BASIC

Lehrbuch

Schneider
COMPUTER DIVISION

BASIC
Lehrbuch

Teil 1
ERSTE SCHRITTE

Vervielfältigung und Weitergabe – auch auszugsweise – dieses Lehrbuches und der dazugehörigen Programmkassetten bedürfen der vorherigen schriftlichen Zustimmung der Schneider Computer Division.

Schneider Computer Division
Silvastraße 1
8939 Türkheim

Schneider BASIC

Lehrbuch und Kassetten

Teil 1: Erste Schritte

SW 111

Erste Ausgabe 1984
Originalausgabe in Englisch
Original Copyright © 1984 Amstrad Consumer Electronics plc
Deutsche Bearbeitung: ESCON GmbH, Freising

INHALT

VORWORT

Dies ist der erste Teil eines Selbstlernkurses für das Programmieren in BASIC unter Anwendung des Schneider CPC 464 Colour Personal Computers. Die zwei Datenkassetten, die zu diesem Buch gehören, enthalten Computer-Programme, die ein wesentlicher Bestandteil des Kurses sind.

Datenkassette A enthält:

- Programme, die dazu dienen, das Prinzip von einfacher und anspruchsvoller Programmierung zu erklären.
- Spiele zur Unterhaltung, die gleichzeitig helfen sollen, daß man sich an den Umgang mit einem Computer gewöhnt.

Datenkassette B enthält:

- Programme, um sich selbst zu testen, ob man die Begriffe, die in den einzelnen Kapiteln erklärt werden, auch verstanden hat.

Fortgeschrittene Programmierungstechniken sind im zweiten Teil dieses Kurses, 'Fortgeschrittenes BASIC', enthalten.

1 WORUM ES SICH EIGENTLICH HANDELT

Wenn Sie diese Worte lesen, sind Sie mit ziemlicher Sicherheit der stolze Besitzer eines Schneider CPC 464 Microcomputers. Seine ausgezeichnete Bildschirmdarstellung und hervorragende Klangqualität wird Ihnen sicher schon eine interessante neue Welt eröffnet haben, die aufregend ist und Spaß macht. Gleichzeitig werden Sie auch erkannt haben, daß es unerläßlich ist, die Sprache des CPC 464 – BASIC – zu erlernen, wenn Sie mehr machen wollen, als nur Standardspiele und Standardprogramme ablaufen zu lassen.

BASIC

BASIC ist die Computersprache, die auf der Welt am meisten verbreitet ist.
Außerdem ist sie die ideale Sprache für den Anfänger, da Sie schon nach ein paar Stunden des Lernens Ihr erstes Programm schreiben können.
Das Erfolgserlebnis, das dadurch erreicht wird, besonders wenn man feststellt, daß man keinen Fehler gemacht hat, ist ganz enorm und programmieren wird Ihnen Spaß machen. Es kann sogar zu einem Hobby werden.
Was ist eigentlich programmieren? Sie wissen, daß ein Computer viele Dinge tun kann, jedoch kann er nicht selbst denken. Deshalb müssen Sie für ihn denken. Dieses Denken – das heißt überlegen, was alles getan werden muß, damit ein gewünschtes Ziel erreicht wird – bezeichnet man in Form einer Folge von Anweisungen als Programm.
Wenn man dieses Programm dann in einen Computer eingibt, verwandelt man ihn dadurch zu einem Tele-Spiel, zu einem Textverarbeiter oder zu einer Maschine, die die Kontoführung macht.

Wahrscheinlich werden Sie feststellen, daß die Programme, die speziell für den CPC 464 geschrieben sind, nicht ohne Abänderung auf anderen Computern laufen können. Der Grund dafür ist, daß das Schneider-BASIC für den CPC 464 viele außergewöhnliche Kommandos und Funktionen enthält, die bei weniger aufwendigen Anlagen nicht zur Verfügung stehen.

BASIC hat seinen eigenen Wortschatz, genau wie jede andere Sprache. Dieser Wortschatz besteht aus 'Schlüsselwörtern' und Sie werden in den folgenden Kapiteln lernen, welche diese Schlüsselwörter sind und was sie für den CPC 464 bedeuten.

Jedesmal, wenn ein neues Schlüsselwort in diesem Handbuch erklärt wird, steht es dick gedruckt am äußeren Rand, damit Sie es schnell wiederfinden, wenn Sie Ihr Wissen darüber auffrischen wollen.

WIE MAN DIESES BUCH ANWENDEN SOLL

Für jedes einzelne Kapitel dieses Buches benötigen sie etwa ein oder zwei Abende an Arbeitszeit. Es besteht immer aus folgenden Teilen:

- Theoretische Erklärungen
- Praktische Anwendung des Computers
- Beispiele, damit Sie selbständig etwas programmieren können.

Außerdem enthält jedes Kapitel Übungen, um das was Sie gelernt haben, zu vertiefen, und Tests zur Selbstüberprüfung Ihres Wissens.

Lassen Sie kein Kapitel aus. Durch das ganze Buch hindurch wird Ihr Wissen stufenweise erweitert, wobei immer auf den vorangegangenen Kapiteln aufgebaut wird. Wenn Sie das Gefühl haben, daß Ihnen ein Kapitel zu schwierig ist, lesen Sie es ein- oder zweimal durch, und arbeiten Sie sich dann noch einmal ganz langsam hindurch. Vergewissern Sie sich mit Hilfe der Tests zur Selbstüberprüfung, daß Sie es wirklich verstanden haben.

Wenn Sie diesen Teil des Kurses beendet haben, sollten Sie in der Lage sein, einfache, aber zuverlässige Programme für Ihren eigenen Gebrauch zu schreiben. Im Teil 2 dieses Kurses, 'Fortgeschrittenes BASIC', werden weitere Besonderheiten des Schneider-BASIC erklärt und Sie lernen, wesentlich schwierigere Programme zu schreiben.

2 AUFBAU UND HERANFÜHRUNG AN DIE ANLAGE

Zuerst müssen Sie Ihren CPC 464 auspacken. Wenn Sie es schon getan haben, können Sie die folgenden Punkte überspringen.
Wenn Sie die Verpackung öffnen, sollten folgende Gegenstände enthalten sein:

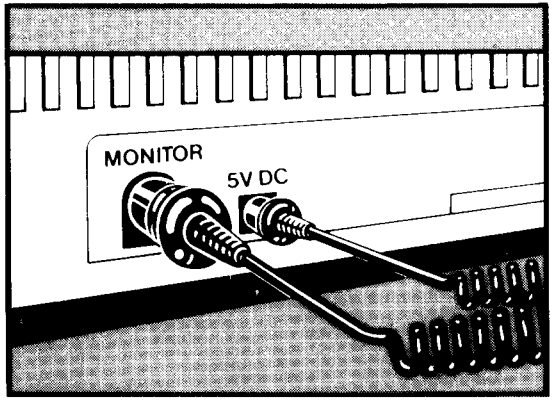
- CPC 464 Colour Personal Computer
- GT 64 Bildschirmgerät, CTM 640 Farbbildschirmgerät
- MP1 Modulator/Stromversorgung (wahlweise)
- Schneider CPC 464 Anwendungsbeschreibung
- Demonstrationskassette

Suchen Sie in einem ruhigen Teil Ihres Hauses nach einem genügend großen Schreibtisch oder Tisch und bauen Sie den CPC 464 und sein Bildschirmgerät (bzw. den MP1 Modulator/Stromversorgung und einen normal gebräuchlichen Fernsehempfänger) darauf auf. Schieben Sie nun vorsichtig die Stecker der zwei Verbindungskabel von der Vorderseite des Bildschirmgerätes bzw. des MP1 in die dafür vorgesehenen Fassungen auf der Rückseite des CPC 464 (siehe Abbildung). Schließen Sie das Bildschirmgerät bzw. den MP1 noch nicht an das Stromnetz an.
Vergewissern Sie sich, daß die Stecker auf der Rückseite des CPC 464 ganz eingesteckt sind, aber wenden Sie nicht zuviel Kraft an. Da die Stecker und Fassungen durch ständiges Ein- und Ausstecken beschädigt werden können, sollten

Sie möglichst einen Stellplatz für den CPC 464 suchen, an dem er ständig stehen bleiben kann, auf alle Fälle für die Zeit dieses Kurses.

Wenn Sie einen gewöhnlichen Fernsehempfänger zusammen mit Ihrem CPC 464 verwenden, stellen Sie sicher, daß Ihnen zwei Netzanschlüsse zur Verfügung stehen oder benutzen Sie einen Doppeladapter, damit Sie den Fernsehempfänger und den MP1 gleichzeitig anschließen können. Dann verbinden Sie das Koaxialkabel des MP1 mit dem Antennenanschluß des Fernsehempfängers und schalten das Fernsehgerät auf Kanal 36.

Suchen Sie sich nun einen bequemen Stuhl zum Sitzen und stellen Sie das Bildschirm – bzw. das Fernsehgerät so auf, daß es etwa einen Meter von Ihrer Nase entfernt ist – wenn Sie zu dicht vor dem Bildschirm sitzen kann das sehr ermüdend sein und sogar Augenschäden verursachen.



Legen Sie sich dieses Buch so zurecht, daß Sie es leicht lesen können, während Sie die Tastatur bedienen. Der Bildschirm sollte dabei so ausgerichtet sein, daß Sie ihn gut sehen. Außerdem kann es von Vorteil sein, wenn Sie die Anwendungsbeschreibung des CPC 464 griffbereit liegen haben

Schließen Sie dann alle Geräte an das Stromnetz an und schalten Sie ein. Folgende Worte sollten nun auf dem Bildschirm erscheinen:

```
Schneider 64K Microcomputer      (vl)

© 1984 Amstrad Consumer Electronics plc
            and Locomotive Software Ltd.

BASIC 1.0

Ready
■
```

Wenn Sie dieses Bild nicht sehen (gelbe Schriftzeichen auf blauem Grund, falls Sie ein CTM 640 Farbbildschirmgerät oder ein Farbfernsehgerät haben) schalten Sie wieder aus und versuchen Sie es noch einmal. Falls Sie ein Fernsehgerät verwenden, prüfen Sie, ob es auch wirklich auf Kanal 36 eingestellt ist. Wenn Sie dann immer noch Schwierigkeiten haben sollten, prüfen Sie, ob die ON-Anzeige des CPC 464 auf rot steht. Andern-

falls vergewissern Sie sich, ob:

- der Ein/Aus-Schalter des CPC 464 auf ON steht
- kein allgemeiner Stromausfall vorhanden ist
- das Stromkabel richtig in den CPC 464 eingesteckt ist

Falls alle diese Überprüfungen fehlgeschlagen, suchen Sie Ihren Händler auf, damit er Ihnen weitere Ratschläge geben kann.

RUN

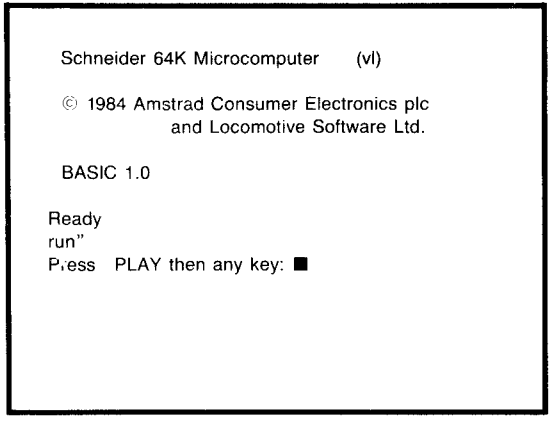
HALLO

Sobald der Rechner betriebsklar ist, können wir anfangen, uns mit ihm zu beschäftigen. 'Ready' bedeutet, daß der Computer darauf wartet, daß Sie Kommandos oder ein Programm eingeben. Das viereckige Zeichen wird als 'Cursor' bezeichnet und zeigt Ihnen, an welcher Stelle des Bildschirms Ihre nächste Eingabe erscheinen wird.

Nun machen wir weiter. Schieben Sie die Kassette 'Erste Schritte in BASIC – Datenkassette A' in den Datenkassettenrekorder und geben Sie folgendes über die Tastatur ein:

run" ENTER

Um das Zeichen " eingeben zu können, müssen Sie eine der Tasten mit der Aufschrift SHIFT, die sich auf beiden Seiten der unteren Buchstabenreihe auf der Tastatur befinden, drücken, und gleichzeitig damit die Taste mit dem Zeichen " (es befindet sich über der '2' auf der linken Seite der oberen Reihe). Durch das Drücken der ENTER-Taste zeigen Sie dem CPC 464 an, daß Ihre Eingabe beendet ist und daß er nun etwas tun soll. Wenn Sie das Kommando richtig eingegeben haben, wird Ihnen der CPC 464 antworten, indem eine weitere Zeile auf dem Bildschirm erscheint:



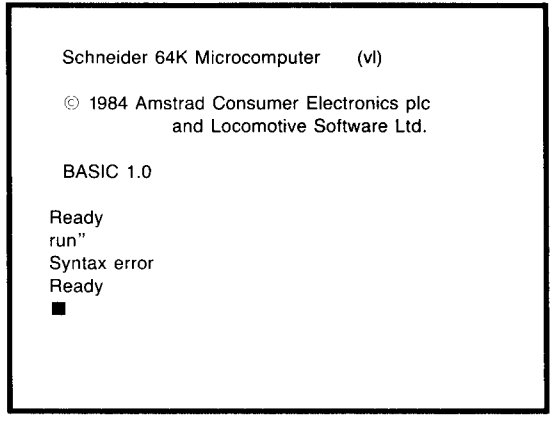
Drücken Sie vorsichtshalber beim Datenkassettenrekorder auf REW, damit das Band auch wirklich bis zum Anfang zurückgespult ist und folgen Sie dann den Anweisungen. Drücken Sie auf PLAY und dann auf die Taste ENTER. Während das Programm vom CPC 464 eingelesen wird, hören Sie ein hohes Geräusch, das aus dem eingebauten Lautsprecher kommt.

Wenn Sie einen Fehler beim Eingeben machen (und bevor Sie auf ENTER drücken) benutzen Sie die

DEL

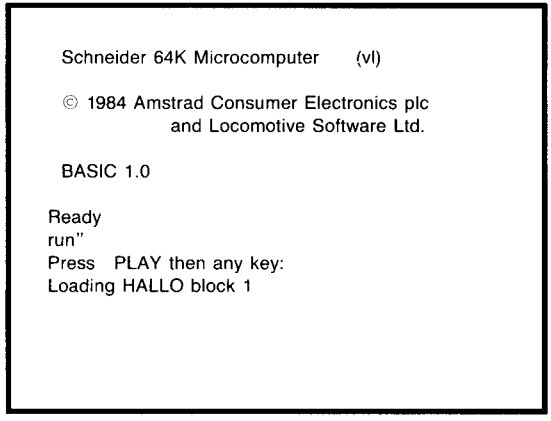
Taste. Damit können Sie zurückgehen und das zuletzt eingegebene Zeichen wieder löschen. Danach können Sie den Buchstaben, den Sie falsch eingegeben haben, nochmal neu tippen.

Falls Sie einen Fehler gemacht und schon auf ENTER gedrückt haben, macht es nicht allzuviel aus. Der CPC 464 wird lediglich eine weitere Zeile auf dem Bildschirm ausgeben, so wie es das folgende Beispiel zeigt:



Wenn Sie diese Mitteilung bekommen, geben Sie einfach die Zeile noch einmal ein.

Wenn alles richtig ist (und wenn Sie das Kommando richtig eingegeben haben), wird die folgende Mitteilung kurz darauf auf dem Bildschirm erscheinen:



Dieser Mitteilung folgt sofort das HALLO-Programm, das Sie auf freundliche Art und Weise in der Computerwelt begrüßt.

Wenn das HALLO-Programm einige Male durchgelaufen ist, schauen Sie auf die linke obere Ecke der Tastatur und Sie werden dort eine hellgraue Taste entdecken, mit der Aufschrift ESC. Damit können Sie das Programm verlassen. Drücken Sie diese Taste zweimal:

ESC ESC

Der CPC 464 wird mit folgender Mitteilung antworten:

```
Break in 140
Ready
■
```

Die Zahl muß nicht 140 sein. Und sie bedeutet nicht, daß Sie, oder der CPC 464, für die nächsten 140 Sekunden, Minuten, Stunden oder Tage aufhören sollen zu arbeiten. Auch erwartet man nicht von Ihnen, daß Sie in das Haus mit der Nummer 140 in Ihrer Straße gewaltsam eindringen.

Was auch immer diese Nummer bedeutet, übergehen Sie sie vorerst einmal. Wir werden zu einem späteren Zeitpunkt etwas über die Zeilennummern erfahren. Auf alle Fälle haben Sie das HALLO-Programm gestoppt und sind nun bereit dafür, den Rest dieses Kapitels anzugehen.

Übrigens sollten Sie sich merken, daß der CPC 464 darauf besteht, daß ein Unterschied zwischen 0 (nichts oder Null) und O (o, wie in 'hallo') gemacht wird. Wenn **nicht** der Buchstabe des Alphabets gemeint ist, wird das Zeichen diagonal durchstrichen.

Spiel Nummer 1

Wir wollen ein Spiel spielen. Zwar ist nicht jeder bereit, für Computer-Spiele seine Zeit und sein Geld zu opfern, aber es steckt doch mehr dahinter, als das, was man mit den Augen sieht. Zuerst einmal werden Sie schnell vertraut mit der Maschine und vermeiden die Langweiligkeit, die von rein formellen Übungen hervorgerufen werden kann und zum zweiten erkennen Sie durch die Spiele, daß der Umgang mit Computern Spaß macht.

Machen Sie diesmal etwas anders. Geben Sie die folgende Zeile in den CPC 464 ein:

load"simon" ENTER

Vergessen Sie nicht, die SHIFT-Taste zu drücken, um die Anführungsstriche (") einzugeben. Der CPC 464 wird mit der folgenden Mitteilung antworten:

Press PLAY then any key

Folgen Sie wiederum den Anweisungen. Drücken Sie beim Datenkassettenrekorder auf PLAY und dann auf die ENTER-Taste der Haupttastatur. Wenn Sie den Namen richtig eingegeben haben und auch alles andere stimmt, wird der CPC 464 die Kassette im Datenkassettenrekorder laufen lassen und davon das Programm mit dem Namen SIMON laden. Programme haben immer Namen, damit man sie findet, wenn man sie braucht.

Diesmal wird das Programm nur in den Speicher des CPC 464 aufgenommen und er macht nichts, bevor Sie ihn nicht dazu auffordern.

Der CPC 464 wird folgende Mitteilung ausgeben:

Loading SIMON block 1

die dann wechselt zu:

Loading SIMON block 2

Denken Sie nicht darüber nach, was diese Mitteilungen bedeuten könnten. Wenn die Kassette gestoppt hat, wird eine weitere Mitteilung ausgegeben:

Ready

Das bedeutet, daß Sie nun den CPC 464 anweisen müssen, was er mit dem Programm, das er in seinem Speicher hat, machen soll. Geben Sie ein:

run ENTER

Viel Spaß.

LOAD

Test

Wenn Sie keine Lust mehr haben SIMON zu spielen, beenden Sie es, indem Sie die Taste ESC zweimal drücken, wie es zuvor erklärt wurde.

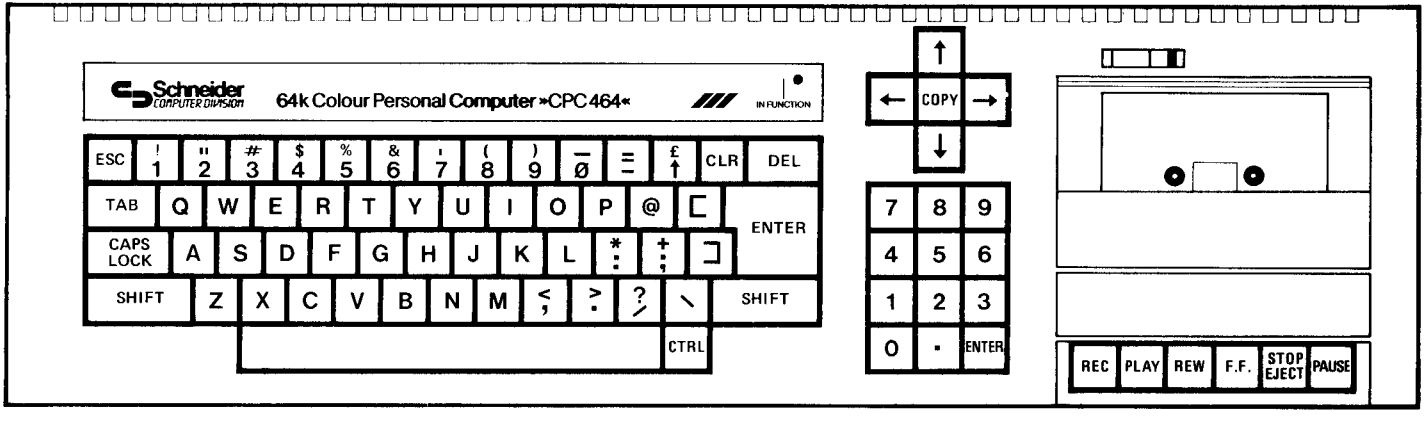
Ihre Ausgabe ist es, daß Sie genauso vorgehen, wie beim Programm SIMON, um den ersten unserer Tests zur Selbstüberprüfung, SAT2, von Datenkassette B zu laden. Wenn Sie dieses Programm laufen lassen, wird es Ihnen Fragen über dieses Kapitel stellen, so daß Sie feststellen können, ob Sie nochmal etwas wiederholen müssen.

