Zeile 3800 beginnen, könnten wir für S die Zufallszahlen 1, 2 und 3 bestimmen und die Zeile 300 wie folgt ändern:

```
300 ON S GOSUB 3000,3500,3800
```

Das ganze Programm

Der folgende Abschnitt zeigt die letzte Version unseres Programms. Es speichert ein paar Meldungen in einer Tabelle, greift eine von ihnen per Zufallsgenerator für die Codierung heraus, codiert sie unter Benutzung einer der zwei Methoden und bittet den Mitspieler um einen Dechiffrierungsversuch.

```
 10 REM-----VERSCHLÜSSELTE
    NACHRICHTEN----------
 20 REM M$(4) : BEREICH FÜR VIER
    NACHRICHTEN
 30 REM J      : ZÄHLER
 40 REM N      : ZUFALLSZAHL NACH-
    RICHTEN
 50 REM S      : ZUFALLSZAHL UNTER-
    ROUTINE
 60 REM R      : ZUFALLSZAHL BUCH-
    STABE
 70 REM L$     : EIN BUCHSTABE IN DER
    NACHRICHT
 80 REM C$     : DIE VERSCHLÜSSELTE
    NACHRICHT
 85 REM M$     : DIE ORIGINALNACH-
    RICHT
 90 REM----------------
100 GOSUB 1000:REM-
    -----NACHRICHTEN SPEICHERN-
200 GOSUB 2000:REM------EINE ZU-
    FÄLLIG AUFGREIFEN
250 S = INT(2 * RND(1)) + 1
300 ON S GOSUB 3000,3500:REM--
    VERSCHLÜSSELUNG
400 GOSUB 4000:REM-----QUIZ---
    --------
500 END
1000 REM------NACHRICHTEN SPEI-
    CHERN
1010 DATA "Einen schönen Tag wünsche
     ich"
1020 DATA "Wir sehen uns morgen"
```

```
1030 DATA "Wir treffen uns an der alten
     Eiche"
1040 DATA "Dies ist eine geheime Nach-
     richt"
1050 DIM M$(4)
1060 FOR J = 1 TO 4
1070     READ M$(J)
1080 NEXT J
1090 RETURN
2000 REM------NACHRICHTENZU-
     FALLSZAHL----------
2010 N = INT(4 * RND(1)) + 1
2020 RETURN
```

Dieses Programm ist schon ganz schön lang, aber da wir es Abschnitt für Abschnitt zusammengebaut haben, dürfte es für Sie nicht schwierig gewesen sein, den einzelnen Entwicklungsschritten zu folgen. Beachten Sie bitte, daß wir die Random-Funktion RND jetzt insgesamt dreimal nutzen: Wir wählen die Nachricht durch Random aus, wir ermitteln die Codierungsunterroutine durch Random und wir fügen die mit Random selektierten Buchstaben in die Nachricht ein.

Variationen

Nachdem Sie alle für dieses Programm notwendigen Schritte kennen, möchten wir Ihnen einige Erweiterungen vorschlagen, mit deren Hilfe Sie selbständig das Programm ausbauen können:
1. Verändern Sie die Anzahl der in der ersten Unterroutine gespeicherten Nachrichten und lassen Sie sie vom Programm auswählen.
2. Wählen Sie mehr als eine Nachricht in einem Durchlauf zur Entzifferung aus.
3. Geben Sie dem Spieler mehr als eine Chance, die Nachricht(en) zu decodieren.

Zwei weitere Varianten, die wir bisher noch nicht erwähnt haben, die aber leicht in das Programm einzubringen sind, wollen wir Ihnen nicht vorenthalten:
1. Anstatt sich darauf zu beschränken, Buchstaben zwischen die Zeichen der Nachricht ein-

zufügen, können Sie jedes Tastatursymbol im Zufallsgenerator zulassen — eingeschlossen den Stern und die Anführungsstriche. HINWEIS: überprüfen Sie die bei CHR$(33) startenden Zeichen bis zu CHR$(65). Das ist wichtig, weil wir in Zeile 3030 die Zahl 65 zur Zufallszahl addiert haben, da die Codezahl 65 in ASCII auf den Buchstaben A verweist.

2. Wenn Sie das Programm durchführen, werden Sie feststellen, daß ab dem zweiten Wort einer Nachricht schon vor dem ersten Buchstaben ein Zufallszeichen generiert wird. Dieses Phänomen ist leicht erklärbar. Eine Leerstelle zwischen zwei Worten wird vom Computer ebenso als Zeichen betrachtet wie ein normales sichtbares Zeichen — unsere Unterroutine wirft also hinter jeder Leerstelle auch ein Zufallszeichen aus. Wenn Sie diesen Nebeneffekt nicht mögen, können Sie den Computer anweisen, Leerstellen zu ignorieren, sobald sie auftreten. Sie brauchen nur die EINFÜGUNGSCODE-Unterroutine um folgendes Kommando zu ergänzen:

```
3525 IF L$ = CHR$(32) THEN C$ = C$ +
     L$:GOTO 3550
```

CHR $ (32) ist die ASCII-Codezahl für eine Leerstelle. Die oben angegebene Zeile weist den Computer an, falls in der Originalnachricht eine Leerstelle auftritt, nur diese der verschlüsselten Nachricht beizufügen. Das Zufallszeichen wird in diesem Fall ausgelassen und die Arbeit mit dem nächsten Zeichen fortgesetzt.

```
3000 REM------UMKEHRUNGSCODE-
     ------------
3005 M$ = M$(N)
3010 FOR J = LEN(M$) TO 1 STEP -1
3020      L$ = MID$(M$,J,1)
3030      C$ = C$ + L$
3040 NEXT J
3050 RETURN
3500 REM-----EINFÜGUNGSCODE---
     ----------------
3505 M$ = M$(N)
3510 FOR J = 1 TO LEN(M$)
3520 L$ = MID$(M$,J,1)
3530 R  = INT(26 * RND(1)) + 65
3540 C$ = C$ + L$ + CHR(R)
3050 NEXT J
3060 RETURN
4000 REM---------QUIZ
4010 PRINT CHR$(147)
4020 PRINT "Bitten Sie Ihren Mitspieler,"
4030 PRINT "diese Nachricht zu entzif-
     fern"
4040 PRINT:PRINT:PRINT C$
4050 PRINT:PRINT "Geben Sie die Antwort
     ein:"
4060 INPUT A$
4070 IF A$ = M$ THEN PRINT "RICHTIG!
     Das war's!"
4080 IF A$ <> M$ THEN PRINT "Schade,
     die Nachricht war: ";M$
4090 RETURN
```

4
Das Rennen der Anfangsbuchstaben

Sozusagen »spielend« haben wir einige wichtige Techniken gelernt: verschiedene Arten, eine Codierung vorzunehmen, Nachrichten zu speichern und auszugeben und die Random-Funktion zu nutzen. Diese Funktionen werden Sie immer und immer wieder in den verschiedensten Spielen finden. Aber bis jetzt haben wir eine Spezialität des Computers völlig außer acht gelassen: die Animation. Bei dieser Spieltechnik geht es um die Bewegung von Bildern auf dem Bildschirm.

Viele Computerspiele sind ohne Animation nicht denkbar. Programmierer verwenden Jahre darauf, die fortschrittlichsten neuen Animationstechniken zu studieren. In diesem Buch wollen wir Sie in die Animation von Zeichen einführen — das heißt, die Benutzung eines beliebigen Zeichens der Tastatur und dessen scheinbare Bewegung auf dem Bildschirm.

Wir beginnen mit einem Rennen zwischen zwei Anfangsbuchstaben, auch Initiale genannt. Auf der linken Seite des Schirms plazieren wir die Initialen von zwei Spielern in einer Ausgangsposition. Von dort bewegen sie sich mit unterschiedlicher Geschwindigkeit nach rechts, sobald einer der Spieler die RETURN-Taste als Startzeichen gedrückt hat. Wenn eines der beiden Zeichen die Ziellinie erreicht hat, zeigt das Programm an, wer gewonnen hat. Jedesmal, wenn Sie dieses Programm starten, wird per Random eine Geschwindigkeit für jeden Anfangsbuchstaben ermittelt, so daß jeder zu jeder Zeit gewinnen kann.

Im nachstehenden Hauptprogramm sind die vier wichtigsten Teile (Unterroutinen) ebenso wie die Variablen, die wir benutzen werden, aufgeführt:

```
10 REM———————ANFANGSBUCH-
   STABENRENNEN———————
20 REM A$   : DER ERSTE ANFANGS-
   BUCHSTABE
25 REM B$   : DER ZWEITE ANFANGS-
   BUCHSTABE
```

```
 30 REM A      : ASCII CODE FOR A$
 35 REM B      : ASCII CODE FOR B$
 40 REM SA     : GESCHWINDIGKEIT VON
    A$
 45 REM SB     : GESCHWINDIGKEIT VON
    B$
 50 REM PA     : POSITION VON A$
 55 REM PB     : POSITION VON B$
 60 REM W$     : DER GEWINNER
 65 REM R$     : EINE ANTWORT (START
    DES RENNENS)
 70 REM J      : EIN ZÄHLER
 75 REM M$     : EIN MELDUNGSTEXT
 80 REM L$     : EIN BUCHSTABE IN M$
 85 REM C      : ASCII CODE FÜR L$
 90 REM———————————————
100 GOSUB 1000:REM———EINGABE AN-
    FANGSBUCHSTABEN
200 GOSUB 2000:REM———AUFBAU DES
    BILDSCHIRMS
300 GOSUB 3000:REM———DAS RENNEN
400 GOSUB 4000:REM———DER GEWIN-
    NER
500 END
```

Die erste Unterroutine speichert die Anfangsbuchstaben der beiden Teilnehmer, die das Rennen gegeneinander austragen. Alles in dieser Routine dürfte Ihnen bekannt sein. Aber sicher nicht das, was nun folgt. Von diesem Punkt an, bis zum Ende des Buches, werden wir Sie mit einer neuen Form der Behandlung von Ein- und Ausgaben auf dem Bildschirm vertraut machen.

Mit dem Ausgabe-Kommando PRINT läßt man normalerweise jeden beliebigen Text auf dem Bildschirm erscheinen. Ist dieser voll und ein PRINT-Befehl wird benutzt, dann verschiebt sich der bis dahin ausgegebene Text um eine Zeile und die oberste Reihe verschwindet. Das ist auch soweit ganz in Ordnung, wenn es sich nur um Texte handelt. In unserem Fall wollen wir jedoch den Schirm als Rennbahn oder Spielfeld benutzen. Der Inhalt muß also für die Dauer des Spieles so bleiben, wie wir ihn zu Beginn gestaltet haben.
Das bedeutet, der Cursor muß sich innerhalb der Bildschirmgrenzen frei bewegen. Beispielsweise könnte es der Spielverlauf fordern, eine Meldung am unteren Bildschirmrand auszugeben und unmittelbar darauf einen Anfangsbuchstaben am oberen Rand anzuzeigen. Eine Möglichkeit, das zu bewerkstelligen, bietet der Commodore 64 mit den Pfeiltasten im Programm. Sie können damit den Cursor in jede beliebige Richtung schieben: nach rechts, links, oben und unten. Oder Sie benutzen die HOME-Funktion, um den Cursor auf die erste Zeile zu bringen (das ist die HOME-CLR-Taste ohne Benutzung der SHIFT-Taste). Diese Methode der Cursor-Bewegung ist wirklich einfach — nur den Cursor auf der gewünschten Position plazieren und dann entweder Meldungen mit PRINT ausgeben oder mit INPUT Daten empfangen.
Wir wollen für unsere Zwecke eine andere Funktion des Commodore benutzen. Das POKE-Kommando, das wir später noch ausführlich bespre-

chen wollen, gibt ein Zeichen auf jeder beliebigen Bildschirmposition aus. Wenn Sie die Textzeichen, die Sie sichtbar machen wollen, kennen, setzen Sie die Bildschirmpositionen fest und führen den POKE-Befehl aus. Für Eingabefunktionen werden wir die GET-Anweisung benutzen, die nur ein Zeichen übernimmt, ohne daß dieses auf dem Bildschirm sichtbar wird. Anschließend können wir mit dem POKE-Befehl das Symbol an jeder beliebigen Stelle aufleuchten lassen.
Sinnvoll mit dieser Technik umgehen zu können, bedeutet für Sie eine erhebliche Erweiterung Ihrer Programmierkenntnisse. Sie versetzt Sie in die Lage, alle Ein- und Ausgabefunktionalitäten in Ihrem Programm unter Kontrolle zu halten. Außerdem bietet sie uns Gelegenheit, die Farbdarstellungsfähigkeit des Commodore auszuprobieren, da wir mit POKE auch die Farbe der darzustellenden Zeichen wählen können.
Die Unterroutine 2000 sorgt für die Farbgestaltung des Bildschirms und legt alle Variablen, die wir für das Rennen benötigen, fest. Die Unterroutine 3000 beinhaltet das Rennen selbst, also die Bewegung der Anfangsbuchstaben, und in der Unterroutine 4000 sorgt das POKE-Kommando für die Ausgabe der Gewinnernachricht. Die neue Eingabetechnik mit GET werden wir allerdings erst in Kapitel 5 benutzen.

