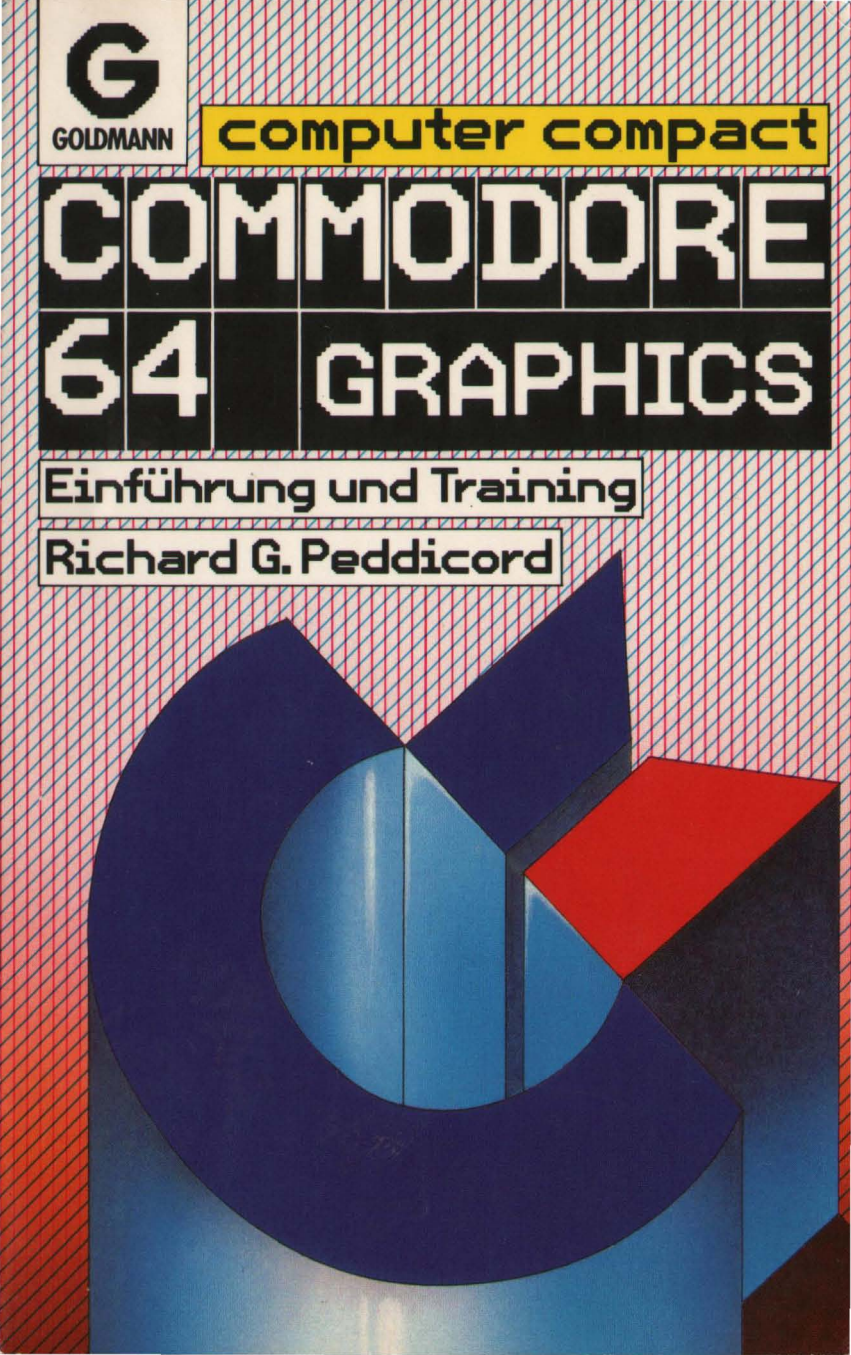
GOLDMANN

computer compact

COMMODORE 64 GRAPHICS

Einführung und Training

Richard G. Peddicord



computer compact

Richard G. Peddicord

COMMODORE 64 GRAPHICS

Einführung und Training

Wir danken der Firma Commodore Büromaschinen GmbH, Frankfurt,
für die freundliche Überlassung des Bildmaterials sowie der Nachdruckrechte
für die auf S. 79f. abgedruckte Tabelle.

Printed in Germany · 12/84 · 1. Auflage · 1110

Umschlaggestaltung: Design Team München
Übersetzung: Sabina Janas
Konzeption und Redaktion: topic GmbH, München-Karlsfeld
Herstellung: Hubert K. Hepfinger/Peter Sturm
Satz: fb Werbeservice
Druck: Presse-Druck Augsburg
Verlagsnummer: 13119

ISBN 3-442-13119-7

INHALT

1. Einführung

Der vorliegende Band aus der Reihe »Goldmann computer compact« wendet sich an den Anfänger, der das Schreiben von Graphikprogrammen in der Programmiersprache BASIC auf dem Commodore 64 erlernen will. Der Commodore 64 ist ein außergewöhnlich leistungsfähiger, dabei aber sehr benutzerfreundlicher Computer, dessen Handhabung aber natürlich die Vertrautheit mit der Maschine wie mit der verfügbaren Software voraussetzt. Einen leichten Einstieg hierzu bieten zwei weitere Bände aus der Reihe »Goldmann computer compact«, nämlich die Titel »Commodore 64« und »Commodore 64 BASIC«. Sie finden darin eine Erläuterung der Tastatur des C 64 wie

Eine typische Systemkonfiguration: Commodore 64, 1701 Monitor, 1520 Drucker, 1541 Diskettenlaufwerk und Kassettenrekorder.

der Inbetriebnahme des Systems und Anweisungen, wie Sie Diskettenlaufwerk, Kassettenrekorder, Drucker und Monitor an das System anschließen sowie Programme laden und sichern. Der Band »Commodore 64 BASIC« gibt einen Überblick über alle BASIC-Befehle, von denen Sie viele auch für die Beispiele und Übungen dieses Bandes benötigen. Dennoch können Sie die hier aufgeführten Programme auch ohne weitere Vorkenntnisse auf Ihrem Commodore 64 durchspielen; vorausgesetzt wird lediglich, daß Sie über einen Monitor sowie ein Diskettenlaufwerk verfügen, und der Computer sich in betriebsbereitem Zustand befindet. In diesem Fall erscheint folgende Meldung auf dem Bildschirm:

```
    ****COMMODORE 64 BASIC V2****
64K RAM SYSTEM 38911 BASIC BYTES FREE
READY
```

Dieser Band gibt eine Einführung in das Schreiben elementarer Graphikprogramme in BASIC, wobei die einzelnen Arbeitsschritte so angelegt sind, daß Sie sie leicht an Ihrem Gerät nachvollziehen können. Beherrschen Sie die hier behandelten Themen, so können Sie dieses Wissen entsprechend Ihren Ansprüchen und Bedürfnissen leicht vertiefen.

2. Los geht's

Bei der Inbetriebnahme des Systems empfiehlt es sich, zuerst das Diskettenlaufwerk und den Monitor einzuschalten und den Geräten damit Zeit zu geben, sich zu erwärmen. Wenden Sie sich dann dem Computer zu, an dessen rechter Seite Sie den On/Off-Schalter finden. Nach einigen Sekunden erhalten Sie die Betriebsmeldung des Systems in hellblauen Zeichen vor dunkelblauem Hintergrund; der Bildschirm wird, sofern die Farbeinstellung Ihres Monitors in Ordnung ist, von einem hellblauen Rand umrahmt. Die Qualität der Graphikdarstellung hängt wesentlich von der Ihres Bildschirms ab. Commodore bietet für den C 64 den 1701 Monitor an, der die Farbinformation auf einem eigenen Kanal erhält und eine optimale Ergänzung zum Rechner selbst darstellt. Natürlich können Sie aber auch jeden anderen Monitor verwenden, sofern er mit dem System kompatibel, also verträglich ist; beraten Sie sich hierzu eingehend mit Ihrem Händler.
Unterhalb der Betriebsmeldung finden Sie auf dem Bildschirm ein blinkendes Rechteck, Kursor (»Cursor«) genannt. Er markiert die Stelle, an der Ihre nächste Eingabe erscheint.
Tippen Sie zunächst den Namen Ihres Lieblingsmusikers ein und bedienen Sie dann die RETURN-Taste, die auf der rechten Seite der Tastatur angeordnet ist. Sie erhalten dann folgende Meldung auf dem Schirm:

```
READY.
MILTON WEATHERSPOON
?SYNTAX ERROR
READY.
```

Antwortet Ihr C 64 auf diese Weise, so arbeitet er zufriedenstellend. Haben Sie ein gültiges Kommando eingegeben, um das System zu prüfen, dann hätte es erst das Kommando ausgeführt und danach mit READY geantwortet. Geben Sie jetzt Ihren Namen ein und beobachten Sie dabei den Bildschirm genauer. Immer dann, wenn Sie eine Taste bedienen, bewegt sich der Kursor um eine Position nach rechts. Drücken Sie die RETURN-Taste, so springt der Kursor an den Anfang der nächsten Zeile; ähnlich arbeitet der Wagenrücklauf einer Schreibmaschine. Nach der Eingabe Ihres Namens meldet sich der Computer mit der Meldung ?SYNTAX ERROR zurück. Auf der Vorderseite der Zifferntasten 1 bis 8 finden Sie die Abkürzungen für acht der insgesamt 16 Farben, die Sie mit dem Commodore64 erzeugen können; die nebenstehende Tabelle hilft Ihnen bei der Enträtselung der Bezeichnungen.
Bedienen Sie die Zifferntaste und die SHIFT-Taste zugleich, so erhalten Sie keine Änderung der Farbe des Monitors, vielmehr erscheint jetzt das oben auf der Zifferntaste abgebildete Punktuationszeichen. Um die Farbe des Kursors und damit die aller Zeichen zu ändern, benötigen Sie die CONTROL(CTRL)-Taste, die Sie zusammen mit einer Zifferntaste drücken. Im Fall der Zifferntaste 2 wandelt sich die Kursorfarbe von hellblau in weiß, und auch jedes von Ihnen eingegebene Zeichen wird nun weiß dargestellt. Versuchen Sie nun, die übrigen der mit CTRL- und der Zifferntasten von 1 bis 8 möglichen Farben zu erzeugen; stimmen diese auf dem Bildschirm nicht mit der Angabe überein, so überprüfen Sie die Farbeinstellung Ihres Monitors.
Sie können aus einer zweiten Menge von acht Farben wählen, wenn Sie die COMMODORE(C=)-Taste zusammen mit den Zifferntasten bedienen; die dadurch möglichen Farben finden Sie ebenfalls in nebenstehender Tabelle aufgelistet. Betätigen Sie die COMMODORE-Taste zusammen mit der Zifferntaste 7, so nimmt der Kursor wie-

der die Farbe Hellblau an. Sicherlich ist Ihr Bildschirm schon jetzt mit vielen Meldungen gefüllt. Sie löschen ihn, indem Sie die SHIFT-und die CLR/HOME-Taste zugleich bedienen.

Tastenkombination zur Farbeinstellung der Zeichen

Ziffer	Mit der CTRL-Taste erzeugte Farben	Mit der C=Taste erzeugten Farben
1	Schwarz	Orange
2	Weiß	Braun
3	Rot	Rosa
4	Blaugrün	Grau 1
5	Purpurrot	Grau 2
6	Grün	Hellgrün
7	Blau	Hellblau
8	Gelb	Grau 3

Oft ist es auch notwendig, den Kursor über den Bildschirm zu bewegen. Zu diesem Zweck finden Sie in der rechten unteren Ecke der Tastatur zwei Tasten, die mit der Abkürzung CRSR (für »Cursor«) und mit Pfeilen gekennzeichnet sind. Drücken Sie die Taste mit den nach rechts und links weisenden Pfeilen, so bewegt sich der Kursor nach rechts. Bedienen Sie zugleich die SHIFT-Taste, so erhalten Sie, wie bei der Schreibmaschinentastatur, stets das oben auf einer Taste angeordnete Zeichen; der Kursor wandert in diesem Fall also nach links. In analoger Weise lassen Sie ihn auch nach oben und unten laufen. Geben Sie nun erneut Ihren Namen ein und drücken Sie die RETURN-Taste. Bewegen Sie anschließend den Kursor nach unten, bis er in der letzten Zeile steht. Betätigen Sie jetzt nochmals die CRSR-Taste, die den Kursor nach unten wandern läßt, so verschiebt sich der gesamte Bildschirminhalt um eine Zeile nach oben. Diesen Vorgang können Sie beliebig oft wiederholen, er wird als »Scrolling« oder »Herausschieben« bezeichnet.

Tippen Sie jetzt das Kommando NEW ein und bedienen Sie die RETURN-Taste. Damit löschen Sie jedes Programm im Speicher. Geben Sie folgende Zeilen ein:

```
10 PRINT "MILTON WEATHERSPOON"
20 GOTO 10
RUN
```

Ist Ihnen während der Eingabe ein Fehler unterlaufen, so betätigen Sie die INST/DEL-Taste, wodurch das letzte Zeichen gelöscht wird. INST steht für »Insert« (Einfügen), DEL für »DELETE« (Löschen). Drücken Sie nach der vollständigen Eingabe des Programms die RETURN-Taste, so arbeitet Ihr Computer gemäß dem Kommando RUN das Programm ab, und es erscheint die Ausgabe MILTON WEATHERSPOON auf dem Monitor, wie es Zeile 10 festgelegt hat. Die Anweisung in Zeile 20 befiehlt dem Computer, erneut in Zeile 10 zu springen, weshalb der Name immer wieder dargestellt wird; man bezeichnet ein solches Programmstück auch als Schleife, in diesem Fall als endlose Schleife, da es keine Anweisung enthält, wann der Computer die Ausführung von Zeile 10 abbrechen soll. Um ein solches Programm zu stoppen, benötigen Sie die RUN/STOP-Taste. Die Maschine meldet sich dann zurück mit der Ausgabe:

```
BREAK IN LINE 20

READY.
```

Mit Hilfe der RUN/STOP-Taste können Sie das System in seinen Ausgangszustand zurückschalten. Der Kursor erscheint dann wieder in hellblauer Farbe, und all das, was sich auf dem Bildschirm befand, wird gelöscht. Versuchen

Sie es! Drücken Sie gleichzeitig die RUN/STOP- und die RESTORE-Taste. Gelingt dies nicht sofort, wiederholen Sie den Vorgang, bis der Ausgangszustand wieder hergestellt ist.