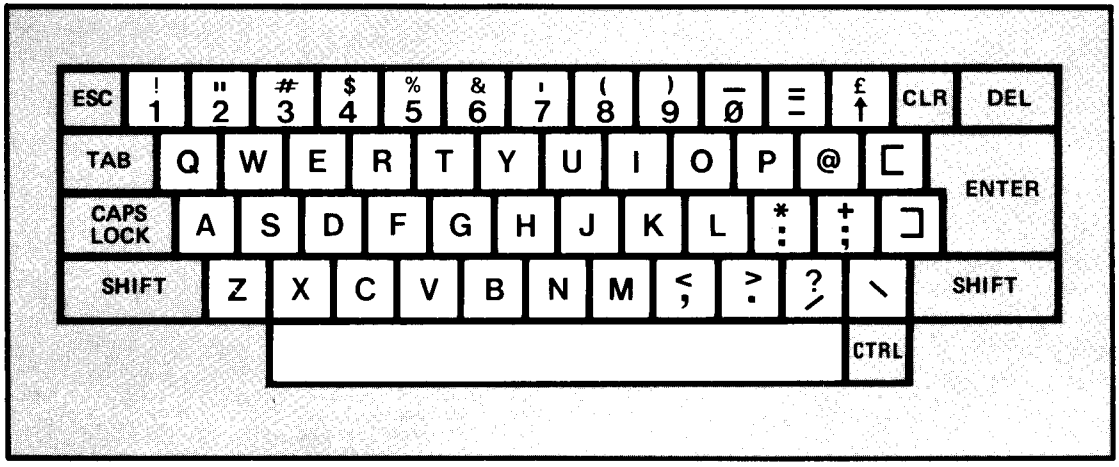
3 DER GEBRAUCH DER TASTATUR

Wir sprachen schon davon, daß Sie die BASIC-Sprache erlernen müssen, um mit dem CPC 464 in Kommunikation treten zu können. Sie teilen dem CPC 464 mit, was er machen soll, indem Sie Wörter und Nummern über die Tastatur eingeben. Dieses Kapitel behandelt alles, was man darüber wissen muß – wo die einzelnen Tasten liegen und wann man sie drücken muß.

Die Tastatur des CPC 464 läßt sich in 5 verschiedene Gruppen einteilen:

- Tasten mit Schriftzeichenaufdruck
- Steuertasten
- Tasten mit Zahlenaufdruck
- Tasten zum Hin- und Herbewegen des Cursors
- Bedienungstasten für den Datenkassettenrekorder

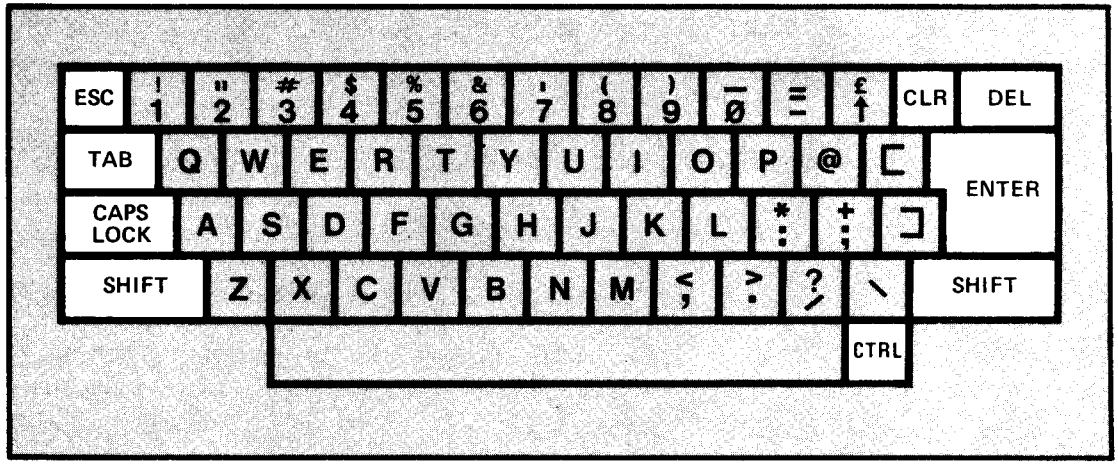




Haupttastatur:
Tasten mit Schriftzeichenaufdruck

Diese Tasten haben Sie schon benutzt, da Sie bestimmt schon einmal auf einer Schreibmaschine geschrieben haben und es ist wohl keine weitere Erklärung dazu nötig. Dieser Teil der Tastatur umfaßt Buchstaben, Zahlen, viele Satzzeichen und eine Taste zum Setzen von Leerstellen.

Wenn Sie eine von diesen Tasten länger als eine halbe Sekunde drücken, wiederholt sich die Eingabe, so, als wenn Sie die Taste erneut gedrückt hätten. Das setzt sich so lange fort, bis Sie den Finger von der Taste nehmen.



Haupttastatur: Steuertasten

ESC

ESC ist die Abkürzung von 'ESCape' (ausbrechen). Wenn Sie diese Taste drücken während ein Programm läuft, stoppen Sie damit den CPC 464 in seinem Ablauf und Sie können das Programm weiterlaufen lassen, indem Sie auf irgendeine andere Taste der Tastatur drücken. Sie können den CPC 464 auch wieder auf die Ausgangsposition vor Ablauf des Programmes (READY-Position) bringen, wenn Sie die ESC-Taste ein zweites Mal drücken.

TAB

Wenn Sie die Taste mit der Aufschrift TAB drücken, erscheint ein nach rechts gerichteter Pfeil auf dem Bildschirm. Diese Taste wird in diesem Teil des Kurses nicht verwendet, sondern erst in Teil 2 des Kurses ausführlich erklärt.

SHIFT

Die Taste SHIFT bewirkt, daß sich die Symbole ändern, wenn Sie eine der Schriftzeichen-Tasten drücken. So erhalten Sie den Großbuchstaben, wenn Sie gleichzeitig auf eine Buchstaben-Taste und auf die SHIFT-Taste drücken. Beim gleichzeitigen Drücken einer der Zahlen/Satzzeichen-Tasten und der SHIFT-Taste erscheint jeweils das obere Zeichen.

CAPS LOCK

Wenn Sie auf die Taste CAPS LOCK drücken, bevor Sie Buchstaben eingeben, erhalten Sie ebenfalls die Großbuchstaben, und zwar so lange, bis Sie erneut auf CAPS LOCK drücken.

Es bewirkt das gleiche, als wenn Sie Ihren Finger auf die SHIFT-Taste halten, nur daß beim Verwenden der Zahlen/Satzeichen-Tasten jetzt immer das untere Zeichen, also die Zahl, erscheint.

DEL

Sie werden sich sicher an die DEL-Taste aus dem vorherigen Kapitel erinnern. Beim Eingeben einer Zeile bewirkt das Drücken der DEL-Taste, daß das zuletzt geschriebene Zeichen gelöscht und der Cursor um eine Stelle zurück nach links gesetzt wird, damit das falsch geschriebene Zeichen neu eingegeben werden kann.

CLR

Die CLR-Taste ist vergleichbar mit der DEL-Taste. Auch durch diese Taste werden Zeichen gelöscht, jedoch nicht das Zeichen links vom Cursor, so wie es die DEL-Taste bewirkt, sondern immer das Zeichen, auf dem der Cursor gerade steht, ohne daß der Cursor seine Position ändert. Alles was rechts vom Cursor steht wird dabei um eine Stelle nach links verschoben.

ENTER

Die ENTER-Taste entspricht der Wagentransporttaste einer elektrischen Schreibmaschine. Sie müssen sie am Ende jeder eingegebenen Zeile drücken, damit der CPC 464 weiß, daß die Zeile zu Ende ist. Normalerweise wird danach der Cursor, der am Ende Ihrer letzten Eingabe steht, an den Anfang der nächstfolgenden Zeile gesetzt. Manchmal gibt der CPC 464 auch folgende Meldung aus:

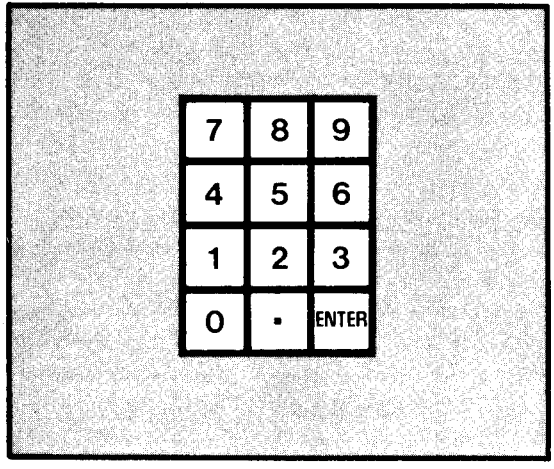
```
Syntax error
```

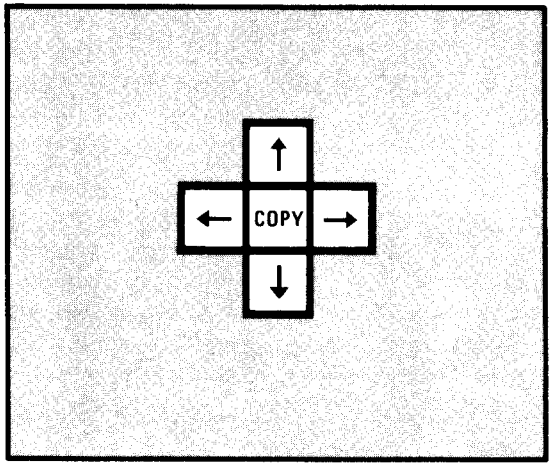
Das ist der Ausdruck in BASIC, um zu sagen, 'Ich habe nicht verstanden', und gilt als eine Fehlermeldung. Wir werden später in diesem Kurs noch mehr darüber erfahren.

CTRL

CTRL bedeutet ConTRoL. Durch das Gedrückthalten der CTRL-Taste erhalten Sie noch eine Reihe weiterer Symbole über die Buchstaben-Tasten und teilweise über die Zahlen-Tasten, zusätzlich zu den Zeichen, die oberhalb der Zahlen aufgedruckt sind.

Verwendet man die CTRL-Taste zusammen mit anderen Kontroll-Tasten, gibt man damit dem CPC 464 Anweisungen, um verschiedene Dinge auszuführen – mehr darüber erfahren Sie jedoch später.





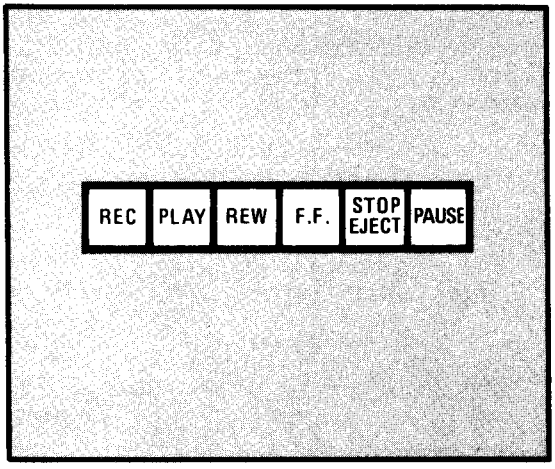
Numerischer Tastenblock

Diese Tasten sind so angeordnet, daß man möglichst einfach viele Zahlen eingeben kann. Abgesehen von der zusätzlichen ENTER-Taste, gleichen sie den Zahlen-Tasten in der oberen Reihe der Haupttastatur. Der einzige Unterschied ist nur, daß sie nicht von SHIFT oder CTRL beeinflußt werden, außer sie sind speziell darauf programmiert. Diese besondere Art der Programmierung wird in diesem Teil des Kurses nicht beschrieben, aber Sie erfahren etwas später in diesem Kapitel, wie die zugehörige ENTER-Taste eine Extrafunktion erhält.

Cursor

Die Tasten mit dem 'Pfeil'-Aufdruck werden dazu benutzt, den Cursor auf dem Bildschirm in den gewünschten Richtungen hin- und herzubewegen.

Die COPY-Taste wird zu einem späteren Zeitpunkt erklärt.



Bedienungstasten des Datenkassettenrekorders

Es gibt zwischen diesen Tasten und den Bedienungstasten eines normalen Kassettenrekorders nur einen Unterschied: Nachdem auf PLAY (oder PLAY und REC) gedrückt wurde, beginnt der Datenkassettenrekorder erst zu laufen, wenn er vom CPC 464 dazu aufgefordert wird.

Praktische Anwendung

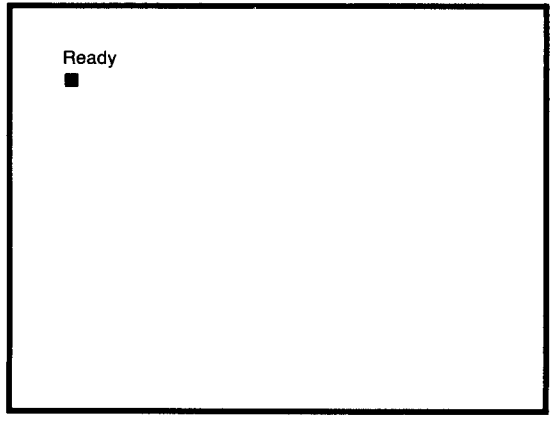
Vergewissern Sie sich, daß die Kassette richtig in den Datenkassettenrekorder eingelegt ist. Wenn auf dem Bildschirm noch der Rest vom Programm SAT2 vorhanden ist, mit dem Sie im vorhergehenden Kapitel gearbeitet haben, können Sie alles verschwinden lassen, wenn Sie folgende Zeile eingeben:

CLS

cls ENTER

Das bedeutet, daß der Bildschirm gelöscht werden soll, und Sie werden sehen, daß genau das auch gemacht wird. Versuchen Sie es. Ganz gleich, was Sie vorher gemacht haben, das Ergebnis sieht folgendermaßen aus:

RUN



Es grenzt an Zauberei, nicht wahr? Jetzt können wir das nächste Programm laufen lassen. Geben Sie folgende Zeile ein:

run "Buchstaben" ENTER

Nun können Sie einfach auf irgendwelche Tasten Ihrer Tastatur drücken, um zu sehen was passiert. Versuchen Sie, gleichzeitig auf mehrere Tasten zu drücken.

Während Sie mit diesem Programm arbeiten, können Sie viele der Zeichen sehen, die zwar auf dem Bildschirm erscheinen, jedoch nicht auf den Tasten aufgedruckt sind. Das ist immer dann der Fall, wenn Sie auf eine der Buchstaben- oder Zahlentasten drücken, während Sie die CTRL-Taste gedrückt halten. Jedoch werden Sie feststellen, daß nicht alle Tasten ein 'verstecktes' Zeichen besitzen.

Wenn Sie einmal ein Computerprogramm zum Laufen gebracht haben, läuft es so lange, bis es das Ende erreicht. Oder es ist dazu bestimmt, sich ständig zu wiederholen, wie es beim Programm 'Buchstaben' der Fall ist. Dann müssen Sie es selbst stoppen. Sie haben inzwischen schon erfahren, daß der Trick den man dabei anwendet, das zweimalige Drücken der ESC-Taste ist. Aber es gibt auch noch einen anderen Weg. Sie können einen neuen Start 'erzwingen'. Das hat den gleichen Effekt, wie wenn Sie den Computer ausschalten und nach ein paar Sekunden wieder einschalten würden. Und so machen Sie es:

Halten Sie

CTRL und **SHIFT**

gedrückt und drücken Sie dann

ESC

Sie werden sehen, daß der CPC 464 wieder das gleiche Bild auf Bildschirm erscheinen läßt, das Sie schon sahen, als Sie den Computer das erste Mal einschalteten. Im folgenden werden wir immer darauf Bezug nehmen, wenn wir schreiben CTRL/SHIFT/ESC.

Warnung: Wenn Sie CTRL/SHIFT/ESC anwenden, 'vergißt' der CPC 464 jedes Programm, das er gespeichert hat.

Ist soweit alles klar? Das nächste Programm ist sehr einfach, jedoch wollen wir es nun über einen neuen Weg zum Laufen bringen. Halten Sie

CTRL

gedrückt und drücken Sie dann auf

ENTER

im numerischen Tastenblock (diesmal **nicht** auf die große ENTER-Taste).

Der CPC 464 wird genauso antworten, als wenn Sie eine ganze Zeile von Befehlen eingegeben hätten. Und wissen Sie auch schon warum? Sie werden sich sicher daran erinnern, daß wir vor kurzem davon sprachen, daß diese ENTER-Taste noch eine zusätzliche Funktion besitzt!

NAMENWIEDERHOLEN (das Programm, das Sie gerade geladen haben) wird Ihnen nicht allzu lange Spaß machen, und so können Sie den CPC 464 mit CTRL/SHIFT/ESC wieder in die Ausgangsposition bringen, um das nächste Programm TASTATUR starten zu können.

Das Programm TASTATUR ist ein Trainingsprogramm, das Ihnen helfen soll, sich an die Tasten auf dem CPC 464 zu gewöhnen. Wenn Sie einige Zeit damit verbracht haben, werden Sie schon auswendig wissen, wo die einzelnen Tasten liegen. Es ist für Sie sicher von Vorteil, wenn Sie ab und zu auf dieses Programm zurückkommen, um schneller eingeben zu können.

Spiel Nummer 2

Sie haben dieses Spiel bestimmt schon auf dem Papier gespielt. Es heißt AM GALGEN. Sie kennen inzwischen schon verschiedene Wege, um ein Programm zu laden und zu starten. Aber um ganz sicher zu gehen, zeigen wir es noch einmal:

run "AM GALGEN" ENTER

Testen

Bevor Sie zum nächsten Kapitel übergehen, laden Sie noch das Programm SAT3 und lassen Sie es laufen.

WIE MAN DIE DINGE RICHTIG PLAZIERT

Wie Sie schon bei der Demonstrations-Kassette und den vorhergehenden Programmen dieses Kurses gesehen haben, besitzt der CPC 464 sehr schöne graphische Darstellungen. In diesem Kapitel erfahren Sie alles, was Sie wissen müssen, um die passenden Linien und Formen Ihrer eigenen Entwürfe auf den Bildschirm zeichnen zu können.

Der CPC 464 zeigt alle Schriftzeichen und Graphiken in einem 'Fenster' auf dem Bildschirm. Dieses 'Fenster' wird umgeben von einem Rand, der niemals benutzt wird. Die Farbe dieses Randes kann durch folgende Eingabe verändert werden:

BORDER

```
border 0 ENTER
```

Der Rand ist nun schwarz. Versuchen Sie nun:

```
border 26 ENTER
```

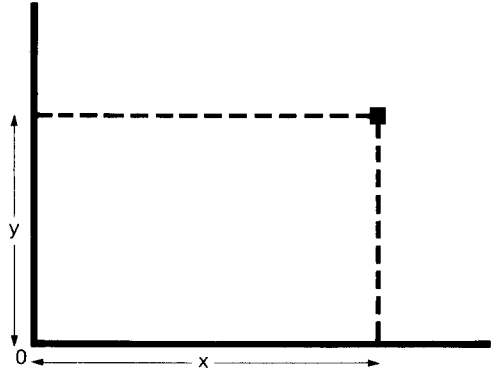
Das andere Extrem. Nun können Sie selbst etwas herumspielen, indem Sie alle Nummern ausprobieren, die dazwischen liegen.

Wenn Sie schon die Gebrauchsanleitung gelesen haben, wissen Sie, daß beim CPC 464 27 Farben zur Auswahl stehen.

Selbst wenn Sie keinen CTM 640 Colour Monitor oder Farbfernseher besitzen ist das Durchgehen der Nummern in der beschriebenen Weise keine Zeitverschwendung, da sich entsprechend zur aufsteigenden Nummernfolge die Grauskala bei schwarz/weiß ändert.

Koordinaten

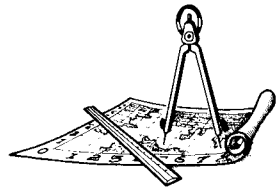
Jede Linie, jede Form und jedes Schriftzeichen, das Sie auf dem Bildschirm sehen, setzt sich aus kleinen Punkten zusammen. Die Lage jedes einzelnen dieser Punkte ist durch seine 'x'- und 'y'-Koordinaten bestimmt.



Die waagrechte Lage wird mit 'x' angegeben, die senkrechte mit 'y'.

Starten Sie das nächste Programm. Es heißt ZEICHNEN. Sie können nun gerade Linien auf dem Bildschirm zeichnen, wenn Sie gleichzeitig auf die TAB-Taste und auf eine der Cursor-Tasten drücken. Wenn Sie ohne zu zeichnen die Position verändern wollen, so drücken Sie nur auf eine der Cursor-Tasten. Diesen Vorgang können Sie durch gleichzeitiges Drücken der SHIFT-Taste beschleunigen.

Wie Sie sehen, gibt Ihnen das Programm die 'x'- und 'y'-Koordinaten für die jeweilige Position des Cursors an. Wenn Sie immer weiter mit dem Cursor nach unten fahren, wird er schließlich aus dem Fenster verschwinden. Merken Sie sich an dieser Stelle den Wert der 'y'-Koordinate. Machen Sie das Gleiche in 'x'-Richtung. Danach werden Sie wissen, daß der graphische Bildschirm des CPC 464 in der Höhe über 400 Stellen und in der Breite über 640 Stellen verfügt. Versuchen Sie, auf dem Bildschirm ein Quadrat in einer bestimmten Lage zu zeichnen, so wie im folgenden Beispiel:



Zeichnen Sie ein Quadrat der Größe 50 x 50 auf den Bildschirm, dessen linke untere Ecke die Koordinaten x = 300 und y = 150 besitzt.

Bewegen Sie zuerst den Cursor in 'x'-Richtung, bis er die Stelle 300 erreicht. Dann in 'y'-Richtung, bis zur Stelle 150. Wenn das Quadrat 50 x 50 groß werden soll, müssen wir von diesem Punkt aus eine Linie bis x = 300 + 50 = 350 ziehen. Zeichnen Sie diese Linie. Wenn Sie zu weit gefahren sind, können Sie es korrigieren, indem Sie mit dem Cursor wieder in die andere Richtung fahren, bis der 'x'-Wert stimmt. Der 'x-y'-Wert ist nun

x = 350 y = 150

Zeichnen Sie nun eine senkrechte Linie bis y = 150 + 50 = 200. Dann ziehen Sie wieder eine Linie in 'x'-Richtung bis x = 350 - 50 = 300 und fahren zum Schluß dann noch bis y = 200 - 50 = 150. Nun sind wir wieder am Ausgangspunkt angelangt.

Verlieren Sie nicht den Mut, falls es beim ersten Mal nicht gleich gelingen sollte – wichtig ist nur, daß Sie das Prinzip verstanden haben.

Schriftzeichengröße

Bevor wir mit den Koordinaten weitermachen, wollen wir noch etwas anderes bezüglich der Bildschirmdarstellung ausprobieren.