Eingabe der Initialen

Zuallererst benötigen wir zwei Anfangsbuchstaben, die das Rennen gegeneinander austragen:

```
1000 REM———EINGABE DER ANFANGS-
     BUCHSTABEN—————————
1010 PRINT CHR$(147)
1020 PRINT "GEBEN SIE DEN ANFANGS-
     BUCHSTABEN DES ERSTEN SPIELERS
     EIN: "
1030 INPUT A$
1040 PRINT:PRINT "NUN DEN ANFANGS-
     BUCHSTABEN DES ZWEITEN SPIE-
     LERS: "
1050 INPUT B$
1060 PRINT:PRINT "DRÜCKEN SIE DIE RE-
     TURN-TASTE ZUM START DES REN-
     NENS: "
1070 INPUT R$
1080 RETURN
```

CHR$(147) ist, wie wir uns erinnern, das Symbol für CLR-HOME. Durch PRINT wird somit der gesamte Schirm gelöscht und der Cursor auf die erste Stelle der ersten Zeile gesetzt. Die Spieler geben ihre Initialen ein, und wir speichern sie als A$ und B$. Wenn jemand die RETURN-Taste drückt, ist die Unterroutine beendet.

Aufbau des Bildschirms

Diese Unterroutine beinhaltet mehrere kleine Schritte, die für die Gestaltung des Bildschirms notwendig sind. Sie wählt einen Zeichenvorrat und die Hintergrundfarbe, die Bildschirmpositionen für die Anfangsbuchstaben und die Zufallszahlen für die Geschwindigkeit der Teilnehmer aus.

```
2000 REM———AUFBAU DES BILD-
     SCHIRMS—————————
2010 POKE 53272,23:REM——ZEICHEN-
     VORRAT 2
2020 POKE 53281, 2:REM——SCHIRMFAR-
     BE ROT
2030 PRINT CHR$(147):REM——LÖ-
     SCHUNG DES BILDSCHIRMS
2040 POKE 53281, 1:REM——SCHIRMFAR-
     BE WEISS
2050 PA=1268:PB=1348
2060 A=ASC(A$):B=ASC(B$)
2070 POKE PA,A:POKE PB,B
2080 SA=INT(2*RND(1))+1
2090 SB=INT(2*RND(1))+1
2100 FOR J=1 TO 1000:NEXT J
2110 RETURN
```

Wir benutzen die POKE-Anweisung in dieser Unterroutine gleich in mehreren Fällen. Wie Sie vielleicht wissen, werden Daten in einem Computer in sogenannten Speicherstellen, die auch Bytes genannt werden, aufbewahrt. Den Wert einer solchen Speicherstelle können Sie durch das POKE-Kommando verändern. In diesem Fall müssen der POKE-Anweisung zwei durch Komma getrennte Zahlen folgen. Die erste Zahl bestimmt die Speicheradresse, die zweite stellt den neuen Wert dar.
Bestimmte Speicherstellen sind im Commodore 64 für spezielle Daten reserviert. Die Adresse 53272 enthält die Information über den zu benutzenden Zeichensatz. Der Commodore ver-

fügt über zwei Zeichensätze, die allerdings nicht gleichzeitig benutzt werden können. POKE 53272,21 schaltet den ersten Zeichensatz, POKE 53272,23 den zweiten. Nebenbei bemerkt: Sie können diesen Vorgang nachvollziehen, wenn Sie die Commodore-Taste links unten und den SHIFT-Schalter benutzen. Wir arbeiten mit dem Zeichenvorrat 2, weil wir ASCII-Zahlen verwenden, um die Zeichen auf dem Bildschirm darzustellen. Sie werden sich an die Aussage in Kapitel 3 erinnern, daß jedes Zeichen einer ASCII-Codezahl entspricht: A ist 65, B ist 66 und so weiter. Im Zeichensatz 2 stehen den Buchstaben A bis Z die Codezahlen 65 bis 90 gegenüber, genau wie im ASCII-Code. Der erste Zeichensatz enthält zwar auch die Großbuchstaben, aber die Codezahlen sind verschieden. Sehen Sie sich am besten im Anhang E des Benutzerhandbuchs die Tabelle der Zeichensätze an.

Eine andere reservierte Speicheradresse ist 53281, in der die Kennzahl für die Bildschirmfarbe festgehalten wird. Es gibt insgesamt sechzehn Farben: Schwarz ist 0, Weiß ist 1, Rot ist 2 und so weiter. POKE 53281, 0 zum Beispiel ändert die Bildschirmfarbe auf Schwarz. Sie können ja mal die anderen Farben ausprobieren, indem Sie die Zahlen bis 15 durch POKE in den Bereich 53281 bringen. Die Zeile 2020 in unseren Unterroutinen ändert die Bildschirmfarbe auf Rot um, dann wird durch die Zeile 2030 der Bildschirm gelöscht und mit der Zeile 2040 die Farbe auf Weiß geändert. Warum wurde überhaupt die Rotumschaltung gemacht? Weil wir eine andere

Farbe für die Textdarstellung benötigen und wir mit Hilfe dieser Methode Rot als unsere Schreibfarbe festsetzen können. Der Text wird in diesem Fall in roten Lettern auf dem weißen Bildschirm ausgegeben. Unser nächster Schritt wäre die Sichtbarmachung der Initialen auf dem Schirm. Der Commodore-64-Bildschirm besitzt 25 Zeilen mit je 40 Stellen pro Zeile. Jede Bildschirmposition hat eine eigene Adresse. Die erste Stelle links wird mit der Zahl 1024, die nächste mit 1025 und so weiter beziffert. Die erste Stelle endet also mit der Zahl 1063. Die zweite Stelle beginnt mit 1064 und endet bei 1103. Die letzte Bildschirmposition rechts unten trägt die Nummer 2023. Damit stehen uns 1000 Speicherstellen zur Verfügung.
Bei der Arbeit mit diesen 1000 unterschiedlichen Bildschirmpositionen ist es sinnvoll, mit einer Übersicht zu arbeiten. Im Anhang G des Commodore-64-Benutzerhandbuchs finden Sie dazu eine Aufstellung. Sie können sich aber auch selbst eine Tabelle anfertigen. Den ersten Anfangsbuchstaben werden wir in Zeile 7, Spalte 5 positionieren. Er hat also die Bildschirmadresse 1268. Der zweite startet zwei Zeilen tiefer, auf Adresse 1348. Die Zeile 2050 der Unterroutine ordnet diese Zahlen den Bereichen PA und PB zu, die für die Positionen von A und B stehen. Wir müssen diese Werte festhalten, weil wir ja später bei der Bewegung der Anfangsbuchstaben die Positionen verändern.
In der Zeile 2060 werden die ASCII-Codezahlen für die zwei Anfangsbuchstaben festgelegt und als A und B bezeichnet. Wir benötigen diese Co-

dezahlen anstelle der Buchstaben in der Befehlszeile 2070. POKE PA,A bedeutet den Übertrag des Zeichens A in die Speicherstelle mit der Adresse 1268. Wir transportieren ein Zeichen auf den Bildschirm, indem wir mit POKE seine Codezahl auf der zugeordneten Bildschirmadresse speichern. POKE PB,B sorgt für die Speicherung der Codezahl von B im Bereich 1348. Der Anfangsbuchstabe B$ wird also an dieser Stelle auf dem Bildschirm erscheinen.
Die Zeilen 2080 und 2090 greifen Zufallszahlen für die Anfangsbuchstaben A und B auf. Jede Zahl kann entweder 1 oder 2 sein. Wir verwenden diese Zahlen, um festzulegen, wie viele Spalten der jeweilige Anfangsbuchstabe auf einmal überspringen darf. Ist die Zufallszahl für den Anfangsbuchstaben A eine 2 und die Zahl für B eine 1, darf A sich um zwei Spalten weiterbewegen und wird das Rennen gewinnen. Die Zeile 2100 unterbricht kurzfristig das Programm, um den Teilnehmern einen Blick auf die Startpositionen zu ermöglichen, bevor das Rennen beginnt.

Animation — die Kunst der Bewegung

Die dritte Unterroutine sorgt für die Bewegung der Anfangsbuchstaben quer über den Bildschirm. Um zu verstehen, wie das funktioniert, sollten wir uns mit der Animation in einem Commodore 64 vertraut machen. Wie wir es schon früher bei neuer Ablauflogik getan haben, werden wir uns nun ein kleines Testprogramm schreiben. Wenn Sie gerade das Programm »Rennen« eingeben, sichern Sie es bitte und geben NEW ein, bevor Sie dieses Testprogramm erfassen.

Es gibt eine Vielzahl von Techniken innerhalb der Computeranimation. In unserem Fall benutzen wir eine Technik, die man »Zeichenanimation« nennt. Das bedeutet, wir suggerieren einen Bewegungsablauf, indem wir ein Zeichen an einer bestimmten Stelle des Bildschirms ausgeben, es

wieder löschen und an einer anderen Stelle, ein wenig entfernt, wieder ausgeben. Durch die ständige Wiederholung dieser Schritte in kurzen Abständen scheint sich das Zeichen zu bewegen — zu »laufen«.
Unser kleines Programm soll dazu dienen, den Buchstaben Z von der Bildschirmposition 1104 zur Position 1124 — das heißt, von der ersten Stelle der dritten Zeile zur Mitte derselben zu bewegen. Wir unterstellen, daß der Bildschirm bereits gelöscht und die Bildschirmfarbe sowie der Zeichensatz 2 festgelegt wurden, wie wir es vorhin vorgeführt haben. Und so sieht das in BASIC aus:

```
10 POKE 1104, 90
20 FOR SL = 1104 TO 1123
30      POKE SL,32
40      POKE SL+1,90
50 FOR J=1 TO 100:NEXT J
60 NEXT SL
```

Zeile 10 setzt ein Z auf die Speicherstelle 1104. Falls uns die Codezahl von Z nicht bekannt gewesen wäre, hätte es auch mit dem Befehl POKE 1104,ASC(»Z«) geklappt. Die Schleife in den Zeilen 20 bis 60 dient zur Löschung des Z auf seiner momentanen Position und zur Ausgabe auf der nächstfolgenden. (SL steht für Screen Location.) Die Schleife wird nur bis zur Position 1123 durchgeführt, weil bei dem letzten Durchlauf, wenn SL den Wert 1123 annimmt, in Zeile 40 das Z auf Position SL+1, also 1124, ausgegeben wird. Das ist

der von uns festgelegte, endgültige Standort von Z. Beachten Sie bitte die kurze Pause, die jeweils durch die Zahl 50 verursacht wird.. Ohne sie würde sich der Buchstabe vermutlich zu schnell bewegen. Sie sollten mit der Länge der Pause, also der Geschwindigkeit, ruhig experimentieren, bis sie Ihnen richtig erscheint.
Ebenso sollten Sie auch mit den anderen Variablen Versuche anstellen: Ändern Sie doch die Start- und Endadressen oder die Codezahl, erhöhen Sie den Bewegungsfaktor SL durch die Angabe STEP 2 oder STEP 3, um sich den Effekt anzusehen. Unsere gesamte Bewegungstechnik wird in ähnlicher Weise funktionieren. Es gibt nur einen kleinen Unterschied: Wir wollen nicht unbedingt immer die gleiche Geschwindigkeit, also keinen regelmäßigen Sprungfaktor. Deshalb greifen wir auf zwei Zufallszahlen (SA und SB) zurück, die diesen Faktor bestimmen sollen.

Das Rennen

Wir haben die Startpositionen für die Anfangsbuchstaben, nämlich 1268 und 1348, festgelegt. Sie starten in der fünften Spalte ihrer Zeilen. Wir müssen nun auch noch die Ziellinie bestimmen. Nach unserer Spielregel soll der Buchstabe, der zuerst 32 Spalten überwunden hat, der Gewinner sein. Das bedeutet, daß die Ziellinie für den Buchstaben A die Position 1300 und für B die Position 1380 ist.
Zusätzlich benötigen wir noch eine Variable für das Rennen. Wenn einer der Anfangsbuchstaben gewonnen hat, wird dieser im Datenfeld W$ gespeichert, damit wir ihn als Gewinner anzeigen können.

```
3000 REM———DAS RENNEN—————
3010 REM: DIE BEWEGUNG DES ERSTEN
     BUCHSTABENS
3020 POKE PA,32
3030 PA=PA+SA
3040 POKE PA,A
3050 IF PA)1300 THEN W$=A$:GOTO 3130
3060 REM: DIE BEWEGUNG DES ZWEITEN
     BUCHSTABENS
3070 POKE PB,32
3080 PB=PB+SB
3090 POKE PB,B
3100 IF PB)1380 THEN W$=B$:GOTO 3130
3110 FOR J=1 TO 100:NEXT J
3120 GOTO 3020
3130 RETURN
```