3. Hintergrund- und Randfarben

Der Commodore 64 erzeugt ein Monitorbild von 25 Zeilen zu je 40 Zeichen. Dies entspricht den Standardwerten der meisten Heimcomputer. Jedes Zeichen wird aus 64 kleinen Punkten, »Pixel« genannt, erzeugt, die in 8 Reihen zu je 8 Spalten angeordnet sind. Der Ausdruck »Pixel« stellt eine Zusammenziehung der Worte »Picture Element« (Bildelement) dar.
Im Randbereich des Bildschirms lassen sich keine Zeichen erzeugen, weshalb er immer nur in einer von 16 möglichen Farben leuchtet:

0	schwarz	8	orange
1	weiß	9	braun
2	rot	10	rosa
3	blaugrün	11	grau 1
4	purpurrot	12	grau 2
5	grün	13	hellgrün
6	blau	14	hellblau
7	gelb	15	grau 3

Die Voreinstellung der Randfarbe ist hellblau. Um sie zu verändern, geben Sie folgenden Befehl ein:

```
POKE 53280,<Randfarbe>
```

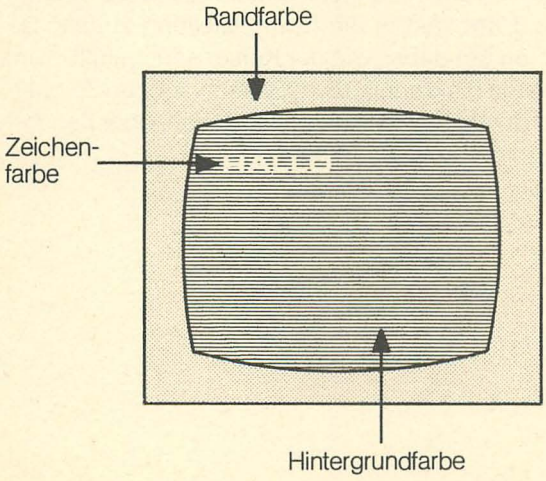
Damit verändern Sie den Inhalt der Speicherzelle, die für die Kontrolle der Randfarbe zuständig ist.
Probieren Sie es aus. Drücken Sie zuerst die RETURN-Taste. Der Computer antwortet mit READY. Geben Sie jetzt die Anweisung POKE 53280, ein Komma und eine beliebige Zahl zwischen 0 und 15 ein. Betätigen Sie erneut die

RETURN-Taste. Das ist schon alles. Der Bildschirmrand wird in der gewählten Farbe aufleuchten.
Wenn Sie den Computer einschalten, erscheint der Hintergrund in der Farbe Dunkelblau. Aber Sie können jede der 16 Farben wählen, indem Sie die entsprechende Nummer in die Speicherzelle eintragen, die die Hintergrundfarbe kontrolliert. Sie hat die Nummer 53281 und der entsprechende Befehl lautet somit:

```
POKE 53281,<Hintergrundfarbe>
```

Natürlich können Sie auch die Farbe der Zeichen verändern. Dazu geben Sie den folgenden Befehl ein:

```
POKE 646,<Zeichenfarbe>
```



Der Commodore-Monitor unterteilt sich in Textbetrieb, Hintergrund und Rand.

Übungen

3.1. Wählen Sie als Randfarbe Blaugrün, als Hintergrundfarbe Rot und als Zeichenfarbe Hellblau. Speichern Sie die entsprechenden Nummern mit einem POKE-Kommando in den richtigen Registern ab.

3.2. Drücken Sie die Commodore-Taste und die Taste mit dem Pluszeichen (+). Auf dem Bildschirm wird eine karierte Miniaturflagge auftauchen. Erstellen Sie unter Verwendung der Kursortasten einen ganzen Block dieser Flaggen in der Mitte des Monitors. Verändern Sie jetzt die Farben des Rands, des Hintergrunds und der Zeichen. Beobachten Sie, was beim Farbwechsel alles geschieht.

3.3. Betrachten Sie durch ein Vergrößerungsglas den blinkenden Kursor. Verändern Sie die Helligkeit und den Kontrast, bis nur noch die grünen Phosphorpunkte leuchten. Sie erkennen jetzt alle 64 Punkte, die den grünen Teil des Kursors erzeugen. Drehen Sie Helligkeit und Kontrast in die Normalstellung zurück. Beobachten Sie dabei, wie der Kursor »ausgefüllt« wird.

3.4. Führen Sie die Übung 3.3. mit verschiedenen Zeichenfarben durch. Sie können dabei einiges über den Aufbau farbiger Zeichen lernen.

4. Handgemachte Designs

Auf der Vorderseite der Tasten des Commodore 64 befinden sich verschiedene graphische Zeichen. Aus ihnen lassen sich einfache Bilder zusammenstellen. Angenommen, Sie wollen ein Herz auf dem Bildschirm zeichnen. Suchen Sie zuerst jene Taste, auf deren Vorderseite ein Herz abgebildet ist. Richtig, es ist die S-Taste. Drücken Sie die SHIFT- und die S-Taste zugleich. Sie haben damit ein Herz erzeugt.

Geben Sie jetzt ein Karo, ein Kreuz, ein Pik, einen vollen und einen leeren Kreis ein. Verändern Sie die Farben. Haben Sie bemerkt, daß beim Wechseln der Zeichenfarben alle schon geschriebenen Zeichen unverändert bleiben?

Es befinden sich immer zwei graphische Symbole auf einer Taste. Um das linke Zeichen zu erhalten, müssen Sie die COMMODORE-Taste zusammen mit der gewünschten Taste drücken. Versuchen Sie es! Betätigen Sie die COMMODORE-Taste und die Taste mit dem Stern (*): Ein kleines Dreieck erscheint auf dem Bildschirm.

Die COMMODORE-Taste ist direkt links neben der SHIFT-Taste angeordnet. Daher wird das linke graphische Symbol zusammen mit dieser Taste erzeugt. Das rechte Zeichen erhält man ja mit der SHIFT-Taste.

Wenn Sie die graphischen Symbole genau betrachten, dann stellen Sie fest, daß viele exakt zusammengefügt werden können. Betrachten Sie beispielsweise die rechten Symbole der Tasten U, I, J und K (diese werden mit Hilfe der SHIFT-Taste erzeugt), die zusammen einen kleinen Kreis formen.

Sie können mit den Spielkartenfarben spielen, wenn Sie die Tasten A (Pik), S (Herz), Z (Karo) und X (Kreuz) verwenden. Mit Hilfe der Tasten U, I, J, und K für die runden Ecken,

SHIFT * für die geraden Kanten und SHIFT + für die senkrechten Kanten erzeugen Sie den Rand einer Spielkarte. Also löschen Sie den Bildschirm und zeichnen Sie die Herz 10. Experimentieren Sie mit graphischen Designs. Der Commodore 64 ist einer der wenigen Computer, mit denen Sie diese Graphiken erstellen können, ohne ein Programm eingeben zu müssen. Sie werden bald feststellen, daß nicht alle möglichen Symbole vorhanden zu sein scheinen. Drücken Sie beispielsweise die COMMODORE-Taste zusammen mit der K-Taste, dann erscheint die linke Hälfte eines Blocks. Wo ist die andere Hälfte? Sie werden sie nicht finden. Es muß also einen Weg geben, das Gegenstück oder die Umkehrung eines Zeichens zu erzeugen. Das ist auch möglich. Betätigen Sie die CONTROL(CTRL)-Taste und die Zifferntaste 9 gleichzeitig. Drücken Sie danach die COMMODORE- und die K-Taste. Jetzt erscheint die rechte Hälfte des Blocks auf dem Monitor. Bedienen Sie also die CTRL-Taste und die Zifferntaste 9 zusammen, so führt der Computer genau das aus, was auf der Vorderseite der Zifferntaste 9 steht. Der Ausdruck RVS ON steht für RESERVE ON und bedeutet, daß Sie die invertierte Darstellung oder den Umkehrmodus einschalten.

Die Herz 10

Übungen

4.1. Wählen Sie die Farben Rot für den Rand, Weiß für den Hintergrund und Schwarz für die Zeichen. Löschen Sie den Bildschirm, bewegen Sie den Kursor in die Mitte

und erzeugen Sie die Zahl 2, bestehend aus Kreuzen. Zeichnen Sie um die Zahl einen Rand mit abgerundeten Ecken.

4.2. Füllen Sie den Bildschirm mit karierten Flaggen (COMMODORE- und +-Taste) aus. Wechseln Sie dabei nach wenigen Spalten jeweils die Farbe. Erzeugen Sie dazu auch verschiedene Hintergrundfarben. Sie können mit etwas Geduld sehr schöne Designs entstehen lassen.

4.3. Entwerfen Sie ein sorgfältig ausgearbeitetes Modell eines Lastzugs. Verwenden Sie dabei die Tasten, mit deren Hilfe Sie auch die Ränder von Spielkarten gezeichnet haben.

4.4. Weben Sie ein Indianertuch aus großen farbigen Rechtecken. Verwenden Sie dabei COMMODORE- und *-Taste, SHIFT- und Y-Taste sowie andere Tasten im Normal- und Umkehrmodus.

5. Fallende Herzen

Im vorangegangenen Kapitel haben Sie gelernt, graphische Designs zu entwerfen, indem Sie die einzelnen Symbole mit Hilfe des Kursors an bestimmten Stellen »plaziert« haben. Mit dieser Methode können Sie jedes Bild zeichnen, aber es dauert relativ lange, und Sie können Ihre Entwürfe nicht abspeichern. Um anspruchsvollere Graphik erzeugen zu können, müssen Sie ein wenig programmieren lernen.
Die Zentraleinheit, das eigentliche »Gehirn« Ihres Computers, arbeitet ausschließlich mit dem binären Zahlensystem, das nur über die Zeichen 0 und 1 verfügt. Eine Binärzahl, auch als Bit (»Binary Digit«) bezeichnet, stellt die kleinste Informationseinheit dar, die ein Computer verarbeiten kann; jeweils 8 Bit werden zu einem Byte zusammengefaßt. Es wäre nun sehr mühselig, in der Sprache des Computers, also in der auf binärer Zahlenbasis laufenden Maschinensprache, ein Programm zu erstellen. Relativ rasch entstanden daher »höhere« Programmiersprachen, wie beispielsweise BASIC, die in einem Kommando jeweils mehrere Befehle in Maschinensprache zusammenfassen und es dem Benutzer erlauben, sich auf einfachem Weg mit dem Computer zu verständigen. Damit dieser die in einer höheren Programmiersprache verfaßten Anweisungen verstehen kann, müssen sie durch einen »Interpreter« oder einen »Compiler« in Maschinensprache übersetzt werden, was natürlich einen gewissen Zeitverlust bedeutet. Er läßt sich begrenzen, verwendet man Programme, die in Assemblersprache geschrieben sind. Diese Programmiersprache steht der Maschinensprache erheblich näher, weshalb Assemblerprogramme schneller ablaufen als in BASIC verfaßte Programme. BASIC zählt jedoch zu den benutzerfreundlichsten und am mei-

sten verbreiteten Programmiersprachen; der Name steht für »Beginners All-purpose Symbolic Instruction Code«. Im weiteren Verlauf dieser Einführung müssen Sie einige Kommandos dieser Sprache erlernen.
Da die Maschine für Sie arithmetische Ausdrücke auswerten muß, ist es nötig, sich kurz damit zu beschäftigen. Angenommen, Sie wollen die Summe von 247 und 591 berechnen. Geben Sie dazu folgendes ein:

```
PRINT 247 + 591
```

Der Computer antwortet mit:

```
838
```

BASIC kennt 5 arithmetische Symbole: + (Addition), – (Subtraktion), ↑ (Multiplikation),/(Division) und ↑ (Exponentiation). Nachfolgend sind einige Beispiele dafür dargestellt:

Beispiel	Ergebnis
PRINT 5+1	6
PRINT 7-2	5
PRINT 4*3	12
PRINT 10/5	2
PRINT 3↑4	81

Denken Sie daran, daß bei der Exponentiation der Ausdruck a^b bedeutet, daß a b-mal mit sich selbst multipliziert wird. Rechnen Sie die folgenden Beispiele erst selbst aus, bevor Sie sie in den Computer eingeben.

```
2+2/2   5+2↑3   3+3*3↑3
```

Sind Sie überrascht? Sie erhalten als Ergebnis des Rechners die Zahlen 3, 13, und 84. Der Computer arbeitet arithmetische Ausdrücke, die verschiedene Rechenoperationen beinhalten, in einer bestimmten Reihenfolge ab; man spricht auch von einer Präferenzordnung, in der die Operationen stehen:

Symbol	Bedeutung	Rangfolge
( )	Klammern	1
↑	Exponentiation	2
*, /	Multiplikation Division	3
+, –	Addition, Subtraktion	4

Stets wird der von Klammern eingeschlossene Ausdruck zuerst berechnet. Als Ergebnis von (2+2)/2 erhalten Sie die Zahl 2, während 2+2/2 die Zahl 3 erbringt. Anschließend nimmt der Computer die Exponentiation vor; (5*(3+2))↑2 ergibt als Ergebnis 625, während 5*(3+2) ↑ 2 gleich 125 ist. Erst dann werden die übrigen Rechenoperationen abgearbeitet, wobei Multiplikation und Division vor Addition und Subtraktion. Arithmetische Operationen mit gleicher Präferenzordnung berechnet die Maschine von links nach rechts.

Der Computer legt Ziffern und Zeichenreihen in Speicherzellen ab, denen er eine symbolische Adresse gibt. Eine solche Adresse bezeichnet man auch als »Variablenname« oder kurz als »Variable«. Sie können die Namen von Variablen frei wählen, sofern Sie darauf achten, keine BASIC-Kommandos, wie RUN, LIST oder PRINT, versehentlich im Namen erscheinen zu lassen. Bezeichnen Sie beispielsweise eine Speicherzelle mit dem Variablennamen A und speichern darin den Wert 5 ab, so geben Sie dazu folgenden BASIC-Befehl ein:

```
A=5
```

Sie haben damit eine Zuweisung (»Assignement«) vorgenommen. Geben Sie jetzt

```
PRINT A
```

ein, dann wird der Computer mit der Meldung

```
5
```

antworten.

Sie haben gerade eine Speicherzelle benannt und darin einen Wert gespeichert. Auf Ihre Anfrage PRINT A hin hat der Computer nachgesehen, was unter der Adresse A im Speicher abgelegt worden ist, und diesen Inhalt dann auf dem Bildschirm dargestellt. Dabei wurde der in der Speicherzelle enthaltene Wert nicht verändert.
Variablen werden häufig verwendet, um Berechnungen durchzuführen. Wählen Sie eine weitere Variable B und geben Sie folgendes Beispiel ein:

```
A=15+7*3
B=1000-A*4
PRINT A
PRINT B
```

Der Computer antwortet mit dem Wert 36 für die Variable A und 856 für die Variable B. Es gibt noch einen einfacheren Weg, zwei Variablen auf einmal auszudrücken: die Zahlen werden nebeneinander und nicht untereinander ausgegeben. Dazu setzen Sie ein Semikolon zwischen die beiden Variablen:

```
PRINT A;B
```

Wollen Sie die Werte der Variablen spaltenweise anordnen, dann geben Sie

```
PRINT A,B
```

ein.
Sie können bis zu 4 Spalten mit je 10 Zeichen auf dem Bildschirm darstellen.
Was passiert, wenn Sie

```
A=A+1
```

eintippen? Der Computer berechnet den Ausdruck A+1 und speichert das Ergebnis seiner Operation wieder in Zelle A ab. Als Resultat erhalten Sie bei obigem Programm dann den Wert 37.
Jetzt sind Sie in der Lage, mit Zahlen zu programmieren. Um sich dies zu beweisen, geben Sie das folgende Programm ein:

```
10 REM FALLENDE HERZEN
20 POKE 53280,6
30 POKE 53281,4
40 A=1
50 A=A+1
60 PRINT TAB(A)"<CTRL 3><SHIFT S>"
70 GOTO 50
```

Mit Ausnahme von Zeile 60 tippen Sie das Programm wie angegeben ein. Wenn Sie bei <CTRL 3> angekommen sind, dann betätigen Sie die CTRL-Taste und die Zifferntaste 3 zusammen, worauf das Symbol für das britische Pfund dargestellt wird.
Bei <SHIFT S> bedienen Sie die SHIFT-Taste und die Taste S gemeinsam; in diesem Fall erscheint ein Herz auf dem Bildschirm.
Lassen Sie jetzt das Programm laufen, indem Sie RUN eingeben. Wenn es jemals ein graphisches Programm gab, dann dieses.
Hier die Bedeutung der einzelnen Befehle:
10 In einer REM-Angabe (»REMark«) geben Sie dem

Programm einen Namen; der Computer ignoriert alle REM-Zeilen, sie dienen lediglich als Information für den Programmierer.