Wenn Sie den CPC 464 das erste Mal anschalten, erscheint die Begrüßungszeile in Schriftzeichen, die doppelt so breit sind, wie die Zeichen auf dieser Seite. Sie werden bestimmt bemerkt haben, daß einige Programme größere Schriftzeichen verwenden. Es gibt tatsächlich drei verschiedene Größen von Schriftzeichen, die durch das Schlüsselwort MODE ausgewählt werden können.

MODE

Probieren Sie es aus. Holen Sie sich das ursprüngliche Bild zurück auf den Bildschirm, durch die Anweisungsfolge CTRL/SHIFT/ESC, an die Sie sich bestimmt noch erinnern. Dann geben Sie ein:

mode 0 ENTER

Jetzt sind die Schriftzeichen doppelt so breit, wie vorher. Versuchen Sie nun:

mode 1 ENTER

Jetzt haben die Schriftzeichen wieder die gleiche Größe wie vorher. Als nächstes probieren Sie:

mode 2 ENTER

Nun erhalten wir Schriftzeichen, die nur halb so breit sind. Also gibt es drei verschiedene Größen von Schriftzeichen:

- Mode 0 – 20 Zeichen pro Zeile
- Mode 1 – 40 Zeichen pro Zeile
- Mode 2 – 80 Zeichen pro Zeile

Merken Sie sich, daß sich nur die Breite ändert und immer 25 Zeilen gleichzeitig auf dem Bildschirm erscheinen können.

Das Schlüsselwort MODE wirkt sich auch auf die graphischen Darstellungen aus. Jedoch sprechen wir in diesem Fall nicht von Schriftzeichen, sondern von 'Pixels'.

Ein 'Pixel' ist die kleinstmögliche Größe eines Punkts, die auf dem Bildschirm darstellbar ist. Wir haben festgestellt, daß der graphische Bildschirm 640 Punkte breit und 400 Punkte hoch ist, jedoch ist die Breite der 'Pixels' abhängig vom jeweiligen Modus:

- Mode 0 – 4 Punkte breit
- Mode 1 – 2 Punkte breit
- Mode 2 – 1 Punkt breit

Die Höhe der Pixel beträgt immer 2 Punkte und ändert sich nicht mit dem Modus.

Weiteres über die Koordinaten

So, nun aber wieder zurück zu den Koordinaten. Später werden wir sehen, wie man Text und Zahlen an die richtige Stelle auf den Bildschirm setzt, aber im Moment wollen wir uns noch mit der graphischen Darstellung befassen. Das nächste Programm heißt KOORGEOM; bevor wir es jedoch laden, wollen wir noch ein sehr nützliches Schlüsselwort vorstellen: CAT.

CAT ist die Abkürzung für 'catalogue' (= Verzeichnis). Dieses Schlüsselwort wird verwendet, um festzustellen, welche Programme sich auf einer Datenkassette befinden. Spulen Sie die Datenkassette an den Anfang zurück, indem Sie die REW-Taste drücken, und geben Sie dann ein:

cat **ENTER**

Der CPC 464 wird mit folgender Mitteilung antworten:

```
Press PLAY and then any key ■
```

Es ist fast das Gleiche, wie bei einem LOAD- oder RUN-Befehl, nur daß der CPC 464 anstatt ein Programm zu laden, eine Bestätigung über das jeweils gefundene Programm ausgibt.

Diese Bestätigung sieht dann folgendermaßen aus:

```
DRAW      block 1 $
```

Auf einer Kassette sind die Programme immer in Blöcken von jeweils 2000 Zeichen gespeichert. Lange Programme bestehen aus vielen Blöcken, die einzeln gespeichert sind. Jedoch gibt der CPC 464 auf dem Bildschirm nicht nur Mitteilungen über die nacheinander folgenden Blöcke aus, sondern überprüft auch, ob keine Aufzeichnungsfehler vorliegen, und gibt dann am Ende der Zeile ein 'Ok' aus.

Wenn Sie dem CPC 464 den Namen eines Programms angeben, das er laden soll, erhalten Sie auch Bestätigungen über alle gefundenen Programme vor dem gewünschten, jedoch wird in diesem Falle keine zusätzliche Bandüberprüfung durchgeführt.

Inzwischen werden Sie auch das Programm KOORGEOM gefunden haben. Laden Sie es und gehen Sie dann weiter.

CAT

Spiel Nummer 3 – BOMBER

Dies ist das Spiel, auf das Sie sicher schon gewartet haben! Sie haben nun die Gelegenheit, ein außerirdisches Raumschiff abschießen zu können. Füttern Sie den Robot-Bomber mit den Koordinaten und feuern Sie dann die Plutonium-Bomben auf das Ziel ab – oder Sie werden vernichtet!

Test

Fall Sie die grausame Schlacht überlebt haben, testen Sie wieder Ihre Fortschritte mit dem Programm SAT4, ehe Sie zum nächsten Kapitel übergehen.

5 ZEICHNEN EINES BILDES

DRAW

PLOT

Das Schlüsselwort PLOT ist der erste BASIC-Befehl, den wir uns näher betrachten wollen. Bis jetzt haben Sie stets Dinge wie RUN und BORDER eingegeben, ohne eigentlich zu erkennen, daß es sich dabei um Befehle handelt, denen eine genaue Ablauffolge zu Grunde liegt.

Um Befehle herstellen zu können, müssen Sie oft an das Schlüsselwort noch einen 'Zusatz' hängen, der dafür sorgt, daß auch alle Einzelheiten der Operation ausgeführt werden. Bezogen auf PLOT gibt dieser Zusatz die gewünschte Position auf dem Bildschirm in x- und y-Koordinaten an. Geben Sie zum Beispiel folgendes ein:

```
plot 319, 199 ENTER
```

Sie erhalten nun einen gelben Punkt in der Größe eines 'Pixels', der sich ziemlich genau in der Mitte des Bildschirms befindet.

Falls Sie nicht mehr wissen sollten, was x und was y ist, denken Sie daran, daß Sie zuerst durch die Türe ins Haus gehen müssen, bevor Sie die Treppe hinaufsteigen können, also links-rechts, vor aufwärts-abwärts.

Nach Ausführung des PLOT-Befehls läßt der CPC 464 den graphischen Cursor auf diesen x/y-Koordinaten stehen, bis er einen anderen Befehl bekommt.

Machen wir mit einem anderen graphischen Befehl weiter:

```
draw 0,0 ENTER
```

Der DRAW-Befehl hat gleiche Struktur wie PLOT, nur daß der Zusatz in diesem Falle den Punkt angibt, zu dem die Linie gezogen werden soll. Dieser Punkt ist natürlich wieder in x/y-Koordinaten angegeben. Die diagonale Linie, die Sie nun auf dem Bildschirm sehen, beginnt im Zentrum (x = 319, y = 199) und endet in der linken unteren Ecke (x = 0, y = 0).

Versuchen Sie nun das gleiche Quadrat zu zeichnen, das wir schon im vorhergehenden Kapitel verwendet haben. Hier sind die Befehle dazu:

```
move 300,150 ENTER
draw 350,150 ENTER
draw 350,200 ENTER
draw 300,200 ENTER
draw 300,150 ENTER
```

Der MOVE-Befehl setzt den graphischen Cursor auf die x/y-Koordinaten, die in seinem Zusatz angegeben sind, ohne daß etwas auf dem Bildschirm gezeichnet wird.

Viereck-Programm

MOVE

Bis jetzt haben wir immer Befehle eingegeben, die direkt vom CPC 464 ausgeführt werden sollten. Nun werden wir lernen, wie man ein Programm eingibt und speichert, um es erst später ausführen zu lassen. Bevor wir dies tun können, müssen wir den Speicher des CPC 464 durch folgende Eingabe löschen:

new ENTER

NEW

Das bewirkt beim Speicher des CPC 464 das Gleiche, wie ein nasser Schwamm auf einer Schultafel. Obwohl der Bildschirm dabei nicht gelöscht wird, tilgt der NEW-Befehl jedes Programm, das vorher geladen oder eingegeben wurde. Benutzen Sie diesen Befehl nur, wenn Sie ganz sicher sind, daß durch ihn kein Schaden angerichtet wird!

Kommen wir nun zu unserem ersten gespeicherten Programm. Es ist wieder das gleiche Quadrat, nur haben wir diesmal Zeilennummern hinzugefügt:

```
10 clg
20 move 300,150
30 draw 350,150
40 draw 350,200
50 draw 300,200
60 draw 300,150
```

CLG

LIST

In Zeile 10 steht ein weiterer neuer Befehl. Während der CLS-Befehl den Bildschirm löscht und danach den Text Cursor in die linke obere Ecke setzt, schiebt der CLG-Befehl den graphischen Cursor nach dem Löschen des Bildschirms in die linke untere Ecke (x = 0, y = 0). Diese zwei Befehle werden Sie vielleicht etwas verwirren, besonders da sie ja scheinbar gleich sind. In Teil 2 werden Sie jedoch dann lernen, wie man gleichzeitig mit Text und graphischer Darstellung auf dem Bildschirm arbeitet und dann werden Sie den unterschiedlichen Sinn dieser beiden Befehle besser verstehen.

Geben Sie die sechs Zeilen dieses Programms genauso ein, wie es Ihnen gezeigt wurde, und drücken Sie am Ende jeder Zeile auf ENTER.

Auch im weiteren Verlauf des Kurses müssen Sie jetzt immer daran denken, am Ende einer Zeile auf die ENTER-Taste zu drücken.

Die Zeilennummern sind notwendig, damit Sie dem CPC 464 angeben können, in welcher Reihenfolge Sie die Befehle gespeichert haben wollen. Solange er keine anderen Anweisungen bekommt, werden die Befehle auch in dieser Reihenfolge ausgeführt. Die Zeilennummern sind in Zehnerschritten aufsteigend, so daß zusätzliche Zeilen eingefügt werden können, falls es notwendig sein sollte. Es spielt keine Rolle, in welcher Reihenfolge Sie die Zeilen eingeben.

Wenn Sie bei der Eingabe einen Fehler gemacht haben, den Sie erst nach dem Drücken der ENTER-Taste bemerken, können Sie die fehlerhafte Zeile im Speicher ganz einfach ersetzen, indem Sie sie noch einmal eingeben. Wenn Sie davon überzeugt sind, daß alles richtig eingegeben wurde, starten Sie das Programm durch folgende Eingabe:

run ENTER

Ist es nicht schön? Der CPC 464 nahm Ihr Programm in seinen Speicher auf und führte es erst aus, nachdem Sie ihm den RUN-Befehl gaben. Und es ist immer noch im Speicher vorhanden. Sie können sich davon überzeugen, indem Sie eingeben:

list ENTER

Das Wechseln der Farben

Zu den drei graphischen Befehlen, die wir gerade gelernt haben, kann noch als weiterer Zusatz INK hinzugefügt werden, der jedoch wahlfrei ist.

Geben Sie die Zeile 50 des Viereckprogramms noch einmal ein und zwar in folgender Weise:

```
50 draw 300,200,3
```

Wenn Sie das Programm dieses Mal laufen lassen, wird die linke und die obere Seite des Quadrats in rot gezeichnet. Durch das Hinzufügen der '3' geben wir die Anweisung, daß bei Ausführung des DRAW-Befehls die Farbe (INK) Nummer 3 verwendet wird. Der CPC 464 fährt dann damit fort, diese Farbe zu benutzen, bis wir ihn anweisen, sie zu wechseln. Dabei hängt der Bereich, aus dem die Farben entnommen werden können, jeweils vom benutzten Modus ab.

Die maximale Anzahl der verschiedenen INKs die für den jeweiligen Modus verwendet werden können ist wie folgt:

- Mode 0 – 16 INKs
- Mode 1 – 4 INKs
- Mode 2 – 2 INKs

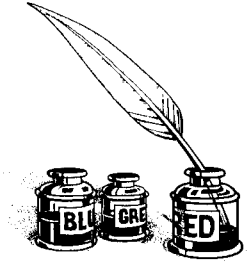
Versuchen Sie nun folgendes:

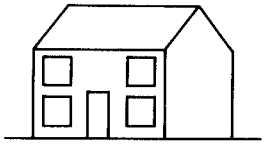
```
ink 3,0
```

ENTER

Sie werden sofort sehen, daß die rote Linie auf schwarz wechselt. Machen Sie das gleiche, was wir im vorhergehenden Kapitel mit BORDER gemacht haben und probieren Sie für INK Nummer 3 andere Farben aus. Auch können Sie versuchen, die INKs 0, 1 und 2 zu verändern.

INK





REM

PAPER

Zeichnen eines Hauses

Das nächste Programm auf der Datenkassette A heißt HAUS. Es verwendet alle Befehle, die wir bis jetzt gelernt haben, und zwei weitere kommen noch dazu.

Der erste Befehl heißt REM (REMark = Bemerkung). Wenn der CPC 464 auf REM stößt, ignoriert er den Rest der Zeile. Dadurch haben Sie die Möglichkeit, Kommentare und Erklärungen in Ihr Programm einzufügen, damit andere Personen (oder auch Sie selbst, wenn Sie es nach einiger Zeit vergessen haben sollten) das Programm verstehen können.

Der andere neue Befehl heißt PAPER. Durch ihn können Sie die Hintergrundfarbe des 'Bildschirm-Fensters' bestimmen.

Die Zusatzangabe bei PAPER muß die Nummer einer Farbe sein, die dem verwendeten Modus entspricht. Wenn Sie vorhin INK 0 veränderten, werden Sie festgestellt haben, daß der CPC 464 diese Farbzuordnung automatisch für den Hintergrund auswählte, als Sie ihn zum ersten Mal einschalteten.

Bitte lesen Sie zuerst die folgende Programmauflistung, bevor Sie das Programm laufen lassen:

```
10 REM Zeichnen eines Hauses
20 MODE 0
30 CLS
40 REM ★★ start ★★
50 BORDER 12
60 INK 0,12:REM gelb
70 INK 1,3:REM rot
80 INK 2,6: REM leuchtendes rot
90 INK 3,9:REM gruen
100 PAPER 0
110 REM zeichnen frontwand
120 MOVE 100,50
130 DRAW 100,250,1
140 DRAW 400,250
150 DRAW 400,50
160 DRAW 100,50
170 REM zeichnen seitenwand
180 MOVE 400,250
190 DRAW 600,250
200 DRAW 600,50
210 DRAW 400,50
220 DRAW 400,250
230 REM zeichnen giebel
240 REM bereits am startpunkt
250 REM darum kein MOVE noetig
260 DRAW 500,350
```

(Fortsetzung)

```
270 DRAW 600,250
280 DRAW 400,250
290 REM  zeichnen dach
300 REM nur zwei  linien noetig
310 MOVE 100,250
320 DRAW 200,350
330 DRAW 500,350
340 REM  zeichnen tuer
350 MOVE 225,50
360 DRAW 225,140,2
370 DRAW 275,140
380 DRAW 275,50
390 REM zeichnen fenster
400 REM links unten
410 MOVE 120,70
420 DRAW 120,130,3
430 DRAW 180,130
440 DRAW 180,70
450 DRAW 120,70
460 REM links oben
470 MOVE 120,170
480 DRAW 120,230
490 DRAW 180,230
500 DRAW 180,170
510 DRAW 120,170
520 REM rechts oben
```

(Fortsetzung)

```
530 MOVE 320,170
540 DRAW 320,230
550 DRAW 380,230
560 DRAW 380,170
570 DRAW 320,170
580 REM rechts unten
590 MOVE 320,70
600 DRAW 320,130
610 DRAW 380,130
620 DRAW 380,70
630 DRAW 320,70
```

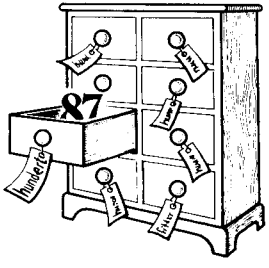
Test

Am besten gehen Sie dieses Kapitel noch einmal durch, bevor Sie SAT5 laufen lassen. Wir haben in relativ kurzer Zeit sehr viele grundlegende Dinge behandelt, kommen nun aber eigentlich erst zur richtigen Programmierung.

6 ZAHLEN, BUCHSTABEN UND WÖRTER

PRINT

LET



Es ist nun an der Zeit, daß wir auch etwas über Schlüsselwörter erfahren, die Zahlen und Wörter auf dem Bildschirm erscheinen lassen. Das erste, das wir besprechen wollen, heißt PRINT. Geben Sie folgendes über die Tastatur ein:

print 2+2 **ENTER**

Wie Sie vielleicht schon erraten haben, wird die Antwort darauf, die natürlich 4 lautet, in der nächsten Zeile auf dem Bildschirm ausgegeben. Versuchen Sie nun folgendes:

print "Hallo" **ENTER**

In diesem Fall müssen wir das, was wir auf dem Bildschirm ausgeben wollen, in Anführungszeichen setzen. Der Grund dafür wird Ihnen später noch klar werden. In der Zwischenzeit wollen wir uns mit einem anderen Schlüsselwort beschäftigen.

Variablenzuweisung

Manchmal erscheint die Computerwelt auf den ersten Blick recht seltsam. BASIC-Wörter sind zwar ganz normale englische Wörter, aber manchmal haben sie für den CPC 464 noch eine ganz besondere Bedeutung. Eines davon ist das Schlüsselwort LET. In der gewöhnlichen Algebra können Sie folgendes schreiben:

Let x=5

Jedoch können Sie auf keinen Fall schreiben:

Let x=x+5

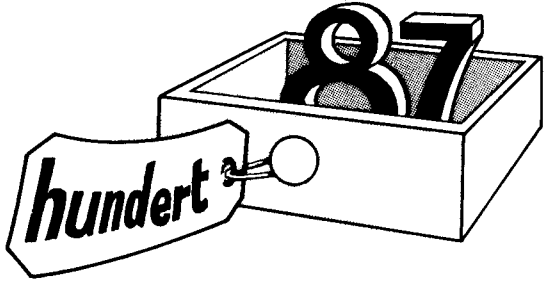
Das wiederum ist in BASIC möglich. Es bedeutet: 'Nehme den ursprünglichen Wert von x, addiere 5 dazu und nehme diese Zahl als neuen Wert von x an'. Warum aber x? Wir sind jetzt auf etwas gestoßen, das man als Variablen (= Platzhalter) bezeichnet.

Variablen enthalten alles das, was man in sie hineinschreibt. Es hindert Sie zum Beispiel nichts daran, in BASIC zu schreiben:

Let hundert = 87

Wenn Ihnen das Wort 'hundert' nicht gefällt, können Sie dafür auch 'h' oder 'C' schreiben, da das, wovon wir sprechen, vergleichbar ist, mit Namensschildchen an leeren Schachteln.

Wenn Sie den LET-Befehl benutzen, um eine Variable zu definieren, wie wir es eben gezeigt haben, schreibt der CPC 464 diesen Namen (der Variablen) auf eine leere Schachtel und füllt die Schachtel mit dem Wert, der rechts vom '=' Zeichen angegeben ist.



Geben Sie folgendes auf Ihrem CPC 464 ein:

Let hundert = 87 **ENTER**

Sie erhalten die Antwort 'Ready'. Geben Sie dann ein:

print hundert **ENTER**

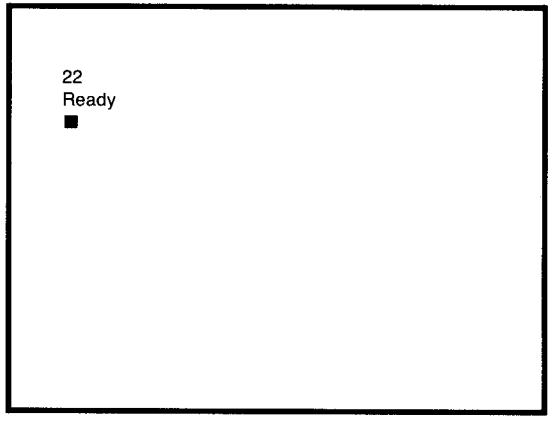
Auf Ihrem Bildschirm erscheinen nun die folgenden Zeilen:

```
Ready
let hundert = 87
Ready
print hundert
87
Ready
■
```

Wir sind nun an dem Punkt angelangt, wo wir unser zweites Programm eingeben können. Die folgenden fünf Zeilen sollen genauso eingegeben werden, wie es gezeigt wird:

```
10 cls
20 let a = 15
30 let b = 7
40 let a = a + b
50 print a
```

Denken Sie daran, daß Sie am Ende jeder Zeile auf ENTER drücken. Versuchen Sie nun, dieses Programm laufen zu lassen. Sie sollten folgendes Ergebnis erhalten:



Das war ganz offensichtlich. Beim nächsten Programm dagegen ist es schon nicht mehr so klar erkennbar:

```
10 let a=5
20 let b=10
30 let a=b
40 let b=a
50 print a+b
```

Versuchen Sie es selbst. Verstehen Sie, warum die Antwort 20 lautet? Vielleicht können Sie es sich besser vorstellen, wenn Sie an Zahlen denken, die von einer Schachtel in die andere geschoben werden. Denken Sie daran, daß immer die Variable links vom '=' Zeichen verändert wird, indem sie den Wert erhält, der rechts vom '=' Zeichen steht, und daß der CPC 464 das Programm Zeile für Zeile abarbeitet.

Text-Variablen

Wir haben gelernt, daß man eine Variable für Zahlen definieren kann und wir haben sie so verwendet, als wenn sie ein konkreter Gegenstand wäre, zum Beispiel eine Schachtel. Jedoch haben wir in BASIC auch die Möglichkeit, Variablen zu definieren, die eine ganze Reihe von Buchstaben, Zahlen und Satzzeichen aufnehmen können. Diese Variablen werden als 'Text-Variablen' bezeichnet.

Eine Text-Variable ist eigentlich das gleiche, wie eine normale Variable, nur daß der Name der Variablen mit einem '$' Zeichen enden muß. Außerdem müssen alle Schriftzeichen, die in die Variable geschrieben werden sollen, in Anführungszeichen gesetzt werden. Wenn das nicht der Fall ist, nimmt der CPC 464 an, daß Sie ihm eine zusätzliche Variable angeben. Geben Sie folgende Zeile ein:

```
let a$=Hallo
```
ENTER

Sie erhalten die Fehlermeldung:

```
Type mismatch
```
ENTER

Der CPC 464 nimmt an, daß Sie den numerischen Wert einer Variablen, genannt 'hallo', in die Text-Variable a$ schieben wollen. Sie hätten eingeben müssen:

```
let a$="Hallo"
```
ENTER

Jetzt können Sie eingeben:

```
print a$
```

Versuchen Sie es.