In den Zeilen 3020 bis 3050 wird der Anfangsbuchstabe A bewegt. Die Zeile 3020 regt die Lö-

schung auf seiner derzeitigen Position (Codezahl 32 entspricht dem Leerwert) an. In Zeile 3030 wird die Randomzahl als Sprungfaktor addiert, das heißt, eine 1 oder eine 2 wird der Bildschirmadresse hinzugefügt. Die Zeile 3040 gibt den Buchstaben auf der neuen Position aus. In der Zeile 3050 wird geprüft, ob der Buchstabe schon gewonnen hat. Trifft das zu, wird das Initial in W$ gespeichert und zum Ende der Unterroutine verzweigt.
In den Zeilen 3070 bis 3100 werden genau die gleichen Programmschritte für den zweiten Anfangsbuchstaben B durchgeführt: Löschung des Buchstabens, Ermittlung seiner neuen Position und an dieser Stelle Ausgabe des Zeichens sowie die Überprüfung, ob er gewonnen hat.
Die Pause in Zeile 3110 setzt die Durchführungsgeschwindigkeit des Programms herab, und in der Zeile 3120 wird zurückverzweigt zur Bewegung des ersten Initials. Die Anfangsbuchstaben werden so lange bewegt, bis einer von beiden die Ziellinie überschreitet, dann übernimmt die letzte Unterroutine den weiteren Programmablauf. Der Gewinner steht fest, und wir wollen eine Nachricht auf dem Bildschirm ausgeben. Da das POKE-Kommando nur mit einem einzigen Zeichen arbeiten kann, ist hierfür eine Reihe von Befehlen erforderlich:

1. Speicherung der Nachricht als Variable (M$)
2. Zeichenweise Verarbeitung von M$
a) Aufgreifen eines Zeichens mit der MID$-Funktion

b) Umsetzung des Zeichens in die Codezahl mit der ASC-Funktion
c) Ausgabe jedes Zeichens hintereinander mit Hilfe von POKE
Zuerst wollen wir den Text »Der Gewinner ist: « in der achtzehnten Zeile des Bildschirms, das heißt auf Position 1704, ausgeben.

```
4000 REM————————DER GEWIN-
     NER—————————
4010 M$="DER GEWINNER IST: "
4020 FOR J=1 TO LEN(M$)
4030      L$=MID$(M$,J,1)
4040      C=ASC(L$)
4050      POKE 1703+J,C
4060 NEXT J
```

Die Schleife in den Zeilen 4020 bis 4060 verarbeitet M$-Zeichen für Zeichen. Erinnern wir uns, daß wir die Länge von M$ nicht ermitteln müssen — das macht die Funktion LEN für uns. Jedesmal, wenn die Schleife durchlaufen wird, wird das nächste Zeichen in L$ gespeichert. C ist die Codezahl für dieses Zeichen. Zeile 4050 gibt das Zeichen mit POKE auf der Adresse 1703+J aus. Da J beim ersten Mal den Wert 1 enthält, wird das erste Zeichen auf 1703+1 oder 1704 angezeigt. Das nächste Zeichen dann auf 1705 und so weiter.

```
4070 POKE 1722,ASC(W$)
4080 RETURN
```

Es bleibt nur noch die Ausgabe des Initials, das das Rennen gewonnen hat, unmittelbar hinter dem angezeigten Text. Sie werden herausfinden,

daß das Initial auf Speicherstelle 1722 erscheinen muß. Beachten Sie bitte, daß wir die ASC-Funktion im POKE-Kommando verwenden, um die Codezahl für W$ speichern zu können.

Das gesamte Programm

```
 10 REM------ANFANGSBUCH-
    STABENRENNEN--------
 20 REM A$   : DER ERSTE ANFANGS-
    BUCHSTABE
 25 REM B$   : DER ZWEITE ANFANGS-
    BUCHSTABE
 30 REM A    : ASCII CODE FOR A$
 35 REM B    : ASCII CODE FOR B$
 40 REM SA   : GESCHWINDIGKEIT VON
    A$
 45 REM SB   : GESCHWINDIGKEIT VON
    B$
 50 REM PA   : POSITION VON A$
 55 REM PB   : POSITION VON B$
 60 REM W$   : DER GEWINNER
 65 REM R$   : EINE ANTWORT (START
    DES RENNENS)
 70 REM J    : EIN ZÄHLER
 75 REM M$   : EIN MELDUNGSTEXT
 80 REM L$   : EIN BUCHSTABE IN M$
 85 REM C    : ASCII CODE FÜR L$
 90 REM--------------------
100 GOSUB 1000:REM---EINGABE AN-
    FANGSBUCHSTABEN
200 GOSUB 2000:REM---AUFBAU DES
    BILDSCHIRMS
```

```
 300 GOSUB 3000:REM———DAS RENNEN
 400 GOSUB 4000:REM———DER GEWIN-
     NER
 500 END
1000 REM———EINGABE DER ANFANGS-
     BUCHSTABEN———————
1010 PRINT CHR$(147)
1020 PRINT "GEBEN SIE DEN ANFANGS-
     BUCHSTABEN DES ERSTEN SPIELERS
     EIN:"
1030 INPUT A$
1040 PRINT:PRINT "NUN DEN ANFANGS-
     BUCHSTABEN DES ZWEITEN SPIE-
     LERS:"
1050 INPUT B$
1060 PRINT:PRINT "DRÜCKEN SIE DIE RE-
     TURN-TASTE ZUM START DES REN-
     NENS:"
1070 INPUT R$
1080 RETURN
2000 REM———AUFBAU DES BILD-
     SCHIRMS——————————
2010 POKE 53272,23:REM——ZEICHEN-
     VORRAT 2
2020 POKE 53281,2:REM——SCHIRMFAR-
     BE ROT
2030 PRINT CHR$(147):REM——LÖ-
     SCHUNG DES BILDSCHIRMS
2040 POKE 53281,  1:REM——SCHIRMFAR-
     BE WEISS
2050 PA=1268:PB=1348
2060 A=ASC(A$):B=ASC(B$)
2070 POKE PA,A:POKE PB,B
```

```
2080 SA=INT(2*RND(1))+1
2090 SB=INT(2*RND(1))+1
2100 FOR J=1 TO 1000:NEXT J
2110 RETURN
3000 REM———DAS RENNEN—————
3010 REM: DIE BEWEGUNG DES ERSTEN
     BUCHSTABENS
3020 POKE PA,32
3030 PA=PA+SA
3040 POKE PA,A
3050 IF PA>1300 THEN W$=A$:GOTO 3130
3060 REM: DIE BEWEGUNG DES ZWEITEN
     BUCHSTABENS
3070 POKE PB,32
3080 PB=PB+SB
3090 POKE PB,B
3100 IF PB>1380 THEN W$=B$:GOTO 3130
3110 FOR J=1 TO 100:NEXT J
3120 GOTO 3020
3130 RETURN
4000 REM————DER GEWINNER————
4010 M$="DER GEWINNER IST:"
4020 FOR J=1 TO LEN(M$)
4030      L$=MID$(M$,J,1)
4040      C=ASC(L$)
4050      POKE 1703+J,C
4060 NEXT J
4070 POKE 1722, ASC (W$)
4080 RETURN
```

Variationen

Wenn Sie einmal wissen, wie der Hase läuft, fällt Ihnen die Animation leichter und leichter. Alles, was Sie brauchen, ist Praxis. Und damit Sie sich nicht langweilen, machen wir Ihnen noch vier Vorschläge zur Optimierung des Programms. Sie haben sicher Phantasie genug, um sich noch weitere auszudenken.

1. Bildschirmformat: Gestalten Sie den Schirm nach Ihren Vorstellungen durch Ausgabe eines Spieltitels oben auf dem Bildschirm, durch Markierung der Start- und der Ziellinie. Sie sollten dabei Ihrer Phantasie freien Lauf lassen.

2. Mehr Farbigkeit: Erweitern Sie die Farbenanzahl auf Ihrem Schirm. Eine andere interessante Ausstattung dieses Computermodells ist die Speicherung der Farbe für jede Bildschirmadresse. In unserem Programm haben wir die Farbe Rot für alle Positionen gespeichert. Die Farbenspeichertabelle entspricht im Aufbau völlig der

Bildschirmadressentabelle mit der Ausnahme, daß sie auf Position 55296 anstatt 1024 beginnt. Das Kommando POKE 55296,6 speichert die Farbe Blau auf der ersten Stelle der ersten Zeile; wenn Sie anschließend den Befehl POKE 1024,ASC(»T«) ausführen, wird dort ein blaues T erscheinen. Eine kleine Subtraktion weist darauf hin, daß die Farbadressnummer um 54272 größer ist als die zugeordnete Bildschirmadresse. In Zeile 4050 geben wir mit POKE das Zeichen auf Adresse 1703+J aus. Gehen wir davon aus, daß wir kurz zuvor die Farbe dieser Stelle verändert haben. Und zwar so: POKE 1703+J+54272,Farbe. Was für eine Farbe? Nehmen Sie eine Zufallszahl von 0 bis 15: R=INT(16+RND(1)).

3. Höhere Geschwindigkeit: Wir haben nur eine Geschwindigkeit von 1 oder 2 gewählt. Sie wünschen sich vielleicht weitere Variationen. Wissen Sie noch, wie Sie die Random-Funktion verwenden müssen, um den Bereich der verfügbaren Zahlen zu verändern?

4. Mehr Teilnehmer am Rennen: Was halten Sie von einem dritten Anfangsbuchstaben als weiteren Rennteilnehmer? Sie müssen bloß drei eingeben, drei Geschwindigkeiten festsetzen, sie auf ihre Startpositionen ausgeben und alle drei zur selben Zeit bewegen. Benutzen Sie die gleichen vier Schritte in der RENNEN-Unterroutine für den dritten Anfangsbuchstaben, wie wir sie für die beiden anderen verwendet haben.

5
Die Seifenblase platzt

Das Rennen der Anfangsbuchstaben war noch ein vergleichsweise einfaches Spiel — es bestand eigentlich nur aus dem Rennen. Animation kann aber ebensogut als Teil komplizierterer Spiele auftreten. Stellen Sie sich folgenden Bildschirminhalt vor: auf der rechten Seite befindet sich ein »Ballon« oder eine »Seifenblase« und auf der linken der Anfangsbuchstabe eines Spielers. In der unteren Hälfte des Bildschirms wird dem Spieler ein arithmetisches Programm genannt, das er oder sie lösen muß. Wenn die Antwort richtig ist, bewegt sich der Anfangsbuchstabe über den Bildschirm und bringt die »Blase«, in der sich eine Nachricht befindet, zum Platzen. Dieses Spiel ist eine Kombination von vielen Elementen, die wir bereits aus früheren Programmen kennen: Ani-

mation, Zufallszahlen, die Benutzung von DATA-Anweisungen, die Möglichkeit mehrerer Antwortversuche. Gute Spiele werden immer aus der Kombination mehrerer solcher Elemente hergestellt, aber keiner dieser Bausteine muß in sich selbst kompliziert sein.
Wir werden uns hier mit einer neuen Programmiertechnik beschäftigen. Wie schon im vorigen Kapitel angedeutet, benutzen wir bei der Bildschirmgestaltung nicht mehr die Kommandos PRINT und INPUT für die Bildschirmaus- beziehungsweise -eingabe. Im vorhergehenden Kapitel haben wir schon eine Methode entwickelt, um Texte oder Meldungen auf dem Bildschirm anzuzeigen. Nun wollen wir den GET-Befehl ausschließlich zur Eingabe eines Zeichens in das Programm verwenden.

```
 5 REM————DIE SEIFENBLASE
   PLATZT—————————
10 REM I$    : ANFANGSBUCHSTABE
15 REM I     : ASCII-CODEZAHL FÜR I$
20 REM SL    : BILDSCHIRMPOSITION
25 REM T$    : MELDUNGSTEXT
30 REM L$    : EIN BUCHSTABE DER MEL-
   DUNG
35 REM A     : ASCII-CODEZAHL DES
   BUCHSTABENS
40 REM J     : EIN ZÄHLER
45 REM K$    : TASTENEINGABE
50 REM A$    : EINGABE DER ANTWORT
60 REM X     : ZUFALLSZAHL
65 REM Y     : ZUFALLSZAHL
```

```
 70 REM X$    : ZEICHENDARSTELLUNG
    VON X
 75 REM Y$    : ZEICHENDARSTELLUNG
    VON Y
 80 REM V     : WERT DER ANTWORT
 85 REM———————————————————
100 GOSUB 1000:REM——BILDSCHIRM-
    AUFBAU
200 GOSUB 2000:REM——FRAGESTEL-
    LUNG(EN)
300 GOSUB 3000:REM——PLATZEN DER
    BLASE
400 END
```

Der Bildschirmaufbau

Diese Unterroutine enthält mehrere kleine Teilbereiche. Zuerst geben wir diesem Spiel einen Namen und setzen den Titel an den oberen Rand

des Bildschirms. Dann zeichnen wir den Ballon und geben den Anfangsbuchstaben ein. Zum Schluß erklären wir dem Mitstreiter das Spiel. Zum besseren Verständnis sehen wir uns jeden Teilbereich im einzelnen an:

```
1000 REM———BILDSCHIRMAUFBAU——
     ——————————
1010 POKE 53272,23:REM——ZEICHEN-
     SATZ 2
1020 POKE 53281, 0:PRINT CHR$(147):
     POKE 53281, 1
1030 T$="DIE SEIFENBLASE PLATZT"
1040 SL=1034:GOSUB 1500
```