20 Sie wählen die Randfarbe Blau.

30 Sie wählen die Hintergrundfarbe Purpur.

40 A wird der Wert 1 zugewiesen.

50 Der Wert von A wird um 1 erhöht.

60 Sie lassen ein rotes Herz in Spalte A drucken.

70 Sie befehlen der Maschine, zu Zeile 50 zurückzugehen.

Wie Sie aus der Programmstruktur erkennen, wird Zeile 60 immer wieder ausgeführt. Dabei erhöht sich der Wert von A jeweils um 1, und bei jedem Schleifendurchlauf verschiebt sich damit die Ausgabe in die nächstfolgende Zeile. Das Programm läuft, bis Sie die STOP-Taste drücken.

Studieren Sie Zeile 60 noch eingehender. Die Funktion TAB wird auf das Argument A angewandt. Sie bewirkt, daß der Kursor auf die Spalte A springt; dann druckt die Maschine die in Anführungszeichen stehende Zeichenreihe aus. Beim ersten Abarbeiten von Zeile 60 hat A den Wert 2. Die Zählung der Spalten auf dem Bildschirm beginnt bei 0. Deshalb wird das erste Symbol in die dritte Spalte gedruckt. Sie können sich davon überzeugen, indem Sie RUN eintippen, auf RETURN drücken und, so schnell Sie können, die RUN/STOP-Taste betätigen. Das erste Herz befindet sich jetzt direkt unter dem N von RUN.

Unmittelbar hinter der Funktion TAB(A) folgt ein Anführungszeichen. Der Computer druckt nun die sich an das Anführungszeichen anschließenden Symbole, bis er auf das nächste Anführungszeichen stößt. In diesem Fall stehen innerhalb der Anführungszeichen zwei seltsame Zeichen: <CTRL 3> und <SHIFT S>. Liest der Computer sie, so gibt er das aus, was auch bei Betätigung der entsprechenden Tasten entsteht. Durch <CTRL 3> verwandelt sich die Zeichenfarbe in Rot, durch <SHIFT S> erhalten Sie ein rotes Herz.

Lassen Sie das Programm nochmals laufen und beachten Sie dabei, daß die dargestellten Herzen mit jeder Zeile etwas weiter nach rechts rutschen. Die TAB-Funktion akzeptiert Werte zwischen 0 und 255. Wenn Sie Glück haben, hält das Programm mit der Meldung

```
?ILLEGAL QUANTITY ERROR IN 60
READY
```

an, sobald A den Wert 256 annimmt.
Noch eine kleine Warnung: Haben Sie bei der Eingabe eines PRINT-Befehls das erste Anführungszeichen geschrieben, so arbeiten die Kursortasten nicht mehr in gewohnter Weise: Die Editierfunktionen stehen Ihnen nicht zur Verfügung, mit deren Hilfe Sie den Kursor auf dem Bildschirm bewegen können. Deshalb müssen Sie Fehler innerhalb der Anführungszeichen mit der INST/DEL-Taste korrigieren. Sie können auch das schließende Anführungszeichen eintippen und dann die Kursortasten zur Korrektur verwenden. Keine Angst, geben Sie einige PRINT-Befehle ein, und Sie werden dies alles verstanden haben.

Übungen

5.1. Ändern Sie im Programm FALLENDE HERZEN die Zeile 60 so ab, daß jeweils 5 Herzen in einer Reihe und in verschiedenen Farben gezeichnet werden. Beachten Sie, daß bei jeder Eingabe von <CTRL Ziffer> ein spezielles Symbol auf dem Bildschirm erscheint. Haben Sie als Zeichenfarbe eines Herzens wie für den Hintergrund Purpur gewählt, dann ist dieses Herz nicht sichtbar.

5.2. Schreiben Sie ein Programm, das eine ganze Zeile (40 Zeichen) mit schwarzen Pik-Zeichen druckt, gefolgt von je einer ganzen Zeile roter Herzen, Karos der Farbe Orange und blauer Kreuze. Wiederholen Sie diese

Anordnung, bis der Bildschirm gefüllt ist. Wählen Sie dabei einen weißen Hintergrund.
Hinweis: Verwenden Sie für jede der vier Zeilen einen PRINT-Befehl. Geben Sie danach ein GOTO-Kommando ein. Jeder dieser PRINT-Befehle wird länger als eine Zeile sein. Der Kursor wird bei der Eingabe automatisch in die nächste Zeile springen, wenn Sie am letzten Zeichen einer Zeile angekommen sind. Betätigen Sie die RETURN-Taste nicht, bevor Sie nicht das komplette Kommando eingetippt haben.

5.3. Schreiben Sie ein Programm, das den gesamten Bildschirm mit gelben Kreisen (SHIFT Q) auf blauem Hintergrund ausfüllt.

5.4. Schreiben Sie ein Programm, das die Kreuz 2 erzeugt.

Die Kreuz 2

6. Sichern von Programmen

Bevor Sie umfangreichere Programme schreiben, sollten Sie deren Sicherung auf Diskette oder Magnetband beherrschen. Ohne Diskettenlaufwerk oder Kassettenrekorder (Datasette) kommt auf Dauer kein Computerbenutzer aus, weil der Arbeitsspeicher Ihres Computers seinen Inhalt verliert, wenn Sie den Rechner ausschalten; man spricht in diesem Fall auch von RAM (Random Access Memory) oder Direkt-Zugriff-Speicher. Sein Vorteil liegt darin, daß Sie in ihn schreiben und aus ihm lesen können, wodurch er sich vom Nur-Lese-Speicher oder ROM (Read Only Memory) unterscheidet, aus dem Sie nur Daten entnehmen können.
Die Flüchtigkeit des Direkt-Zugriff-Speichers macht externe Speichermedien, wie Diskette oder Magnetband, erforderlich, um Information dauerhaft zu sichern. Wie Sie diese Zusatz- oder Peripheriegeräte an Ihren Commodore 64 anschließen, finden Sie in dem Band »Commodore 64« aus der Reihe »Goldmann computer compact« erläutert. Sie wissen vermutlich, wie man eine Kassette in einen Kassettenrekorder einlegt. Das Einschieben einer Diskette in ein Diskettenlaufwerk ist nicht wesentlich schwieriger. Nehmen Sie die Diskette aus ihrer Schutzhülle und vermeiden Sie es, die Oberfläche der Diskette selbst zu berühren; schon geringste Schmutzspuren können zu Übertragungsfehlern führen. Achten Sie darauf, daß die ovale Öffnung zum Laufwerk hin und das Etikett nach oben zeigt. Drücken Sie leicht gegen die Diskette, bis sie ganz im Laufwerk verschwunden ist. Schließen Sie dann das Laufwerk, indem Sie die Klappe nach unten drücken und einrasten lassen. Zum Entnehmen der Diskette öffnen Sie zuerst diese Klappe. Die Diskette wird automatisch so weit herausgeschoben, daß sie leicht zu greifen ist.

Angenommen, Sie haben ein Programm mit dem Namen KUNDENLISTE geschrieben und wollen es jetzt auf Magnetband sichern.
Geben Sie dazu das Kommando

```
SAVE "KUNDENLISTE"
```

ein. Achten Sie darauf, nicht den Vorlauf des Bandes zu überspielen.

Mit dem Kommando

```
SAVE "KUNDENLISTE",8
```

sichern Sie das Programm auf Diskette. Die Zahl 8 stellt das Codewort für das Diskettenlaufwerk dar. Löschen Sie nun den Speicher Ihres Computers durch die Eingabe eines NEW-Kommandos, so steht Ihnen das Programm nur noch auf dem externen Speichermedium zur Verfügung. Um damit arbeiten zu können, müssen Sie es erneut in den Speicher Ihres Rechners laden.
Haben Sie Ihr Programm auf Kassette gesichert, so spulen Sie dazu das Band an den Anfang zurück, drücken die PLAY-Taste und geben das Kommando

```
LOAD "KUNDENLISTE"
```

oder nur

```
LOAD
```

ein. In beiden Fällen leuchtet der Bildschirm hellblau auf, und der Computer antwortet mit der Meldung:

```
FOUND "KUNDENLISTE"
```

Haben Sie nur das Kommando LOAD eingetippt, dann hält der Kassettenrekorder jetzt an. Nach Betätigung der Commodore-Taste läuft das Band weiter. Ist das aber nicht Ihr Wunsch, so drücken Sie auf die STOP-Taste.

Verwenden Sie eine Diskette, dann geben Sie folgenden Befehl ein:

```
LOAD "KUNDENLISTE",8
```

Die Diskette wird im Laufwerk kurz rotieren und dann wieder anhalten. Unabhängig davon, ob Sie Kassette oder Diskette verwenden, meldet sich der Computer wie gewohnt mit READY zurück. Das Programm mit dem Namen KUNDENLISTE ist wieder im Rechner, wie Sie durch folgendes Kommando leicht überprüfen können:

```
LIST
```

Das Programm erscheint am Bildschirm exakt so, wie Sie es eingegeben haben. Nur war jetzt der Aufwand, es erscheinen zu lassen, wesentlich geringer.
Sie können den oben beschriebenen Vorgang mit jedem Programm wiederholen. Verwenden Sie nur die Kommandos SAVE "NAME" oder SAVE "NAME",8 und LOAD "NAME" oder LOAD "NAME",8. Einen wichtigen Punkt gilt es bei der Benutzung von Kassetten noch zu beachten. Haben Sie ein Programm am Anfang eines Bandes aufgenommen, dann müssen Sie beim Laden das Band auch von Anfang an abspielen. Lassen Sie zwischen zwei Programmen mindestens 1 – 2 Zählernummern Platz frei, um ein Programm nicht versehentlich zu löschen.

7. Neun kleine Kreise

Ein äußerst wichtiges Kommando in BASIC stellt die IF...-THEN-Anweisung dar, mit deren Hilfe Sie überprüfen können, ob ein Ausdruck den Wert »wahr« oder »falsch« besitzt.
Das folgende Programmstück gibt ein gutes Beispiel dafür:

```
NEW
10 REM NEUN KLEINE KREISE
20 A=0
30 PRINT TAB (2*A)"<SHIFT U><SHIFT I>"
40 PRINT TAB (2*A)"<SHIFT J><SHIFT K>"
50 IF A=9 THEN STOP
60 A=A+1
70 GOTO 30
```

Lassen Sie das Programm laufen, indem Sie das Kommando RUN eingeben; auf dem Bildschirm erscheinen nun neun kleine Kreise, diagonal angeordnet. Dabei sind mehrere Punkte zu beachten.
Zuerst einmal haben Sie innerhalb der Klammern von TAB die Variable A mit 2 multipliziert. Dadurch wird jeder Kreis in einer Zeile etwas weiter nach rechts geschoben, wenn die Variable A einen neuen Wert annimmt. Zum anderen gibt es für jeden Wert von A zwei PRINT-Befehle. Damit fügen Sie den oberen und den unteren Halbkreis zusammen.
Zeile 50 enthält eine IF...THEN-Anweisung. So lange die Variable A noch nicht den Wert 9 angenommen hat, ist der Ausdruck A=9 falsch, und die Maschine arbeitet die nächste Zeile des Programms ab, in diesem Fall also die Zeile 60. Wird A aber der Wert 9 zugewiesen, dann nimmt der Ausdruck den Wert »wahr« an, und der Rechner führt die

Anweisung aus, die auf das Kommando THEN folgt. In diesem Fall wird der Programmablauf beendet.
Das gleiche Ergebnis erhalten Sie auch bei Verwendung einer FOR...NEXT-Schleife. Das Programm nimmt dann folgende Form an:

```
10 REM NEUN KLEINE KREISE VERSION 2
20 FOR A=0 TO 9
30 PRINT TAB(2*A)"<SHIFT U><SHIFT I>
40 PRINT TAB(2*A)"<SHIFT J><SHIFT K>
50 NEXT A
```

Geben Sie das Programm ein und lassen Sie es laufen, um sich selbst überzeugen zu können. Beim ersten Abarbeiten von Zeile 20 weist der Computer der Variablen A den Wert 0 zu. Dann führt er die Anweisung zwischen FOR und NEXT aus. In Zeile 50 addiert der Rechner 1 zu A und überprüft, ob der Wert von A immer noch kleiner oder gleich 9 ist, wie in Zeile 20 festgelegt. Ist der Wert von A kleiner als 9, dann springt der Computer in die der FOR...NEXT-Anweisung folgenden Zeile, in diesem Falle also in Zeile 30. Er arbeitet weiter, bis er wieder auf ein NEXT-Kommando stößt, und der Vorgang wiederholt sich.
Hat A den Wert 9 erreicht, dann addiert die Maschine beim nächsten Erreichen des Kommandos NEXT den Betrag 1 zu A. Damit übersteigt der Wert von A die spezifizierte Größe 9. Die Programmkontrolle springt zu der auf das NEXT-Kommando folgenden Zeile. In diesem Fall gibt es keine weitere Anweisung, und der Programmlauf ist beendet. Die Gesamtmenge der Anweisungen von FOR bis zu NEXT nennt man FOR...NEXT-Schleife, eine Konstruktion, die Sie häufig benötigen werden.
Geben Sie einmal folgendes Programm ein, in dem eine FOR...NEXT-Schleife eingebaut ist:

```
NEW
10 REM GESCHACHTELTE FOR...NEXT-SCHLEIFE
20 POKE 53280,5:POKE 53281,2
```

```
30 PRINT "<CTRL 6>"
40 FOR L=1 TO 20
50 FOR C=1 TO 39
60 PRINT "<SHIFT W>";
70 NEXT C
80 NEXT L
```

Betrachten Sie zunächst Zeile 60. Die Eingabe SHIFT W erzeugt einen kleinen Ring auf dem Bildschirm. Das Semikolon weist die Maschine an, in dieser Zeile noch weitere Zeichen zu schreiben. Das bedeutet, daß das nächste Zeichen rechts neben den Ring gedruckt wird. Ohne das Semikolon würde die Maschine die neue PRINT-Anweisung in die folgende Zeile schreiben.
Die äußere FOR...NEXT-Schleife, bestehend aus den Programmzeilen 40 bis 80, druckt 20 Zeilen. Die innere FOR...NEXT-Schleife, bestehend aus den Zeilen 50 bis 70, erzeugt 40 Ringe in jeder Zeile.
Sie haben gerade viel gelernt. Zur Vertiefung führen Sie die folgende Übung aus.