Geben Sie nun das folgende Programm ein. Bevor Sie es jedoch laufen lassen, sollen Sie versuchen herauszufinden, was das Ergebnis sein wird.

```
10 let a$ = "5"
20 let b$ = "12"
30 let a = 5
40 let b = 12
50 let c = a + b
60 let c$ = a$ + b$
70 print c
80 print c$
```

Namen für Variablen

Sie können als Variable jeden Namen angeben, den Sie wollen. Nur bei der Verwendung eines BASIC-Schlüsselwortes erhebt der CPC 464 Einspruch. Er akzeptiert zum Beispiel nicht:

```
let save = s + p
```

ENTER

Sie erhalten die Mitteilung 'syntax error'. Sie müssen den Namen der Variablen abändern in 'savers' oder 'savings'. Am Ende der Gebrauchsanweisung für den CPC 464 finden Sie eine Auflistung aller Schlüsselwörter, wenn Sie wissen wollen, ob ein Name als Variablenbezeichnung zulässig ist oder nicht.

Sie haben sicher schon bemerkt, daß es für den CPC 464 keine Rolle spielt, ob Sie die Befehle in Klein- oder Großbuchstaben eingeben. Unabhängig davon, wie Sie eingegeben haben, werden bei einer Programmauflistung die Schlüsselwörter immer in Großbuchstaben ausgegeben. Daher fallen die Namen der Variablen besser ins Auge, wenn sie in Kleinbuchstaben geschrieben sind.

Sicherung der Programme

Bis jetzt haben Sie immer Programme von den beiden mitgelieferten Datenkassetten geladen, oder selbst etwas über die Tastatur eingegeben. Sie haben nun die Möglichkeit das zuvor eingegebene Programm zu sichern, so daß wir es in diesem oder im nächsten Kapitel noch einmal verwenden können, ohne nochmal alles eingeben zu müssen. Nehmen Sie die Datenkassette aus dem Kassettenrekorder und legen Sie dafür eine neue Kassette ein, auf die man aufnehmen kann. Sie können ganz gewöhnliche Kassetten verwenden, nur sollten Sie C60-Kassetten verwenden, da längere Bänder zu dünn sind. Achten Sie außerdem darauf, daß das Band keinen Vorspann hat.

Spulen Sie die Kassette an den Anfang zurück und geben Sie folgendes ein:

SAVE

save "variable" ENTER

Sie können anstatt VARIABLE auch eine andere Bezeichnung wählen, mit Groß- und Kleinbuchstaben und Leerstellen zwischen den Wörtern. Nur müssen Sie am Anfang und am Ende des Namens Anführungszeichen setzen. Der CPC 464 wird folgendes antworten:

Press REC and PLAY and then any key

Folgen Sie den Anweisungen. Sie erhalten dann die Mitteilung:

Saving VARIABLE block 1

Der CPC 464 startet daraufhin den Datenkassettenrekorder und Sie hören, wie Ihr Programm auf die Kassette übertragen wird. Während dieser Zeit verschwindet der Cursor vom Bildschirm und erscheint erst wieder, nachdem der CPC 464 die Übertragung beendet hat und die 'ready'-Mitteilung bringt. Es ist immer ratsam, ein Programm zweimal zu sichern, falls etwas mit der Kassette passieren sollte. Wiederholen Sie deshalb den ganzen Vorgang noch einmal. Vergessen Sie nicht, danach wieder die Aufnahmetaste des Datenkassettenrekorders zu lösen.

Der PRINT-Befehl

Wenn Sie das zuletzt eingegebene Programm gerade haben laufen lassen, hat der CPC 464 noch die Variablen in seinem Speicher. Andernfalls laden Sie bitte das Programm noch einmal und lassen Sie es laufen, damit wir die Variablen verwenden können, um den PRINT-Befehl weiter zu untersuchen. Geben Sie folgendes ein:

```
print a,b,c ENTER
```

Sie sehen, daß dieser Befehl die drei Zahlen in einer Zeile auf dem Bildschirm erscheinen läßt, jedoch mit 13 Stellen Abstand voneinander. Das ist der einfachste Weg, um Zahlen spaltengerecht auflisten zu können, aber es ist nur unter Anwendung von 'mode 1' oder 'mode 2' möglich. Versuchen Sie das Gleiche mit 'mode 0', um es feststellen zu können.

Wenn Sie nicht wollen, daß die Zahlen durch Leerstellen voneinander getrennt sind, können Sie folgendes machen:

```
print a;b;c ENTER
```

Auch diesmal werden die Zahlen in der gleichen Zeile ausgegeben, jedoch ohne jede Trennung, so daß wir oft Leerstellen einfügen müssen, wie im folgenden Beispiel:

```
print "Der Wert von c ist ";c;" nicht ";c$
```

Wir können mit Hilfe des Schlüsselwortes LOCATE überall auf dem Bildschirm Buchstaben und Zahlen erscheinen lassen. Der Aufbau des Befehls sieht folgendermaßen aus:

```
locate x,y
```

Das kommt Ihnen sicher bekannt vor, nicht wahr? Jedoch haben diese x- und y-Koordinaten nichts mit den graphischen Koordinaten zu tun. Der Befehl LOCATE bewegt den Text-Cursor an diejenige Stelle auf dem Bildschirm, die durch seine Zusatzangaben bestimmt ist. Text-Koordinaten haben ihren Ausgangspunkt in der linken oberen Ecke des Bildschirms (der die Koordinaten x = 1, y = 1 besitzt), und nach unten gezählt.

Geben Sie folgendes ein:

```
75 locate 20,13 ENTER
```

Löschen Sie den Bildschirm mit dem CLS-Befehl. Wenn Sie nun das Programm mit dieser neu eingefügten Zeile laufen lassen, werden Sie feststellen, daß der Wert von c$ nun in der Mitte des Bildschirm ausgegeben wird, beginnend bei Stelle 20 in Zeile 13.

BALKENDIAGRAMM

Der Name des nächsten Programms auf der Datenkassette lautet BALKENDIAGRAMM. Darin werden vier Zahlen zwischen 0 und 290 optisch dargestellt – so wie Sie es von Wahlergebnissen oder Meinungsumfragen her kennen. Diese Art von Programm wartet an bestimmten Punkten auf Zahlen, die über die Tastatur eingegeben werden müssen, bevor es weitermachen kann. Das Schlüsselwort dafür ist INPUT und ein typischer Befehl der folgende:

INPUT

```
10 INPUT name$
```

Der CPC 464 wartet geduldig in Zeile 10, bis etwas eingegeben und danach die ENTER-Taste gedrückt wird. Die Information wird dann in die Text-Variable 'name$' geschoben und das Programm läuft weiter.

Lassen Sie das Programm BALKENDIAGRAMM laufen, bevor Sie sich die folgende Programmauflistung näher ansehen. Sie werden feststellen, daß der CPC 464 ein Fragezeichen (?) setzt, hinter dem der Cursor steht, wenn er auf eine Eingabe wartet. Außerdem wird ein 'Stichwort' angegeben, damit der Anwender weiß, was als Eingabe erwartet wird. Sie machen diese Eingabe, indem Sie eine Mitteilung hinter das INPUT-Schlüsselwort setzen:

```
INPUT "Wert (1-290)";a
```

Die Stichwort-Angabe muß in Anführungszeichen stehen und von der Variablen durch einen Strichpunkt (;) getrennt sein.

Sie sehen, daß viele Zeilen im Programm BALKENDIAGRAMM mehrere Befehle enthalten, die durch Doppelpunkte (:) voneinander getrennt sind. Diese Doppelpunkte bedeuten das Gleiche, als würde man einen Befehl mit Zeilennummer beginnen und mit ENTER abschließen. Das bietet die Möglichkeit, mehrere Befehle, die miteinander in Beziehung stehen, zu verbinden, besonders dann, wenn sie sehr kurz sind.

Eine sehr gute Anwendung des Doppelpunktes ist die folgende:

```
50 b=50:REM Strichlaenge
```

Wenn man mit dem REM-Befehl eine Bemerkung neben einen anderen Befehl setzt, um dadurch seine Bedeutung zu erklären, kann man das Programm zu einem späteren Zeitpunkt wesentlich leichter lesen.

```
10 REM 3D BALKENDIAGRAMM
30 MODE 1
40 BORDER 14:INK 0,14:INK 1,0:INK 2,3:INK 3,24
50 b=50:REM balkengroesse
60 LOCATE 1,23:INPUT "Wert (1-290)";a
70 x=100:PLOT x,55,1
80 DRAW x-b★2,55:DRAW x-b★2,a+55
90 DRAW x-b,a+55+b:DRAW x+b,a+55+b
100 DRAW x,a+55:DRAW x-b★2,a+55
110 MOVE x+b,a+b+55:DRAW x+b,b+55
120 DRAW x,55:DRAW x,55+a
130 LOCATE 1,23
140 PRINT"                                                    "
150 LOCATE 1,23:INPUT "Wert (1-290)";a
160 x=260:PLOT x,55,2
170 DRAW x-b★2,55:DRAW x-b★2,a+55
180 DRAW x-b,a+55+b:DRAW x+b,a+55+b
190 DRAW x,a+55:DRAW x-b★2,a+55
200 MOVE x+b,a+b+55:DRAW x+b,b+55
210 DRAW x,55:DRAW x,55+a
220 LOCATE 1,23
230 PRINT"                                                    "
240 LOCATE 1,23:INPUT "Wert (1-290)";a
250 x=420:PLOT x,55,3
260 DRAW x-b★2,55:DRAW x-b★2,a+55
270 DRAW x-b,a+55+b:DRAW x+b,a+55+b
```

(Fortsetzung)

```
280 DRAW x,a+55:DRAW x-b★2,a+55
290 MOVE x+b,a+b+55:DRAW x+b,b+55
300 DRAW x,55:DRAW x,55+a
310 LOCATE 1,23
320 PRINT"                                                    "
330 LOCATE 1,23:INPUT "Wert (1-290)";a
340 x=580:PLOT x,55,1
350 DRAW x-b★2,55:DRAW x-b★2,a+55
360 DRAW x-b,a+55+b:DRAW x+b,a+55+b
370 DRAW x,a+55:DRAW x-b★2,a+55
380 MOVE x+b,a+b+55:DRAW x+b,b+55
390 DRAW x,55:DRAW x,55+a
```

Spiel Nummer 4

Manche Leute verwenden oft hochtrabende Worte – besonders auf dem Computer-Sektor. Unser automatischer WORTVERMISCHUNGs-Generator wird Ihnen helfen, damit Sie mit den gleichen Worten zurückgeben können.

Test

Lassen Sie SAT6 laufen, um festzustellen ob Sie einen Teil dieses Kapitels noch einmal überarbeiten müssen, und gehen Sie dann zum nächsten Kapitel über.

7 KORREKTUREN UND ÄNDERUNGEN

Der größte Teil dieses Kapitels befaßt sich mit Korrekturen und Programmänderungen. Das Programm, das Sie im vorhergehenden Kapitel eingegeben haben, eignet sich hervorragend für diesen Zweck. Spulen Sie deshalb die Kassette zurück und laden Sie das Programm auf's Neue:

load "variable" **ENTER**

VARIABLE ist der Name, den Sie dem Programm gaben, als Sie es mit dem SAVE-Befehl auf der Datenkassette speicherten.

Ändern einer Zeile

Wenn Sie ein Programm über die Tastatur eingeben, können Sie sicher sein, daß Ihnen dabei Fehler unterlaufen, auch wenn Sie nur etwas abschreiben. Der häufigste Fehler ist für gewöhnlich, daß Sie sich vertippen. Meistens bemerken Sie es jedoch gleich und können vor dem Drücken der ENTER-Taste zurückgehen, um den Fehler mit Hilfe der DEL-Taste zu verbessern.

Während der Eingabe einer Zeile (sie wird als 'laufende' Zeile bezeichnet) können Sie den Cursor über die Cursor-Tasten so oft in der Zeile hin- und herbewegen, wie Sie wollen. Neue Zeichen können eingefügt werden, indem man sie einfach eintippt, und mit den CLR- und DEL-Tasten werden Zeichen gelöscht.

Im vorhergehenden Kapitel haben wir schon gesehen, daß man vollständige Zeilen ersetzen kann, indem man sie einfach noch einmal neu eingibt. Der CPC 464 ersetzt dann automatisch die alte Zeile durch die neue Zeile.

Wenn Sie einmal mehrere eigene Programme geschrieben haben, werden Sie feststellen, daß sie beim ersten Mal meistens nicht fehlerfrei laufen. Der CPC 464 wird Ihnen einige Fehlermeldungen ausgeben, wenn Sie Zeilen eingeben und einige weitere, wenn Sie versuchen, das Programm laufen zu lassen. Aber er kann Ihnen nicht mitteilen, ob Sie zum Beispiel Befehle ausgelassen haben oder vergessen haben, den Cursor in eine neue Stellung zu bringen, bevor Sie eine Linie zeichnen. Daher der Grund, warum die Zeilennummern, die wir verwendet haben, in Zehnerschritten aufsteigend sind. Im vorhergehenden Kapitel konnten wir dadurch eine Zeile zwischen 70 und 80 einfügen, indem wir eingaben:

```
75 locate 20,13
```

Im Extremfall können Sie bis zu neun Zeilen an dieser oder einer anderen Stelle einfügen. Genausogut können Sie aber auch Zeilen löschen, indem Sie eine leere Zeile eingeben. So kann die Zeile, die wir im letzten Beispiel eingefügt haben, durch folgende Eingabe wieder gelöscht werden:

75 **ENTER**

Der CPC 464 läßt daraufhin die Zeile 75 aus dem Programm verschwinden. Versuchen Sie es und lassen Sie sich dann das Programm auflisten, um zu sehen, was passiert ist.

Änderung eines eingegebenen Programms

Wenn Sie ein bereits eingegebenes oder geladenes Programm ändern wollen, können Sie das mit der EDIT-Funktion bewirken. Benutzen Sie noch einmal das vorhergehende Beispiel und geben Sie ein:

EDIT

edit 40 **ENTER**

Daraufhin erscheint die Zeile 40 auf dem Bildschirm. Der Cursor steht auf dem ersten Zeichen dieser Zeile. Benutzen Sie dann die Cursor-Tasten für die Rechts- und Linksbewegung und bringen Sie den Cursor damit zu dem Teil der Zeile, den Sie ändern wollen. Sie können dabei genauso vorgehen, als wären Sie gerade in der 'laufenden' Zeile bei der Eingabe eines Programms. Probieren Sie es selbst aus, indem Sie den Wert von b von 15 auf 26 ändern. Schieben Sie den Cursor auf die '1' der Zahl '15' und drücken Sie dann zweimal auf die CLR-Taste. Dadurch wird die Zahl '15' gelöscht. Geben Sie nun '26' ein und drücken Sie dann auf die ENTER-Taste. Der Cursor muß dabei nicht unbedingt am Ende der Zeile stehen. Wenn Sie das Programm nun auflisten (LIST) oder laufen (RUN) lassen, werden Sie feststellen, daß der CPC 464 die Zeile genauso abgeändert hat, wie Sie es wollten.

Es gibt aber auch noch einen leichteren Weg, um Zeilen zu ändern, und zwar durch die COPY-Taste. Mit Hilfe eines zweiten Cursors, 'COPY Cursor' genannt, kann man an jeder Stelle des

Bildschirms ganze Zeilen oder Teile von Zeilen herausnehmen.

Der COPY-Cursor erscheint, wenn man auf SHIFT drückt und gleichzeitig auf eine der Cursor-Tasten. Lassen Sie sich das Programm noch einmal auflisten und versuchen Sie, den COPY-Cursor an den Anfang einer der Zeilen zu setzen. Der normale Cursor verändert dabei seine Position nicht. Drücken Sie auf die COPY-Taste und kopieren Sie dadurch jedes Zeichen der 'laufenden' Zeile, auf der der normale Cursor steht. Wenn Sie dabei die COPY-Taste unten halten, wird alles automatisch noch einmal geschrieben!

Sie können an jeder Stelle der Originalzeile den Kopiervorgang anhalten und neue oder geänderte Informationen in die 'laufende' Zeile schreiben, bevor Sie mit dem Kopieren fortfahren.

Wenn Sie sich an den COPY-Cursor gewöhnt haben, werden Sie den EDIT-Befehl nicht mehr häufig verwenden. Das Ergebnis ist im Prinzip das gleiche, aber Sie haben bei Verwendung der COPY-Taste den Vorteil, daß Sie in einer Zeile diejenigen Zeichen, die nicht mehr erscheinen sollen, weglassen können.

Der LET-Befehl

Wahrscheinlich haben Sie es im letzten Kapitel als sehr ermüdend empfunden, ständig LET am Anfang einer Zeile einzugeben. Nun müssen wir Ihnen jedoch ein Geständnis machen; in Schneider-BASIC ist es nicht unbedingt notwendig, daß man dieses Schlüsselwort verwendet.

Verändern Sie das Programm VARIABLE noch einmal, indem Sie alle LET-Befehle herausnehmen. Wenn Sie dann das Programm laufen lassen, werden Sie feststellen, daß sich nichts geändert hat!

Das Überspringen von Zeilen

Wir haben schon einmal erwähnt, daß die Zeilennummern dem CPC 464 angeben, in welcher Reihenfolge die Befehle gespeichert werden sollen, aber wir sagten auch, daß sie nicht unbedingt in dieser Reihenfolge ausgeführt werden müssen. Manchmal wollen wir im Programm vorwärts und rückwärts springen und dafür verwenden wir das Schlüsselwort GOTO. Der Befehl hat als Zusatz eine Zeilennummer nangabe. Zum Beispiel:

```
70 goto 130
```

Durch diese Zeile sagen Sie dem Programm, daß es alle Zeilen zwischen 70 und 130 überspringen soll. Sie können das Gleiche auch umgekehrt machen:

```
130 goto 70
```

Das Programm geht nun zurück zu Zeile 70 und führt die Zeile aus, indem es auf Zeile 130 springt und das setzt sich so lange fort, bis Sie den CPC 464 ausschalten oder auf die ESC-Taste drücken.

Was wird als nächstes gemacht?

Von uns werden in unserem täglichen Leben ständig Entscheidungen verlangt, auch wenn es sich dabei oft um unwichtige Dinge handelt, zum Beispiel, ob man Kaffee oder Tee trinken soll, oder welche Schuhe man zum Ausgehen anziehen soll. Wenn wir uns dann entschieden haben, ob wir Kaffee, Tee oder garnichts trinken wollen und ob wir schwarze, rote oder überhaupt keine Schuhe anziehen wollen, entspricht unsere darauffolgende Handlungsweise der Entscheidung. Der CPC 464 arbeitet die verschiedenen Möglichkeiten der Handlungsweise durch das Schlüsselwort IF aus.

GOTO

IF

Die folgende Zeile könnte aus einem Programm entnommen sein, das den Kontostand Ihres Sparkontos prüft. Dabei gibt die Variable 'Geld' den aktuellen Stand an:

```
IF Geld = 0 THEN PRINT "leere Kasse"
```

THEN

Das Interessante dabei ist, daß der PRINT-Befehl nur ausgeführt wird, wenn die Variable 'Geld' den Wert 0 hat – ansonsten geht der CPC 464 gleich weiter zur nächsten Zeile des Programms.

Ein noch besserer Weg, um das Gleiche zu erreichen, ist die zusätzliche Verwendung des Schlüsselwortes ELSE in folgender Art und Weise:

ELSE

```
IF Geld>0 THEN PRINT "reich" ELSE PRINT "leere Kasse"
```

Das Zeichen '>' bedeutet 'größer als' oder 'mehr als' und prüft, ob Sie der Bank Geld schulden oder nicht! Wenn der Wert der Variablen 'Geld' Null oder weniger beträgt, ignoriert der CPC 464 das, was zwischen THEN und ELSE steht, und führt gleich die Anweisungen aus, die auf ELSE folgen. ELSE bietet die Möglichkeit einer alternativen Handlungsweise, bevor zur nächsten Zeile des Programms weitergegangen wird.

Im vorhergehenden Kapitel untersuchten wir ein Programm mit dem Namen BALKENDIAGRAMM, das die folgende Eingabe-Anweisung enthielt:

```
Wert (0-290)?
```

Wenn Sie versucht haben, eine Zahl größer als 290 einzugeben, schien es das Programm nicht zu bemerken und zeichnete den Strich über den oberen Rand des Bildschirms hinaus. Diese Möglichkeit könnten Sie ausschließen, wenn Sie die Variable 'a' auf einen maximalen Wert beschränken würden. Zum Beispiel:

```
IF a>290 THEN a=290
```

Wie Sie sehen, ist der IF...THEN Befehl nicht an PRINT-Anweisungen gebunden und diese Zeile begrenzt 'a' auf einen Wert, der auf den Bildschirm paßt.

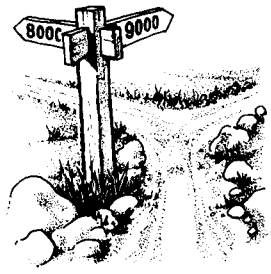
Die GOTO-Anweisung

Das Schlüsselwort IF kommt erst richtig zur Geltung, wenn Sie es in Verbindung mit der GOTO-Anweisung verwenden. Es ist wie ein Wegweiser, der dem CPC 464 angibt, welchen Teil des Programms er als nächstes ausführen soll. Angenommen, Sie wollen verhindern, daß beim Programm BALKENDIAGRAMM höhere Zahlen als 290 eingegeben werden. Das wird durch die folgenden Zeilen erreicht:

```
60 INPUT "Wert (0-290)";a
65 IF a>290 GOTO 60
```

So lange, bis eine Zahl kleiner oder gleich 290 eingegeben wird, springt der CPC 464 auf Zeile 60 zurück. Diese 'Falle' verhindert die Eingabe nicht 'zulässiger' Zahlen und bewirkt, daß sie auf die richtige Größe reduziert werden.

Beseitigung von Logikfehlern



Sie können sich sicher vorstellen, daß der CPC 464 beim kleinsten Fehler in den vorhergehenden Anweisungen an eine falsche Stelle im Programm springt. Damit sind wir bei einem klassischen Ausdruck in der Computersprache angelangt – dem Logikfehler.