Die Zeile 1010 wählt den zweiten Zeichensatz für die Datenanzeige aus. Mit Zeile 1020 bestimmen wir die Hintergrundfarbe Weiß (Farbe 1) und die Zeichenfarbe Schwarz (Farbe 0). Außerdem wird der Bildschirm gelöscht. Übrigens haben wir diese Anweisungen schon im letzten Kapitel ausführlich besprochen.
Wir wollen also den Text DIE SEIFENBLASE PLATZT in der ersten Zeile des Bildschirms ausgeben. Hierzu legen wir zuerst den Text in einer Variablen T$ fest, ordnen die Startposition SL zu und rufen eine separate Unterroutine auf. Die Unterroutine 1500 tut genau das, was die vierte Unterroutine in Kapitel 4 (DER GEWINNER) auch getan hat — sie gibt T$ ab der Position SL, die wir zugeordnet haben, auf dem Bildschirm aus. Wir werden uns das am Ende dieses Abschnitts noch einmal anschauen.

```
1050 DATA 1175,1176,1213,1214,1217,1218
1060 DATA 1252,1259,1292,1299,1332,1339
1070 DATA 1373,1374,1377,1378,1415,1416
1080 FOR J=1 TO 18
1090      READ SL:POKE SL,ASC("O")
1100 NEXT J
```

Der oben aufgezeigte Programmbaustein zeichnet die »Seifenblase« auf der rechten Seite des Bildschirms. Die Blase setzt sich aus »O«-Buchstaben zusammen und nimmt 18 Bildschirmadressen in Anspruch. Wie Sie sehen können, werden diese 18 Positionen als DATA festgelegt. Die Zeile 1090 speichert den DATA-Inhalt mit READ im Bereich SL und zeigt mit POKE den Buchstaben O auf jeder definierten Bildschirmposition an.

```
1110 T$="GEBEN SIE IHREN ANFANGS-
     BUCHSTABEN EIN:"
1120 SL=1544:GOSUB 1500
1130 GET I$:IF I$="" THEN 1130
1140 I=ASC(I$)
1150 POKE 1268,I
```

Die Zeilen 1110 und 1120 machen den Text GEBEN SIE IHREN ANFANGSBUCHSTABEN EIN auf dem Bildschirm sichtbar: Er beginnt auf der Adresse 1544, also auf der vierzehnten Zeile. Das Kommando GET I$ in Zeile 1130 stellt dem Programm eine Tastatureingabe zur Verfügung — wenn jetzt ein Zeichen eingegeben wird, wird es in I$ gespeichert. Wird kein Zeichen eingetastet, setzt

das Programm seine angefangene Arbeit fort; deshalb muß geprüft werden, ob der Bereich I$ leer ist (beachten Sie bitte, daß zwischen den Anführungsstrichen keine Leerstelle sein darf.) In diesem Fall wird so lange der Befehl auf 1130 wiederholt, bis eine Eingabe erfolgt ist. Durch diesen »Eingabezwang« ist der weitere, richtige Programmverlauf gewährleistet.
Die Zeile 1140 setzt den Inhalt von I$ in die ASCII-Codezahl um, damit das Initial in der Folgezeile gegenüber der Seifenblase, auf der linken Seite des Schirms, sichtbar gemacht werden kann.

```
1160 T$="BEANTWORTEN SIE EINE FRAGE
     RICHTIG, UND"
1170 SL=1624:GOSUB 1500
1180 T$="IHR ANFANGSBUCHSTABE
     BRINGT DIE SEIFENBLASE ZUM PLAT-
     ZEN."
1190 SL=1664:GOSUB 1500
1200 FOR J = 1 TO 2000:NEXT J
1210 RETURN
```

Die zusätzlich ausgegebenen zwei Textzeilen werden im Bereich T$ definiert und erhalten eine Bildschirmposition SL. Anschließend an diese Definition wird jeweils wieder die Unterroutine 1500 zur Anzeige der Texte aufgerufen. Um sicher zu sein, daß der Spieler wirklich genug Zeit hat, den Text zu lesen, benutzen wir wieder die »Pausen«-Schleife in Zeile 1200, die den Computer zwingt, bis 2000 zu zählen und erst dann mit dem Programm fortzufahren.

Bevor wir uns nun dem eigentlichen Spiel zuwenden, betrachten wir die Unterroutine 1500. Sie ist fast identisch mit dem Ablauf, den wir am Ende des vierten Kapitels verwendet haben:

```
1500 REM------TEXTAUSGABE--
1510 FOR J=1 TO LEN(T$)
1520      L$=MID$(T$,J,1)
1530      A=ASC(L$)
1540      POKE SL+J-1,A
1550 NEXT J
1560 RETURN
```

Der einzige tatsächliche Unterschied ist die Bestimmung, auf welcher Adresse jedes einzelne Zeichen sichtbar werden soll. Nehmen wir an, die Startadresse soll 1544 sein, und diese Nummer wird in SL festgelegt. Beim ersten Schleifendurchlauf hat J den Wert 1. Nun ergibt ja 1544+1 den Wert 1545, also müssen wir grundsätzlich eine 1 subtrahieren, um die richtige Adresse zu erhalten. Wenn Sie diese Formel einmal gedanklich nachvollziehen, werden Sie feststellen, daß die Addition von SL und J und die anschließende Subtraktion einer 1 wirklich »astrein« ihre Funktion erfüllt.

Wie die Frage gestellt wird

In diesem Programmteil werden dem Spieler so lange Rechenaufgaben gestellt, bis er eine davon richtig beantwortet hat. Die Fragestellung setzt die Auswahl zweier Zufallszahlen, X und Y genannt, voraus und besteht darin, die Summe aus X + Y zu ermitteln. Anschließend wird das vom Spieler eingegebene Ergebnis auf Richtigkeit überprüft:

```
2000 REM———FRAGESTELLUNG(EN)——
2010 X = INT(100 * RND(1)) + 1
2020 Y = INT(100 * RND(1)) + 1
2030 GOSUB 2500:REM——UNTEREN
     SCHIRMTEIL LÖSCHEN
2040 X$=STR$(X):Y$=STR$(Y)
2050 T$="WIEVIEL IST "+X$+" + "+Y$+"?"
2060 SL=1544:GOSUB 1500
2070 SL=1630:GOSUB 2700:REM—BILD-
     SCHIRMEINGABE
```

```
2080 V=VAL(A$)
2090 IF V <> X + Y THEN 2010
2100 GOSUB 2500
2110 RETURN
```

In den Zeilen 2010 und 2020 werden die beiden Zufallszahlen ermittelt, und zwar zwischen 1 und 100. Die Unterroutine auf Zeile 2500 löscht den unteren Teil des Bildschirms, damit wir eine neue Aufgabe machen können. Mit diesem Vorgang werden wir uns später noch genauer beschäftigen.
Nun kommt es darauf an, die Fragestellung auf dem Schirm richtig zu formulieren. Der hierzu benötigte Text besteht aus fünf Teilen:

1. dem Textbaustein WIEVIEL IST
2. der ersten Zahl X
3. einem Pluszeichen (+)
4. der zweiten Zahl Y
5. einem Fragezeichen (?)

Wenn X den Wert 34 hat und Y den Wert 87, dann liest sich der Text folgendermaßen: WIEVIEL IST 34 + 87? Wir bezeichnen diesen Text wieder als T$, damit wir die Unterroutine 1500 benutzen können, um ihn auf dem Schirm anzuzeigen. Das geht aber nur dann, wenn alle Textteile in T$ Zeichenketten sind und keine Rechenfelder. Deshalb müssen vor der Textdefinition die Rechenfelder X und Y in Zeichenkettenformat umgewandelt werden. Das ist die Aufgabe der STR$-Funktion in Zeile 2040 — sie setzt den numerischen Inhalt des in Klammern angegebenen Feldes in ein alphanumerisches Datenfeld um. Nach der Umwandlung werden in Zeile 2050 die fünf kleinen Textbausteine zusammengefügt, als T$ bezeichnet und mittels Aufruf der Unterroutine 1500 in Zeile 2060 ausgegeben.
In Zeile 2070 wird in eine andere Unterroutine verzweigt, in der mit dem GET-Kommando die Eingabe des Ergebnisses durch den Spieler vollzogen werden soll. Auch diese Unterroutine sehen wir uns noch genauer an. Für den Moment soll genügen, daß wir die Antwort in einem Bereich A$ erhalten.
Im Anschluß an die Eingabe des Ergebnisses wird der Inhalt des Antwortbereichs A$ mit der im Programm ermittelten Summe aus X und Y verglichen. Da aber eine Eingabe im Zeichenkettenformat nicht mit einer Rechenoperation verglichen werden kann, muß vorher das Feld A$ durch die VAL-Funktion in Zeile 2080 in ein Rechenfeld, genannt V, übertragen werden. Dies ist genau das Gegenteil dessen, was die Funktion

STR$ bewirkt. Alle numerischen Zeichen einer Kette werden in ein Zahlenfeld übertragen. Anschließend können wir dann das Feld V mit der Summe aus X + Y vergleichen. Bei Übereinstimmung, das heißt, der richtigen Lösung, wird der untere Bildschirmteil gelöscht und die Unterroutine beendet. Andernfalls erfolgt ein Rücksprung auf die Zeile 2010, und das uns schon bekannte Programm beginnt von vorn. Die Schleife wird so lange durchlaufen, bis der Spieler die korrekte Antwort eingibt — bis V der Rechenoperation X + Y entspricht.

Die Löschung des unteren Bildschirmteils

Mit folgender Unterroutine löschen Sie die Zeilen 14 bis 17 auf dem Bildschirm:

```
2500 REM-----LÖSCHUNG DES UNTE-
     REN SCHIRMTEILS-------
2510 FOR SL = 1544 TO 1703
2520      POKE SL,32
2530 NEXT SL
2540 RETURN
```

Sie sehen, es ist ganz einfach. Wir übertragen auf jede Bildschirmstelle, die gelöscht werden soll, einen Leerwert (Codezahl 32).

Die Eingabe der Antwort

Wir haben bereits das GET-Kommando zum Empfang eines einzelnen Zeichens benutzt. Die folgende Unterroutine besteht aus mehreren Wiederholungen dieses Befehls und der Anzeige der eingetasteten Zeichen auf dem Bildschirm, bis der Spieler die RETURN-Taste drückt.

```
2700 REM----------ERGEBNISEIN-
     GABE----------
2710 A$=""
2720 GET K$:IF K$="" THEN 2720
2730 IF ASC(K$)=13 THEN 2780
2740 A$=A$+K$
2750 POKE SL,ASC(K$)
2760 SL=SL+1
2770 GOTO 2720
2780 RETURN
```

Der Ablauf empfängt die über Tastatur eingegebenen Ziffern des Ergebnisses einzeln und fügt sie im Bereich A$ zu einer Zeichenkette zusammen. Zeile 2710 stellt sicher, daß A$ zu Beginn völlig leer ist — denn wir wollen ja nicht, daß dieses Feld die letzte, möglicherweise falsche Antwort enthält. In Zeile 2720 wird die Tastatureingabe mit GET übernommen und außerdem überprüft, ob überhaupt eine Eingabe gemacht wurde, damit das Programm seine Arbeit nicht einfach fortsetzt. Anschließend erfolgt die Kontrolle, ob der Spieler die RETURN-Taste gedrückt hat (Codezahl 13). In diesem Fall wird die Eingabe als beendet betrachtet und die Routine beendet.
War die gedrückte Taste nicht RETURN, wird das eingegebene Zeichen der Zeichenkette in A$ beigefügt, also Zeichen für Zeichen in A$ gespeichert. Anschließend wird das letzte Zeichen in Zeile 2750 auf dem Bildschirm angezeigt, deshalb haben wir vor Beginn dieser Unterroutine in SL den Wert 1630 gespeichert. Durch Addition ei-

ner 1 auf den in SL gespeicherten Wert erreichen wir nun, daß das nächstfolgende Zeichen auf der nächstfolgenden Bildschirmposition sichtbar wird, bevor es nach Rücksprung auf Zeile 2720 eingegeben werden kann.
Unangenehm ist allerdings die Tatsache, daß bei vorliegendem Ablauf der Spieler keine Korrekturmöglichkeit hat, falls er sich vertippt. Aber einige zusätzliche Befehle ermöglichen auch diesen Luxus. Können Sie sich schon vorstellen, wie das aussehen müßte? Wir geben Ihnen im Abschnitt »Variationen« am Ende des Kapitels ein paar Tips dazu.

Die Seifenblase zum Platzen bringen

Die FRAGESTELLUNG-Routine endet nur dann erfolgreich, wenn die eingegebene Zahl richtig war. Tritt der Fall ein, ist es an der Zeit, für diesen Fall die Blase zum Platzen zu bringen. Die dazu be-

nötigte Unterroutine muß aus drei Bausteinen bestehen:
1. den Anfangsbuchstaben über den Bildschirm bewegen
2. die Seifenblase verschwinden lassen
3. den Text RICHTIG! "innerhalb" der Blase ausgeben
Wir werden uns nun diese drei »Jobs« einzeln ansehen.

```
3000 REM————————PLATZEN DER SEIFEN-
     BLASE————————
3010 FOR SL = 1268 TO 1291
3020      POKE SL,32
3030      POKE SL+1,I
3040      FOR J = 1 TO 50:NEXT J
3050 NEXT SL
```

Diese FOR-NEXT Schleife sieht genauso aus wie unser einfaches Animationsprogramm im letzten Kapitel. Wir wissen genau, wo das Initial seine »Reise« starten muß (Position 1268) und wo die Fahrt zu Ende ist (Position 1292). Die Ausgabe auf Adresse 1268 erfolgte mit POKE schon zu einem früheren Zeitpunkt, und die Speicherstelle 1292 enthält eines der »O« der Seifenblase (siehe auch die DATA-Anweisungen). Die Schleife wird allerdings nur bis zur Adresse 1291 durchgeführt, da wir jedesmal eine Position löschen und den Anfangsbuchstaben auf der nächstfolgenden wieder anzeigen. Beim letzten Durchlauf wird also SL 1291 enthalten und das Zeichen auf 1292 sichtbar gemacht.

```
3060 RESTORE
3070 FOR J=1 TO 18
3080      READ SL:POKE SL,32
3090 NEXT J
```

Und wieder ein neuer Befehl: RESTORE bewirkt die Rückkehr zum Start der DATA-Definitionen, so daß die einzelnen Angaben erneut mit READ in SL eingelesen werden können. Somit lesen wir nun alle 18 Adressen der Seifenblase genauso, wie wir es zur Darstellung auf dem Bildschirm getan haben, mit dem Unterschied, daß wir nun keine »O« ausgeben, sondern mit Leerwert die Blase löschen.

```
3100 T$="RICHTIG"
3110 SL=1293:GOSUB 1500
3120 RETURN
```

Das Ende des Ablaufs besteht aus der Ausgabe des Textes RICHTIG! auf einer Position, die sich innerhalb der ehemaligen Grenzen der Seifenblase befindet.