Übungen

7.1. Verändern Sie das Programm NEUN KLEINE KREISE so, daß die Variable A mit dem Betrag 4 innerhalb der Klammern von TAB multipliziert wird. Versuchen Sie auch, die Variable A mit 20 zu multiplizieren.

7.2. Verändern Sie das Programm NEUN KLEINE KREISE so, daß das Programm anhält, wenn A den Wert 100 erreicht.

7.3. Verändern Sie das Programm NEUN KLEINE KREISE VERSION 2 so, daß die Variable A Werte von 2 bis 20 annimmt.

7.4. Verändern Sie das Programm GESCHACHTELTE FOR...NEXT-Schleifen so, daß die Ringe die Farben Blau und Grün annehmen.

7.5. Benutzen Sie die Struktur des Programms GESCHACHTELTE FOR...NEXT-Schleifen, um den Bildschirm mit Kreisen der Form zu füllen, die das Programm NEUN KLEINE KREISE erzeugt.

8. Weitere BASIC-Anweisungen

Um Computer in zeitsparender und effizienter Weise programmieren zu können, benötigen Sie höher entwickelte Methoden als die bisher dargestellten. BASIC verfügt über zahlreiche weitere Kommandos, die die Anwendungsmöglichkeiten dieser Sprache wesentlich erweitern.
Angenommen, Sie wollen 100 Variablen Werte zuweisen. Dies selbst vorzunehmen, wäre äußerst zeitintensiv, und würde Sie dazu zwingen, stets ein Verzeichnis dieser Zuweisungen mit sich zu tragen. Es gibt jedoch einen sehr viel einfacheren Weg, um mehreren Variablen verschiedene Werte zuzuweisen, indem Sie READ- und DATA-Anweisungen verwenden. Folgendes Beispiel veranschaulicht dies:

```
10 REM DAS EINLESEN VON VIER WERTEN
20 READ A, B, C, C1
30 DATA 5, 19,8, 0
40 PRINT A, B, C, C1
RUN
5          19          8          0
READY.
```

Die READ-Anweisung ordnet jeder Variablen genau einen Wert zu. Was passiert, wenn mehr Daten als Variable vorliegen? Ganz einfach: Die restlichen Daten bleiben übrig. Stößt der Computer beim Abarbeiten des Programms auf ein weiteres READ, so liest er das nächste Datenelement ein.
Stellen Sie sich vor, Sie betreiben die Wetterstation auf der Zugspitze. Die höchsten Temperaturen in einer Woche betrugen beispielsweise 12, 15, 13, 9, 14, 17 und 18 °C. Sie sind

nun an der Durchschnittstemperatur interessiert. Schreiben Sie alle Werte in eine DATA-Anweisung. Lesen Sie jeweils einen Wert ein und addieren Sie die Werte zu einer Gesamtsumme. Teilen Sie jetzt die Summe durch 7, so erhalten Sie die durchschnittliche Temperatur. Das folgende Programm löst diese Aufgabe.

```
NEW
10 REM DURCHSCHNITTLICHE TEMPERATUR
20 SUM=0
30 FOR I=1 TO 7
40 READ NUM
50 SUM=SUM+NUM
60 NEXT
70 PRINT "DER DURCHSCHNITT IST:";SUM/7
80 DATA 12, 15, 13, 9, 14, 17, 18
RUN
DER DURCHSCHNITT IST: 14

READY.
```

Zuerst wird die Variable SUM gleich 0 gesetzt. Die Maschine durchläuft die Schleife siebenmal. Bei jedem Durchgang wird ein Wert gelesen und zur Variablen SUM addiert, schließlich der Durchschnitt ausgedruckt. Der Computer führt nichts aus, wenn er die DATA-Anweisung liest, sie kann nur durch ein READ-Kommando benutzt werden.

Weitere Anwendungsmöglichkeiten eröffnen Zeichenreihenvariablen. Sie haben Zeichenreihen, keine Ziffern als Wert. Der Name einer solchen Variablen unterscheidet sich von dem einer numerischen Variablen durch das Dollarsymbol an seinem Ende. Geben Sie folgende Zeilen ein und achten Sie darauf, was geschieht:

```
A$="<SHIFT CLR/HOME>"
PRINT A$
```

Der Bildschirm wird gelöscht! Haben Sie dadurch irgendetwas zerstört? Nein! Sie haben nur den Monitor für neue Eingaben freigemacht, indem Sie die Zeichenreihenvariable mit dem Wert CLR ausdrucken ließen. Auch ganze Worte stellen eine Zeichenreihe dar:

```
A$="<CTRL 2>COMMODORE 64<Commodore 7>"
PRINT A$
```

Wie können Sie aber von der Tastatur aus etwas in Ihr Programm eingeben? Dazu müssen Sie eine INPUT-Anweisung verwenden. Versuchen Sie folgendes Programm:

```
NEW
10 REM EINGABE-BEISPIEL
20 PRINT "WAS SOLL ICH DRUCKEN?";
30 INPUT P$
40 PRINT P$
RUN
WAS SOLL ICH DRUCKEN? WEN INTERESSIERT ES?
WEN INTERESSIERT ES?

READY.
```

Wird die INPUT-Anweisung in Zeile 30 abgearbeitet, dann erscheint ein Fragezeichen auf dem Bildschirm. Der Computer gibt damit an, daß er auf Ihre Eingabe wartet. Tippen Sie diese ein und bedienen Sie die RETURN-Taste, dann weist er sie der Variablen P$ als Wert zu.
Die INPUT-Anweisung impliziert stets auch ein PRINT-Kommando, so daß Sie dieses nicht gesondert vor der INPUT-Anweisung eingeben müssen. Es ist auch nicht notwendig, die Frage mit einem Fragezeichen zu versehen, da die INPUT-Anweisung automatisch ein solches setzt.

```
10 INPUT "WIE ALT SIND SIE";A
20 PRINT "SIE WERDEN"A+2"IN ZWEI
   JAHREN SEIN."
```

Allerdings benötigt zu dieser Auskunft niemand einen Computer, außer Kindern vielleicht, aber sie sind dann wohl nicht in der Lage, Eingaben dieser Art zu bewerkstelligen. Und Erwachsene unterhalten sich lieber mit Computerspielen, als solche Programme einzugeben. Angenommen, Sie erstellen selbst ein Arkadenspiel. Wollen Sie, daß das Programm anhält und Sie fragt:

```
WOLLEN SIE FEUERN?
```

Oder wollen Sie, daß das Programm feststellt, wann Sie die Taste F gedrückt und damit »gefeuert« haben? Das Programm muß überprüfen, ob eine bestimmte Taste betätigt wurde. Geben Sie folgendes Beispiel ein:

```
NEW
10 REM FARBDEMONSTRATION
20 C=0
30 POKE 646,C
40 PRINT "FARBEN - BETAETIGEN SIE EINE
   BELIEBIGE TASTE"
50 GET A$
60 IF A$<>"" THEN C=C+1
70 IF C=16 THEN C=0
80 GOTO 30
```

Die GET-Anweisung registriert die Eingaben auf der Tastatur. Drücken Sie eine Taste, so wird das Zeichen dieser Taste zum Wert der Variablen A$. Ist keine Eingabe vorgenommen worden, so setzt die Maschine die Variable A$

gleich der leeren Zeichenreihe, die durch zwei nebeneinander stehende Anführungszeichen dargestellt wird; zwischen beiden steht kein Ausdruck, auch keine Leerstelle! Soll das Programm warten, bis eine Taste gedrückt wird, dann muß ständig – wie im folgenden Beispiel – die Tastatur überprüft werden.

```
NEW
10 REM TASTATUR WARTESCHLEIFE
20 PRINT "WOLLEN SIE PROGRAMMIEREN
   (DRUECKEN SIE J ODER N)"
30 GET A$: IF A$="" THEN 30
40 IF A$="N" THEN 20
50 PRINT "DANKE"
RUN
```

Was passiert, wenn die gedrückte Taste eine der Funktionstasten – beispielsweise f7 – war, die sich rechts auf der Tastatur befinden? Sie können nicht

```
IF A$="f7" THEN ...
```

einsetzen, wohl aber:

```
IF A$=CHR$(136) THEN ...
```

CHR$ wandelt eine Zahl gemäß der ASCII-Tabelle in eine bestimmte Zeichenreihe um. ASCII ist eine Abkürzung für »American Standard Code for Information Interchange« (Amerikanischer Standardcode für Informationsaustausch). Dieses international verwendete Codierschema entstand in den sechziger Jahren und dient vor allem dazu, den Informationsaustausch zwischen zwei Computern verschiedener Hersteller und unterschiedlicher Betriebssysteme zu ermöglichen. Jedes durch die Tastatur darstellbare Zeichen besitzt somit eine bestimmte ASCII-Codezahl. Eine Übersichtstabelle finden Sie in Kapitel 13.

So ist CHR$ (48) gleich 0, CHR$ (65) gleich A, CHR$ (133) gleich der f1-Taste usw.
Das folgende Beispiel veranschaulicht die Anwendung der CHR$-Funktion

```
NEW
10 REM UMWANDLUNG VON ASCII-CODE
   IN ZEICHEN
20 INPUT "GEBEN SIE EINE ZAHL EIN";AS
30 PRINT CHR$(AS)
40 GOTO 20
RUN
```

Geben Sie eine Zahl kleiner als 0 oder größer als 255 ein, dann reagiert der Computer mit der Meldung:

```
?ILLEGAL QUANTITY ERROR IN 30
```

Wie können Sie zu einem vorgelegten Zeichen den entsprechenden ASCII-Code bestimmen? Das nachfolgende Programm beantwortet diese Frage:

```
NEW
10 REM UMWANDLUNG VON ZEICHEN IN
   ASCII-CODE
20 PRINT "BETAETIGEN SIE IRGENDEINE TASTE"
30 GET A$: IF A$="" THEN 30
40 CH=ASC(A$)
50 PRINT A$" HAT DEN ASCII CODE "CH
60 GOTO 30
```

Die ASC-Funktion liefert den ASCII-Code für jedes Zeichen. Lassen Sie das Programm laufen und bedienen Sie die Funktionstasten, <HOME>, <SHIFT CLR/HOME>, <RUN/STOP> und <SHIFT RUN/STOP>.

Beim Schreiben von Programmen ist es oft vorteilhaft, Zeichenreihen zusammfügen zu können, was man auch als »Konkatenation« bezeichnet. Das nachfolgende Programm demonstriert diesen Vorgang:

```
NEW
10 REM KONKATENATION
20 A$="KON"
30 B$="KAT"
40 C$="EN"
50 D$="ATI"
60 E$="ON"
70 F$=A$+B$+C$+D$+E$
80 PRINT F$
RUN
KONKATENATION

READY.
```

Die Variable F$ erhalten Sie, wenn Sie A$, B$, C$, D$ und E$ zusammenfügen. Das ist wirklich einfach.
Geben Sie jetzt dieses Programm ein:

```
NEW
10 REM LIEBLINGSFARBE
20 PRINT "NENNEN SIE IHRE
   LIEBLINGSFARBE";C
30 IF C=1 THEN PRINT CHR$(144)
40 IF C=2 THEN PRINT CHR$(5  )
50 IF C=3 THEN PRINT CHR$(28 )
60 IF C=4 THEN PRINT CHR$(159)
70 IF C=5 THEN PRINT CHR$(156)
80 IF C=6 THEN PRINT CHR$(30 )
90 IF C=7 THEN PRINT CHR$(31 )
100 IF C=8 THEN PRINT CHR$(158)
```

```
110 PRINT "DAS IST DIE FARBE"C
120 GOTO 20
RUN
```

Es gibt noch eine Möglichkeit, die gleiche Aufgabe schneller zu erfüllen:

```
NEW
10 REM LIEBLINGSFARBE VERSION 2
20 INPUT "NENNEN SIE IHRE
   LIEBLINGSFARBE";C
30 ON C GOTO 40, 50, 60, 70, 80, 90,
   100, 110
40 PRINT CHR$(144):GOTO 120
50 PRINT CHR$(5 ):GOTO 120
60 PRINT CHR$(28 ):GOTO 120
70 PRINT CHR$(159):GOTO 120
80 PRINT CHR$(156):GOTO 120
90 PRINT CHR$(30 ):GOTO 120
100 PRINT CHR$(31 ):GOTO 120
110 PRINT CHR$(158)
120 PRINT "DAS IST DIE FARBE"C
130 GOTO 20
```

Beachten Sie dabei die »ON C GOTO 40, 50, 60, ...«-Anweisung in Zeile 30. Bei ihrer Ausführung bestimmt der Computer in Abhängigkeit des Wertes von C die Zeile, die er als nächste abarbeitet. Ist C gleich 1, dann springt der Rechner nach Zeile 40, ist C gleich 2, dann geht er nach Zeile 50 usw. Übersteigt der Wert von C die Anzahl der Springadressen, dann arbeitet der Computer die nächste Zeile ab.
Der GOSUB-Befehl arbeitet auf die gleiche Weise. Wie Sie weiter unten sehen werden, kann man damit Programme weiter verkürzen. Die Anweisung ON GOSUB arbeitet wie

GOSUB, jedoch kann man damit den Sprung zu mehreren Zeilen programmieren.

Übungen

8.1. Geben Sie eine Zeichenreihe mit Hilfe einer INPUT-Anweisung ein und schreiben Sie den Bildschirm mit dieser Zeichenreihe (horizontal) voll.

8.2. Versuchen Sie, Übung 8.1. mit graphischen Symbolen durchzuführen.

8.3. Fragen Sie »Wollen Sie weitere Anweisungen (J oder N)?«. Verwenden Sie die GET-Anweisung, um gedrückte Tasten zu erkennen. Ist die Taste J oder N nicht betätigt worden, dann überprüfen Sie weiterhin die Tastatur. Wurde die Taste J gedrückt, dann geben Sie die Nachricht »Betätigen Sie J für Ja oder N für Nein« aus und springen an den Programmanfang zurück. Wurde die Taste N gedrückt, dann beenden Sie das Programm.

8.4. Finden Sie heraus, wie sich der ASCII-Wert einer Taste verändert, wenn Sie diese zusammen mit <CTRL>, <SHIFT> oder <C=> betätigen.

8.5. Drucken Sie eine Tabelle (mit Hilfe einer FOR...NEXT-Schleife) der ASCII-Codeworte von 0 bis 255. Geben Sie in einer Spalte die Zahl an. In einer zweiten Spalte drucken Sie das entsprechende Zeichen aus. Machen Sie sich keine Gedanken, wenn der überwiegende Teil dieser Tabelle aus dem Monitor herausgeschoben wird. Durch Drücken der CTRL-Taste verlangsamen Sie das Herausschieben und können somit die Tabelle kontrollieren.

8.6. Fragen Sie nach einer Zahl zwischen 1 und 15. Verwenden Sie drei verschiedene ON GOTO Anweisungen, von denen jede 5 Zahlen verarbeiten kann. Springen Sie jeweils in eine Zeile, in der die eingegebene Zahl ausgedruckt wird, beispielsweise PRINT 7, wenn die Zahl 7 eingegeben wurde.