Maschinen können falsche Ergebnisse erzeugen, Menschen machen Fehler. Wir sagten einmal, daß Computer nicht denken können, und daß deshalb der Programmierer das Denken für ihn übernehmen muß. Wenn nun der Programmierer einen Denkfehler macht, und dagegen ist auch der Beste nicht gefeit, bemerkt er ihn meistens erst beim ersten Testlauf. Der CPC 464 macht genau das, was ihm gesagt wird, aber es muß nicht unbedingt mit dem übereinstimmen, was der Programmierer ursprünglich beabsichtigte.

Das Programm muß nun auf 'Logik-Fehler' hin überprüft werden. Aber es ist durchaus keine Schande, wenn man logische Fehler oder Gedankensprünge macht. Zwar haben erfahrene Programmierer in einem fertigen Programm weniger Logikfehler als Anfänger, aber das ist nur eine Frage des Wissens und der praktischen Erfahrung. Auch Ihnen werden mit der Zeit immer weniger Logikfehler unterlaufen.

Es ist ratsam, ein Programm noch einmal in Gedanken durchzugehen, bevor man es laufen läßt. Man macht zum Beispiel einen 'Trocken-Lauf', indem man von Zeile zu Zeile geht und jedesmal die Werte der Variablen und die Ergebnisse der Summen aufschreibt.

Probieren wir es an folgendem Beispiel:

```
10 a=0
20 print a
30 a=a+1
40 if a<4 then goto 20
```

Bevor Sie das in den CPC 464 eingeben, sollten Sie sich ein Stück Papier nehmen und es in drei Spalten einteilen, mit den folgenden Überschriften:

Schritt Zeilennummer Wert von a

Gehen Sie nun das Programm durch und führen Sie in Gedanken jede Anweisung genauso aus, wie es der CPC 464 tun würde.

Und so sollte es aussehen:

Schritt	Zeilennummer	Wert von a
1	10	0
2	20	0
3	30	1
4	40	1
5	20	1
6	30	2
7	40	2
8	20	2
9	30	3
10	40	3
11	20	3
12	30	4
13	40	4

Durch diesen 'Trockentest' werden Sie gezwungen, das Programm aus der Sicht des Computers zu betrachten, und irgendwelche Unklarheiten kommen sofort ans Tageslicht. Das Hauptproblem nach der Fertigstellung eines Programms ist meistens, daß Sie zwar genau wissen, was es machen soll, aber einfach nicht herausfinden, warum es nicht funktioniert.

Eine andere Technik zur Programmüberprüfung ist der vorübergehende Einbau von PRINT-Anweisungen an bestimmten Stellen des Programms, damit Sie mitverfolgen können, wie sich der Wert der Variablen ändert.

Die folgende Zeile könnte man zum Beispiel ganz leicht in das vorhergehende Programm einfügen:

```
35 print "Zeile 35 a= ";a
```

Sie können auch den STOP-Befehl verwenden:

```
35 stop
```

Der CPC 464 unterbricht dann an diesem Punkt den Programmlauf und Sie können sich die Variablen, die Sie interessieren, durch einen direkten PRINT-Befehl ausgeben lassen.

Durch die folgende Eingabe wird der Programmlauf fortgesetzt:

```
cont
```
ENTER

Denken Sie jedoch daran, daß dieser CONT-Befehl nicht funktioniert, wenn nach dem Stoppen des Programms irgendeine Zeile hinzugefügt oder gelöscht wurde.

Hausrenovierung

Kehren wir wieder zum Haus zurück. Diesmal wollen wir ihm etwas Farbe geben. Nehmen Sie deshalb Ihre Farbtabelle zur Hand und laden Sie das Programm DEKO.

STOP

Es ist kein richtiges Spiel, sondern verbindet auf sehr unterhaltsame Art und Weise alle Befehle, die Sie bis jetzt kennengelernt haben. Vielleicht wollen Sie das Programm auch etwas abändern, um Ihrem Haus eine persönliche Note zu geben. Inzwischen müßten Sie eigentlich genug wissen, um mit Hilfe der beschriebenen Techniken eine Programmänderung vornehmen zu können.

CONT

Machen Sie sich keine Gedanken über irgendwelche unbekannten Schlüsselwörter in der Auflistung des Programms DEKO. Die Erklärungen dafür werden in den späteren Kapiteln folgen.

```
10 REM deko
20 MODE 0
30 CLS
40 REM ★★ start ★★
50 BORDER 12
60 INK 0,12:REM gelb
70 INK 3,3:REM rot
80 INK 6,6:REM leuchtend rot
90 INK 9,9:REM gruen
100 PAPER 0
110 REM zeichnen frontwand
120 MOVE 100,50
130 DRAW 100,250,3
140 DRAW 400,250
150 DRAW 400,50
160 DRAW 100,50
170 REM zeichnen seitenwand
180 MOVE 400,250
190 DRAW 600,250
200 DRAW 600,50
210 DRAW 400,50
220 DRAW 400,250
230 REM zeichnen giebel
240 REM bereits am startpunkt
250 REM darum kein MOVE noetig
260 DRAW 500,350
270 DRAW 600,250
280 DRAW 400,250
```

(Fortsetzung)

```
290 REM  zeichnen dach
300 REM nur zwei linien noetig
310 MOVE 100,250
320 DRAW 200,350
330 DRAW 500,350
340 REM  zeichnen tuer
350 MOVE 225,50
360 DRAW 225,140,6
370 DRAW 275,140
380 DRAW 275,50
390 REM  zeichnen fenster
400 REM links unten
410 MOVE 120,70
420 DRAW 120,130,9
430 DRAW 180,130
440 DRAW 180,70
450 DRAW 120,70
460 REM  links oben
470 MOVE 120,170
480 DRAW 120,230
490 DRAW 180,230
500 DRAW 180,170
510 DRAW 120,170
520 REM rechts oben
530 MOVE 320,170
540 DRAW 320,230
550 DRAW 380,230
560 DRAW 380,170
```

(Fortsetzung)

```
570 DRAW 320,170
580 REM rechts unten
590 MOVE 320,70
600 DRAW 320,130
610 DRAW 380,130
620 DRAW 380,70
630 DRAW 320,70
640 REM ★★★ DEKO ★★★
650 r$ = CHR$(18)
660 LOCATE 1,25
670 PRINT "Farbnummer? (1-15)";
680 FOR i = 1 TO 15
690 INK i,i:NEXT i
700 LOCATE 1,1:PRINT r$;
710 INPUT "Dachfarbe";r
720 FOR i = 107 TO 399 STEP 2
730 MOVE i,252:DRAW i + 96,349,r
740 NEXT i
750 LOCATE 1,1:PRINT r$;
760 INPUT "Giebelfarbe";g
770 FOR i = 252 TO 346
780 MOVE i + 154,i:DRAW 848-i,i,g
790 NEXT i
800 LOCATE 1,1:PRINT r$;
810 INPUT "Seitenwandfarbe";e
820 FOR i = 52 TO 248 STEP 2
830 MOVE 404,i:DRAW 598,i,e
840 NEXT i
```

(Fortsetzung)

```
850 LOCATE 1,1:PRINT r$;
860 INPUT "Frontwandfarbe";f
870 FOR i=52 TO 248 STEP 2
880 MOVE 104,i:DRAW 398,i,f
890 NEXT i
900 LOCATE 1,1:PRINT r$;
910 INPUT "Tuerfarbe";d
920 FOR i=52 TO 138
930 MOVE 229,i:DRAW 268,i,d
940 NEXT i
950 LOCATE 1,1:PRINT r$;
960 INPUT "Fensterfarbe";w
970 FOR j=0 TO 100 STEP 100
980 FOR i=70+j TO 130+j
990 MOVE 120,i:DRAW 180,i,w
1000 MOVE 320,i:DRAW 380,i
1010 NEXT i
1020 NEXT j
1030 END
```

Test

Mit Hilfe des Programms SAT7 können Sie überprüfen, ob Sie alles in diesem Kapitel verstanden haben. Denken Sie sich nichts dabei, wenn Sie im Buch zurückblättern müssen, um eine Frage beantworten zu können. Die meisten Programmierer haben beim Arbeiten eine Programmierrichtlinie neben sich liegen.

8 VERBESSERUNGEN BEI DER HAUSDARSTELLUNG

In der Programmierung gibt es einige Aufgaben, vergleichbar mit Hausaufgaben, die sich ständig wiederholen. Für diese Art von Aufgaben ist der CPC 464 genau wie alle anderen Computer sehr gut einsetzbar. Wenn wir ihn mit dem entsprechenden Programm arbeiten lassen, wiederholt er es so lange, bis wir ihm andere Anweisungen geben.

Kehren wir nun zu unserem Hausbeispiel zurück und versuchen es noch weiter auszubauen, ohne dabei viel programmieren zu müssen.

Laden Sie das Programm 'VILLA', das als nächstes auf der Datenkassette folgt. Auf den ersten Blick scheint es wieder das gleiche alte Haus zu sein, jedoch wollen wir diesmal versuchen, mit Hilfe von 'Schleifen' und 'Subroutinen' möglichst geschickte optische Eindrücke zu erzeugen.

Die Programmschleife

Es wird Sie vielleicht überraschen, aber Sie haben im vorhergehenden Kapitel schon eine Programmschleife kennengelernt. Nur haben wir sie dort noch nicht als solche bezeichnet. Das folgende Programm enthält eine Schleife:

```
10 a=0
20 print a <----------
30 a=a+1              |
40 if a<4 then goto 20 ----------
```

Der Sinn dieses Programms ist folgender: 'Drucke den Wert von a aus, springe zu Zeile 20 zurück und wiederhole das so oft, bis der Wert von a größer als 3 ist!

Wir könnten das Programm auch folgendermaßen schreiben:

```
10 for a=0 to 3
20 print a
30 next a
```

Wenn Sie sich die Auflistung des Programms 'VILLA' ansehen, werden Sie feststellen, daß es nicht bei Zeile 640 endet. Wir wollen uns zuerst die neu dazugekommenen Zeilen von Zeilennummer 640 bis einschließlich 680 näher betrachten.

Wenn Sie das Programm laufen lassen, zeichnet es das Haus wieder genauso, wie wir es schon gesehen haben. Wenn es beim STOP-Befehl angelangt ist, können Sie auch noch den letzten Teil laufen lassen, wenn Sie eingeben:

CONT **ENTER**

Versuchen Sie es. Wie gefällt Ihnen der Zaun? Nun kann der Hund des Nachbarn nicht mehr in den Garten laufen!

Den Schlüssel zu dieser Operation finden Sie in Zeile 650, betrachten wir sie deshalb einmal genauer:

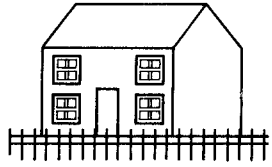
```
650 FOR F=0 TO 620 STEP 20
```

Durch den FOR-Befehl wird dem CPC 464 mitgeteilt, daß er nun eine Programmschleife durchläuft. 'F=0' gibt an, wo die Schleife beginnt; 'TO 620' gibt das Ende an. Durch die Anweisung 'STEP 20' wird der Abstand zwischen den einzelnen Zaunpfosten festgelegt, indem der CPC 464 aufgefordert wird, bei jedem Durchlauf der Schleife den Wert von 'F' um 20 zu erhöhen.

In Zeile 660 wird der aktuelle Wert von 'F' benutzt, um den graphischen Cursor an die Stelle des nächsten Zaunpfostens zu setzen, damit die Linie des Zaunpfostens gezeichnet werden kann.

Die Anweisung NEXT, die Sie in Zeile 670 finden, vervollständigt die Schleife. Sie bedeutet: "Gebe mir den nächsten Wert von 'F' an, nachdem 'F', wie schon beschrieben, durch 'step' verändert wurde.

Um es noch einmal kurz zusammenzufassen, der CPC 464 durchläuft eine Programmschleife für die Werte F = 20, 40, 60, 80... usw, bis der Wert F = 640 erreicht wird. Dann übergeht der CPC 464 die Anweisung NEXT und führt Zeile 680 aus, indem er die Querbalken des Zaunes zeichnet.



FOR

STEP

NEXT

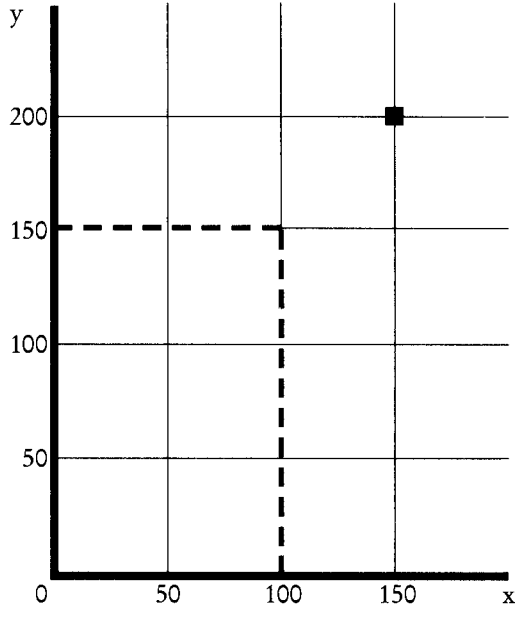
PLOTR

MOVER

DRAWR

Abhängige Koordinaten

Bevor wir weitergehen wollen wir die Betrachtung der graphischen Anweisungen mit den Befehlen PLOT, DRAW und MOVE abschließen. Sie erinnern sich, daß die Zusatzangaben für die graphischen Befehle stets die x- und y-Koordinaten sind, ausgehend vom graphischen Nullpunkt, beschrieben durch x = 0 und y = 0.



Die Zusätze der Befehle PLOTR, DRAWR und MOVER sind jedoch die erforderlichen Verschiebungen in x- und y-Richtung, ausgehend von dem aktuellen Stand des graphischen Cursors.

Auf dem Diagramm können Sie sehen, daß die folgende Anweisung den graphischen Cursor in die Position x = 150, y = 200 bringt:

```
mover 50,50 ENTER
```

Das gleiche Beispiel ist auch auf die Befehle DRAWR und PLOTR anwendbar. Ihre Zusatzangaben sind relative Koordinaten, die den aktuellen Stand des Cursors als Nullpunkt verwenden. Dadurch muß der Programmierer die Koordinaten nicht jedesmal neu berechnen. Welchen praktischen Vorteil das mit sich bringt, werden Sie etwas später in diesem Kapitel erfahren.

Vervollständigung der Fenster

Sie haben bestimmt schon bemerkt, daß wir mit dem Haus noch nicht fertig sind. Nach der Programmschleife, durch die der Zaun gezeichnet wird, finden Sie einen weiteren STOP-Befehl in Zeile 685. Die darauffolgenden Zeilen enthalten einen neuen Befehl – GOSUB. Lassen Sie das Programm bis zum Ende durchlaufen, indem Sie CONT eingeben. Nun wird es Zeit, daß wir Glasscheiben in die Fenster einsetzen. Das erfolgt mit Hilfe einer 'Subroutine'. Der Ausdruck 'Subroutine' wird in der Programmierung für eine Anweisungsfolge verwendet, die wiederholt aufgerufen werden kann, immer dann, wenn sie gebraucht wird. Wenn wir beim Zeichnen der Fensterscheiben nicht mit einer Subroutine arbeiten würden, müßten wir die gleiche Anweisungsfolge 16-mal schreiben. Durch den Befehl GOSUB springt das Programm auf die erste Zeile des Unterprogramms, deren Zeilennummer als Zusatz von GOSUB angegeben ist.

GOSUB

```
10 REM villa
20 MODE 0
30 CLS
40 REM ★★ start ★★
50 BORDER 12
60 INK 0,12:REM gelb
70 INK 3,3:REM rot
80 INK 6,6:REM leuchtendes rot
90 INK 9,9:REM gruen
95 INK 15,15:REM orange
100 PAPER 0
110 REM zeichnen frontwand
120 MOVE 100,50
130 DRAW 100,250,3
```

(Fortsetzung)

```
140 DRAW 400,250
150 DRAW 400,50
160 DRAW 100,50
170 REM zeichnen seitenwand
180 MOVE 400,250
190 DRAW 600,250
200 DRAW 600,50
210 DRAW 400,50
220 DRAW 400,250
230 REM zeichnen giebel
240 REM bereits am startpunkt
250 REM darum kein MOVE noetig
260 DRAW 500,350
270 DRAW 600,250
280 DRAW 400,250
290 REM zeichnen dach
300 REM nur zwei linien noetig
310 MOVE 100,250
320 DRAW 200,350
330 DRAW 500,350
340 REM zeichnen tuer (in rot)
350 MOVE 225,50
360 DRAW 225,140,6
370 DRAW 275,140
380 DRAW 275,50
390 REM zeichnen fenster (in gruen)
400 REM links unten
410 MOVE 120,70
```

(Fortsetzung)

```
420 DRAW 120,130,9
430 DRAW 180,130
440 DRAW 180,70
450 DRAW 120,70
460 REM links oben
470 MOVE 120,170
480 DRAW 120,230
490 DRAW 180,230
500 DRAW 180,170
510 DRAW 120,170
520 REM rechts oben
530 MOVE 320,170
540 DRAW 320,230
550 DRAW 380,230
560 DRAW 380,170
570 DRAW 320,170
580 REM rechts unten
590 MOVE 320,70
600 DRAW 320,130
610 DRAW 380,130
620 DRAW 380,70
630 DRAW 320,70
635 STOP
640 REM zaun
650 FOR F=0 TO 620 STEP 20
660 MOVE F,0:DRAW F,60,15
670 NEXT F
680 MOVE 0,45:DRAW 620,45
```

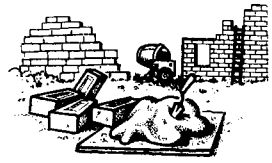
(Fortsetzung)

```
685 STOP
690 REM  fensterkreuz
700 REM  verwendung eines unterprogramms
705 REM  zum zeichnen einen quadrats
710 groesse=18
720 MOVE 130,78:GOSUB 900
730 MOVE 156,78:GOSUB 900
740 MOVE 130,103:GOSUB 900
750 MOVE 156,103:GOSUB 900
760 MOVE 130,178:GOSUB 900
770 MOVE 156,178:GOSUB 900
780 MOVE 130,203:GOSUB 900
790 MOVE 156,203:GOSUB 900
800 MOVE 330,78:GOSUB 900
810 MOVE 356,78:GOSUB 900
820 MOVE 330,103:GOSUB 900
830 MOVE 356,103:GOSUB 900
840 MOVE 330,178:GOSUB 900
850 MOVE 356,178:GOSUB 900
860 MOVE 330,203:GOSUB 900
870 MOVE 356,203:GOSUB 900
880 END
890 REM  unterprogramm fuer quadrat
900 DRAWR 0,groesse,9
910 DRAWR groesse,0
920 DRAWR 0,-groesse
930 DRAWR -groesse,0
940 RETURN
```

Nun werden Sie verstehen, daß wir zuerst die relativen graphischen Befehle besprechen mußten, um mit diesem Teil des Kapitels weitermachen zu können. Wenn wir für eine bestimmte Aufgabe eine Standard-Subroutine verwenden, (in diesem Fall für das Zeichnen der Fensterscheiben), muß diese Aufgabe in einer allgemein gültigen Form beschrieben werden. In unserer 'Fensterscheiben-Subroutine' haben wir durch Verwendung des Befehls DRAWR und der Variablen 'Groesse' die Möglichkeit, ein Quadrat in jeder beliebigen Größe, an jeder gewünschten Stelle des Bildschirms zu zeichnen.

Eine Subroutine sollte immer mit einer Bemerkung (REM) beginnen, in der erklärt wird, was sie macht. Besonders bei einem sehr langen Programm mit vielen verschiedenen Subroutinen ist es von großem Nachteil, wenn keine erklärenden Bemerkungen gemacht worden sind. In Kapitel 9 werden wir sehen, wie wichtig das ist und außerdem erfahren, welche zusätzlichen Informationen erforderlich sind. Das Ende einer Subroutine wird immer durch den Befehl RETURN gekennzeichnet.

Wenn der CPC 464 auf einen GOSUB-Befehl stößt, macht er sich einen Vermerk, bis wohin er das Programm durchlaufen hat, bevor er die Anweisungen der Subroutine ausführt. Der RETURN-Befehl am Ende der Subroutine ist vergleichbar mit der NEXT-Anweisung in einer FOR-Schleife, nur daß man durch NEXT an den Anfang der Schleife zurückspringt und RETURN den CPC 464 zu der nächstfolgenden Anweisung nach dem GOSUB-Befehl bringt, zu dem Punkt also, den er sich gemerkt hat.



Ein Programm kann Subroutinen zum Zeichnen eines Hauses verwenden, genauso wie der Bauunternehmer mit Zulieferfirmen arbeitet, um es zu bauen. Handwerker, wie Maurer, Zimmerleute, Klempner und Fliesenleger werden beauftragt, bestimmte Aufgaben zu übernehmen. Und diese Facharbeiter machen selbst Nebenverträge für einen Teil ihrer Aufgaben. Zum Beispiel wird ein Maurer immer einen Arbeiter beschäftigen, der ihm den Mörtel mischt. Und auch die Zeiten, in denen er selbst Ziegelsteine hergestellt hat, sind längst vorbei. Jedoch kann dieser eine Arbeiter gleichzeitig für mehrere Handwerker tätig sein.

RETURN

Sie können ein Programm in der gleichen Art und Weise aufbauen. Eine Subroutine, die durch GOSUB aufgerufen wurde, kann wiederum eine andere Subroutine aufrufen, die einen Teil der Aufgaben übernimmt, und das setzt sich immer weiter fort. Der CPC 464 kann sich gleichzeitig viele Programmstellen merken, an denen er in Subroutinen gesprungen ist, und findet immer wieder dahin zurück.

Das Programmende

Subroutinen werden immer an den Schluß eines Programms gestellt. Was geschieht aber, nachdem der letzte Befehl ausgeführt wurde? Wenn Sie keine Vorsichtsmaßregeln getroffen haben, wird das Programm unverzüglich zum Anfang der ersten Subroutine zurückkehren, mit manchmal lustigen, meistens jedoch absurden Ergebnissen. Um das zu vermeiden setzt man an den Schluß des Programms einen END-Befehl. Wie einfach das ist, können Sie beim Programm 'Villa' sehen:

END

```
800 END
```

Wenn der CPC 464 diese Zeile erreicht, beendet er den Programmlauf und ist für neue Aufgaben bereit.