Das ganze Programm

```
  5 REM-----DIE SEIFENBLASE
    PLATZT----------
 10 REM I$    : ANFANGSBUCHSTABE
 15 REM I     : ASCII-CODEZAHL FÜR I$
 20 REM SL    : BILDSCHIRMPOSITION
 25 REM T$    : MELDUNGSTEXT
 30 REM L$    : EIN BUCHSTABE DER
    MELDUNG
 35 REM A     : ASCII-CODEZAHL DES
    BUCHSTABEN
 40 REM J     : EIN ZÄHLER
 45 REM K$    : TASTENEINGABE
 50 REM A$    : EINGABE DER ANTWORT
 60 REM X     : ZUFALLSZAHL
 65 REM Y     : ZUFALLSZAHL
 70 REM X$    : ZEICHENDARSTELLUNG
    VON X
 75 REM Y$    : ZEICHENDARSTELLUNG
    VON Y
 80 REM V     : WERT DER ANTWORT
 85 REM----------
    -------
100 GOSUB 1000:REM-
    -BILDSCHIRMAUFBAU
200 GOSUB 2000:REM-
    -FRAGESTELLUNG(EN)
300 GOSUB 3000:REM--PLATZEN DER
    BLASE
400 END
1000 REM-----BILDSCHIRMAUFBAU-
    ----------
```

```
1010 POKE 53272,23:REM——ZEICHEN-
     SATZ 2
1020 POKE 53281,  0:PRINT CHR$(147):
     POKE 53281, 1
1030 T$="DIE SEIFENBLASE PLATZT"
1040 SL=1034:GOSUB 1500
1050 DATA 1175, 1176, 1213, 1214, 1217,
     1218
1060 DATA 1252, 1259, 1292, 1299, 1332,
     1339
1070 DATA 1373, 1374, 1377, 1378, 1415,
     1416
1080 FOR J=1 TO 18
1090       READ SL:POKE SL,ASC ("O")
1100 NEXT J
1110 T$="GEBEN SIE IHREN ANFANGS-
     BUCHSTABEN EIN:"
1120 SL=1544:GOSUB 1500
1130 GET I$:IF I$="" THEN 1130
1140 I=ASC(I$)
1150 POKE 1268,I
1160 T$="BEANTWORTEN SIE EINE FRAGE
     RICHTIG, UND"
1170 SL=1624:GOSUB 1500
1180 T$="IHR ANFANGSBUCHSTABE
     BRINGT DIE SEIFENBLASE ZUM PLAT-
     ZEN."
1190 SL=1664:GOSUB 1500
1200 FOR J = 1 TO 2000:NEXT J
1210 RETURN
1500 REM——————TEXTAUSGABE———
     ——————————
1510 FOR J=1 TO LEN(T$)
```

```
1520      L$=MID(T$,J,1)
1530      A=ASC(L$)
1540      POKE SL+J-1,A
1550 NEXT J
1560 RETURN
2000 REM------FRAGESTEL-
     LUNG(EN)------------
2010 X = INT(100 * RND(1)) + 1
2020 Y = INT(100 * RND(1)) + 1
2030 GOSUB 2500:REM--UNTEREN
     SCHIRMTEIL LÖSCHEN
2040 X$=STR$(X):Y$=STR$(Y)
2050 T$="WIEVIEL IST "+X$+" + "+Y$+"?"
2060 SL=1544:GOSUB 1500
2070 SL=1630:GOSUB 2700:REM-
     -BILDSCHIRMEINGABE
2080 V=VAL(A$)
2090 IF V <> X + Y THEN 2010
2100 GOSUB 2500
2110 RETURN
2500 REM-----LÖSCHUNG DES UNTE-
     REN SCHIRMTEILS--
2510 FOR SL = 1544 TO 1703
2520      POKE SL,32
2530 NEXT SL
2540 RETURN
2700 REM-------ERGEBNISEIN-
     GABE---------
2710 A$=""
2720 GET K$:IF K$="" THEN 2720
2730 IF ASC(K$)=13 THEN 2780
2740 A$=A$+K$
2750 POKE SL,ASC(K$)
```

```
2760 SL=SL+1
2770 GOTO 2720
2780 RETURN
3000 REM------PLATZEN DER SEIFEN-
     BLASE----------
3010 FOR SL = 1268 TO 1291
3020      POKE SL,32
3030      POKE SL+1,I
3040      FOR J = 1 TO 50:NEXT J
3050 NEXT SL
3060 RESTORE
3070 FOR J=1 TO 18
3080      READ SL:POKE SL,32
3090 NEXT J
3100 T$="RICHTIG!"
3110 SL=1293:GOSUB 1500
3120 RETURN
```

Variationen

Auch bei diesem Programm können Sie selbständig Änderungen vornehmen Sie könnten

zum Beispiel das Bildschirmformat durch detailliertere Zeichnung der Seifenblase verfeinern, das Spielfeld weiter ausschmücken oder die Farbe der Seifenblase verändern. (Schlagen Sie bitte in den Variationen des Kapitels 4 nach, dort finden Sie die Hinweise auf die Farbgestaltung.) An dieser Stelle wollen wir nur zwei grundsätzliche Änderungen erörtern:

1. Wir stellten vorhin fest, daß die Eingaberoutine keine Veränderung beziehungsweise Korrektur bereits eingegebener Zeichen zuläßt. Mit Hilfe einer Anzahl von Modifikationen kann dies jedoch geändert werden. Schon in der Routine wird geprüft, ob der Spieler die RETURN-Taste gedrückt hat. Es könnte aber auch eine Prüfung auf eine »Löschtaste« erfolgen. Sie können jede beliebige Taste zum Löschen bestimmen. Wir haben uns hier für das Sternchen entschieden. Das Drücken dieser Taste soll bewirken, daß das vorher eingegebene Zeichen gelöscht wird. Zwischen den Zeilen 2730 und 2740 fügen Sie bitte eine Zeile mit der Prüfung IF ASC (K$) = ASC ("*")...ein. Ist die Bedingung erfüllt, dann muß 1 von SL subtrahiert und mit POKE eine Leerstelle auf der Adresse SL ausgegeben werden. Anschließend wird zurückverzweigt zur Eingabe eines weiteren Zeichens. Mit diesem Hinweis sind Sie sicher in der Lage, das Problem zu lösen.

2. Stellen Sie anstatt einer Additions- eine Subtraktions-, Multiplikations- oder Divisionsaufgabe. Nur zwei Zeilen müssen in der FRAGESTELLUNG-Unterroutine verändert werden: Die Stelle, an der die Frage auf dem Bildschirm gezeigt

wird, und die Stelle, an der die Antwortüberprüfung erfolgt.

6
Verwirrte Staaten

Alle Techniken, die wir bis jetzt gelernt haben, plus einiger neuer, kombinieren wir, um ein noch umfangreicheres Spielprogramm herzustellen als diejenigen, die wir bislang durchgearbeitet haben. In diesem Spiel geht es darum, mit drei Bewegungen vom Start zum Ziel zu gelangen. Die Bewegung findet aber nur dann statt, wenn der Spieler die richtige Antwort auf die Frage gibt, um welchen europäischen Staat es sich handelt. Der Name des Staates wird ihm in völlig verdrehter Form auf dem Bildschirm gezeigt, und der Spieler muß schon einiges an Kombinationsfähigkeit aufbieten, um die Frage richtig zu beantworten. Wir geben dem Spieler insgesamt sechsmal die Chance, die drei Bewegungen aufgrund einer richtigen Eingabe durchzuführen. Wenn er oder sie damit die Ziellinie nicht erreicht, ist das Spiel beendet.
Die Namen der Variablen und die Liste der Unterroutinen sehen folgendermaßen aus:

```
 4 REM-----VERWIRRTE STAATEN--
   --------
10 REM SL     : BILDSCHIRMPOSITION
16 REM T$     : MELDUNGSTEXT
20 REM L$     : BUCHSTABE DES TEXTES
```

```
 24 REM A       : ASCII-CODEZAHL DES
    BUCHSTABEN
 32 REM J       : EIN ZÄHLER
 36 REM K$      : TASTENEINGABE
 40 REM R$      : EINE ANTWORT
 44 REM T$      : ANFANGSBUCHSTABE
    DES SPIELERS
 52 REM S$(10) : TABELLE FÜR ZEHN
    LÄNDERNAMEN
 56 REM U$(20): TABELLE DER BENUTZ-
    TEN BUCHSTABEN
 60 REM A$      : DIE ANTWORT
 64 REM N$      : NAME EINES LANDES
 68 REM T       : DURCHLAUFNUMMER
 72 REM P       : POSITION DES INITIALS
 80 REM O$      : SPIELENDEZEICHEN
 84 REM L       : LÄNGE DES NAMENS N$
 88 REM R       : ZUFALLSZAHL FÜR
    BUCHSTABEN
 92 REM————————————————
100 GOSUB 1000:REM———BILDSCHIRM-
    AUFBAU—————
200 GOSUB 2000:REM———STAATEN-
    TABELLE———
300 GOSUB 3000:REM———SPIEL—————
    ———
400 END
```

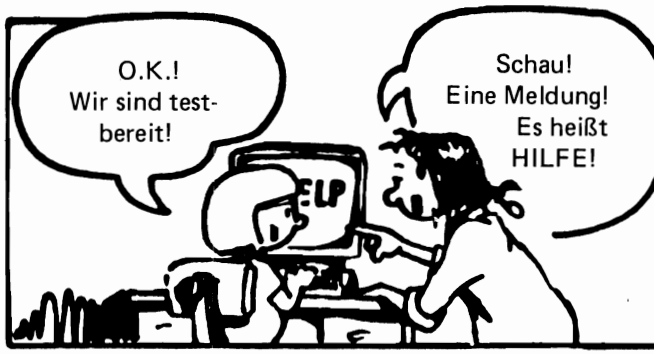
Wir verwenden in diesem Spiel weder neue BASIC-Abläufe noch neue Programmbefehle, sondern benutzen all unsere bisher erworbenen Kenntnisse in einer weitaus komplexeren Form. Sie können an der langen Liste der Variablen

schon erkennen, daß wir eine Vielzahl von Daten zu speichern haben. Wir sehen im Hauptprogramm nur drei Unterroutinen, obwohl wir viel mehr benötigen. Die hier sichtbaren Unterroutinen rufen nämlich ihrerseits weitere auf. Da wir insgesamt dann acht Routinen haben werden, ist es sinnvoll, diese aufzuzeigen, damit wir uns immer wieder darauf beziehen können:

1000 Bildschirmaufbau
1500 Textausgabe
2000 Ländertabelle
2300 Löschung des unteren Bildschirmteils
2700 Eingabe der Antworten
3000 Das Spiel
3500 Durcheinanderwirbeln eines Ländernamens
3800 Bewegung des Anfangsbuchstabens

Einige dieser Routinen werden Ihnen bekannt vorkommen. Mit der Eingabe, dem Löschen des Bildschirms, der Textausgabe, der Zeichenbewegung und dem Laden eines Textbereichs haben wir bereits gearbeitet. Diese Unterroutinen sind sehr nützlich — sie können als eine Art Bibliothek, auf die der Programmierer zur wiederholten Benutzung zurückgreift, bezeichnet werden. Die einzigen Programmteile, die wir noch nicht genau kennen, sind das Spiel und das »Durcheinanderwirbeln« eines Textes.