9. Feldgraphiken

Felder benötigen Sie in der Regel, um Daten in geordneter Weise aufzulisten. Sie sind jedoch auch bei der Darstellung von Graphik sehr nützlich. Geben Sie das folgende Programm ein, auch wenn Sie es im Moment nicht vollständig verstehen.

```
NEW
10 REM ORANGE KAROS
20 DIM S$(38,21),C$(38,21)
30 POKE 53281,0
40 FOR V=0 TO 21
50 FOR H=0 TO 38
60 GOSUB 120
70 PRINT C$(H,V);S$(H,V);
80 NEXT H
90 PRINT "<CTRL 2>"
100 NEXT V
110 END
120 C$(H,V)="<COMMODORE 1>"
130 S$(H,V)="<SHIFT Z>"
140 RETURN
```

Bewegen Sie den Kursor in die letzte Zeile des Bildschirms, bevor Sie das Programm laufen lassen. Tippen Sie dann RUN ein und betätigen Sie die RETURN-Taste. Der Bildschirm wird sich mit Karos der Farbe Orange füllen.
Zählen Sie die Karos, so stellen Sie fest, daß sich in 22 Zeilen je 39 Karos befinden.
Warum wurden diese Zahlen gewählt? In einer Zeile befinden sich 40 Spalten, die von 0 bis 39 numeriert sind. Das heißt also, in jeder Zeile befindet sich ein Karo weniger, als möglich wäre.

Warum 22 Zeilen? Die Zeilen werden hintereinander geschrieben und von unten nach oben auf dem Bildschirm verschoben. Ist das Programm beendet, führt der Computer einen Zeilenvorschub aus, druckt READY und läßt den Kursor in der nächsten Zeile aufblinken. Die Leer-, die READY- und die Kursorzeile ergeben zusammen 3 Zeilen. Auf dem Monitor sind 25 Zeilen darstellbar, und deshalb wurden der Darstellung genau 25 – 3 = 22 Zeilen Platz eingeräumt.

Betrachten Sie zunächst die Logik des Programms, bevor Sie sich den Einzelheiten zuwenden. Es werden zehn verschiedene, zweidimensionale Felder verwendet. Ein Feld (S$) speichert die Zeichen, das andere (C$) die Farben, wobei jedes Feld über 22 Zeilen und 39 Spalten verfügt. Das bedeutet, daß für jede Position auf dem Bildschirm genau ein Element in jedem Feld vorhanden ist.

Zeile 20 des Programms reserviert den Speicherplatz für die Felder. Das Schlüsselwort DIM steht für Dimension, Zeile 20 beinhaltet somit eine Dimensions-Anweisung. Durch die Angabe von zwei Zahlen in den Klammern werden zweidimensionale Felder definiert; die zwei Indeces reichen aus, um jedes Element des Feldes anzusprechen. Der erste Index läuft von 0 bis 38, der zweite von 0 bis 21. Zeile 30 färbt den Hintergrund schwarz, und in Zeile 40 beginnt eine FOR...NEXT-Schleife, mit der die vertikale Koordinate V kontrolliert wird. V nimmt Werte zwischen 0 und 21 an. Die dazugehörige NEXT-Anweisung steht in Zeile 100. Zeile 50 weist der horizontalen Koordinate H Werte von 0 bis 38 (für jedes V) zu. Die Schleife wird durch die NEXT-Anweisung in Zeile 80 geschlossen.

Die Anweisung in Zeile 60 haben Sie bis jetzt noch nicht kennengelernt. Das Kommando GOSUB 120 bewirkt einen ähnlichen Effekt wie eine GOTO-Anweisung. Trifft der Computer beim Abarbeiten des Programms auf diese Anweisung, dann springt er in Zeile 120 und führt das dort beginnende Unterprogramm, auch als SUBroutine be-

zeichnet, aus. Enthält das Unterprogramm eine RETURN-Anweisung, dann geht der Rechner von dort aus wieder in Zeile 60 zurück. Stellen Sie sich einfach vor, daß alle Anweisungen, beginnend mit Zeile 120, bis zur nächsten RETURN-Anweisung zwischen die Zeilen 60 und 70 eingefügt werden. Sie können den Sprung zu einem solchen Unterprogramm von mehreren Stellen eines Programms aus mit einer GOSUB-Anweisung vorprogrammieren. Dadurch ersparen Sie sich viel Zeit beim Eintippen des Programms.

In Zeile 70 werden die gewünschten Zeichen in die Felder für die Farbe (C$(H,V)) und den Bildschirm (S$(H,V)) eingetragen. Die einzelnen »Zellen« der Felder legen Sie durch die Zeilennummer (Wert von H) und die Spaltennummer (Wert von V) fest.

Die Anweisung NEXT H in Zeile 80 kennen Sie schon. Entweder bekommt H den nächsten Wert zugewiesen oder die Schleife wird beendet.

In Zeile 90 wechselt die Farbe des Kursors zu Weiß, und der Kursor selbst springt an den Anfang der nächsten Zeile. Nur beim letzten Durchlaufen der V-FOR-NEXT-Schleife bemerken Sie den Farbwechsel. In allen anderen Fällen werden die Symbole in der nächsten Zeile wieder orange dargestellt. Wichtig ist jedoch der Zeilenvorschub. Die PRINT-Anweisung in Zeile 70 beschreibt eine Bildschirmzeile bis auf das letzte Zeichen. In Zeile 90 wird jetzt dieses letzte Zeichen markiert, und der Kursor springt in die nächste Zeile, da die PRINT-Anweisung kein Semikolon enthält. Ist die Graphik vollständig dargestellt, dann erscheint das READY auf dem Bildschirm wieder in weißer Schrift.

Die V-FOR-NEXT-Schleife wird in Zeile 100 und das ganze Programm durch das END in Zeile 110 beendet. Jetzt meldet sich der Computer durch die oben erwähnte Meldung READY wieder zurück. In Zeile 120 beginnt das Unterprogramm, das ein <Commodore 1> in das C$-Feld und ein

<SHIFT Z in das S$-Feld in der Zeile schreibt, die durch Zeile V und Spalte H bestimmt wird. Zeile 120 trägt die Zeichenfolge Orange in das C$-Feld ein, Zeile 130 schreibt ein Karo in das S$-Feld. Die RETURN-Anweisung in Zeile 140 bewirkt einen Sprung an die Stelle, von der aus das Unterprogramm aufgerufen wurde; dies geschah in Zeile 60. Dort wird dann die Abarbeitung des Programms fortgesetzt.
Wollen Sie nun die Ausgabe eines Feldgraphikprogramms ändern, dann müssen Sie nur das Unterprogramm ändern, durch das die Werte in die Felder eingetragen werden. Modifizieren Sie das Unterprogramm in folgender Weise:

```
10 REM GRUENE DREIECKE
30 POKE 53281,2
120 C$(H,V)="<Commodore 6>"
130 S$(H,V)="<Commodore *>"
140 IF H=V then S$(H,V)=" "
    :REM DAS IST EIN LEERZEICHEN
150 RETURN
```

Zeile 30 färbt den Bildhintergrund rot, Zeile 120 legt die Zeichenfarbe Hellgrün fest. In Zeile 130 wird ein dreieckiges Symbol in S$(H,V) eingetragen, und Zeile 140 überprüft die Gleichheit von V und H. Wenn H=V ist, wird ein Leerzeichen anstelle des Dreiecks eingetragen. Was bedeutet das? Finden Sie dazu heraus, wann V=H ist. Die Variablen haben den gleichen Wert, wenn V=O, H=O, wofür man auch (O,O) schreibt, sowie bei (1,1), (2,2), (3,3) bis hin zu (21,21). Das heißt, die Diagonale von oben links nach unten rechts auf dem Bildschirm besteht aus Leerzeichen. In Zeile 150 finden Sie nichts Unbekanntes. Sie befiehlt den Rücksprung an die Aufrufstelle des Unterprogramms. Lassen Sie das Programm jetzt laufen.

Hoffentlich sind Sie nicht enttäuscht; aber Sie finden unten ein aufregenderes Beispiel.
Wieder wird nur das Unterprogramm verändert. Geben Sie folgendes ein:

```
10 REM BLAUE X
30 POKE 53281,7
120 C$(V,H)="<CTRL 7>"
130 IF V/2=INT(V/2) THEN C$(V,H)=
    "<Commodore 7>"
140 S$(V,H)="<SHIFT V>"
150 RETURN
```

Dieses Mal füllt sich der Bildschirm mit dem Zeichen SHIFT-V, das wie ein X aussieht. Dabei wechselt die Farbe ständig zwischen Hellblau und Blau.
Nur die Programmzeile 130 sollte für Sie ein kleines Problem darstellen. Sie prüft, ob die Zeilennummer ganzzahlig durch 2 teilbar ist. Die Funktion INT (die Abkürzung steht für »Integer«, auf deutsch: ganze Zahl) nimmt als Argument eine Zahl X und berechnet als Wert die größte ganze Zahl, die kleiner oder gleich X ist. Wenn nun V/2 gleich INT (V/2) ist, so handelt es sich bei V um eine gerade Zahl; gilt beispielsweise V = 3, dann ist V/2 = 1.5 und INT (V/2) = 1. Da 1.5 ungleich 1 ist, folgt daraus, daß 3 ungerade ist. Aber die Hälfte von 4 ist 2, und dies ist gleich INT (4/2). Somit stellt 4 eine gerade Zahl dar.
Deshalb nimmt jede Zeile mit gerader Nummer (0,2,4,6,...20) die Farbe Hellblau, und jede Zeile mit ungerader Nummer die Farbe Blau an. Geben Sie nun das Kommando RUN ein.

Übungen

9.1. Verändern Sie das Feldgraphikprogramm mit dem Symbol X so, daß bei ungeraden Zeilennummern ein rotes und bei geraden Zeilennummern ein weißes Zeichen erscheint. Sie können im Programm auch Zeilen mit Nummern 125 oder 142 verwenden, ohne daß sich dadurch etwas ändert.

9.2. Verändern Sie das Programm 9.1. in folgender Weise: wenn (IF) die horizontale Position kleiner als 12 und (AND) die vertikale Position kleiner als 10 ist, dann (THEN) wird die Farbe Blau gewählt.

9.3. Die Graphik, die bei Übung 9.2. entsteht, sieht wie eine amerikanische Flagge mit zuvielen Streifen aus, bei der die Sterne vergessen wurden. Belassen Sie die Streifen in dieser Form, Sie können ein genaues Abbild der Flagge nicht erzeugen. Es sieht viel besser aus, wenn alle Streifen die gleiche Breite besitzen. Fügen Sie in das Programm (unmittelbar nach der Zeile, die in Übung 9.2. hinzugekommen ist) eine weitere IF-THEN-Anweisung mit folgender Bedeutung ein: Wenn (IF) die Farbe Blau ist, und (AND) die Zeilennummer und (AND) die Spaltennummer ungerade sind, dann (THEN) sollte die Farbe Weiß sein.

9.4. In Übung 9.3. haben Sie 20 Sterne eingefügt. Es gibt aber 50 Staaten in den USA, und jeder wird in der Flagge durch einen Stern repräsentiert. Deshalb benötigen Sie 30 weitere Sterne. Fügen Sie eine weitere IF-THEN-Anweisung ein, die abfragt, ob die Zeilennummer gerade und kleiner als 11 ist. Wenn ja, dann soll die Farbe Weiß gewählt werden. Damit werden 6 mal 5 gleich 30 zusätzliche Sterne erzeugt, die sogar genauso wie auf dem Banner der USA angeordnet sind.

9.5. Ändern Sie schließlich das Symbol SHIFT-V in ein umgekehrtes Leerzeichen ab; es muß im Umkehrmodus sein, damit die Farben dargestellt werden.
Hinweis: Erinnern Sie sich an RVS ON und RVS OFF?

10. Die Verwendung von Funktionen

Im letzten Abschnitt haben Sie viele neue BASIC-Anweisungen kennengelernt. Dazu zählen auch Funktionen, wovon es in BASIC weit mehr gibt als nur ASC (x$) und CHR$(x).

Folgende Liste führt die in Commodore 64 BASIC verfügbaren Funktionen auf:

ABS(X)	Erbringt den Absolutbetrag einer Zahl X, wobei als Ergebnis immer ein positiver Betrag oder eine Null erscheint.
ASC(X$)	Erbringt den ASCII-Code des ersten Zeichens der Reihe X$.
ATN(X)	Erbringt den Arcustangens von X.
CHR$(X)	Erbringt das Zeichen, dessen ASCII-Code X ist. Es handelt sich also um die Umkehrfunktion zu ASC(X$).
COS(X)	Erbringt den Kosinus von X.
EXP(X)	Erbringt die X-te Potenz der Konstante e (2.71827183).
FRE(X)	Erbringt die Anzahl freier Bytes im Speicher.
FNxx(X)	Erbringt das Resultat einer Funktion FNxx, die vom Benutzer durch eine DEF FN Anweisung festgelegt wurde.
INT(X)	Erbringt die ganze Zahl vor dem Dezimalpunkt; die folgenden Stellen werden gestrichen (Zahlen mit negativen Vorzeichen werden dabei dem Betrag nach größer, wie aus dem Beispiel unten hervorgeht).
LEFT$(X$,X)	Erbringt eine Zeichenreihe, die vom String X$ genau X Zeichen, von links gesehen, erfaßt.

LEN(X$)	Gibt die Anzahl der Zeichen in der Zeichenreihe X$ an.
LOG(X)	Erbringt den natürlichen Logarithmus von X zur Basis e.
MID$(X$,S,X)	Erbringt X Zeichen aus der Zeichenreihe X$, beginnend mit dem S-ten Zeichen von links gelesen.
PEEK(X)	Erbringt den Inhalt der Speicheradresse X, wobei X im Bereich von 0–65535 liegt.
POS(X)	Erbringt die Position in der Zeile des Bildschirms (Zeile 0–39), an der die nächste PRINT-Anweisung ausgeführt wird.
RIGHT$(X$,X)	Erbringt X Zeichen aus der Zeichenreihe X$, von rechts gelesen.
RND(X)	Erbringt eine Zufallszahl. Der Wert des Parameters X wird unten erläutert.
SGN(X)	Erbringt das Vorzeichen von X; bei einer positiven Zahl erscheint das Ergebnis 1, bei einer negativen das Ergebnis –1. Die Zahl Null liefert das Ergebnis 0.
SPC(X)	Die Position, an der die nächste PRINT-Anweisung ausgeführt wird, verschiebt sich um X Zeichen
STR$(X)	Wandelt die Zahl X in eine Zeichenreihe um.
SQR(X)	Erbringt die Quadratwurzel von X.
TAB(X)	Gibt an, in welcher Spalte X das nächste Zeichen ausgegeben wird.
TAN(X)	Erbringt den Tangens von X.
VAL(X$)	Wandelt die Zeichenreihe X$ in eine Zahl um; stellt die Umkehrfunktion zu STR$(X) dar.