Anstatt des END-Befehls kann man aber auch die folgende Schleife einbauen:

```
800 GOTO 800
```

Diese Schleife wird so lange durchlaufen, bis Sie entweder auf die ESC-Taste drücken oder den CPC 464 ganz abschalten und wieder einschalten. Man wendet diese Möglichkeit dann an, wenn man nicht will, daß READY auf dem Bildschirm erscheint.

Übungen

Versuchen Sie mit Hilfe des Fenster-Unterprogramms eine schöne große Panoramascheibe in den rechten Teil des Hauses zu setzen, damit man den Garten gut überblicken kann.

Schreiben Sie eine Subroutine, die einen Schornstein zeichnet, und verwenden Sie sie dann, um den Schornstein irgendwo auf das Dach zu setzen.

Test

Diesmal werden Sie nicht nur auf die Themen dieses Kapitels hin überprüft, sondern wir wollen auch Dinge aus den vorhergehenden Kapiteln wiederholen. Dabei dürfen Sie selbstverständlich im Buch zurückblättern, um eine Frage beantworten zu können. Niemand kann von Ihnen verlangen, daß Sie sich alle Informationen, die Sie in so kurzer Zeit erhalten haben, auf's erste Mal merken können.

9 PROGRAMMERSTELLUNG

Neben dem Wissen über die Anwendung der Befehle erfordert die Programmierung auch ein hohes Maß an Geschicklichkeit. Sie werden sich erinnern, daß wir einmal sagten, ein Computer könne nicht denken. Die Kunst der Programmierung liegt nun darin, ein Problem oder eine Aufgabe in ihre Einzelheiten zu zerlegen und daraus dann ein logisch zusammenhängendes Programm zu formen, das nicht zuviel Speicherkapazität erfordert und verläßlich das erwartete Ergebnis liefert.

Es ist klar, daß dafür Wissen und Erfahrung notwendig sind. Dieses Kapitel vermittelt Ihnen bestimmte Techniken, die für gute Programmierung wesentlich sind und Ihnen in späteren Jahren vielleicht von Nutzen sein werden.

Sachliches Vorgehen bei der Programmerstellung

Für jedes Programm, das länger als 20 Zeilen werden soll, gibt es nur einen Weg, um die Aufgabe zu bewältigen. Sie müssen es in überschaubare Einzelteile aufgliedern, denn es ist nicht möglich, alle Gesichtspunkte eines langen Programms gleichzeitig im Kopf zu behalten. Es spielt dabei keine Rolle, wie gut Sie in der Programmierung sind. Auch Berufsprogrammierer arbeiten in dieser Art und Weise.

Also, bevor wir mit einem Programm beginnen, müssen wir es zuerst in einzelne Teile zerlegen und uns aufschreiben, was in jedem Teil gemacht werden soll. Am besten legen Sie ein 'Projekt-Buch' an, um alle diese Parameter festzuhalten, die für den Aufbau des Programms und seiner Unterprogramme notwendig sind.

Wenn Sie es unterlassen, kann es passieren, daß Sie mitten in einem Programm sind, und plötzlich vergessen haben, warum eine bestimmte Voraussetzung erfüllt werden muß oder was ein Unterprogramm für eine Funktion hat.

Diese allgemeine Programmbeschreibung müssen Sie dann in eine logisch geordnete Folge von Einzelaufgaben bringen. Jetzt werden Sie fragen, wie man das macht. Wir wollen Ihnen von den verschiedenen Methoden, die es gibt, eine näher erklären. Dabei wird eine sogenannte 'Sprache des Programmablaufs' angewendet. Nehmen wir ein bekanntes Beispiel. Wenn ein Postbote seine Runde macht, muß er viele Entscheidungen treffen und bestimmte Handlungen vollführen. Nun stellen Sie sich vor, Sie müßten einen Roboter-Postboten so programmieren, daß er das Gleiche macht. Und da Roboter von Computern gesteuert werden, müssen Sie dem Computer genaue Anweisungen geben, was er tun soll. Sehen Sie sich den folgenden Ablaufplan einmal an:

Programm für einen Roboter-Postboten

```
Gehe zum ersten Haus in der Straße
Für (FOR) jedes Haus in der Straße gilt
    Wenn (IF) du etwas für dieses Haus hast
    Dann wiederhole (THEN):
                Nehme eine Postsendung aus dem Sack
                Bis du nichts mehr für dieses Haus hast
        Wenn (IF) es ein Gartentor gibt
        Dann (THEN) öffne es
                gehe hindurch
                schließe das Tor
        Falls du noch nicht an der Haustür bist
                gehe auf die Tür zu
        Wenn (IF) es ein Einschreiben/Nachporto/Päckchen ist
        Dann (THEN) betätige die Klingel
                Warte auf Antwort
                Für (FOR) jede Postsendung
                    Im Fall:
                            Einschreiben: verlange Unterschrift
                            Nachporto:    verlange Geld
                            Anderenfalls: nicht aushändigen
        Sonst (ELSE) werfe alles in den Briefkasten
        Falls er nicht am Eingang ist
                gehe hinein
        Wenn (IF) es ein Gartentor gibt
        Dann (THEN) öffne das Tor
                gehe hindurch
                schließe das Tor
    Sonst (ELSE) gehe zum nächsten Haus
Nächstes (NEXT) Haus
```

Wir haben einen Programmablaufplan geschrieben, der in einer formalen Art aufgebaut ist.

Anhand des vorhergehenden Beispiels können wir eine Menge wichtiger Dinge feststellen. Als erstes sehen wir, daß das Programm aus mehreren Bausteinen besteht, von denen Sie einige als BASIC-Befehle erkennen werden. Dann wird durch den Text verdeutlicht, welche Handlungsabläufe innerhalb der Schleifen oder der IF-THEN-ELSE-Blöcke stattfinden. Achten Sie darauf ganz besonders, wenn Sie Ihr eigenes Programm schreiben, da Sie sonst schnell durcheinander kommen!

Die einzelnen Programmbausteine werden auch als 'Routinen' bezeichnet. Diesen Ausdruck wollen wir näher erläutern. Eine Reihe von Handlungen (oder Befehlen), die zur Ausführung einer ganz bestimmten Aufgabe notwendig sind, werden zusammengefaßt und bilden eine 'Routine'. Betrachten wir die zweite Zeile in unserem Beispiel:

Für (FOR) jedes Haus in der Straße gilt

Das ist der Beginn einer Routine, die für jedes x-beliebige Haus in jeder x-belieben Straße wiederholt werden könnte, wenn im weiteren Verlauf des Programms der restliche Teil des jeweils unterschiedlichen Problems behandelt werden würde.

Innerhalb einer Routine finden wir mehrere 'Unterroutinen', oder auch 'Subroutinen'. Folgendes Beispiel ist typisch dafür:

Falls du noch nicht an der Haustür bist
    gehe auf die Tür zu

Sie sehen, daß das von einer Subroutine übernommen werden kann, die sich nur damit beschäftigt, daß man von einem Ausgangspunkt zu einem Endpunkt gelangt.

Der wichtigste Punkt bezüglich unseres Beispiels ist der, daß zum gegenwärtigen Zeitpunkt nicht allzuviel Wert darauf gelegt wurde, jede Handlung in ihren Einzelheiten zu beschreiben. Wenn Sie fortfahren würden, dieses Programm in Einzelschritte zu zerlegen, kämen Sie schnell zu der Feststellung, daß allein die genaue Erklärung des Wortes 'gehen' einige Seiten Arbeit erfordern würde. Aber wir wollen es im Moment bei einer allgemeinen Beschreibung belassen.

Wenn alle Aufgaben nacheinander zerlegt worden sind, erreichen wir schließlich einen Punkt, wo jede Handlung im Programm einer einzelnen Handlung unseres Roboter-Sensors oder des Steuermechanismus' entspricht.

Wenn Sie das zuvor Erklärte abschrecken sollte, können wir Sie beruhigen. Ein mittleres Programmiererteam würde mehrere Jahre brauchen, um das Programm schreiben zu können. Aus diesem Grund sollten Sie es lieber von der Liste Ihrer Vorhaben streichen. Wir haben es nur gewählt, um Ihnen zu zeigen, wie man selbst die kompliziertesten Aufgaben einfach beschreiben kann.

Übungen

Bei dem vorhergehenden Beispiel fehlen eine Reihe von Parametern im Programmentwurf. Versuchen Sie die Subroutinen zu entwerfen, die folgende Dinge behandeln:

- Es erfolgt keine Reaktion auf das Klingeln
- Es ist kein Briefkasten vorhanden
- Der Postbote hat keine Hand frei, um das Gartentor öffnen zu können
- In der Straße haben die Häuser keine Hausnummern, sondern nur Namensschilder

Sie können sich aber genausogut auch etwas anderes ausdenken. Seien Sie nicht entmutigt, wenn Sie viele davon nicht in Subroutinen umwandeln können. Dafür schaffen Sie bestimmt wieder andere Dinge spielend leicht.

Programmbausteine

Wir wollen Ihnen nun anhand von Beispielen zeigen, wie man einige Bausteine des Postboten-Programms in Schneider-BASIC umsetzen kann.

Für (FOR) jedes Haus in der Straße gilt
Nächstes (NEXT) Haus

```
800 FOR Haus = Ersteshaus TO Letzteshaus
.....
870 NEXT
```

Wiederhole
Nehme eine Postsendung aus dem Sack
Bis du nichts mehr für dieses Haus hast

```
1000 GOSUB 12000:REM nehme Sendung aus Sack
.....
1110 IF Letztesendung = 0 THEN GOTO 1000
```

Das setzt voraus, daß die Subroutine in Zeile 12000 die Variable 'Letztesendung' setzt.

Falls du noch nicht an der Haustür bist
gehe auf die Tür zu

```
2000 GOSUB 13000:REM feststellen, ob wir an der Tuer sind
2010 IF Antuer <> 0 then 2040
2020 GOSUB 14000:REM auf Tuer zugehen
2030 GOTO 2000
2040 REM Ende von 'falls'
```

Die Subroutine in Zeile 13000 setzt die Variable 'Antuer'.

Im Fall:
Einschreiben: verlange Unterschrift
Nachporto: verlange Geld
Anderenfalls: nicht aushändigen

```
3000 GOSUB 15000:REM stelle Art der Sendung fest
3010 IF Sendungsart = Einschreiben THEN GOSUB 16000:GOTO 3040
3020 IF Sendungsart = Nachporto THEN GOSUB 17000:GOTO 3040
3030 GOSUB 18000:REM nicht aushaendigen
3040 REM Ende von 'im Fall'
```

Die Subroutine in Zeile 15000 setzt die Variable 'Sendungsart' und den Variablen 'Einschreiben' und 'Nachporto' müssen die geeigneten Werte zugewiesen werden.

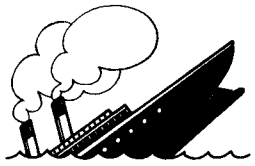
Wenn (IF) es ein Einschreiben/Nachporto/Päckchen ist
Dann (THEN) betätige die Klingel
Sonst (ELSE) werfe alles in den Briefkasten

Hier eine der möglichen Übersetzungen:

```
4000 GOSUB 10000:REM feststellen ob Einschreiben/Nachporto/Päckchen
.....
4070 IF Klingelnoetig <> 0 THEN GOSUB 20000 ELSE GOSUB 21000
```

Wenn Sie wollen, daß aus Gründen der Klarheit weniger mit Subroutinen gearbeitet wird und der Programmteil sich mehr an die IF-THEN-ELSE Befehle hält, könnte Ihre Übersetzung folgendermaßen aussehen:

```
4000 IF Klingelnoetig = 0 THEN GOTO 4100
.....: REM Antwort erhalten usw.
4090 GOTO 4200
4100 REM Klingeln nicht notwendig
.....: REM alles in den Briefkasten werfen
4200 REM Ende von "IF Klingelnoetig"
```



Aufbau von Routinen

Wir haben im folgenden die Punkte aufgelistet, die Sie zuerst ausarbeiten sollten, bevor Sie eine Routine schreiben:

- Bezeichnung der Routine (für Ihren eigenen Gebrauch – nicht für den CPC 464)
- Namen der Variablen, denen ein Wert zugewiesen werden muß, bevor sie in der Routine verwendet werden können
- Auswirkung der Routine auf die Variablen
- Nebenauswirkungen der Routine

Wenn Sie ein Programm erstellen, sollten Sie für jede Routine ein getrenntes Blatt verwenden. Diese können Sie dann später in ein Ringbuch einordnen, so daß Ihr 'Projekt-Buch' entsteht. Den Kopf der einzelnen Blätter sollte eine Kurzinformation über die jeweilige Routine darstellen. (wie oben beschrieben). Das hat zwei Vorteile:

1. Sie sehen auf einen Blick, was Sie zu schreiben beabsichtigen.
2. Sie können leichter erkennen, ob Sie zwei ganz ähnliche Subroutinen schreiben wollten, die auch zu einer einzigen vereint werden könnten, um dadurch die doppelte Ausführung ein und derselben Sache zu vermeiden.

Erinnern Sie sich an die Routine, die die Glasscheiben in die Hausdarstellung zeichnete? Sie begann mit folgender Bemerkung:

REM subroutine zum Quadratzeichnen

Die einzige Variable, der vor dem Aufruf der Subroutine ein Wert zugewiesen wurde, war die Variable 'Groesse', und die Subroutine verwendete sie mehrfach, ohne ihren Wert zu verändern. Somit wirkte sich die Subroutine nicht auf andere Teile des Programms oder auf den CPC 464 direkt aus.

Wenn Sie Routinen auf diese Weise aufbauen, geraten Sie nie in Schwierigkeiten. Sie wissen genau, wie man die Routinen anwendet, was sie machen und welche Auswirkungen sie auf das übrige Programm haben können. Es hilft Ihnen auch, schneller logische Fehler zu finden und Sie vermeiden unerwartete Ergebnisse, die sich durch das ganze Programm ziehen können und es schneller 'untergehen' lassen, als die **Titanic**, jedoch mit dem gleichen Katastropheneffekt.

Wenn Sie ein Programm schreiben, indem Sie nach dieser Methode vorgehen und eine Routine nach der anderen entwerfen, haben Sie am Ende einen ganzen Stoß Papier vor sich liegen, der Ihnen genaue Anweisungen gibt, was Sie tun müssen.

Dokumentation

Als Amateur-Programmierer arbeiten Sie wahrscheinlich ganz alleine. Folglich sind alle Programme, die Sie schreiben, Ihr eigener stolzer Besitz, den Sie weder an andere preisgeben, noch einer kritischen Prüfung unterziehen lassen wollen. Berufsmäßige Programmierer dagegen arbeiten meistens in einem Team an dem gleichen Programm oder Programmsystem und müssen deshalb in der Lage sein, ohne Schwierigkeiten alles lesen zu können, was ein anderer geschrieben hat.

BASIC ist eine einfach anwendbare, tolerante Sprache, die von einem Programm keine starre Struktur oder strengen Formalismus erfordert. Wenn Sie gerade anfangen zu programmieren, ist das alles schön und gut, sobald die Aufgaben jedoch etwas komplizierter werden, ist es erforderlich, die Programme übersichtlich aufzubauen, damit sie leicht zu verstehen sind. Sonst kann es Ihnen passieren, daß Sie ein fertiges Programm noch einmal schreiben müssen, weil Sie nicht mehr wissen, wo Anfang und Ende ist und deshalb keine Möglichkeit haben, es zu ändern.

Das Geheimnis des Erfolgs liegt im Befehl REM und immer wieder REM. Wenn Sie eine Verzweigung oder einen GOSUB-Befehl in Ihrem Programm haben, fügen Sie einfach einen REM ein, der Ihnen sagt, warum und an welche Stelle gesprungen wird.

Wenn Sie zum Beispiel GOSUB 4000 schreiben, ist das akzeptabel, solange Sie nicht 50 andere Subroutinen verwenden. In diesem Fall sollten Sie schreiben:

```
GOSUB 4000:REM Cursor in Position bringen
```

Es soll also möglichst klar erläutert werden, was die Routine machen soll und wozu die darin vorkommenden Variablen verwendet werden. Man macht das am besten, indem man alle zugehörigen Informationen in Form von REM-Befehlen an den Anfang der Routine setzt. Das hat keine besonderen Auswirkungen auf die Laufzeit, erhöht aber die Lesbarkeit ganz erheblich, und Sie selbst oder andere Programmierer können leichter nachvollziehen, was das Programm macht.

10 AKUSTISCHE MÖGLICHKEITEN

Wenn wir eingangs sagten, daß man die BASIC-Programme, die für den CPC 464 geschrieben worden sind, nicht auf anderen Computern laufen lassen kann, bezog sich das hauptsächlich auf die Klang-Befehle, auch Sound-Befehle genannt. Denn es gibt nur wenige andere Computer, die den gleichen Klang-Bereich und die gleiche Lautstärke zur Verfügung haben, wie der CPC 464.

Das bringt jedoch auch ein Problem mit sich. Theoretisch könnte man allein über die Sound-Befehle ein eigenes Buch schreiben. Da Sie jedoch die Grundlagen der Schneider-BASIC erlernen sollen, müssen wir in diesem Kapitel viele Informationen auslassen.

Nachdem Sie das nun wissen, laden Sie das nächste Programm von der Kassette. Es heißt ZAP-POW, und zeigt Ihnen, was Sie mit den Sound-Befehlen alles zustande bringen können. Am Ende dieses Kapitels finden Sie die Auflistung des Programms, damit Sie die interessanten Teile in Ihr eigenes Programm übernehmen können.

Tonerzeugung

Wie Sie wahrscheinlich schon erraten haben, heißt der Befehl, der den Ton erzeugt 'SOUND'. Geben Sie folgendes ein:

```
sound 1,478
```
ENTER

Das war das mittlere 'C'. Die Zahl 478 wird als 'Periode' bezeichnet und bestimmt den Ton, der gespielt werden soll. In der Gebrauchsanweisung des CPC 464 finden Sie eine komplette Auflistung der Noten und der damit verbundenen Perioden. Die 1 steht für den Sound-Kanal, der benutzt werden soll. In Teil 2 dieses Kurses wird dann beschrieben, wie man die beiden anderen Kanäle verwendet.

Geben Sie nun folgendes ein:

```
Sound 1,478,200
```
ENTER

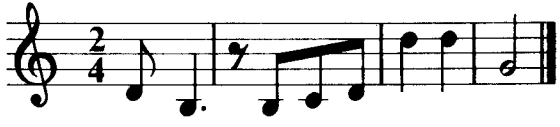
Wenn Sie diesmal die ENTER-Taste drücken, ertönt das 'C' genau 2 Sekunden lang. Das wird durch die Zahl 200 gesteuert, die die 'Dauer' des Tons in 1/100 Sekunden angibt. Wenn Sie diesen Teil des Befehls auslassen, nimmt der CPC 464 an, daß Sie eine Tondauer von 20/100 Sekunde wünschen. Sie können nun ein Programm schreiben, durch das eine bekannte Melodie gespielt wird. Zum Beispiel:

```
10 rem Eine bekannte Melodie
20 sound 1,213
30 sound 1,253,60
40 sound 1,0,40
50 sound 1,253
60 sound 1,239
70 sound 1,213
80 sound 1,127,40
90 sound 1,0,1
100 sound 1,127,40
110 sound 1,159,60
120 end
```

SOUND

Sehen wir uns die Zeilen 40 und 90 einmal näher an. Wenn Sie eine 'Pause' zwischen zwei Noten setzen wollen, müssen Sie einen Befehl einschieben, dessen Periode Null ist. In Zeile 40 haben wir eine Pause für die Dauer von zwei Schlägen, in Zeile 90 eine extrem kurze Pause, die nur verhindern soll, daß die zwei aufeinanderfolgenden gleichen Noten zu einer einzigen Note zusammenschmelzen.

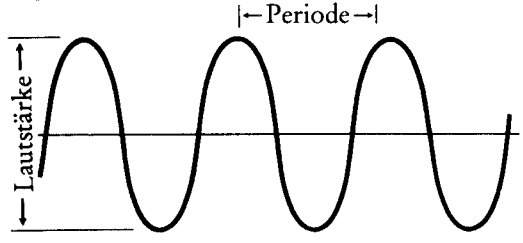
Eine bekannte Melodie



Für diejenigen unter Ihnen, die mit Musik vertraut sind, zeigen wir die gleiche Melodie auch noch in Notenschreibweise, damit Sie sie mit dem Programm vergleichen können.

Tonaufbau in BASIC

Bevor wir weitergehen, wollen wir uns näher damit beschäftigen, wie die Töne überhaupt aufgebaut sind. In seiner einfachsten Form kann ein Ton durch eine Sinus-Kurve graphisch dargestellt werden, wie es die folgende Abbildung zeigt:

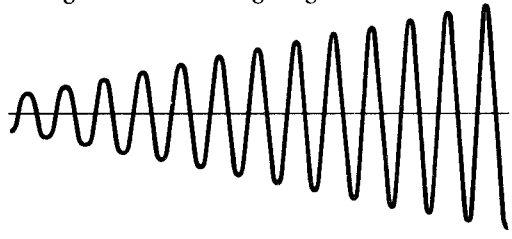


Vielleicht wird Ihnen jetzt klar, was der 'Perioden'-teil des Sound-Befehls eigentlich bedeutet. Er gibt die Zeit an, die zwischen zwei Wellenbergen vergeht, gemessen in Schritten von 8 Mikrosekunden (millionstel einer Sekunde). Je kürzer die Zeitdauer zwischen den Wellenbergen, desto höher der Ton und je länger die Zeitdauer, desto tiefer der Ton.