Der Bildschirmaufbau



Wir beginnen mit der Spielvorbereitung, indem wir auf dem Bildschirm drei Positionen markieren, die den jeweiligen Startpunkt für die drei Bewegungen darstellen, die der Anfangsbuchstabe des Spielers machen muß, um die Ziellinie zu erreichen. Auch in diesem Fall bleibt es Ihnen überlassen, den Bildschirm durch ein ausgetüfteltes »Spielbrett« zu verschönern. Sie können leicht alle möglichen dekorativen Elemente auf dem Bildschirm darstellen. Wir beschränken uns hier auf die Markierung der drei Startpositionen durch Pluszeichen (+), positionieren den Anfangsbuchstaben am Startpunkt der Bewegungslinie und erklären den Spielverlauf:

```
1000 REM———BILDSCHIRMAUFBAU——
     ————————
1010 POKE 53272,23
1020 POKE 53281,6:PRINT CHR$(147)
1030 POKE 53281,1
```

Wie in den letzten beiden Spielen wählen wir den zweiten Zeichensatz aus und bestimmen die Hintergrundfarbe Weiß (1) und die Farbe Blau (6) für die Zeichendarstellung.

```
1040 T$="START"
1050 SL=1187:GOSUB 1500
1060 T$="ZIEL"
1070 SL=1216:GOSUB 1500
1080 FOR SL=1268 TO 1298 STEP 10
1090       POKE SL,ASC("+")
1100 NEXT SL
```

Die Unterroutine in Zeile 1500 gibt uns wieder den Text auf den Bildschirm aus. Sie ist identisch mit der Routine 1500 aus Kapitel 5, wo sie mit Hilfe der Variablen T$ und SL für den Text und die Startpositionen verantwortlich war: Hier wird sie benutzt für die Anzeige der START- und ZIELposition in Zeile 5 und vier Pluszeichen in Zeile 7 des Bildschirms. Die Pluszeichen befinden sich auf den Adressen 1268, 1278, 1288 und 1298. Deshalb können wir eine Schleife in Zehnerschritten (STEP 10) ausführen. Die beiden Zeilen sehen nun so aus:

```
START                    ZIEL
  +       +       +       +
```

Nun wird der Anfangsbuchstabe des Spielernamens positioniert, und zwar mit den Befehlen GET und POKE:

```
1110 T$="GEBEN SIE BITTE IHREN AN-
     FANGSBUCHSTABEN EIN: "
1120 SL=1544:GOSUB 1500
1130 GET I$:IF I$ = " " THEN 1130
1140 POKE 1268,ASC(I$)
```

Am Ende der Unterroutine geben wir jetzt noch eine Erklärung des Spielablaufs beziehungsweise der Regeln auf dem Bildschirm aus. Eigentlich benötigen wir nun pro Textzeile zwei Befehlszeilen: Eine speichert den Text in T$, die nächste bestimmt die Bildschirmadresse und verzweigt auf 1500. Wir werden es aber mit einer optimaleren Programmierung, die kürzer ist und in der DATA-Anweisung Verwendung findet, versuchen:

```
1150 DATA "SIE HABEN 6 VERSUCHE, UM
     ANS ZIEL ZU GELANGEN."
1160 DATA "UM SICH ZU BEWEGEN, MÜS-
     SEN SIE DEN"
1170 DATA "NAMEN EINES STAATES IN EU-
     ROPA ENTZIFFERN."
1180 DATA "DRÜCKEN SIE BITTE ZUM START
     DIE RETURN-TASTE:"
1190 FOR SL=1624 TO 1744 STEP 40
1200       READ T$:GOSUB 1500
1210 NEXT SL
```

Die Schleife versieht SL mit Werten und wird in Vierzigerschritten durchgeführt, da jede Bildschirmzeile vierzig Stellen hat. Beim ersten Durchlauf hat SL also den Wert 1624, und wir lesen mit READ in T$ die erste DATA-Definition ein,

T$ wird dann auf Adresse 1624 ausgegeben. Beim zweiten Mal ist in SL 1664 gespeichert, und an dieser Stelle wird der zweite DATA-Eintrag angezeigt. Auf diese Art und Weise werden die Texte in aufeinanderfolgenden Zeilen sichtbar.

```
1220 GET R$:IF R$="" THEN 1220
1230 RETURN
```

Die letzte Textzeile verlangte vom Spieler das Drücken der RETURN-Taste. In der Zeile 1220 empfängt das Programm ein Zeichen. Es interessiert uns allerdings nicht, welche Taste gedrückt wird, denn die Unterroutine wird nach Drücken einer beliebigen Taste beendet.

Eingabe und Ausgabe

Die hier benötigten Unterroutinen sind identisch mit denen, die wir im letzten Spiel verwendet haben. Wir zeigen sie nur noch einmal der Vollständigkeit halber.

```
1500 REM———————TEXTAUSGABE———
     ———————————
1510 FOR J=1 TO LEN(T$)
1520      L$=MID$(T$,J,1)
1530      A=ASC(L$)
1540      POKE SL+J—1,A
1550 NEXT J
1560 RETURN
2700 REM———ERGEBNISEINGABE———
     ———————
2710 A$=" "
2720 GET K$:IF K$=" " THEN 2720
2730 IF ASC(K$)=13 THEN 2780
2740 A$=A$+K$
2750 POKE SL,ASC(K$)
2760 SL=SL+1
2770 GOTO 2720
2780 RETURN
```

Jetzt wollen wir eine Zeile einfügen, damit der Spieler Gelegenheit hat, bei fehlerhafter Eingabe die Zeichen zu korrigieren. (Dies haben wir schon in Kapitel 5 besprochen.) Wir nehmen also das Sternchen als Löschsymbol. In Zeile 2730 wird geprüft, ob der Spieler die RETURN-Taste gedrückt hat. Ist das nicht der Fall, müssen wir feststellen, ob die Eingabe ein Sternchen ist. Trifft das zu, subtrahieren wir 1 von SL, um eine Bild-

schirmposition zurückgehen zu können, löschen diese Stelle mit POKE und einem Leerwert und verzweigen zurück zur Eingabe des nächsten Zeichens:

```
2735 IF ASC(K$)=ASC("*") THEN SL=SL—1:
     POKE SL,32:GOTO 2720
```

Wenn Sie diese Zeile in das Programm einfügen, sollten Sie allerdings die Erweiterung in Ihre Spielregeln aufnehmen und dem Spieler mitteilen, daß das Sternchen eine Löschfunktion in bezug auf das zuletzt eingegebene Zeichen hat.

Die Speicherung der Staaten

Können Sie sich noch an die Unterroutine in Kapitel 1 erinnern, mit der wir mehrere Zukunftsvoraussagen in einer Tabelle gespeichert haben?

In diesem Fall handelt es sich um zehn Staaten, deren Namen wir in DATA-Anweisungen angeben. Beachten Sie bitte, daß pro DATA-Zeile mehrere Dateneinträge, das heißt Ländernamen dargestellt werden können, solange die einzelnen Zeichenketten durch Kommata getrennt werden. Wir nehmen auch hier den READ-Befehl, um die Ländernamen in eine Tabelle zu laden, die wir mit S$ (S als Abkürzung für Staat) bezeichnet haben. Bevor wir diese Tabelle jedoch benutzen können, müssen wir sie DIMensionieren — das heißt, der Computer wird angewiesen, zehn Leerbereiche für unsere Tabelle zu reservieren.

```
2000 REM———STAATENTABELLE————
     ——————————————
2010 DATA "BUNDESREPUBLIK","ENG-
     LAND","FRANKREICH","HOLLAND",
     "LUXEMBURG","BELGIEN","SPANIEN"
2020 DATA "ITALIEN","GRIECHENLAND"
2030 DATA "ÖSTERREICH"
2040 DIM S$(10)
2050 FOR J = 1 TO 10
2060      READ S$(J)
2070 NEXT J
2080 RETURN
```

Die Schleife in den Zeilen 2050 bis 2070 bewirkt zehn Lesevorgänge (READs) zur Speicherung der zehn Ländernamen in unserer Tabelle. Der Name "BUNDESREPUBLIK" wird damit auf Platz 1 der Tabelle, also S$(1), der Name "FRANKREICH" auf S$(3) und der Name "ÖSTERREICH"

auf S$(10) gespeichert. Im weiteren Verlauf des Programms werden wir diese Tabelle in derselben Weise verwenden wie die Weissagungstabelle. Wir greifen eine Zufallszahl auf und benutzen diese zur Auswahl einer der Datenzellen. Anschließend wird der in dieser Zeile gespeicherte Ländername herausgenommen und durcheinandergeschüttelt.

Das eigentliche Spiel

Wir haben vorhin festgelegt, daß der Spieler sechs Möglichkeiten erhält, die Ziellinie zu erreichen. Jede Runde besteht aus dem Versuch, einen zufällig angezeigten Ländernamen zu entziffern, das heißt, die Buchstaben in die richtige Reihenfolge zu bringen. Wir müssen unsere Aufmerksamkeit also noch auf weitere Dinge richten:

1. wo sich der Anfangsbuchstabe zur Zeit befindet,

2. ob der Anfangsbuchstabe inzwischen die Ziellinie erreicht hat,

3. Wie viele Versuche der Spieler bereits hinter sich hat.

Die SPIEL-Unterroutine ist ziemlich komplex. Sie liest sich wie ein eigenständiges kleines Programm und enthält selbst die Verzweigung auf weitere Unterroutinen. Bevor wir uns das in BASIC anschauen, wollen wir uns in allgemeinverständlicher deutscher Sprache die Schritte verdeutlichen, die wir durchführen müssen:

Überprüfung, ob der Spieler schon gewonnen hat ODER die maximal mögliche Anzahl der Versuche überschritten ist.

Wenn eine dieser beiden Bedingungen zutrifft, geben wir die entsprechende Meldung aus.

Wenn keine von beiden erfüllt ist, führen wir folgende Aktionen durch:

1. Löschung des unteren Bildschirmteils,

2. per Zufallsgenerator einen Ländernamen aufgreifen,

3. die Buchstaben des ausgewählten Ländernamens mischen,

4. den Spieler zur Entschlüsselung oder Richtigstellung auffordern,
5. ermitteln, ob die eingegebene Antwort richtig oder falsch ist;
a) wenn sie richtig ist, den Anfangsbuchstaben um zehn Stellen weiterbewegen und überprüfen, ob der Spieler nun gewonnen hat;
b) wenn sie NICHT richtig ist, dem Spieler den richtigen Ländernamen mitteilen;
c) in jedem Fall addieren wir eine 1 zu der Zahl der bereits erfolgten Versuche und kehren zurück zum ersten Schritt.
Überprüfung, ob der Spieler gewonnen hat...)

Die Zeilen 3030 bis 3170 sind die gleichen Schritte in BASIC. Die Zeilen 3010 und 3020 initialisieren vier notwendige Variablen. U$ (20) ist ein Bereich, den wir bei der Veränderung des Ländernamens benötigen. T steht für die Anzahl der Versuche und wird zu Beginn des Spiels auf 1 gesetzt, da wir uns ja im ersten Versuch befinden. P steht für Position und enthält 1268 als erste Bildschirmadresse des Anfangsbuchstabens des Spielers. O$ sagt uns, ob das Spiel beendet ist, weil der Spieler gewonnen hat. Dieses Feld erhält als Anfangswert den Text »NEIN«.

```
3000 REM————SPIEL————————————
     ——
3010 DIM U$(20)
3020 T = 1:P = 1268:O$ = "NEIN"
3030 IF T › 6 OR O$ = "JA" THEN 3180
3040    GOSUB 2300:REM—
        ——TEILLÖSCHUNG DES SCHIRMS
```

```
3050    GOSUB 3500:REM——MIX DES
        LÄNDERNAMENS
3060    T$="BITTE EINGABE DER
        ANTWORT:"
3070    SL=1664:GOSUB 1500
3080    SL=1708:GOSUB 2700:REM——EIN-
        GABE
3090    SL=1784
3100    IF A$ = N$ THEN 3140
3110        T$="SCHADE! DER STAAT IST
            "+N$
3120        GOSUB 1500
3130        GOTO 3160
3140        T$="RICHTIG! ":GOSUB 1500
3150        GOSUB 3800:REM——
            ANIMATION
3160    T = T + 1
3170    GOTO 3030
3180 GOSUB 2300:REM——TEILLÖSCHUNG
     DES SCHIRMS
3190 SL=1544
3200 IF O$ = "JA" THEN 3230
3210    T$="SCHADE, DIE VERSUCHE SIND
        BEENDET!"
3220    GOSUB 1500:GOTO 3250
3230    T$="GLÜCKWUNSCH! SIE HABEN
        GEWONNEN!"
3240    GOSUB 1500
3250 RETURN
```

Zeile 3030 prüft ZWEI Bedingungen: Ob die maximale Versuchsanzahl erreicht ist oder ob der Spieler gewonnen hat, in beiden Fällen springt

das Programm zur Zeile 3180. Ist das Spiel nicht vorbei, wird der untere Bildschirmteil gelöscht (Unterroutine 2300), ein neuer Ländername aufgegriffen, gemixt und die Fragestellung auf dem Bildschirm ausgegeben (Routine 3500), dann wird die Antwort eingegeben (2700) und geprüft. Wenn A$ (die Antwort) dem Inhalt von N$ (dem eigentlichen Namen) entspricht, hat der Spieler recht, und wir geben den Text RICHTIG! aus, andernfalls erhält er die Nachricht SCHADE! DER STAAT IST und die Länderbezeichnung.
Früher oder später, wenn der Spieler entweder gewonnen oder zu viele Versuche unternommen hat, verzweigt das Programm zur Zeile 3180, wo der untere Bildschirmteil gelöscht wird und die Entscheidung fällt, ob der Spieler gewonnen hat oder nicht. Wenn O$ den Wert JA enthält, bedeutet das, der Spieler hat gewonnen, also wird die Glückwunschnachricht angezeigt. Wenn O$ jedoch nicht JA enthält, muß der Spieler die Anzahl der möglichen Versuche ausgeschöpft haben, und der entsprechende Text wird ausgegeben.