Eine ganze Zahl weist nur Nullstellen nach dem Dezimalpunkt auf, beispielsweise 3.00000. Die INT-Funktion wandelt eine Zahl X in die größte ganze Zahl um, die kleiner

oder gleich X ist; die Stellen nach dem Dezimalpunkt werden also gleichsam gestrichen, wobei Zahlen mit negativem Vorzeichen dem Absolutbetrag nach zunehmen, wie folgende Beispiele zeigen:

X	INT(X)
12.3456	12
-12.3456	-13
5.99999	5
6.00001	6
-1	-1

Geben Sie dieses Programm ein:

```
NEW
10 REM DIE INTEGER-FUNKTION
20 INPUT "GEBEN SIE EINE ZAHL EIN";N
30  PRINT "DER INTEGERWERT VON "N" IST
   "INT(N)"."
40 GOTO 20
```

Probieren Sie das Programm mit positiven und negativen Zahlen aus. Dann tippen Sie folgende Zeilen ein:

```
NEW
10 REM INTEGER TEST
20 INPUT "GEBEN SIE EINE ZAHL EIN";N
30 IF N=INT(N) THEN PRINT N"
   IST EINE GANZE ZAHL.":GOTO 20
40 PRINT N" IST KEINE GANZE ZAHL . "
50 GOTO 20
```

Der Absolutbetrag einer Zahl gibt ihren Abstand von der Zahl 0 an. Das heißt, der Absolutbetrag ist immer positiv, auch wenn die betreffende Zahl ein negatives Vorzeichen

aufweist. So ist der Absolutbetrag von –3 gleich 3, und der Absolutbetrag der Zahl 3 ebenfalls gleich 3.
Geben Sie folgendes Programm ein:

```
NEW
10 REM ABSOULTBETRAG
20 INPUT "GEBEN SIE EINE ZAHL EIN";N
30 PRINT "DER ABSOLUTBETRAG VON
   "N" IST "ABS(N)"."
40 GOTO 20
```

Lassen Sie das Programm mit mehreren Zahlen laufen. Tippen Sie dann diese Zeilen ein:

```
NEW
10 REM NEGATIVTEST
20 INPUT "GEBEN SIE EINE ZAHL EIN";N
30 IF N=ABS(N) THEN PRINT N" IST
   NICHT NEGATIV.":GOTO 10
40 PRINT N" IST NEGATIV."
50 GOTO 20
```

Nachdem Sie mit dem Programm ein wenig experimentiert haben, versuchen Sie herauszufinden, warum die Anweisung GOTO 10 am Ende der Zeile 30 steht. Finden Sie dafür keine Erklärung, dann löschen Sie einfach die Anweisung GOTO 10 und beobachten, was dann passiert.
Das Vorzeichen einer Zahl ist entweder negativ oder positiv, die Zahl Null trägt kein Vorzeichen. So erbringt die Funktion SGN(–2001) den Wert –1, SGN(84) den Wert 1 und SGN(0) den Wert 0. Geben Sie folgendes Programm ein:

```
NEW
10 REM DAS VORZEICHEN EINER ZAHL
20 INPUT "GEBEN SIE EINE ZAHL EIN";N
```

```
30 PRINT N" IST ";:REM VERGESSEN SIE
   DIE LEERZEICHEN NICHT
40 ON SGN(N)+2 GOTO 50, 60, 70
50 PRINT "NEGATIV.":END
60 PRINT "NULL.":END
70 PRINT "POSITIV.":END
```

Beachten Sie Zeile 30. Die Funktion SGN erbringt ausschließlich die Werte -1, 0 und 1. Addieren Sie dazu den Betrag 2, so erhalten Sie die Werte 1, 2 und 3, also genau jene Zahlen, die Sie für das Kommando ON GOTO benötigen. Unterbrechen Sie an dieser Stelle kurz die Betrachtung der in Commodore 64 BASIC verfügbaren Funktionen und werfen Sie einen Blick auf die innere Uhr des Commodore 64, die mit den Kommandos TIME und TIME$ angesprochen wird.

Geben Sie das Kommando

```
PRINT TIME
```

ein, dann erhalten Sie eine Zahl, beispielsweise:

```
9528931
```

Die Einheit dieser Zahl ist 1/60 Sekunde. Sie gibt die Zeit an, die seit Einschalten des Geräts vergangen ist. Das Kommando TIME$ rechnet den Betrag in Sekunden, Minuten und Stunden um. Tippen Sie

```
PRINT TIME$
```

ein, und der Computer reagiert beispielsweise mit der Meldung:

```
001527
```

Das bedeutet, Sie arbeiten seit 00 Stunden, 15 Minuten und 27 Sekunden an Ihrer Maschine. Sie können diese Zeitangabe auch in die aktuelle Tageszeit umwandeln, indem Sie

```
TIME$ = "032517"
```

eingeben, wenn die Uhrzeit gerade 3 Uhr 25 Minuten und 17 Sekunden ist. Der Computer verwendet eine 24-Stunden-Uhr. Ist es 3 Uhr nachmittags, müssen Sie daher die Information »152517« eintippen. Der Wert von TIME kann jedoch nicht verändert werden.
Diese Anweisungen sind wichtig, wollen Sie mit Hilfe der RND-Funktion eine Zufallszahl erzeugen oder den Zufallszahlengenerator initialisieren. Ist X in dem Ausdruck RND(X) eine positive Zahl, dann stellt das Ergebnis eine Zufallszahl dar. Ist X gleich 0, dann liefert RND(X) die gleiche Zufallszahl wie beim letzten Aufruf. Handelt es sich bei X jedoch um eine negative Zahl, so wird der Zufallszahlengenerator mit dem Absolutbetrag von X initialisiert. Verwenden Sie stets die gleiche Zahl, so erhalten Sie natürlich immer wieder die gleiche Folge von Zufallszahlen. Um sich verändernde Zufallszahlen zu erzeugen, sollten Sie den Generator durch die Anweisung

```
10 X=RND(-TIME)
```

initialisieren. TIME liefert die Zeit, die Sie bis zur Ausführung der obigen Anweisung am Rechner benötigt haben. Der Wert dieser Anweisung wechselt daher fortwährend. Der Aufruf von RND(1) liefert eine dezimale Zufallszahl, die kleiner als 1 und größer oder gleich 0 ist. Aber genug der Theorie! Geben Sie folgende Zeile ein:

```
NEW
10 REM ZUFALLSZAHLEN
20 X=RND(-TIME)
```

```
30 R=RND(1)
40 PRINT R
50 GOTO 30
```

Versuchen Sie erst gar nicht, ein und dieselbe Zahl zweimal in einer Folge zu finden. Es ist sehr unwahrscheinlich, daß dies vorkommt.
Es gibt Fälle, da wünschen Sie sich vielleicht eine Zahl zwischen 0 und 10, nicht zwischen 0 und 1. Üblicherweise beginnen Denksportaufgaben nicht mit der Anweisung:

```
DENKEN SIE SICH EINE ZAHL
ZWISCHEN 0 UND 1.
```

Ändern Sie deshalb das Programm wie folgt ab:

```
10 X=RND(-TIME)
20 R=RND(1)*10
30 PRINT R,
40 GOTO 20
```

Das sieht schon besser aus, aber die erzeugten Zahlen nehmen Werte zwischen 0 und 9.999999 an; die Zahl 10 selbst ist niemals dabei. Addieren Sie jedoch den Betrag 1 zum Ergebnis, so erhalten Sie Zahlen zwischen 1 und 10.99999. Ändern Sie daher Zeile 20 in folgender Weise:

```
20 R=RND(1)*10+1
```

Natürlich wäre es ideal, die Ziffern hinter dem Komma streichen zu können. Verändern Sie Zeile 20 erneut:

```
20 R=INT(RND(1)*10)+1
```

Endlich erhalten Sie ganze Zufallszahlen zwischen 1 und 10. Es gibt aber natürlich auch eine Formel, mit der Sie

Zufallszahlen zwischen zwei beliebigen Zahlen X und Y erzeugen können:

```
N = INT( RND( 1) * ( Y - X ) + X) + 1
```

Mit Hilfe dieser Anweisung können Sie Zufallsgraphiken erstellen. Geben Sie folgendes Programm ein:

```
NEW
10 REM UNORDNUNG
20 PRINT CHR$(INT(RND(1)*255)+1);
   :GOTO 20
```

Die Funktion SQR(X) zieht die Quadratwurzel von X. Betrachten Sie das nächste Programm:

```
10 INPUT "GEBEN SIE EINE ZAHL EIN";N
20 PRINT SQR(N)" MAL "SQR(N)" IST GLEICH
   "N"."
```

Lassen Sie es mit den Zahlen 20, 50, 100, 144, 625 und 900 laufen. Die Funktionen SIN(X), COS(X), TAN(X) und ATN(X) berechnen den Sinus, den Kosinus, den Tangens und den Arcustangens von X. Zeichnet man SIN(X) und COS(X) in ein Koordinatensystem, dann entsteht ein wellenförmiges Gebilde, das man in verschiedenen Graphiken gut verwenden kann, wie Sie gleich sehen werden.

Geben Sie dieses Programm ein:

```
NEW
10 REM SINUSWELLE
20 FOR I = 0 TO 43.75 STEP .25
30   S=17+17*SIN(I)
40   A$="WELLE"
50   IF I*2 = INT(I*2) THEN A$="SINUS"
```

```
60   PRINT TAB(S);A$
70 NEXT I
```

Das ist mit Abstand die beste Graphik, die Sie bis jetzt gezeichnet haben. Aber was bedeutet der Ausdruck STEP (»Stufe«) in der FOR-NEXT-Schleife? Er teilt dem Computer mit, daß der Wert von I bei jedem Schleifendurchgang um .25 und nicht um 1 erhöht wird. In diesem Fall »zählt« die Schleife 0, .25, .5, .75, 1, 1.25 usw. Durch diese feinere Abstufung erhalten Sie eine exaktere Darstellung der Wellenform. Versuchen Sie es zur Unterscheidung einmal ohne den STEP-Befehl. Von nun an ist eine FOR-NEXT-Schleife eine FOR-NEXT-STEP-Schleife.
Wenn Sie später hochauflösende Graphiken zeichnen, dann konstruieren Sie mit Hilfe von SIN und COS einen Kreis. Schon im letzten Kapitel haben Sie die Funktionen ASC(X$) und CHR$(X) zur Darstellung der Beziehung zwischen ASCII-Zeichen und ASCII-Code benutzt. Sie bedürfen deshalb hier keiner weiteren Erläuterung.
Dagegen kennen Sie die Funktion LEN(X$) noch nicht; sie liefert die Anzahl der Zeichen, die die Zeichenkette X$ enthält. Lassen Sie folgendes Programm laufen:

```
NEW
10 REM LAENGENFUNKTION
20 INPUT "GEBEN SIE EINE ZEICHENREIHE
   EIN";A$
30 PRINT N$" IST "LEN(N$)" ZEICHEN
   LANG."
40 GOTO 20
```

Die Funktion LEFT$(X$,X) erbringt eine Zeichenkette, die vom String X$ genau X Zeichen, von links gelesen, erfaßt; entsprechend liefert die Funktion RIGHT$(X$,X) von rechts gelesen X Zeichen der Zeichenkette X$. Zeichenketten lassen sich aber noch auf eine andere Art verändern, die

vor allem bei Geschäftsprogrammen und bei Spielen häufig verwendet werden. Testen Sie dieses Programm:

```
NEW
10 REM LINKS MITTE RECHTS
20 INPUT "GEBEN SIE EINE ZEICHENREIHE
   EIN";N$
30 PRINT LEFT$(N$,5)" SIND DIE ERSTEN 5
   ZEICHEN VON LINKS"
40 PRINT RIGHT$(N$,5)" SIND DIE ERSTEN 5
   ZEICHEN VON RECHTS"
50 PRINT MID$(N$,3,2)" SIND DAS 3. UND
   4. ZEICHEN"
60 GOTO 20
```

Darüber hinaus können Sie auch Zahlen in Zeichenreihen und umgekehrt verwandeln. Die Funktion STR$(X) liefert eine Zeichenreihe in der Form, als ob X mittels einer PRINT-Anweisung ausgedruckt worden wäre. VAL(X$) wandelt X$ in eine Zahl um. Ist das erste Zeichen von X$ keine Ziffer, dann ist der Wert von VAL(X$) gleich 0. Lassen Sie die folgenden Programme laufen:

```
10 INPUT "GEBEN SIE EINE ZAHL EIN";N
20 N$=STR$(N)
30 IF LEN(N$)<10 THEN PRINT LEFT$
   ("          ",10-LEN(N$));
40 PRINT N$
```

```
10 INPUT "GEBEN SIE EINE ZEICHENREIHE
   EIN";S$
20 PRINT "DER WERT DAVON IST "VAL(S$)
```

Natürlich können Sie auch eigene Funktionen definieren. Betrachten Sie dieses Beispiel:

```
10 DEF FNS(X)=17+17*SIN(X)
20 FOR I = 0 TO 43.75 STEP .25
30   A$="WELLE"
40   IF I*2=INT(I*2) THEN A$="SINUS"
50   PRINT TAB(FNS(I));A$
60 NEXT I
```

In Zeile 10 definieren Sie eine Funktion mit dem Namen FNS. Die Funktion verwendet intern die Variable X, sie hat keinerlei Auswirkungen auf eine im Programm verwendete Variable gleichen Namens. Die Funktion akzeptiert jede Zahl, berechnet das Produkt aus 17 und dem Sinus dieser Zahl und addiert dazu 17. Das ist eine Formel für eine Sinuswelle. Lassen Sie das Programm laufen, und Sie werden auf dem Monitor eine Sinuswelle erhalten.
Zeile 50 verwendet die Funktion FNS, um die Zeichenreihe A$ TABellarisch anzuordnen. In einem Programm kann eine vom Benutzer definierte Funktion beliebig oft aufgerufen werden.
Abschließend finden Sie noch einige Beispiele für Funktionsdefinitionen zusammen mit ihrer Bedeutung:

Funktion	Bedeutung
`DEF FNPI(X)= ATN(1)*4*I`	Sie erhalten den Ausdruck
`DEF FNRA(DG)= DG*FNPI(1)/180`	Sie wandelt einen Winkel DG in Bogenmaß um.
`DEF FNRN(X)= INT(RND(1)*X)+1`	Sie erbringt Zufallszahlen zwischen I und X.

Übungen

10.1. Berechnen Sie die folgenden Funktionswerte zuerst ohne, dann, zur Überprüfung, mit dem Rechner:
Integerwert von 3.29 und −3.29
Absolutwert von 15 und −15
Vorzeichenwert von 14, −91 und 0.