Die Höhe der Wellen gibt die Lautstärke an, die Sie ebenfalls im Sound-Befehl angeben können. Geben Sie noch einmal die Zeile für das mittlere 'C' ein, und setzen Sie diesmal die Zahl 2 an das Ende des Befehls:

```
Sound 1,478,200,2
```
ENTER

Dadurch wird der Ton sehr leise. Die Lautstärke kann zwischen 0 und 7 geregelt werden, 0 bedeutet, daß man nichts hört (das kann in bestimmten Fällen von Nutzen sein) und 7 steht für maximale Lautstärke. Der Ton, den wir zuerst betrachteten, behält die ganze Zeit die gleiche Lautstärke bei. In der Realität sieht es jedoch anders aus. Normalerweise verändert sich die Lautstärke zwischen Anfang und Ende des Tons, so wie es die folgende Darstellung zeigt:

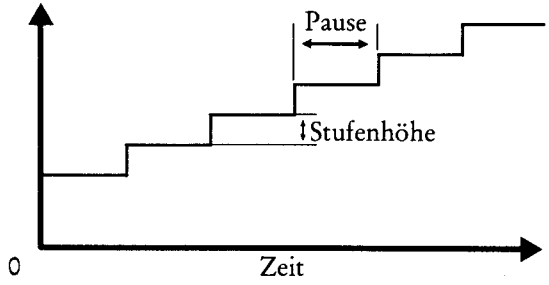


ENV

Beim CPC 464 kann man das mit dem Befehl ENV erreichen, der eine 'Variation' liefert, die angibt, in welcher Form die Lautstärke eines Tons während seiner Dauer verändert werden soll. Geben Sie die folgenden zwei Zeilen ein und lassen Sie sie laufen:

```
10 env 1,6,1,30
30 sound 1,478,180,1,1
```

Wahrscheinlich müssen Sie zuerst das nachstehende Diagramm genau betrachten, bevor Sie verstehen können, wie das Ganze vor sich geht. Stören Sie sich nicht daran, daß die Zeile 20 ausgelassen wurde – wir kommen zu gegebener Zeit darauf zurück.



Sie können in einem Programm bis zu 15 verschiedene 'Variationen' für Lautstärkenveränderung verwenden und die erste 1 in dem vorhergehenden ENV-Befehl sagt lediglich aus, daß es sich um die erste Angabe einer Variation handelt. Die nächste Zahl, 6, bedeutet, daß die Lautstärke sich in sechs gleichen Stufen ändern soll. Die darauffolgende Zahl 1 gibt den Wert an, um den die Lautstärke erhöht werden soll. Wenn wir vor die 1 ein Minuszeichen gesetzt hätten (-1), so würden wir dadurch die Lautstärke vermindern. Die Zahl 30 am Ende des ENV-Befehls bestimmt die Länge der einzelnen Stufen in 1/100 Sekunden.

Seien Sie vorsichtig mit der Lautstärkenangabe im SOUND-Befehl. Wenn eine Variation für die Lautstärke bestimmt wird, kann der CPC 464 mit **sechzehn** verschiedenen Lautstärkestufen arbeiten, daß heißt, der Bereich geht dann von 0 bis 15. Dabei entsprechen 0, 2, 4, 6, 8, 10, 12 und 14 den Stufen 0 bis 7, die zur Verfügung stehen, wenn keine Variation angegeben ist.

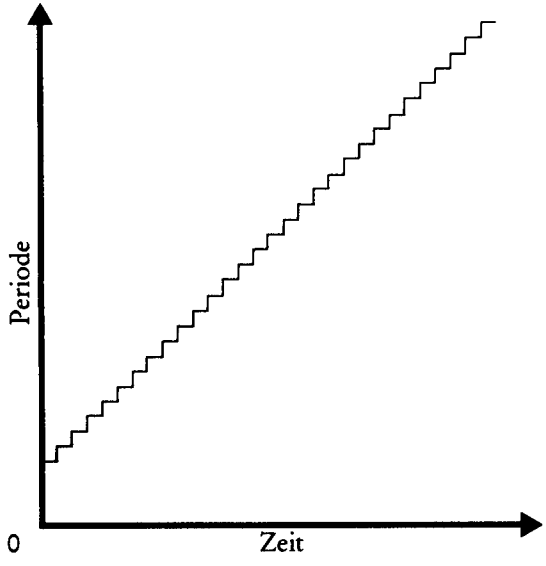
Wir wollen nun untersuchen, wie die Befehle SOUND und ENV in unserem Beispiel zusammenarbeiten. Im SOUND-Befehl ist die Lautstärke auf 1 gesetzt und ENV 1 bestimmt, daß die Lautstärke in sechs Schritten um jeweils eine Stufe erhöht wird, sodaß wir als Ergebnis eine Lautstärke von 1 + 6 = 7 erhalten. Dabei ist wichtig zu wissen, daß der erste Schritt unmittelbar nach Ausführung des SOUND-Befehls stattfindet, das bedeutet, die Ausgangslautstärke ist bereits 1 + 1 = 2. Außerdem dürfen die sechs Schritte der vorgegebenen Variation multipliziert mit der Pause insgesamt nicht länger sein, als die im SOUND-Befehl angegebene Tondauer, da sonst das Ende der Variation verloren geht.

Kommen wir nun auf die fehlende Zeile 20 zurück. In der gleichen Weise, wie die Lautstärkestufen im SOUND-Befehl durch die Lautstärke-Lautstärken-Hullkurve modifiziert werden, kann die Frequenz der Töne durch eine 'Ton-Variation' abgeändert werden. Dafür wird der Befehl ENT verwendet. Setzen wir nun die Zeile 20 zurück an ihren Platz:

```
10 env 1,6,1,30
20 ent 1,180,1,1
30 sound 1,478,180,1,1,1
```

Wie Sie sehen, hat unser SOUND-Befehl noch eine weitere 1 dazubekommen, die anzeigt, daß ENT 1 verwendet werden soll.

Wie Sie aus dem folgenden Diagramm ersehen können, ist der Aufbau des ENT-Befehls dem ENV sehr ähnlich. Jedoch ist es diesmal die Periode des Tons, die in 180 Schritten erhöht wird, wobei jede einzelne Stufe 1/100 Sekunde andauert.



Falls Sie es noch nicht getan haben, lassen Sie das vorhergehende Programm ein paarmal laufen. Ändern Sie dann bei den Befehlen ENV und ENT den Wert, der die Stufenhöhe angibt, von 1 auf -1 und die Ausgangslautstärke beim SOUND-Befehl auf 7. Lassen Sie nun das Programm wieder laufen.

Der Zusatz von Geräuschen

Der CPC 464 kann jedem Ton einen Geräuschanteil beifügen, um ihm damit einen besonderen Reiz zu verleihen.

Wahrscheinlich werden Sie jetzt sagen, daß der SOUND-Befehl durch das Hinzufügen der ENV- und ENT-Nummern inzwischen schon wie ein Weihnachtsbaum aussieht. Das 'Geräusch' ist nun endgültig der letzte Zusatz, der an den SOUND-Befehl angefügt wird.

Ändern Sie Zeile 30 Ihres Programms folgendermaßen:

```
30 sound 1,478,180,1,1,1,5
```

Wenn Sie nun das Programm laufen lassen, werden Sie eine Veränderung feststellen. Probieren Sie auch die übrigen Geräuscharten durch, von denen insgesamt 31 zur Verfügung stehen. Sie bestimmen ein Geräusch, indem Sie die entsprechende Nummer an die dafür vorgesehene Stelle im SOUND-Befehl setzen. Wenn Sie keine Nummer oder die Nummer 0 angeben, bedeutet das, daß kein Geräusch beigemengt wird.

Übungen

Fügen Sie Ton- und Lautstärkevariationen zu der 'bekannten Melodie' hinzu, und ändern Sie sie dann, um verschiedene Klangfarben zu erhalten. Geben Sie dann für einige SOUND-Befehle noch verschiedene Geräuscharten an.

Schreiben Sie nun Ihr eigenes Programm für ein paar Takte einer bekannten Melodie. Selbst wenn Sie nicht allzu viel von Musik verstehen, erreichen Sie durch den Versuch eine ganze Menge und lernen durch Ihre Fehler. Falls Sie ein Musikspezialist sind, sollten Sie trotzdem nicht zuviel Ehrgeiz entwickeln. Ein Kinderlied reicht für den Anfang vollkommen aus.

Zeit für Spiele

Ganz so ist es nun auch wieder nicht. Das nächste Programm auf der Datenkassette heißt ORGEL. Sie können damit nicht nur Töne direkt über die Tastatur erzeugen, sondern haben auch die Möglichkeit, das Programm zu analysieren, um zu sehen, wie das Ganze vor sich geht. Sie sollten inzwischen genug von BASIC verstehen, um aus der Auflistung eines Programms erkennen zu können, was es macht, obwohl dieses spezielle Programm sehr weit forgeschrittene Programmiertechniken enthält. Haben Sie keine Angst davor, etwas abzuändern, um zu sehen, was passiert.

Wie versprochen folgt nun die Auflistung des Programms ZAPPOW:

```
10 REM Zappow
30 MODE 1
35 INK 0,1:INK 1,25
40 LOCATE 13,4:PRINT "TON-DEMO"
50 LOCATE 10,7:PRINT"1. Explosion"
60 LOCATE 10,8:PRINT"2. Hundbellen"
70 LOCATE 10,9:PRINT"3. Sirene"
80 LOCATE 10,10:PRINT"4. Wasserspuelung"
90 LOCATE 10,11:PRINT"5. Kuckuck"
100 LOCATE 10,12:PRINT"6. Maschinengewehr"
110 LOCATE 10,13:PRINT"7. Weltraumangreifer"
120 LOCATE 4,17:PRINT"Waehlen Sie ein Geraeusch (1 - 7)"
130 IF INKEY$>""THEN 130
140 a$=INKEY$:IF a$="" THEN 140
150 IF a$<"1" OR a$>"7" THEN 140
160 a=VAL(a$)
170 LOCATE 20,19:PRINT a$
180 IF a=1 THEN GOSUB 280
190 IF a=2 THEN GOSUB 330
200 IF a=3 THEN GOSUB 380
210 IF a=4 THEN GOSUB 430
220 IF a=5 THEN GOSUB 480
230 IF a=6 THEN GOSUB 530
240 IF a=7 THEN GOSUB 580
250 FOR j=0 TO 1000:NEXT
260 LOCATE 20,19:PRINT "  "
270 GOTO 130
```

(Fortsetzung)

```
280 REM Explosion
290 ENV 1,11,-1,25
300 ENT 1,9,49,5,9,-10,15
310 SOUND 1,145,255,0,1,1,12
320 RETURN
330 REM hundebellen
340 ENV 1,4,7,10
350 ENT 1,7,-8,3,6,24,2
360 SOUND 1,120,33,8,1,1,3
370 RETURN
380 REM Sirene
390 ENV 1,2,9,45
400 ENT 1,2,9,45
410 SOUND 1,150,90,6,1,1
420 RETURN
430 REM wasserspuelung
440 ENV 1,3,-2,85
450 ENT 1,5,-1,51
460 SOUND 1,150,254,11,1,1,8
470 RETURN
480 REM kuckuck
490 ENV 1,4,12,11
500 ENT 1,5,12,8
510 SOUND 1,165,40,13,1,1
520 RETURN
530 REM Maschinengewehr
540 ENV 1,21,-5,4
550 ENT 1
```

(Fortsetzung)

```
560 SOUND 1,162,82,15,1,1,11
570 RETURN
580 REM weltraumangreifer
590 ENV 1,4,30,19
600 ENT 1,9,49,5,1,-10,26
610 SOUND 1,136,68,15,1,1,0
620 RETURN
```

Test

Es wäre sehr überraschend, wenn Sie nicht in diesem Kapitel zurückblättern müßten, um die Fragen in SAT10 beantworten zu können. Lassen Sie sich dadurch nicht verunsichern. Wenn Sie schon alles über die SOUND-Befehle des CPC 464 wüßten, würden Sie dieses Buch ja garnicht lesen.

11 VERARBEITUNG VON ZAHLEN

Wie wir schon öfters erwähnten, eignet sich der CPC 464 genau wie jeder andere Computer hervorragend für Aufgaben, die ständig wiederholt werden müssen. Das trifft vor allem für die Verarbeitung von Zahlen zu, ja, man könnte sogar sagen, daß Computer hauptsächlich darauf spezialisiert sind. Alles, was wir bisher gesehen haben wurde vom CPC 464 durch die interne Verarbeitung riesiger Zahlenmengen in unglaublich kurzer Zeit zustande gebracht. Auch die graphischen und tonerzeugenden Funktionen laufen in dieser Art und Weise ab.

Wenn Sie nicht besonders gut in Mathematik sind, wird dieses Kapitel nicht ganz leicht für Sie sein. Aber versuchen Sie es auf alle Fälle. Die einfache Arithmetik ist nicht allzu schwer und ihre Darstellung in BASIC ist eigentlich recht unkompliziert. Der wirklich schwierige Stoff der Arithmetik wird erst in Teil 2 des Kurses behandelt.

Arithmetik in BASIC

Der CPC 464 verwendet beim Rechnen die vier arithmetischen Grundfunktionen, jedoch haben zwei davon ein anderes Zeichen:

Funktion	Bedeutung	BASIC
+	addieren	+
-	subtrahieren	-
x	multiplizieren	★
÷	dividieren	/

Aus diesem Grund sehen Rechenaufgaben, die in BASIC geschrieben sind, etwas anders aus, als Sie es gewöhnt sind.

In Kapitel 4 machten wir folgende Eingabe:

```
print 2+2
```

Das gleiche können Sie auch mit den folgenden Rechenaufgaben machen, wenn Sie sich an die vorhergehende Liste der Funktionen halten und die angegebenen BASIC-Symbole verwenden:

```
3 + 7          17 - 8
15 x 4         15 ÷ 3
8 ÷ 2          20 x 17
```

Um kompliziertere Rechenaufgaben lösen zu können, müssen wir dem CPC 464 mitteilen, daß bestimmte Rechenoperationen vor anderen ausgeführt werden sollen. Das folgende Beispiel ist nach arithmetischen Regeln ganz klar:

$$\frac{15 \times 7}{7 - 2}$$

In BASIC dürfen Sie jedoch **nicht** schreiben:

```
15*7/7-2  [= 13 Falsch!]
```

Wir müssen dem CPC 464 angeben, daß die Multiplikation 15 * 7 und die Subtraktion 7 - 2 vor der Division ausgeführt werden müssen. Um das zu erreichen, müssen Klammern gesetzt werden:

```
(15*7)/(7-2)  [= 21 richtig!]
```

Jetzt müßten Sie eigentlich in der Lage sein, die folgenden Rechenaufgaben in BASIC umsetzen zu können:

$$\frac{30}{3 + 2} \qquad \frac{15 \times 6}{9}$$

$$24 - (4 \times 3) \qquad 5 \times (2 + 8)$$

Die gleichen Regeln gelten auch für Variablen. Geben Sie das folgende Programm ein und verändern Sie dann die Zeile 40 so oft, bis Sie alle nachstehenden Rechenaufgaben gelöst haben. Vergessen Sie nicht, die zulässigen BASIC-Symbole zu verwenden und an den richtigen Stellen Klammern zu setzen!

```
10 a=2
20 b=5
30 c=10
40 print a+b+c
```

$$\frac{a \times a}{c - (a \times b)} \qquad \frac{b \times c}{b - a}$$

$$\frac{c}{b - a} \qquad \frac{c - b}{c + b}$$

Die beiden ersten Rechenaufgaben waren recht einfach. Was halten Sie jedoch von dem Ergebnis der anderen drei? Sie haben dadurch einen Einblick in die Arbeitsweise des CPC 464 erhalten. Wenn Sie bei den letzten drei Ergebnissen die Stellen vor und nach dem Dezimalpunkt zusammenzählen, stellen Sie fest, daß es immer neun sind. Das ist die normale Darstellung von Zahlen auf dem Bildschirm, die vom CPC 464 verwendet wird. Der Haken dabei ist nur, daß bei einer Reihe von Berechnungen möglicherweise ein Ergebnis entsteht, das etwas ungenau ist. Statt 5.0 erhalten Sie vielleicht:

```
5.00000001
```

oder:

```
4.99999999
```

Raffiniert, nicht wahr? Glücklicherweise benötigt man diese große Genauigkeit nicht allzu oft.

ROUND

Manchmal wollen wir jedoch ganze Zahlen haben, da zum Beispiel die Differenz zwischen 276.25 und 276 nicht sehr viel aussagt. Das Schlüsselwort, mit dessen Hilfe wir als Ergebnis eine ganze Zahl erhalten, heißt ROUND. Ein typischer Befehl wäre zum Beispiel:

```
x = round(26 * 17)/1.6
```

Probieren Sie es aus. Wenn der Wert hinter dem Dezimalpunkt kleiner als 0.5 ist, wird das Ergebnis abgerundet, ist der Wert aber gleich oder größer 0.5, wird auf die nächste ganze Zahl aufgerundet.

Wenn der CPC 464 die Zahlen auf 9 Stellen genau über den Bildschirm ausgibt, hat er meistens noch weitere Stellen in seinem Speicher stehen. Sie können zum Beispiel zu folgendem Wert gelangen:

```
5.000000001 (zehn Stellen)
```

Der CPC 464 wird immer behaupten, daß es 5.0 ist, ganz gleich, wie oft Sie ihn danach fragen! Wenn jedoch der nächste Befehl folgendermaßen lautet:

```
IF a = 5 THEN GOTO...
```

dann wir der GOTO nicht ausgeführt.

Um solche Probleme erst garnicht entstehen zu lassen, wird eine IF-Anweisung in dieser Form vermieden und dafür folgendes geschrieben:

```
Wert = 0,0000001
IF ABS(a-5) < Wert THEN GOTO...
```

Machen Sie sich weiter keine Gedanken über das Schlüsselwort ABS. Es gibt nur die Differenz zwischen 'a' und 5 an, wie wir dann in Teil 2 sehen werden.

Das nächste Programm auf der Datenkassette A ist eine gute Multiplikationsübung, sowie ein Beispiel für die Anwendung einfacher Mathematik auf dem CPC 464. Lassen Sie sich die Tabelle der Multiplikationsergebnisse von 13.87 ausgeben! Nach dem Programmlauf sollten Sie sich das Programm auf dem Bildschirm auflisten lassen, um zu sehen, wie es aufgebaut ist.

Elementare Logik

In früheren Kapiteln haben wir oft Dinge erwähnt, die dann nicht näher erklärt wurden. Der Grund dafür ist ganz einfach. Zu diesen Zeitpunkten hätte eine Erklärung Sie eher durcheinandergebracht als Ihnen geholfen. Eines dieser Dinge ist die Verwendung von 'logischen Operatoren'.

Erinnern Sie sich zurück an die Besprechung der Programmschleifen. Wir verwendeten dabei Zeichen, wie '<' und '>'. Diese Symbole werden als 'logische Operatoren' bezeichnet, und es gibt davon eine ganze Liste.

```
<   kleiner als
>   größer als
=   gleich
<>  ungleich
<=  kleiner oder gleich
>=  größer oder gleich
```

Auch wenn diese Symbole ungewohnt für Sie sein sollten, werden Sie ganz klar erkennen, was damit gemeint ist. Wenn Sie zum Beispiel zwei Variablen vergleichen müssen um festzustellen, ob eine Schleife beendet oder zwischen zwei Wegen entschieden werden soll, können Sie das mit Hilfe dieser 'logischen Operatoren' bewerkstelligen. Der IF-Befehl, über den wir in Kapitel 7 sprachen, ist vollständig von diesen Bedingungen abhängig.

Zusätzlich sei noch zu bemerken, daß sich die logischen Operatoren danach richten, ob es sich um eine negative oder um eine positive Zahl handelt. Wenn a = 5 und b = 3:

```
a>b
```

Wenn aber a = -5 und b = 3:

```
a<b
```

Logische Textprüfung

Als wir begannen, sagten wir, daß der CPC 464 nur mit Zahlen arbeiten kann. Nun werden Sie sicher fragen, wie er dann Schriftzeichen aus den Textvariablen speichern und verarbeiten kann, wie wir es in Kapitel 6 lernten?

Die Antwort darauf ist, daß jedes Schriftzeichen durch eine Nummer gekennzeichnet ist, den sogenannten 'Schriftzeichencode'. Dadurch können die Schriftzeichen genauso wie eine Reihe von Zahlen behandelt werden. Auch können auf diese Weise Wörter durch die logischen Operatoren auf ihre alphabetische Reihenfolge hin überprüft werden. Dabei ist aufgrund der numerischen Werte des Schriftzeichencodes 'A' kleiner als 'Z'. Wenn wir das an folgendem Befehl ausprobieren,

```
IF "Apfel"<"Orange" THEN PRINT "Zitrone"
```

erhalten wir als Antwort 'Zitrone', da 'Apfel' im Alphabeth vor 'Orange' kommt. Großbuchstaben sind kleiner als Kleinbuchstaben, und eine kurze Schriftzeichenfolge ist kleiner als eine lange, wenn die Anfangsbuchstaben gleich sind. Zum Beispiel:

```
"Anfangswert"<"Awert"      (richtig!)
"Text"<"Textfolge"         (richtig!)
"variable"<"VARIABLE"      (falsch!)
"Test">"Testergebnis"      (falsch!)
"rechnen">"Rechnung"       (richtig!)
```

Der CPC 464 prüft jeden Buchstaben eines Wortes auf seinen numerischen Wert hin und vergleicht ihn mit dem numerischen Wert desjenigen Buchstabens, der in dem Wort rechts vom Bedingungssymbol an der gleichen Stelle steht.

Haus und Garten

Unser Haus hat inzwischen Gestalt angenommen. Es besitzt an der einen Seite eine Panoramascheibe, einen Schornstein und einen Zaun, um den Hund des Nachbarn abzuhalten.

Da es sich jedoch beim CPC 464 um eine Maschine zur praktischen Anwendung handelt, benutzt man ihn nicht zur Gestaltung einer attraktiven Blumenumrandung. Dagegen ist er für die Planung des Gemüsebeets sehr gut einsetzbar. Laden Sie das nächste Programm von der Datenkassette A. Es heißt 'GARTEN'.

Der Garten ist 6 Meter lang und 4 Meter breit und wir wollen in Reihen von 4 Metern anpflanzen. Verschiedene Gemüsearten müssen in unterschiedlich breiten Reihen gesät werden, und der Ertrag pro Quadratmeter ist bei jeder Sorte verschieden. Wenn Sie das Programm laufen lassen, wird es Sie fragen, wie viele Reihen Sie von jeder Gemüsesorte anlegen wollen und Ihnen dann mitteilen, ob noch freier Platz für weitere Reihen vorhanden ist.