Das Löschen des unteren Bildschirmbereichs

Diese Unterroutine ist größtenteils identisch mit der entsprechenden aus Kapitel 5, die den unteren Bildschirmteil löschen soll. An dieser Stelle werden wir sieben Zeilen des Schirms mit Leerwerten versehen:

```
2300 REM———TEILLÖSCHUNG DES BILD-
     SCHIRMS—————————
2310 FOR SL = 1554 TO 1823
2320      POKE SL,32
2330 NEXT SL
2340 RETURN
```

Das Durcheinanderwirbeln des Ländernamens

Wir haben eine Tabelle, bestehend aus zehn Staaten, die im Programm mit dem Symbol S$(10) bezeichnet ist. Immer wenn der Zeitpunkt für eine erneute Fragestellung gekommen ist, wollen wir einen Ländernamen aufgreifen und die Zeichenkette durcheinanderbringen. Das erste, was bekannt sein muß, ist die Anzahl der Buchstaben des herausgegriffenen Ländernamens, den wir mit N$ bezeichnet haben. Die LEN-Funktion, so erinnern wir uns, gibt uns über die Gesamtzahl der Zeichen in einer Kette Auskunft. Also stellt LEN(N$) die Anzahl der Zeichen des aktuellen Ländernamens dar. Wir reden von Zeichen und nicht von Buchstaben, da die Namen ja auch Bindestriche oder Leerstellen enthalten können.
Die Zeichenanzahl im Namen markieren wir mit dem Variablennamen L. Um sie zu mischen, müssen wir eine Serie von Schritten auslösen:

1. Aufgreifen einer Zufallsganzzahl (genannt R) zwischen 1 und L,
2. Ausgabe des Zeichens mit der Positionsnummer R auf dem Bildschirm,
3. Aufgreifen einer weiteren Zufallsganzzahl R zwischen 1 und L,
4. Ausgabe eines weiteren Zeichens mit der Positionsnummer R als Folgezeichen unmittelbar hinter dem bereits angezeigten Zeichen.

Werden die oben angeführten Schritte insgesamt L mal wiederholt, dann erscheinen L-Zeichen auf dem Bildschirm. Diese Schrittfolge enthält ein Problem, auf das wir Sie noch nicht aufmerksam gemacht haben. Sie müßten nun wissen, daß der Computer bei der Bereitstellung einer Zufallszahl zwischen 1 und 7 beispielsweise mehrfach dieselbe Zahl angeben könnte. Wenn also HOLLAND der zu erratende Staat ist, und der Computer gibt ständig die 4 an, würde die gemixte Ausgabe von HOLLAND aussehen wie »LLLLLLL«. Und das würde natürlich den weiteren Spielverlauf erheblich behindern.
Aus diesem Grund muß ein Weg gefunden werden, den Computer anzuweisen, die Zufallszahl aus dem Bereich 1 bis 7 zu bilden, aber keine Zahl zweimal zu verwenden. Mit anderen Worten, wir müssen die schon verwendeten Zahlen aufzeichnen. Die beste Art, das zu tun, besteht aus der Festlegung einer weiteren Tabelle, die wir mit U$ bezeichnen. Bevor wir mit dem Mix beginnen, speichern wir in jeder Zelle von U$ das Wort »NEIN« ab. Es zeigt uns an, daß noch keine

Zahl benutzt wurde. Wenn nun eine Zahl aufgegriffen wurde, speichern wir in der zugeordneten Zelle das Wort »JA« ab. Übrigens DIMensionieren wir die Tabelle zu Beginn der SPIEL-Unterroutine mit dem Wert 30. Nun wollen wir unsere Schrittfolge diesen Überlegungen anpassen und sehen, wie sich diese Ergänzung auswirkt:

1. Aufgreifen einer Zufallszahl zwischen 1 und L,

2. Überprüfung, ob diese Zahl schon verwendet wurde:

a) wenn ja, zurück zu Schritt 1.

b) wenn nein, Speicherung von »JA« in der Zelle von U$, die der Zufallszahl entspricht, und Ausgabe des entsprechenden Zeichens aus dem Ländernamen.

Wiederholung der Schrittfolge L mal.

Der ganze Bereich sieht dann in BASIC so aus:

```
3500 REM ———MIX DER LÄNDER-
     NAMEN————————
3510 R = INT(10 * RND(1)) + 1
3520 N$ = S$(R)
3530 L = LEN(N$)
3540 FOR J = 1 TO L
3550      U$(J) = "NEIN"
3560 NEXT J
3570 T$="WELCHER STAAT IST DAS?"
3580 SL=1544:GOSUB 1500
3590 FOR J = 1 TO L
3600      R = INT(L * RND(1)) + 1
3610      IF U$(R) = "JA" THEN 3600
3620      U$(R) = "JA"
```

```
3630      A=ASC(MID$(N$,R,1))
3640      POKE J+1587,A
3650 NEXT J
3660 RETURN
```

R ist eine Zufallsganzzahl von 1 bis 10. Warum 10? Weil wir genau zehn Staaten haben. Wenn R eine 1 enthält, dann ist S$(R) gleich BUNDESREPUBLIK, also der erste Staat in der DATA-Auflistung. Im Programm leichter verständlich ist die Speicherung von S$(R) in N$. Wenn N$ den Namen FRANKREICH enthält, hat L den Wert 10, nämlich die Anzahl aller Zeichen in FRANKREICH.
Die Schleife in den Zeilen 3540 bis 3560 initialisiert die Tabelle U$. Sie bewirkt den Eintrag von »NEIN« in den 10 Zellen von U$. Also enthalten U$(1), U$(2), U$(3) und so weiter bis U$(10) zu Beginn das Wort »NEIN«. Um bei unserem Beispiel zu bleiben: Das Wort FRANKREICH besteht aus zehn Buchstaben und alle wurde noch NICHT benutzt.
Nach der Frage WELCHER STAAT IST DAS? gelangen wir in den Teil der Unterroutine, der tatsächlich die Mischung der Zeichenfolge vornimmt und diese dann in verdrehter Reihenfolge ausgibt. Die FOR/NEXT-Schleife in den Zeilen 3590 bis 3650 wird L mal durchgeführt. In unserem Beispiel mit FRANKREICH also zehnmal — aufgrund der zehn Buchstaben.
Jedesmal, wenn nun eine Zufallszahl von 1 bis 10(L) aufgegriffen wird, erfolgt also die Prüfung, ob die entsprechende Zelle von U$ schon ein »JA«

enthält, weil die Zahl bereits verwendet wurde. In diesem Fall kehrt das Programm zur Auswahl zurück, bis es eine Zahl findet, in deren zugeordneter Zelle das Wort »NEIN« enthalten ist. Daraufhin wird es den Befehl in Zeile 3610 überpringen und mit der Verarbeitung zweier Anweisungen fortfahren. Erstens wird in der Zelle ein »JA« eingetragen, so daß derselbe Buchstabe nicht zweimal durch den Computer ausgegeben wird. Zweitens wird durch die Benutzung der MID$-Funktion das der Zufallszahl zugeordnete Zeichen aus N$ ausgegeben.
Die Zeile 3630 definiert eine Codezahl (A) für das Zeichen, das wir nun ausgeben wollen. Wenn R die Zahl ist, die eine Zeichenposition in N$ beziffert, dann ist MID$(N$,R,1) genau das Zeichen. Deshalb nehmen wir in Zeile 3630 die ASCII-Codezahl des Zeichens auf und geben diese in Zeile 3640 auf der gewünschten Adresse aus. Die Formel J+1587 führt zur Ausgabe ab Speicherstelle 1588, da J den Initialwert 1 enthält, und fährt mit der Anzeige der Zeichen nebeneinander fort, bis L-Zeichen sichtbar geworden sind.
Es wäre sinnvoll, wenn Sie an dieser Stelle die einzelnen Schritte der Unterroutine mehrmals für sich nachvollziehen, um zu sehen, was jeder Schritt im einzelnen bewirkt. Vergegenwärtigen Sie sich, daß bei jedem Aufruf dieser Routine ein neuer Ländername, hier genannt N$, durch den mehrmaligen Durchlauf einer Schleife per Zufallsgenerator gemischt und verkehrt ausgegeben wird. Diese Unterroutine dient nur diesem einen Zweck. Blättern Sie noch einmal zur ei-

gentlichen SPIEL-Unterroutine zurück. Genau vor dem Aufruf der Mixroutine löschen wir den unteren Bildschirmbereich. Unmittelbar danach gibt der Spieler seine Antwort ein, die dann auf Richtigkeit überprüft wird. Wenn diese stimmt, müssen wir mit einem weiteren Ablauf beginnen.

Die Bewegung des Anfangsbuchstabens

Jede richtige Antwort bringt den Anfangsbuchstaben dem Ziel einen Schritt näher. Wir haben Pluszeichen zur Markierung dieser Schritte auf dem Bildschirm ausgegeben, und zwar in einem Abstand von 10 Spalten auf den Positionen 1278, 1288 und 1298 (dem Ziel). Erinnern wir uns auch daran, daß wir zu Beginn der SPIEL-Unterroutine die Variable P mit 1268 initialisiert haben, um die Startposition des Anfangsbuchstabens aufzuzeichnen.

Zur Bewegung des Initials durchläuft unser Programm die gleichen Basisschritte, die wir in den letzten beiden Programmen schon verwendet haben:
1. Löschung des Zeichens auf seiner zuletzt erreichten Position,
2. Ausgabe des Zeichens auf der nächsten Position.
Wir wiederholen diese Schritte zehnmal, um den Anfangsbuchstaben um zehn Spalten vorwärtszubewegen. Dann tun wir zwei weitere wichtige Dinge. Wir addieren die Zahl 10 auf P, um die neue Position des Zeichens festzuhalten, und wir prüfen, ob P den Wert 1298 erreicht hat. Falls das zutrifft, heißt das, der Spieler hat gewonnen, und wir speichern das Wort »JA« in der Variablen O$.

```
3800 REM———ANIMATION————————————
     ——————
3810 FOR SL = P TO P + 9
3820       POKE SL,32
3830       POKE SL+1,ASC(I$)
3840       FOR J = 1 TO 50:NEXT J
3850 NEXT SL
3860 P = P + 10
3870 IF P = 1298 THEN O$ = "JA"
3880 RETURN
```

Die Bewegungstechnik ist die gleiche wie im letzten Kapitel.
Wir setzen einen Leerwert, wo sich das Initial befindet, und geben den Buchstaben eine Stelle

weiter wieder aus, bis wir ihn um zehn Stellen fortbewegt haben.
O$ wurde am Anfang des Spiels auf den Wert »NEIN« gesetzt. Sobald P den Wert 1298 annimmt, wird in O$ ein »JA« gespeichert. Zurück in der SPIEL-Abteilung, trifft nun die Bedingung in Zeile 3030 zu. Dadurch wird das Programm die nächste Schleife überspringen. Wenn es in der Zeile 3200 anlangt, wird der Text »GRATULATION!. . .« ausgegeben, da O$ dem Wort »JA« entspricht, und das Spiel ist somit beendet.