10.2. Schreiben Sie ein Programm, das Zufallszahlen zwischen 1 und 100 einschließlich der Grenzen erzeugt.
10.3. Berechnen Sie die Hälfte der Quadratwurzel von 2.
10.4. Welchen Betrag erhalten Sie, wenn Sie den Arcustangens von 1 mit der Zahl 4 multiplizieren.
10.5. Schreiben Sie ein Programm, das eine Nachricht (eine Zeichenkette) mittels eines INPUT-Befehls einliest. Die Nachricht soll eine Länge von 40 Zeichen aufweisen; ist dies nicht der Fall, verwenden Sie entsprechend viele Leerzeichen. Merken Sie sich die Länge der eingegebenen Nachricht in einer Variablen, da Sie diese Größe in den folgenden Übungen benötigen werden.
10.6. Erweitern Sie das Programm von Übung 10.5. wie folgt: Erstellen Sie eine neue Zeichenkette aus den ersten 40 Zeichen der eingegebenen Nachricht. Drucken Sie die neue Zeichenreihe in der obersten Zeile Ihres Bildschirms aus (verwenden Sie dabei die HOME, nicht die CLR-Taste).
10.7. Erweitern Sie das Programm von Übung 10.6. wie folgt: Es sei I die Länge der eingegebenen Zeichenreihe. Verändern Sie diese Zeichenreihe so, daß I-1 Zeichen von rechts und dann das erste Zeichen von links zu einer Zeichenreihe zusammengefaßt werden. Verfahren Sie mit der so erhaltenen Zeichenreihe wie in Übung 10.6. Sie werden eine andere graphische Form erhalten.

11. Die Funktionen PEEK und POKE

Der Commodore 64 verfügt, wie viele andere Mikrocomputer seiner Klasse, über 65536 Speicherzellen, die von 0 bis 65535 durchnumeriert sind. Durch die Funktion POKE haben Sie bereits die Inhalte der Speicherzellen 646 (Zeichenfarbe), 53280 (Randfarbe) und 53281 (Hintergrundfarbe) kennengelernt.
Die kleinste Informationsmenge, die ein Computer verarbeiten kann, ist, wie Sie bereits aus Kapitel 5 wissen, ein Bit: es besitzt entweder den Wert 0 oder 1. Ein Zahlensystem, das nur diese beiden Werte kennt, wird auch als binäres Zahlensystem bezeichnet. Unterschiedliche Information kennzeichnet ein Computer nun dadurch, indem er Bits auf verschiedene Weise miteinander kombiniert; entscheidend in diesem Zusammenhang ist die Frage, welche Anzahl an Bits ein Computer gleichzeitig handhaben kann. Beim Commodore 64 handelt es sich um einen 8-Bit-Computer, womit ausgedrückt wird, daß er 8 Bit zugleich miteinander kombinieren kann. Wieviele Kombinationsmöglichkeiten entstehen dadurch? Richtig, genau 256 (2^8) Kombinationen, die von 0 bis 255 durchnumeriert sind. Jede einzelne Speicherzelle kann somit eine Zahl zwischen 0 und 255, also ein Zeichen mit einer Länge von 8 Bit, speichern; um größere Zahlen im Speicher abzulegen, sind entsprechend mehr Speicherzellen notwendig.
Mit Hilfe der Funktion PEEK erhalten Sie die Inhalte der Speicherzellen Ihres Computers. Versuchen Sie folgendes Beispiel:

```
NEW
10 REM SPEICHERAUSZUG
20 FOR I = 0 TO 65535
30   M = I:IF I>32767 THEN M = I-65536
```

```
40   B = PEEK(M)
50   PRINT B, CHR$(B)
60 NEXT I
```

Wenn Sie genügend Geduld aufbringen, erhalten Sie auch die Werte der Zellen ausgegeben, in denen sich das BASIC-Programm befindet. Sie stoßen dabei auf die Meldungen READY und ?SYNTAX ERROR, die natürlich auch im Speicher liegen.
Werden die Zellen ab Nummer 2048 durchlaufen, so sollte Ihnen außer den unlesbaren Schlüsselwörtern, wie FOR, TO, PRINT, +, = oder CHR$, alles bekannt sein.
Sie haben bereits gelernt, die Zeichenfarbe auf dem Bildschirm zu verändern. Dazu speichern Sie mittels der POKE-Funktion eine bestimmte Zahl im Arbeitsspeicher Ihres Computers ab. Jedoch können Sie auch den Inhalt des Bildschirms durch die POKE-Funktion verändern. Untenstehende Tabelle gibt eine Zusammenstellung aller Möglichkeiten, die Ihnen die Funktion POKE bietet:

Mit der POKE-Funktion durchführbare Änderungen:

Änderung	Speicherzelle	Inhalt
Zeichenfarbe	646	(0–15) Farbe
Bildschirm-RAM	1024–2023	ASCII Code (0–255)
Randfarbe	53280	(0–15) Farbe
Hintergrundfarbe	53281	(0–15) Farbe
Farb-RAM	55296–56295	(0–15) Farbe

Sie wissen schon alles über die Wahl der Zeichen-, Rand- und Hintergrundfarbe; betrachten Sie daher nun den Bildschirm selbst. Er besteht aus 25 Zeilen mit je 40 Spalten, was zusammen 1000 Elemente ergibt. Die Speicherzelle 1024 entspricht dem Element in Zeile 0 und Spalte 0, Zelle 1025 dem Element in Zeile 0 und Spalte 1, bis hin zur Speicherzelle 2023, die dem Element in Zeile 24 und Spalte 39 entspricht. Ist R die Zeilen- und C die Spaltennummer

eines Bildschirmelements, dann erhalten Sie die entsprechende Speicherzelle durch die Formel:

```
ME = 1024 + R * 40 + C
```

Geben Sie das folgende Programm ein:

```
NEW
10 REM BILDSCHIRM-POKE
20 PRINT "<SHIFT CLR/HOME>"
30 INPUT "<CLR/HOME> ZEILE      ";R
40 INPUT "<CLR/HOME> SPALTE     ";C
50 INPUT "<CLR/HOME> ASCII CODE";A
60 POKE 1024 + R * 40 + C, A
70 GOTO 20
```

Verwenden Sie eine Zeilennummer kleiner als 16 sowie die Spaltennummer 0. Sie bemerken, daß das Zeichen gelöscht wird, sobald die Fragen erneut gestellt werden. Geben Sie niemals eine Spaltennummer kleiner als 0 oder größer als 39 oder eine Zeilennummer kleiner als 0 oder größer als 24 ein. Sie könnten dadurch Ihr Programm zerstören! Die POKE-Funktion würde dann Werte in Speicherzellen eintragen, die nicht im Bildschirm-RAM liegen. Aber jede Speicherzelle hat eine ganz bestimmte Aufgabe. Sie können die Ausgabe auch farbig gestalten.
Führen Sie folgende Änderungen durch:

```
50 INPUT "<CLR/HOME> FARBE      ";FARBE
70 POKE 55296 + R * C, FARBE
```

Wählen Sie Farbwerte zwischen 0 und 15. Bleibt der Wert unter 0, so erhalten Sie die Meldung ?ILLEGAL QUANTITY ERROR. Ist der Wert größer als 15, aber kleiner als 256, dann wird von der eingegebenen Zahl 15 abgezogen und der neue Wert als Farbcode genommen.

Der Farb-RAM beginnt bei Speicherzelle 55296 und endet bei Zelle 56295. In diesem Bereich werden lediglich Werte zwischen 0 und 15 gespeichert. Versuchen Sie einmal, einen Wert, der größer als 15 ist, mit der POKE-Funktion ein- und mit der PEEK-Funktion wieder auszulesen. Sie werden immer eine Zahl zwischen 0 und 15 erhalten.
Was haben Sie nun davon, daß Sie mit Hilfe der POKE-Funktion etwas auf den Monitor schreiben können? Sie hätten ja auch eine PRINT-Anweisung benutzen können. Haben Sie jedoch schon bemerkt, daß sich bei Verwendung von POKE der Kursor nicht weiterbewegt? Dadurch können Sie an beliebiger Stelle und in beliebiger Farbe etwas auf den Bildschirm schreiben, ohne den sonstigen Inhalt des Bildschirms zu beeinflussen. Allerdings läßt sich mittels einer POKE-Anweisung immer nur genau ein Bildschirmelement ändern.
Das sieht wie eine Aufgabe für eine FOR-NEXT-Schleife aus. Wie wäre es mit einer Zufallsgraphik? Versuchen Sie dieses Beispiel:

```
NEW
10 REM 10 X = INT( RND( 1 ) * 40)
30 Y = INT( RND( 1 ) * 25)
40 COLOR = INT( RND( 1 ) * 16)
50 CH = INT( RND( 1 ) * 256)
60 POKE 1024 + X + Y * 40, CH
70 POKE 55296 + X + Y + 40, COLOR
80 GOTO 10
```

Die Werte für die Spaltennummer (X), die Zeilennummer (Y), die Farbe (C) und das Zeichen (CH) werden als Zufallszahl bestimmt. Dann tragen Sie das Zeichen in den Bildschirm-RAM und die Farbe in den Farb-RAM mittels der POKE-Funktion ein. Sofort beginnt der ganze Vorgang erneut. Lassen Sie das Programm laufen.

Übungen

11.1. Verändern Sie unter Verwendung einer FOR-NEXT-Schleife die Rand- und Hintergrundfarbe von Schwarz zu Grau 3. Um die Farben einen kurzen Augenblick lang sehen zu können, müssen Sie eine Verzögerungsschleife einbauen. Stellen Sie dann wieder die normalen Farben ein, also dunkelblauer Hintergrund und hellblauer Rand.

11.2. Verändern Sie die Farbe des Kursors und fragen Sie danach den Farbwert ab, der die Werte 2 oder 6 besitzt, nicht »rot« oder »blau«. Die Anwort soll »richtig« oder »falsch« sein.

11.3. Finden Sie die Anzahl der Kombinationen von Hintergrund- und Randfarbe heraus. Erhalten Sie das nicht korrekte Ergebnis 255, dann überprüfen Sie Ihr Vorgehen erneut.

11.4. Welche Kombination empfinden Sie als die angenehmste?

11.5. Schreiben Sie mit der POKE-Funktion über 16 Zeilen ein X im Umkehrmodus auf den Bildschirm. Die Zeilen sollen abwechselnd die Farben Schwarz und Weiß annehmen.
Hinweis: Um ein Zeichen im Umkehrmodus mit der POKE-Funktion darstellen zu können, addieren Sie den Wert 128 zum ASCII-Code des Zeichens.

12. Sprite-Graphiken

Bis jetzt waren Ihre graphischen Objekte auf dem Bildschirm fixiert. Sie konnten nicht einfach einen Ball oder ein Herz über den Monitor schieben. Haben Sie einmal versucht, mehrere Zeichen gleichzeitig zu bewegen, dann war das schwierig und dauerte zudem noch sehr lange. Ein »Sprite« löst beide Probleme.

Unter einem Sprite versteht man einen beweglichen Objektblock (MOB für »moveable object block«), der aus einer Zeichnung oder aus einem Bild besteht. Eine solche Zeichnung kann an jeden beliebigen Ort auf dem Bildschirm verschoben werden, ohne irgendwelche andere Zeichen auf dem Monitor zu verändern.

Ein Sprite weist eine Höhe von 21 und eine Breite von 24 Pixeln auf. Jedes dieser Pixel kann »an« oder »aus« sein. Zudem stehen Ihnen für den gesamten Sprite vier verschiedene Farben zur Verfügung.

Versuchen Sie einmal, das Emblem der Firma Commodore zu zeichnen.

Übersetzen Sie jetzt Ihre Zeichnung in BASIC-Anweisungen. Verwenden Sie für jede Zeile der Zeichnung eine DATA-Anweisung. Stellen Sie einen eingeschalteten Pixel durch ein 0 und einen ausgeschalteten Pixel durch ein Leerzeichen dar. Sie erhalten ein Programm ähnlich dem Beispiel auf Seite 69.

Sichern Sie das Programm unter dem Namen SPRITE0. Sollte Ihnen anschließend ein Fehler unterlaufen, dann müssen Sie diese Anweisungen nicht erneut eingeben. Tippen Sie die folgenden Anweisungen Zeile für Zeile ein und lassen Sie nach jeder Zeile das Programm mit RUN laufen. Überzeugen Sie sich, daß jede Anweisung gemäß der Beschreibung ausgeführt wird, bevor Sie zu der nächsten Zeile übergehen.

```
NEW
099 REM  ,SPRITE0.................
100 DATA"                        "
110 DATA"                        "
120 DATA"                        "
130 DATA"                        "
140 DATA"                        "
150 DATA"          0000000       "
160 DATA"         00000000       "
170 DATA"        000      0000    "
180 DATA"       000       000     "
190 DATA"       000              "
200 DATA"       000       000     "
210 DATA"        000      0000    "
220 DATA"         00000000       "
230 DATA"          0000000       "
240 DATA"                        "
250 DATA"                        "
260 DATA"                        "
270 DATA"                        "
280 DATA"                        "
290 DATA"                        "
300 DATA"                        "
310 REM  123456789012345678901234
```

Aus Gründen der Übersichtlichkeit wurde bei diesem Programm jede Zeile in ihre individuellen Kommandos unterteilt. Geben Sie nicht gewohnheitsgemäß das Kommando NEW ein, da Sie sonst alle DATA-Anweisungen löschen.

```
10 PRINT "<SHIFT CLR/HOME>" :
FOR I = 0 to 62 :
POKE 832 + I, 0 :
NEXT I
```

Der Sprite 0 belegt in diesem Programm den Speicherbereich von 832 bis 894. In Zeile 10 wird dieser Bereich mit Nullen vorbelegt. Im allgemeinen kann der Sprite an beliebiger Stelle in den Speicher eingetragen werden.

```
20 FOR R = 0 to 20 :
   READ S$ :
   FOR C = 0 TO 2 :
   T = 0 :
   FOR P = 0 to 7 :
   B = 0
```

In Zeile 20 beginnen drei Schleifen mit den folgenden Aufgaben: Eine Datenzeichenreihe wird mit dem READ-Kommando eingelesen, die 24 Zeichen werden in drei Mengen zu je 8 Zeichen zerlegt und jede dieser Mengen wird in eine Zahl zwischen 0 und 255 umgewandelt. Die Variablen T und B stellen Abkürzungen für »Total« und »Bit« dar. Ein Bit kann, wie gesagt, den Wert 0 oder 1 annehmen. In Zeile 40 wird bei jedem Schleifendurchlauf die Variable B mit 2 (7-J) multipliziert und zu T addiert.

```
30 IF MID$(S$,P+C*8+1)<>" "THEN B=1
```

In Zeile 30 wird der Variablen B der Wert 1 zugewiesen, wenn das Zeichen in der Zeichenreihe kein Leerzeichen darstellt.

```
40 T = T+B*2↑(7-J) :
   NEXT P :
   POKE 832+R*3+C,T :
   NEXT C :
   NEXT R
```

Nach der Ausführung von Zeile 40 hat T (Total) einen Wert zwischen 0 und 255, die Schleife P (Position des Bits) ist beendet, der Gesamtwert von T (die Summe aller Bits) ist

in den Speicherbereich des Sprites 0 eingetragen, und die Schleifen C (»Column«; auf deutsch: Spalte) und R (»Row«; auf deutsch: Zeile) sind beendet.

```
50 BA = 53248:
 POKE BA + 21, 1:
 POKE BA + 39, 14:
 POKE 2040, 13
```

In Zeile 50 wird die Variable BA (BAsis des Sprite-Registers) mit 53248 belegt. BA+21 ist das Sprite-Zustandsregister, das den Wert »ein« oder »aus« aufnimmt. In dieses Register wird mit der POKE-Funktion der Wert 1 (»ein«) eingetragen. Die Speicherzelle 2040 bestimmt, unter welcher Adresse die Spritedaten gespeichert werden sollen — in unserem Beispiel in Block 13, oder zwischen 832 und 894.

```
60 POKE BA + 23, 1:
   POKE BA + 29, 1
```

In Zeile 60 wird eine 1 in die entsprechenden Register eingetragen. Die Höhe und die Breite des Sprites werden dadurch verdoppelt.

```
70 FOR X = 1 TO 200
80 POKE BA, X :
   POKE BA + 1, X
90 NEXT :
```

In den Zeilen 70 bis 90 wird der Sprite von der oberen linken Ecke des Bildschirms (Position (0,0)) nach (200,200) verschoben. Die Speicherzelle BA enthält die X-Koordinate und die Speicherzelle BA+1 die Y-Koordinate des Sprite 0.
Mit dem letzten RUN haben Sie das vollständige Pro-

gramm gestartet. Sichern Sie es jetzt unter dem Namen SPRITE DEMO 1.