Vorausgesetzt, daß der Boden gut ist und es wettermäßig ein normales Jahr wird, erhalten Sie dann eine Übersicht, die Ihnen den zu erwartenden Ertrag pro Gemüsesorte in kg angibt. Nachstehend finden Sie die Auflistung des Programms GARTEN. Die darin enthaltene Mathematik ist recht einfach und hat folgende Tabelle zur Grundlage:

Gemüse	Reihenbreite (Meter)	Ertrag pro Reihe (kg)
Zwiebeln	0.30	9
Karotten	0.30	3
Kartoffeln	1.0	50
Kohl	0.60	8
Bohnen	1.0	30
Pastinaken	0.50	11

Falls Sie keine Pastinaken mögen, können Sie das Programm abändern, und dafür etwas anderes einsetzen.

```
10 REM Garten
20 MODE 1
30 INK 0,0:BORDER 0
40 INK 1,26
50 laenge=6
60 CLS
```

(Fortsetzung)

```
70 PRINT "GARTEN"
80 PRINT"Verbleibende Laenge : ";laenge; "Meter"
90 PRINT"Welche Gemuesesorten wollen Sie anbauen"
100 PRINT
110 PRINT"                  Breite        Ertrag       Reihen"
120 PRINT"1. Zwiebeln        0.3m           9Kg        ";zwiebeln
130 PRINT"2. Karotten        0.3m           3Kg        ";karotten
140 PRINT"3. Kartoffeln        1m          50Kg        ";kartoffeln
150 PRINT"4. Kohl            0.6m           8Kg        ";kohl
160 PRINT"5. Bohnen            1m          30Kg        ";bohnen
170 PRINT"6. Pastinaken      0.5m           1Kg        ";pastinaken
180 PRINT
190 PRINT "Zahl zwischen 1 und 6 eingeben"
200 PRINT"oder 7, fuer die Gesamtproduktion"
210 INPUT gemuese
220 IF gemuese= <OR gemuese    >7 THEN GOTO 190
230 IF gemuese =7 THEN GOTO 320
240 IF gemuese =1 THEN GOSUB 470
250 IF gemuese =2 THEN GOSUB 540
260 IF gemuese =3 THEN GOSUB 610
270 IF gemuese =4 THEN GOSUB 680
280 IF gemuese =5 THEN GOSUB 750
290 IF gemuese =6 THEN GOSUB 820
300 PRINT "-------------------------------------"
310 GOTO 60
320 REM Uebersicht
340 PRINT "UEBERSICHT"
350 PRINT "ERTRAG DES GARTENS IN KILO"
```

(Fortsetzung)

```
370 PRINT
380 PRINT"Zwiebeln   :";zwiebeln ★9;"Kg"
390 PRINT"Karotten   :";karotten ★3;"Kg"
400 PRINT"Kartoffeln :";kartoffeln ★50;"Kg"
410 PRINT"Kohl       :";kohl ★8;"Kg"
420 PRINT"Bohnen     :";bohnen ★30;"Kg"
430 PRINT"Pastinaken:";pastinaken ★11 ;"Kg"
450 GOTO 450
460 :
470 REM Zwiebeln
480 PRINT"Zwiebeln"
490 reihenbreite = 0.3
500 GOSUB 890
510 zwiebein = reihen
520 RETURN
530 :
540 REM Karotten
550 PRINT"Karotten"
560 reihenbreite = 0.3
570 GOSUB 890
580 karotten = reihen
590 RETURN
600 :
610 REM Kartoffeln
620 PRINT"Kartoffeln"
630 reihenbreite = 1
640 GOSUB 890
650 kartoffeln = reihen
```

(Fortsetzung)

```
660 RETURN
670 :
680 REM Kohl
690 PRINT"Kohl"
700 reihenbreite = 0.6
710 GOSUB 890
720 kohl = reihen
730 RETURN
740 :
750 REM Bohnen
760 PRINT"Bohnen"
770 reihenbreite = 1
780 GOSUB 890
790 bohnen = reihen
800 RETURN
810 :
820 REM Pastinaken
830 PRINT"Pastinaken"
840 reihenbreite = 0.5
850 GOSUB 890
860 pastinaken = reihen
870 RETURN
880 :
890 REM Einzelheiten
900 INPUT "In wievielen Reihen wollen Sie anbauen";reihen
910 testlaenge = laenge-reihen * reihenbreite
920 IF testlaenge < 0 THEN PRINT"Kein Platz":GOTO 900
930 laenge =testlaenge
950 RETURN
```

Test

Es ist sehr wichtig, daß Sie SAT11 laufen lassen, um dadurch ein Gefühl für den Umgang mit der mathematischen Arbeitsweise des CPC 464 zu bekommen. Da die meisten Programme einen Teil der elementaren Mathematik verwenden, sollten Sie sich die Mühe machen und soviel Zeit wie möglich aufwenden, um wirklich zu verstehen, wie der CPC 464 auf diesem Gebiet arbeitet. Vergessen Sie nicht, daß die meisten Programme für Telespiele, die Sie für den CPC 464 kaufen können, auf mathematischen Grundlagen beruhen, die sehr schnell durchlaufen werden.

12 TELESPIELE

RND

Bei den meisten Spielen, nicht nur bei Telespielen, messen Sie Ihre körperlichen und geistigen Fähigkeiten an einem Gegenspieler, an einem zufälligen Ereignis oder an der Zeit. Manchmal müssen Sie sogar gegen alle drei spielen. Mit den Telespielen oder einem Schachcomputer können Sie sich viele Stunden in dieser Weise beschäftigen, ohne daß Sie etwas davon verstehen müssen, wie man die Programme für diese Spiele schreibt.

Wenn Sie selbst Spiele entwerfen wollen, müssen wir Ihnen einen guten Rat geben. Nehmen Sie sich Ihr Mathematikbuch noch einmal vor. Sie können kein Programm schreiben über einen Ball der von einer Wand abprallt, wenn Sie sich nicht an die Formel zur Berechnung des Abprallwinkels, der Abprallgeschwindigkeit und der resultierenden Flugbahn erinnern. Jedoch wollen wir in diesem Abschnitt des Kurses nicht bis zu diesem hohen Niveau der Einzelheiten vordringen. Sie sollen lediglich erkennen, daß die zwei Spiele, die in diesem Kapitel beschrieben werden, nur einen geringen Teil von dem enthalten, was der CPC 464 wirklich kann.

Zufallsergebnisse

Bei dem Spiel BOMBER in Kapitel 4 erschien das außerirdische Raumschiff immer an verschiedenen Stellen auf dem Bombenzielgerät. Das wurde mit der Funktion RND erreicht, durch die ein Zufallselement in das Programm eingebracht wird. Im folgenden Beispiel wird gezeigt, wie RND angewendet werden kann:

```
move rnd*639,rnd*399
```

Durch RND erhalten Sie eine Dezimalzahl zwischen 0 und 1. Diese müssen Sie dann mit der größten Zahl, die Sie in dem Befehl oder in der Routine erwarten, multiplizieren. Im oben dargestellten Fall wird der graphische Cursor in eine durch den Zufall bestimmte Position auf dem Bildschirm gesetzt, da wir den maximalen Wert der x- und y-Koordinaten mit RND multipliziert haben.

Versuchen Sie ein kurzes Programm zu schreiben, das ein kleines Quadrat in eine durch den Zufall bestimmte Lage auf dem Bildschirm bringt.

Die Zeit läuft ab

Eine andere nützliche Funktion des CPC 464 ist TIME, und das nicht nur für Spiele. Ab dem Moment des Einschaltens oder des Neubeginns zählt der CPC 464 die ablaufende Zeit in 3/100 Sekunden mit und speichert sie in TIME ab. Dieses ständige Mitzählen der Zeit wird nur beim Laden des Programms und beim Abspeichern auf einer Datenkassette unterbrochen. Das folgende Programm ist ein Beispiel dafür, wie man es verwenden kann:

```
10 print "irgendeine Taste druecken"
20 if inkey$ = "" goto 20
30 for t = 1 to rnd * 5000:next
40 a = time
50 print "wiederholen"
60 if inkey$ = "" goto 60
70 b = time
80 print "Reaktionszeit = ";(b-a)/300;" Sekunden"
90 end
```

In diesem Beispiel wird der Wert von TIME vor und nach dem Erscheinen des Wortes 'wiederholen' genommen, und die Reaktionszeit durch die Differenz der beiden Werte ermittelt.

In Zeile 20 und 60 finden Sie ein neues Schlüsselwort, INKEY$. Es ist vergleichbar mit INPUT, nur daß es nur ein Schriftzeichen überträgt und nicht von ENTER gefolgt werden muß. Wir verwenden es in dem Programmbeispiel, um festzustellen, ob nach der Aufforderung 'irgendeine Taste druecken' auf eine Taste außer SHIFT, CTRL, CAPS LOCK und ESC gedrückt wurde.

TIME

INKEY$

BLACKJACK

Das Kartenspielen macht mit dem CPC 464 viel mehr Spaß, als wenn man für sich allein eine Patience legt. Das nächste Programm auf der Datenkassette A heißt BLACKJACK. Falls Sie das Spiel nicht kennen sollten, hier eine kurze Erklärung: Sie verlangen so lange nach einer neuen Karte, bis der Gesamtwert der Karten 21 ist, oder sehr knapp unter dieser Zahl liegt. Bei mehr als 21 haben Sie verloren. Wenn das nicht der Fall ist, 'gibt' sich der CPC 464 selbst Karten, und versucht näher als Sie an 21 heranzukommen, ohne den Wert zu überschreiten.

Weitere Regeln:

- 5 Karten, die zusammenaddiert 21 oder weniger ergeben, gelten mehr als alles andere
- As zählt 1 oder 11
- Bube, Dame, König zählt 10

```
10 REM ★★ BLACKJACK ★★
20 REM
30 REM ★★   BEGINN   ★★
40 MODE 1:BORDER 4
50 INK 0,17:INK 1,0
60 LOCATE 16,5:PRINT "BLACKJACK"
70 LOCATE  3,12
80 PRINT" Starten beliebige Taste druecken
90  folge$ = CHR$(226) + CHR$(227) + CHR$(228) + CHR$(229)
100 karte$ = "A23456789ZBDK
110 K5$ = "5 Karten Trick  -  Ich gewinne"
120  meinas = 0:deinas = 0
130  meinspiel = 0:deinspiel = 0
140 WHILE INKEY$ = "":WEND
```

(Fortsetzung)

```
150 CLS
160 REM ★★ DU BIST DRAN ★★
170 deinekarten = 0:deinas = 0
180 deinehand = 0
190 LOCATE 14,1
200 PRINT" Meine Spiel:";Meinespiel;
210 PRINT  "Deine:";deinespiel
220 y = 20:x = 5
230 deinekarten = 0
240 GOSUB 770
250  deinekarten = deinekarten + 1
260  IF wert = 1 THEN deinas = deinas + 1
270  deinehand = deinehand + wert
280 GOSUB 830:x = x + 5
290 IF deinehand >21 THEN GOTO 690
300 GOSUB 930
310 einashand = deinehand
320 IF  deinas > = 1 THEN einashand = deinehand + 10
350 IF  deinekarten = 5 THEN 440
360 IF  deinehand = 21 THEN 440
370 IF einashand = 21 THEN 420
380 IF  deinas = 0 AND deinehand< = 11 THEN 240
390 IF deinekarten = 1 THEN 240
400 INPUT"Noch eine Karte (J/N)";q$
410 IF UPPER$(q$) = "J" THEN GOTO 240
420 IF einashand< = 21 THEN deinehand = einashand
440 REM ★★ ICH BIN DRAN ★★
450 y = 10:x = 5
```

(Fortsetzung)

```
460 meinehand =0: meinekarten = 0: meinas =0
470 GOSUB 770
480 meinekarten = meinekarten + 1
490 IF wert = 1 THEN meinas = meinas + 1
500 meinehand = meinehand + wert
510 GOSUB 830:x = x + 5
520 FOR warten = 0 TO 1000:NEXT
530 IF  meinhand >21 GOTO 720
540 IF meinekarten =5 THEN GOSUB 930:PRINT k5$: GOTO 710
550 IF  deinekarten = 5 THEN 470
560 meinA = meinehand
570 IF mein as> = 1 AND meinehand < 12 THEN meinA = meinehand+ 10
600 IF meinehand > = deinehand THEN 640
610 IF meinA > = deinehand THEN meinehand = meinA:GOTO 640
630 GOTO 470
640 REM ★★ ENDERGEBNIS VERGLEICHEN ★★
650 GOSUB 930
660 PRINT"Ich habe";meinehand;
670 PRINT"und Du hast";deinehand
680 IF meinehand< deinehand GOTO 730 ELSE GOTO 700
690 GOSUB 930:PRINT"Du hast verloren!"
700 PRINT"Ich habe gewonnen"
710 meinspiel = meinspiel+ 1:GOTO 140
720 GOSUB 930:PRINT"Ich habe verloren!"
730 PRINT"Du hast gewonnen"
740 deinespiel = dienespiel + 1:GOTO 140
750 END
760 REM ★★ KARTEN MISCHEN ★★
```

(Fortsetzung)

```
770 karte = INT(RND★13)+1
780 folge = INT(RND★4)+1
790  wert = karte
800 IF wert >10 THEN wert = 10
810 RETURN
820 REM ★ ★KARTE AUSDRUCKEN★ ★
830 LOCATE x,y
840 PRINT CHR$(24);"        ";CHR$(24)
850 LOCATE x,y + 1
860 PRINT CHR$(24);"  ";
870 PRINT MID$(karte$,karte,1);
880 PRINT MID$(folge$,folge,1);
890 PRINT"  ";CHR$(24)
900 LOCATE x,y + 2
910 PRINT CHR$(24);"        ";CHR$(24)
920 RETURN
930 LOCATE 1,24:PRINT SPACE$(40)
940 LOCATE 1,24:RETURN
```

Simple Simon

In der nachfolgenden Programmauflistung können Sie sehen, daß es nicht ganz einfach ist! Wieder werden Sie einige Befehle finden, die erst in Teil 2 dieses Kurses erklärt werden. Jedoch wollen wir Ihnen eines der Schlüsselwörter erklären; es ist CHR$.

CHR$

In Kapitel 3 wurde beschrieben, daß zusätzliche Zeichen, die nicht auf den Tasten aufgedruckt sind, über den Bildschirm ausgegeben werden können. Mit dem Schlüsselwort CHR$ können Sie diese Zeichen durch ihren Zeichen-Code aufrufen. Wenn Sie zum Beispiel ein kleines Raumschiff auf dem Bildschirm erscheinen lassen wollen, würde der Befehl folgendermaßen lauten:

```
print chr$(239)
```

Das Gesamtverzeichnis der CPC 464-Zeichen finden Sie in der **Schneider CPC 464-Anwendungsbeschreibung**

```
10 cr$ = CHR$(13)
20 REM Simon
30 REM ★★★★ ANWEISUNGEN ★★★★
40 MODE 1:BORDER 20:INK 0,20:INK 1,1
50 LOCATE 16,2:PRINT CHR$(24);"Simon";CHR$(24)
60 PRINT:PRINT
70 PRINT" In diesem Spiel sollen Sie die"
80 PRINT"aufleuchtenden Kreise beobachten und"
90 PRINT"sich ihre Positionen merken. Ist die"
100 PRINT "Folge zu Ende, wiederholen Sie sie mit"
110 PRINT "den Cursortasten. Nach jeder richtigen"
120 PRINT "Antwort verlaengert sich die Folge um"
130 PRINT "einen weiteren Kreis.":PRINT
140 PRINT "Ein Kreis oben auf dem Schirm wird mit"
150 PRINT "der ↑ - Cursortasken eingegeben. Die"
160 PRINT "Cursortasten sind ueber dem numerischen"
170 PRINT "Tastenblock und sehen so aus:":PRINT
```

(Fortsetzung)

```
180 PRINT TAB(20);CHR$(240)
190 PRINT TAB(19);CHR$(242);" ";CHR$(243)
200 PRINT TAB(20);CHR$(241)
210 LOCATE 7,22:PRINT "Zum Weitermachen [ENTER]"
220 LOCATE 6,24:PRINT"Es folgt eine kurze Pause!"
230 WHILE INKEY$ < > cr$:WEND
240 REM ★★★★ BETRIEBSPARAMETER ★★★★
250 MODE 0
260 WINDOW 7,15,10,16
270 b = 17:f = 3:REM Hintergrund/Vordergrund
280 BORDER b
290 INK 0,17
300 FOR i = 1 TO 15:INK i,b:NEXT
310 x = 320:y = 70:c = 2:GOSUB 940
320 y = 330:c = 1:GOSUB 940
330 x = 120:y = 200:c = 3:GOSUB 940
340 x = 520:c = 4:GOSUB 940
350 INK 5,f:PEN 5
360 RANDOMIZE TIME
370 a$ = ""
380 REM ★★★★ ANZEIGEFOLGE ★★★★
390 a$ = a$ + CHR$(RND★3 + 1)
400 FOR i = 1 TO LEN(a$)
410 FOR j = 1 TO 200:NEXT
420 x = ASC(MID$(a$,i,1))
430 INK x,2★x + 1
440 SOUND 1,10 + x★100
450 FOR j = 1 TO 200:NEXT
```

(Fortsetzung)

```
460 INK x,b
470 NEXT
480 FOR i=1 TO 100:NEXT
490 REM ★★★★ ANTWORT AUSWERTEN ★★★★
500 FOR i=1 TO LEN(a$)
510 WHILE k$>"":k$=INKEY$:WEND
520 FOR L=1 TO 2000:k$=INKEY$
530 IF k$>"" THEN 560
540 NEXT
550 k$=" "
560 k=ASC(k$)-239:IF k<1 OR k>4 THEN 520
570 x=ASC(MID$(a$,i,1))
580 IF k<>x GOTO 730:REM falsch
590 INK x,2★x+1
600 SOUND 1,10+x★100
610 FOR j=1 TO 80:NEXT
620 INK x,b
630 For j=1 TO 20:NEXT
640 NEXT
650 REM ★★★★ RICHTIG! ★★★★
660 CLS:PRINT" RICHTIG!
670 PRINT:PRINT:PRINT " PUNKTE:"
680 PRINT:PRINT"  ";LEN(A$)
690 FOR j=1 TO 600:NEXT
700 LOCATE 1,1:PRINT"          "
710 GOTO 390
720 REM ★★★★ FALSCH ★★★★
730 SOUND 1,2000
```

(Fortsetzung)

```
740 CLS:PRINT"  Falsch"
750 FOR j=1 TO 300:NEXT
760 PRINT:PRINT " Folge     war:
770 FOR i=1 TO LEN(a$)
780 x=ASC(MID$(a$,i,1))
790 INK x,2*x+1
800 SOUND 1,10+x*100
810 FOR j=1 TO 200:NEXT
820 INK x,b
830 FOR j=1 TO 200:NEXT
840 NEXT
850 REM **** ENDE &  NEUSTART  ****
860 CLS
870 PRINT" Anzahl"
880 PRINT" Spiele"
890 PRINT:PRINT"   ";LEN(a$)
900 PRINT:PRINT "DRUECKE"
910 PRINT "[ENTER]"
920 WHILE INKEY$<>CR$:WEND
930 GOTO 360
940 REM **** KREISE ****
950 r=60
960 FOR i=-r TO r STEP 2
970 h=SQR(r*r-i*i)
980 MOVE x-h, i+y:DRAW x+h,i+y,c
990 NEXT
1000 RETURN
```

Test

Bevor Sie Teil 2 des Kurses, **Fortgeschrittenes BASIC**, angreifen, sollten Sie möglichst viele eigene Programme geschrieben und den Test SAT12 durchlaufen haben. In diesem letzten Test stellen wir Ihnen Fragen über den gesamten Teil 1 des Kurses, damit Sie feststellen können, ob Ihnen etwas noch nicht ganz geläufig ist.

Viel Glück!

VERZEICHNIS DER SCHLÜSSELWÖRTER

Im folgenden sind Kapitel für Kapitel alle Schlüsselwörter des Schneider BASIC, die in diesem Buch behandelt wurden, aufgelistet. Dabei wurden jedoch nicht alle Variationen und Erweiterungen berücksichtigt, da dies ja nur ein Buch für den Anfänger sein soll. Teil 2 des Kurses, **Fortgeschrittenes BASIC,** enthält weitere Schlüsselwörter und fortgeschrittenere Programmiertechniken.

Eine Liste aller Schlüsselwörter finden Sie in der **Schneider CPC 464-Anwendungsbeschreibung.**

Kapitel 2

RUN
LOAD

Kapitel 3

CLS
RUN

Kapitel 4

BORDER
MODE
CAT

Kapitel 5

CLG
INK
DRAW
LIST
MOVE
NEW
PAPER

PLOT
REM

Kapitel 6

INPUT
LET
LOCATE
PRINT
SAVE

Kapitel 7

CONT
EDIT
ELSE
GOTO
IF
STOP
THEN

Kapitel 8

DRAWR
END
FOR
GOSUB
MOVER

NEXT
PLOTR
RETURN
STEP

Kapitel 10

ENT
ENV
SOUND

Kapitel 11

ROUND

Kapitel 12

CHR$
INKEY$
RND
TIME

VERZEICHNIS DER PROGRAMME

Die Datenkassette A enthält die folgenden Programme, die in der gleichen Reihenfolge aufgelistet sind, wie in diesem Buch Bezug auf sie genommen wird. Auf Datenkassette B finden Sie die Tests zur Selbstüberprüfung (SAT), die der Leser am Ende jedes Kapitels, außer Kapitel 1 und 9, durcharbeiten soll.

Kapitel 2

HALLO
SIMON (siehe auch Kapitel 12)

Kapitel 3

BUCHSTABEN
NAMENWIEDERHOLUNG
TASTATUR
AM GALGEN

Kapitel 4

ZEICHNEN
KOORGEOM
BOMBER

Kapitel 5

HAUS

Kapitel 6

BALKENDIAGRAMM
WORTVERMISCHUNG

Kapitel 7

DEKO

Kapitel 8

VILLA

Kapitel 10

ZAPPOW
ORGEL

Kapitel 11

MULTIPLIKATION
GARTEN

Kapitel 12

BLACKJACK
SIMON

INDEX