Das gesamte Programm

Das Spiel »Verwirrte Staaten« ist bei weitem das längste, das wir geschrieben haben. Aber es wird nichts benutzt, das Sie nicht schon kennen. Und jetzt wollen wir Ihnen nochmals das gesamte Programm vorstellen:

```
  4 REM————VERWIRRTE STAATEN—
    ————————
 10 REM SL      : BILDSCHIRMPOSITION
 16 REM T$      : MELDUNGSTEXT
 20 REM L$      : BUCHSTABE DES TEX-
    TES
 24 REM A       : ASCII-CODEZAHL DES
    BUCHSTABEN
 32 REM J       : EIN ZÄHLER
 36 REM K$      : TASTENEINGABE
 40 REM R$      : EINE ANTWORT
```

```
  44 REM I$       : ANFANGSBUCHSTABE
     DES SPIELERS
  52 REM S$(10) : TABELLE FÜR ZEHN
     LÄNDERNAMEN
  56 REM U$(20): TABELLE FÜR BENUTZ-
     TEN BUCHSTABEN
  60 REM A$       : DIE ANTWORT
  64 REM N$       : NAME EINES LANDES
  68 REM T        : DURCHLAUFNUMMER
  72 REM P        : POSITION DES INITIALS
  80 REM O$       : SPIELENDEZEICHEN
  84 REM L        : LÄNGE DES NAMENS
     N$
  88 REM R        : ZUFALLSZAHL FÜR
     BUCHSTABEN
  92 REM———————————————
 100 GOSUB 1000:REM—
     ——BILDSCHIRMAUFBAU——————
 200 GOSUB 2000:REM———STAATEN-
     TABELLE————————
 300 GOSUB 3000:REM———SPIEL————
     ————————
 400 END
1000 REM—————BILDSCHIRMAUFBAU—
     ———————————————
1010 POKE 53272,23
1020 POKE 53281,6:PRINT CHR$(147)
1030 POKE 53281,1
1040 T$="START"
1050 SL=1187:GOSUB 1500
1060 T$="ZIEL"
1070 SL=1216:GOSUB 1500
1080 FOR SL=1268 TO 1298 STEP 10
```

```
1090      POKE SL,ASC("+")
1100 NEXT SL
1110 T$="GEBEN SIE BITTEN IHREN AN-
     FANGSBUCHSTABEN EIN:"
1120 SL=1544:GOSUB 1500
1130 GET I$:IF$ = " " THEN 1130
1140 POKE 1268,ASC(I$)
1150 DATA "SIE HABEN 6 VERSUCHE, UM
     ANS ZIEL ZU GELANGEN."
1160 DATA "UM SICH ZU BEWEGEN, MÜS-
     SEN SIE DEN"
1170 DATA "NAMEN EINES STAATES IN EU-
     ROPA ENTZIFFERN."
1180 DATA "DRÜCKEN SIE ZUM START BIT-
     TE DIE RETURN-TASTE:"
1190 FOR SL=1624, TO 1744 STEP 40
1200      READ T$:GOSUB 1500
1210 NEXT SL
1220 GET R$:IF R$=" " THEN 1220
1230 RETURN
1500 REM————TEXTAUSGABE———
     ——————
1510 FOR J=1 TO LEN(T$)
1520      L$=MID$(T$,J,1)
1530      A=ASC(L$)
1540      POKE SL+J-1,A
1550 NEXT J
1560 RETURN
2000 REM———STAATENTABELLE————
     ——————
2010 DATA "BUNDESREPUBLIK","ENG-
     LAND","FRANKREICH","HOLLAND",
     "LUXEMBURG","BELGIEN","SPANIEN"
```

```
2020 DATA "ITALIEN","GRIECHENLAND"
2030 DATA "ÖSTERREICH"
2040 DIM S$(10)
2050 FOR J = 1 TO 10
2060      READ S$(J)
2070 NEXT J
2080 RETURN
2300 REM———TEILLÖSCHUNG DES BILD-
     SCHIRMS—————————
2310 FOR SL = 1544 TO 1823
2320      POKE SL,32
2330 NEXT SL
2340 RETURN
2700 REM————ERGEBNISEINGABE——
     ———————
2710 A$=" "
2720 GET K$:IF K$=" " THEN 2720
2730 IF ASC(K$)=13 THEN 2780
2735 IF ASC(K$)=ASC("*") THEN SL=SL-1:
     POKE SL,32:GOTO 2720
2740 A$=A$+K$
2750 POKE SL,ASC(K$)
2760 SL=SL+1
2770 GOTO 2720
2780 RETURN
3000 REM———SPIEL————————
3010 DIM U$(20)
3020 T = 1:P = 1268:O$ = "NEIN"
3030 IF T › 6 OR O$ = "JA" THEN 3180
3040      GOSUB 2300:REM——
          —TEILLÖSCHUNG DES SCHIRMS
3050      GOSUB 3500:REM——MIX DES
          LÄNDERNAMENS
```

```
3060    T$="BITTE EINGABE DER ANT-
        WORT:"
3070    SL=1664:GOSUB 1500
3080    SL=1708:GOSUB 2700:REM——
        EINGABE
3090    SL=1784
3100    IF A$ = N$ THEN 3140
3110        T$="SCHADE! DER STAAT IST"
            +N$
3120        GOSUB 1500
3130        GOTO 3160
3140        T$="RICHTIG!" :GOSUB 1500
3150        GOSUB 3800:REM——
            ANIMATION
3160    T = T + 1
3170    GOTO 3030
3180 GOSUB 2300:REM——TEILLÖSCHUNG
     DES SCHIRMS
3190 SL=1544
3200 IF O$ = "JA" THEN 3230
3210    T$="SCHADE, DIE VERSUCHE SIND
        BEENDET!"
3220    GOSUB 1500:GOTO 3250
3230    T$="GLÜCKWUNSCH! SIE HABEN
        GEWONNEN!"
3240    GOSUB 1500
3250 RETURN
3500 REM———MIX DER LÄNDERNA-
     MEN————————
3510 R = INT(10 * RND(1)) + 1
3520 N$ = S$(R)
3530 L = LEN(N$)
3540 FOR J = 1 TO L
```

```
3550      U$(J) = "NEIN"
3560 NEXT J
3570 T$="WELCHER STAAT IST DAS?"
3580 SL=1544:GOSUB 1500
3590 FOR J = 1 TO L
3600      R = INT(L * RND(1)) + 1
3610      IF U$(R) = "JA" THEN 3600
3620      U$(R) = "JA"
3630      A=ASC(MID$(N$,R,1))
3640      POKE J+1587,A
3650 NEXT J
3660 RETURN
3800 REM———ANIMATION————————
     ——
3810 FOR SL = P TO P + 9
3820      POKE SL,32
3830      POKE SL+1,ASC(I$)
3840      FOR J = 1 TO 50:NEXT J
3850 NEXT SL
3860 P = P + 10
3870 IF P = 1298 THEN O$ = "JA"
3880 RETURN
```

An dieser Stelle liegen nun alle Variationen bei Ihnen. In den ersten fünf Kapiteln haben wir uns noch erlaubt, eigene Verbesserungen vorzuschlagen. Sie können fast alle Hinweise auch in diesem Spiel verwirklichen, wenn Sie wollen. Aber wichtiger für Sie ist, sich eigene Möglichkeiten der Verbesserung auszudenken und in die Tat umzusetzen. Gleiches gilt für die Erfindung weiterer Spiele. Entwerfen Sie sich Ihre eigenen Codierungsroutinen. Erfinden Sie weitere Wege, die Random-Funktion auszunutzen. Gestalten Sie die Animation nach eigenem Geschmack so phantasievoll, wie Sie nur können.
Wir haben eine Vielzahl von Kommandos und Ausstattungen des Commodore 64 erarbeitet, es bleibt jedoch für Sie noch genug zu lernen und zu entdecken. Insbesondere eröffnet sich Ihnen die ganze Welt der Farbenspiele. Sie können so-

gar den Computer und Ihre Programme mit Tönen ausstatten.
Programmierung ist eine Kombination aus der Benutzung dessen, was Sie bereits erlernt haben und beherrschen, und der Nutzung der eigenen Vorstellungskraft. Wir haben versucht, Sie mit einigen wesentlichen Abläufen und Techniken vertraut zu machen. Natürlich müssen Sie immer noch eine Menge dazulernen — das gilt sogar für die erfahrensten Programmierer. Wir hoffen, daß Sie wenigstens einen Teil dessen, was wir Ihnen vorgeführt haben, zur Erfindung weiterer spannender Spiele verwenden können und wir Ihnen einen Anreiz zum zukünftigen Ausbau Ihrer Kenntnisse bieten konnten. Das Wichtigste, was Sie dazu beitragen können, ist der eigene Spaß an der Programmierung.

Kleines Lexikon und Index

Ein Verzeichnis der in diesem Buch behandelten neuen Abläufe und Fachausdrücke mit Querverweisen auf die Kapitel, in denen sie angesprochen wurden. In BASIC reservierte Begriffe sind in Großbuchstaben dargestellt.

NAME *BEDEUTUNG UND VERWENDUNG*

ARRAY
: deutsch: Tabelle, ein Variablentyp, in dem mehr als ein Datenelement gespeichert und unter Verwendung desselben Namens erarbeitet wird. (1,3,6)

Animation
: bewirkt die scheinbare »Bewegung« irgendeines Objekts, beispielsweise eines Zeichens über den Bildschirm durch schnelles Ausgeben und Löschen auf aufeinanderfolgenden Bildschirmpositionen. (4 — 6)

ASC-Funktion
: setzt ein gegebenes Zeichen in die ASCII-Codezahl um. Beispiel: PRINT ASC ("A") bewirkt die Ausgabe der Codezahl 65. (3)

CHR$-Funktion
bewirkt die Umsetzung einer gegebenen ASCII-Codezahl in das zugeordnete Zeichen. Beispiel: PRINT CHR$(65) bewirkt die Ausgabe des Buchstabens A. (3)

DIM
ein Kommando, das eine gegebene Anzahl von Datenzellen für eine Tabelle reserviert. Beispiel: DIN M$(15) legt 15 Datenzellen im Speicher für die Tabelle M$ fest. (1,3,6)

FOR/NEXT
FOR und NEXT werden am Anfang und am Ende einer Wiederholungsschleife plaziert. (1 — 6)
Rückwärtszählung durch Benutzung negativer Schritte (2)
Leere Schleifen für die Programmpause (4 — 6)

GET
ein Eingabekommando, das nur ein einziges Zeichen oder eine einzige Ziffer als Eingabe zuläßt. (4 — 6)

GOSUB
(siehe Unterroutine)

Initialisierung
die Festlegung und Speicherung von Anfangs- oder Initialisierungswerten. (1 — 6)

INT-Funktion
rundet jede Zahl auf die nächstniedrigere Ganzzahl ab. Wird in Verbindung mit der RND(1)-Funktion angewendet. (1 — 6)

LEN-Funktion
stellt die Gesamtanzahl von Zeichen in einer gegebenen Zeichenkette bereit. Beispiel: LEN(»HALLO«) ergibt eine 5. (2 — 3)

MID$-Funktion
bewirkt die Wiedergabe einer definierten Zeichenkette innerhalb einer gegebenen Zeichenkette. Beispiel: MID$(»COMMODORE«,4,2) beginnt beim vierten Zeichen, gibt also MO wieder. (2 — 3)

ON/GOSUB
unterstützt ein Auswahlverfahren für die Durchführung von Unterroutinen. (3)

POKE
eröffnet die Möglichkeit der direkten Adressierung von Speicherstellen und Abspeicherung einzelner Zeichen. (4 — 6)

random
Die RND(1)-Funktion wählt Zufallszahlen, wie es in Kapitel 1 erklärt wurde, aus. Wir haben diese Funktion wie folgt genutzt: Zufallszahlen in Tabellenbereichszellen. (1,3,6) Zufallszeichenzahlen in Verbindung mit der CHR$-Funktion. (3)

Zufallszahl für ein ON/GOSUB-Kommando. (3)
Zufallszahl für die Bewegung auf dem Bildschirm. (4)
Zufallszahlen für arithmetische Probleme in Verbindung mit der MID$-Funktion zur Darstellung eines Zufallszeichens in einer Zeichenkette. (6)

READ/DATA
Das READ-Kommando stellt Daten aus DATA-Anweisungen, wie in Kapitel 1 erklärt, bereit. Es wurde benutzt in Verbindung mit: Daten in Zeichenketten. (1,3,6)
Positionsdaten für den Bildschirmaufbau. (5)

RETURN (siehe Unterroutine)

RESTORE
erneute Positionierung auf den Beginn der DATA-Anweisungen. (5)

STR$
Umsetzung eines Zahlenfeldes in ein Zeichenkettenfeld. (5)

Unterroutine
Bezeichnung für einen Programmteil oder -baustein, der durch GOSUB aufgerufen wird. Unterroutinen müssen mit einem RETURN-Kommando abgeschlossen werden. (1 — 6)

VAL

Umsetzung eines Zeichenkettenfeldes in ein Zahlenfeld. (5)

Computer / Neue Medien

Als Band mit der Bestellnummer 63073 erschien:

Kommunikationsfähige Textautomaten im Büro lösen allmählich die Schreibmaschinen ab. Die elektronische Datenverarbeitung nimmt mehr und mehr eine Schlüsselfunktion im Informationswesen ein. Satelliten und Netzwerke lassen ein weltumspannendes Kommunikationsnetz enststehen. Wir stehen an der Schwelle eines Zeitalters der Telematik (= Telekommunikation und Informatik).
Der Autor beschreibt die Entwicklungen der neuen Medien und analysiert die damit verbundenen möglichen gesellschaftlichen und arbeitsmarktpolitischen Auswirkungen. Zahlreiche Tabellen und Grafiken erleichtern den Einstieg in diese zum Teil hochkomplizierte Materie.
Ein sachlicher und verständlicher Beitrag zur aktuellen Mediendiskussion.

BASTEI
LÜBBE

Bildschirmtext

Als Band mit der Bestellnummer 63077 erschien:

Hinter dem neuen Medium Bildschirmtext (Btx) stecken nahezu unbegrenzte Kommunikationsmöglichkeiten. Von zu Hause oder vom Arbeitsplatz aus hat man Zugang zu den verschiedensten Informationen, kann Datenverarbeitungsanlagen nutzen oder Nachrichten absenden. Das Telefonnetz ist die Basis für den preiswerten Datentransport, der Fernseher erlaubt die Wiedergabe dieser Daten überall dort, wo es gewünscht wird.
Eine allgemeinverständliche Darstellung der Möglichkeiten und Auswirkungen von Btx für jeden, dem die bereits vorhandene Literatur zu fachlich ist.

BASTEI
LÜBBE

Howard Budin

Computer-spiele

mit Commodore 64

Sie verfügen bereits über Grundkenntnisse der Programmsprache BASIC? Dann können Sie auch für Ihren Commodore 64 einige fabelhafte Spiele programmieren!

Mit einfachen, leichtverständlichen Anweisungen werden Sie schrittweise durch geheimnisvolle und abenteuerliche Programme geführt, die Sie mit Hilfe dieses Buches erweitern und verbessern können. Was Sie daraus machen, liegt bei Ihnen – Ihrer Phantasie sind keine Grenzen gesetzt!

BASTEI LÜBBE

Deutsche Erstveröffentlichung

Computer/ Neue Medien

ISB N 3-404-63084-X DM +006.80

T 3-28-00

00680

Österreich S 53,–
Niederlande f 9,15
Frankreich FF 25,–
Italien L 6800

9 783404 630844