Eine schnellere Version

Nehmen Sie an Ihrem SPRITEDEMO 1-Programm folgende Veränderungen vor:

```
10 PRINT "<SHIFT CLR/HOME>"
20 FOR R = 0 TO 62 :
   READ T :
   POKE 832 + R, T :
   NEXT
30 REM STREICHEN
40 REM STREICHEN
100 DATA 0 , 0 , 0
110 DATA 0 , 0 , 0
120 DATA 0 , 0 , 0
130 DATA 0 , 0 , 0
140 DATA 0 , 127 , 0
150 DATA 0 , 255 , 0
160 DATA 1 , 192 , 240
170 DATA 3 , 128 , 224
180 DATA 3 , 128 , 0
190 DATA 3 , 128 , 224
200 DATA 1 , 192 , 240
210 DATA 0 , 255 , 0
220 DATA 0 , 127 , 0
230 DATA 0 , 0 , 0
240 DATA 0 , 0 , 0
250 DATA 0 , 0 , 0
260 DATA 0 , 0 , 0
270 DATA 0 , 0 , 0
280 DATA 0 , 0 , 0
290 DATA 0 , 0 , 0
300 DATA 0 , 0 , 0
RUN
```

Diese Version läuft schneller, aber woher kommen alle diese Zahlen? Sie haben lediglich die Umwandlung des Bildes in Zahlen von Hand ausgeführt und nicht, wie in der ersten Version, dem Programm überlassen. Verwenden Sie zum Arbeiten mit Sprite-Graphiken das im folgenden beschriebene Schema.
Jede Zeile des Bildes wird in 3 Byte unterteilt, wobei sich in jedem Byte maximal 8 Punkte befinden. Jeder Position im Byte ist eine Zahl zugeordnet, nämlich, von rechts beginnend, die Zahlen 1, 2, 4, 8, 16, 32, 64 und 128. Befinden sich nun beispielsweise Punkte in den Spalten 128, 64 und 32, dann nimmt die Zahl für dieses Byte den Wert 128+64+32 = 224 an. Ist überhaupt kein Punkt im Byte vorhanden, dann ist diese Zahl gleich 0. Sind alle 8 Punkte im Byte, dann ist die Zahl gleich 128+64+32+16+8+4+2+1 = 255. Sie erhalten damit den niedrigsten und den höchsten Wert eines Bytes.
In der DATA-Anweisung wurden zur Steigerung der Lesbarkeit jeweils drei Zahlen in eine Zeile geschrieben. Sie können aber auch mehr Zahlen in eine Zeile setzen und damit Speicherplatz einsparen. Probieren Sie es selbst aus.
Was können Sie sonst noch mit Sprites unternehmen? Wie wäre es mit einem Farbwechsel? Die Farbe von Sprite 0 wird durch den Inhalt der Speicherzelle BA+39 bestimmt.
Führen Sie diese Veränderungen aus:

```
25 BA = 53248 :
   POKE BA + 21, 1 :
   POKE BA + 39, 14 :
   POKE 2040, 13
30 POKE BA + 23, 1 :
   POKE BA + 29, 1
40 FOR C = 0 TO 15
```

```
50 POKE BA + 39, C
60 FOR X = 0 TO 11
70 POKE BA, C * 12 + X :
   POKE BA + 1, C * 12 + X
80 NEXT
90 NEXT
```

Jetzt wechselt der Sprite nach jeweils 12 Schritten die Farbe. Können Sie aber auch die Größe des Sprites verändern? Schon in Zeile 30 haben Sie ihn auf seine vierfache Größe ausgedehnt, indem Sie seine Breite und seine Höhe jeweils verdoppelten. Streichen Sie Zeile 30 und lassen Sie das Programm erneut laufen.
Jetzt erscheint der Sprite wesentlich kleiner. Experimentieren Sie ein wenig und lassen Sie den Sprite breiter, aber nicht höher werden oder umgekehrt.
In diesem Bildschirmbereich können Sie den Sprite verschieben:

X-Achse (horizontal): 0 – 347(24 – 347 sichtbar)
Y-Achse (vertikal): 0 – 255(50 – 250 sichtbar)

255 ist die größte Zahl, die mit der POKE-Funktion in eine Speicherzelle eingetragen werden kann. Wie bewegen Sie dann den Sprite in eine X-Position, deren Wert den der Zahl 255 übersteigt? Sie sagen einfach dem Computer, er soll den Sprite in den nächsten Bildschirmabschnitt setzen und ihn dort bewegen. Verändern Sie dazu Ihr Programm wie folgt:

```
60 FOR X = 0 TO 19
70 XP = C * 20 + X :
   YP = XP / 2 :
   GOSUB 500
```

Zeile 70 weist der Variablen XP den Wert von C mal 20 plus X zu; die Variable steht für »Color« (Farbe). Als Wert von YP

wird die Hälfte des Wertes von XP festgesetzt. Dann rufen Sie ein in Zeile 500 beginnendes Unterprogramm auf.

```
499 END
```

Zeile 499 verhindert im Fehlerfall eine Ausführung des Unterprogramms. Diesen Schutzmechanismus sollten Sie in jedes Programm einbauen.

```
500 IF YP < 0 OR YP > 255 THEN STOP
```

In Zeile 500 wird überprüft, ob sich der Wert von YP im erlaubten Bereich der POKE-Funktion befindet. Ist das nicht der Fall, dann STOPpt die Programmausführung. Damit verhindern Sie Fehler, die Ihr Programm außer Kontrolle geraten lassen.

```
510 IF XP < 0 OR XP > 511 THEN STOP
```

Zeile 510 stellt denselben Schutzmechanismus für die Variable XP dar wie Zeile 500 für die Variable YP.

```
210 POKE BA + 1, YP
```

In Zeile 520 tragen Sie den Wert von YP in die Speicherzelle des Sprites ein, die die Y-Koordinate bestimmt.

```
530 IF XP < 256 THEN POKE BA + 16, 0 :
    POKE BA, XP :
    RETURN
```

In Zeile 530 wird überprüft, ob der Wert von XP kleiner als 256 ist. Wenn ja, dann wird der »höchste Bit«, auch MSB (»Most Significant Bit«) genannt, mit 0 und die X-Koordinate mit XP belegt, jeweils unter Verwendung der POKE-

Funktion. Im Anschluß daran folgt ein Rücksprung (RETURN) zur Aufrufstelle des Unterprogramms.

```
540 POKE BA + 16, 1 :
    POKE BA, XP - 256
```

Gelangt die Programmausführung zu Zeile 540, dann soll der Sprite im Bereich MSB=1 sein. Deshalb wird der MSB mit 1 und die X-Koordinate mit XP-256 (dadurch erhält man einen Wert zwischen 0 und 255) belegt. Dies geschieht erneut mit Hilfe der POKE-Funktion.

```
550 RETURN
```

Sie können auch feststellen, ob ein Sprite mit einem anderen Sprite oder einem Hintergrundzeichen zusammenstößt. Löschen Sie alles, mit Ausnahme der DATA-Anweisungen (100-300), und geben Sie folgendes Programm ein:

```
10 PRINT "<SHIFT CLR/HOME>"
:
   FOR I = 0 TO 62 :
   READ T :
   POKE 12288 + I, T :
   NEXT
20 POKE 2040, 192 :
   POKE 2041, 192
30 BA = 53248 :
   POKE BA + 21, 3 :
   POKE BA + 39, 10 :
   POKE BA + 40, 13
```

Das Commodore Emblem wird in den Block 192 eingetragen, die Verweise für Sprite 0 und 1 zeigen auf diesen Block. Beide Sprites haben das gleiche Bild, da sie die gleichen Daten verwenden. Zeile 30 legt die BAsis fest,

beide Sprites werden eingeschaltet (2 ↑ 0+2 ↑ 1=1+2=3) und die Farben Rosa und Hellgrün ausgewählt.

```
40 FOR I = 1 to 24:
   PRINT SPC(15)"<SHIFT
   V"SPC(14)"<SHIFT V>" :
   NEXT
```

Zeile 40 druckt zwei Zeilen Text aus, um den Zusammenstoß eines Sprites mit Hintergrundzeichen demonstrieren zu können.

```
50 FOR X = 0 to 347
60 S = 0 :
   XP = X :
   YP = SIN( X / 64) * 90 + 140 :
   GOSUB 500
70 S = 1 :
   XP - X :
   YP - COS( X / 64 ) * 90 + 140 :
   GOSUB 500
80 GOSUB 400
90 NEXT
```

Die Zeilen 50 bis 90 bewegen die Sprites. Es wird die Position von Sprite 0 entsprechend einer Sinuswelle festgelegt, die Position von Sprite 1 entsprechend einer Kosinuswelle bestimmt. Dann rufen Sie das in Zeile 400 beginnende Unterprogramm auf. Dieses Unterprogramm stellt den Zusammenstoß zweier Sprites oder eines Sprites mit einem Hintergrundzeichen fest.

```
399 END
400 PRINT "<HOME>          " :
    PRINT "      " :
    REM 10 & 6 LEERZEICHEN
```

```
410 IF PEEK( BA + 30 ) <> 0 THEN PRINT
    "<HOME><CRSR DOWN>SPRITE"
420 IF PEEK( BA + 31 ) <> 0 THEN PRINT
    "<HOME>BACKGROUND"
430 RETURN
499 END
```

Das Unterprogramm löscht zuerst jede Nachricht, die sich in der Ecke links oben befindet. Stoßen zwei Sprites zusammen, dann erscheint die Meldung SPRITE in der zweiten Zeile des Bildschirms. Das CRSR-DOWN-Kommando bewegt den Kursor nach unten, bevor etwas gedruckt wird. Stößt ein Sprite mit einem Zeichen auf dem Monitor zusammen, dann erscheint die Meldung BACKGROUND (Hintergrund) in der ersten Zeile.

```
500 IF YP < 0 OR YP > 255 THEN STOP
510 IF XP < 0 OR XP > 511 THEN STOP
520 IF S < 0 OR S > 7 THEN STOP
530 POKE BA + S * 2 + 1, YP
540 MSB=0 :
    IF XP > 255 THEN MSB = 1 :
    XP = XP - 256
550 IF MSB = 1 THEN POKE BA + 16, PEEK
    ( BA + 16 ) OR 2↑S
560 IF MSB = 0 THEN POKE BA + 16, PEEK
    ( BA + 16 ) AND (NOT 2↑S)
570 POKE BA + S * 2, XP
580 RETURN
```

Das in Zeile 500 beginnende Unterprogramm leistet im Prinzip dasselbe wie das Unterprogramm in Zeile 400 – 499, allerdings arbeitet es jetzt mit mehreren Sprites. Zeile 530 verändert die Y-Koordinate des Sprites

S. In Zeile 540 erhält eine Variable MSB den Wert des MSB von XP. Der Wert von XP wird in den Bereich zwischen 0 und 255 transformiert, so daß XP von POKE benutzt werden kann. In den Zeilen 550 und 560 wird das entsprechende Bit im MSB-Register ein- bzw. ausgeschaltet. Für die Sprites 0 und 1 werden die Bits 0 und 1 verwendet. Dann tragen Sie mit Hilfe der POKE-Funktion die X-Koordinaten der entsprechenden Sprites ein.
Sie können dieses Programm auch benutzen, um mehrere Sprites auf dem Monitor zu bewegen. Das Programm läuft schneller, wenn die DATA-Anweisungen am Ende, beispielsweise ab Zeile 1000, stehen.
Sichern Sie jetzt dieses Programm. Sie werden es häufig gebrauchen können.

13. ASCII- und CHR$-Codes

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
	0		27	6	54	Q	81
	1	RED	28	7	55	R	82
	2	CRSR →	29	8	56	S	83
	3	GRN	30	9	57	T	84
	4	BLU	31	:	58	U	85
WHT	5	SPACE	32	;	59	V	86
	6	!	33	<	60	W	87
	7	"	34	=	61	X	88
blockiert SHIFT C=	8	#	35	>	62	Y	89
entriegelt SHIFT C=	9	$	36	?	63	Z	90
	10	%	37	@	64	[	91
	11	&	38	A	65	£	92
	12	'	39	B	66	]	93
RETURN	13	(	40	C	67	↑	94
UMSCHALTUNG KLEINBUCHST.	14	)	41	D	68	←	95
	15	*	42	E	69		96
	16	+	43	F	70		97
CRSR ↓	17	,	44	G	71		98
RVS ON	18	–	45	H	72		99
CLR HOME	19	.	46	I	73		100
INST DEL	20	/	47	J	74		101
	21	0	48	K	75		102
	22	1	49	L	76		103
	23	2	50	M	77		104
	24	3	51	N	78		105
	25	4	52	O	79		106
	26	5	53	P	80		107

Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$	Zeichen	CHR$
	108	orange	129	lt. red	150		171
	109		130	grey 1	151		172
	110		131	grey 2	152		173
	111		132	lt. green	153		174
	112	f1	133	lt. blue	154		175
	113	f3	134	Grey 3	155		176
	114	f5	135	PUR	156		177
	115	f7	136	← CRSR	157		178
	116	f2	137	YEL	158		179
	117	f4	138	CYN	159		180
	118	f6	139	SPACE	160		181
	119	f8	140		161		182
	120	SHIFT RETURN	141		162		183
	121	UMSCHALTUNG GROSSBUCHST.	142		163		184
	122		143		164		185
	123	BLK	144		165		186
	124	CRSR ↓	145		166		187
	125	RVS OFF	146		167		188
	126	SHIFT CLR HOME	147		168		189
	127	INST DEL	148		169		190
	128	brown	149		170		191

Codes	**192-233**	**wie Codes**	**96-127**
Codes	**224-254**	**wie Codes**	**160-190**
Code	**255**	**wie Code**	**126**

COMMODORE 64 GRAPHICS

Wer dieses Gerät besitzt, will seine Leistungsfähigkeit auch im Bereich Grafik ausschöpfen.

Mit diesem Buch schafft jeder den Einstieg. Es setzt bei einfachsten Grafikblöcken und Displays ein und endet bei hochauflösender Grafik.

- **Einführung und Training**
- **Programmieren und Anwenden**
- **Einfache Operationen, praktische Übungen**
- **Functions, Displays und Sprites**

GOLDMANN VERLAG

ISB N 3-442-13119-7 DM +009.80

T 3-28-00