Schneider
COMPUTER DIVISION

Der Schneider Computerkurs

Endlich,
der Schneider Computerkurs
für alle, die Computer
wirklich verstehen
und beherrschen
wollen!

Kursbuch

VORWORT

Die Idee zur Erstellung dieses interaktiven Lehrprogrammes ist bei den zahlreichen Schulungen herangereift, die zur Einführung in den Umgang mit Computern von **HOMESOFT** abgehalten wurden. Die bei diesen Schulungen gestellten Fragen stellen eine ständige Wiederholung immmer wiederkehrender Probleme dar.

Da Computer bekanntlich dazu gedacht sind, immer wiederkehrende Aufgaben zu verrichten, war es gedanklich nur noch ein kleiner Schritt, das von **Hans Stettmeier** entwickelte Schulungskonzept computergerecht umzuschreiben.

Systemaufbau	**Martin Sandweg**
Projektleitung	**Laci Bencze**
Programmierung Stufe 1	**Michael Geisler**
Programmierung Stufe 2 bis 6	**Laci Bencze &** **Günter Leupold &** **Mourad Boulouednine**
Ausarbeitung des Manuskriptes	**Gabriele Jaroschka &** **Beate Wollnitza &** **Katrin Reinert**
Verpackungskonzeption und Gestaltung	**Pfeiffer Agentur** für Werbung und Öffentlichkeitsarbeit

München, im August 1985

Ihr HOMESOFT-Team

BEDIENUNGSANLEITUNG FÜR DIE PROGRAMMDISKETTE

Beim CPC 464 muß das Diskettenlaufwerk (bzw. wenn Sie zwei haben, dann beide Diskettenlaufwerke) mit dem Computer verbunden und eingeschaltet sein. Haben Sie ihren Computer vor dem Diskettenlaufwerk eingeschaltet, dann müssen Sie Ihr Diskettenlaufwerk zuerst aktivieren. Dies geschieht mit dem Befehl **!DISC** (! = SHIFT und "Klammeraffe").

Laden und Starten des Lernprogramms:

Tippen Sie ein:

RUN "Start" **<ENTER>**

Jetzt lädt Ihr Computer das Startprogramm, während das Schneider-Logo angezeigt wird. Kurze Zeit später erscheint die **Hauptauswahl:**

SCHNEIDER Computerkurs

Hauptauswahl

Stufe	1	(Alles über meinen Computer.)	... 1
Stufe	2	(Wie arbeitet mein Computer?)	... 2
Stufe	3	(Ich lasse meinen Computer arbeiten!)	... 3
Stufe	4	(Noch mehr Arbeit für meinen Computer!)	... 4
Stufe	5	(Programmieren in BASIC.)	... 5
Stufe	6	(Programme für meinen Computer!)	... 6
Übersicht über die einzelnen Stufen			... 7
Einstellung von Farben			... 8
Beenden des Programmes und zurück zum Basic			... e

Bitte wählen Sie

Auswahl einer Lektion:

Die Diskette sollte im Laufwerk bleiben und während des Ladens auch nicht entfernt werden!

Mit den **Tasten 1 bis 6** werden die Stufen 1 bis 6 ausgewählt. Jede dieser Stufen ist in mehrere Teile untergliedert, die zu Beginn der Stufe ausgewählt werden können. Dabei ist keine bestimmte Reihenfolge vorgeschrieben, es empfiehlt sich jedoch, stets mit Teil 1 anzufangen.
Jeder dieser Teile besteht aus mehreren Bildschirmseiten, in denen Sie "blättern" können:

Taste "Cursor rechts" zur nächsten Seite
Taste "Cursor links" zur vorherigen Seite
Taste "TAB" zur Auswahl der entsprechenden Stufe

Bei den jeweiligen Stufen haben Sie in der Auswahl ferner die Möglichkeit, mit der entsprechenden Zifferntaste zur Hauptauswahl zu "blättern".

Wenn Sie keine Lust haben, die geforderten Programmzeilen, Befehle oder sonstigen Eingaben einzutippen (vielleicht weil Sie es schon einmal gemacht haben), dann brauchen Sie nur die "Weiter-Taste" (Cursor rechts) zu drücken, und die geforderte Eingabe erscheint von selbst. Mindestens einmal sollten Sie jedoch jede Eingabe selber gemacht haben.

Auswahl der Übersicht:

Mit der **Taste 7** bekommen Sie eine Übersicht über die einzelnen Stufen und die Teile aus denen sie bestehen. Sie erhalten also eine Art Inhaltsverzeichnis der Programmdiskette.

Auswahl der Einstellungen:

Die **Taste 8** startet in der Hauptauswahl einen Programmteil, mit dem Sie Systemeinstellungen (Farben) verändern können.

Folgende Einstellungen sind möglich:

1 Systemtextfarbe
2 Textfarbe
3 Rahmenfarbe
4 Hintergrundfarbe
5 Melodie (anhören)
6 Grundeinstellungen
8 zurück (zur Hauptauswahl)

Da innerhalb der Stufen Texte farbig hervorgehoben werden, sollten Sie mit der Einstellung der Hintergrund- und der Textfarbe sehr vorsichtig umgehen. Sind nämlich die Textfarbe und die Hintergrundfarbe gleich, so bleibt der Text unsichtbar. Dies gilt auch für die Meldungen des Programmes (Systemtextfarbe).
Die von Ihnen vorgenommenen Änderungen bleiben während des gesamten Programmablaufes gültig, also auch in jeder Stufe. Beim Laden der Hauptauswahl jedoch werden wieder die Standardwerte eingestellt.

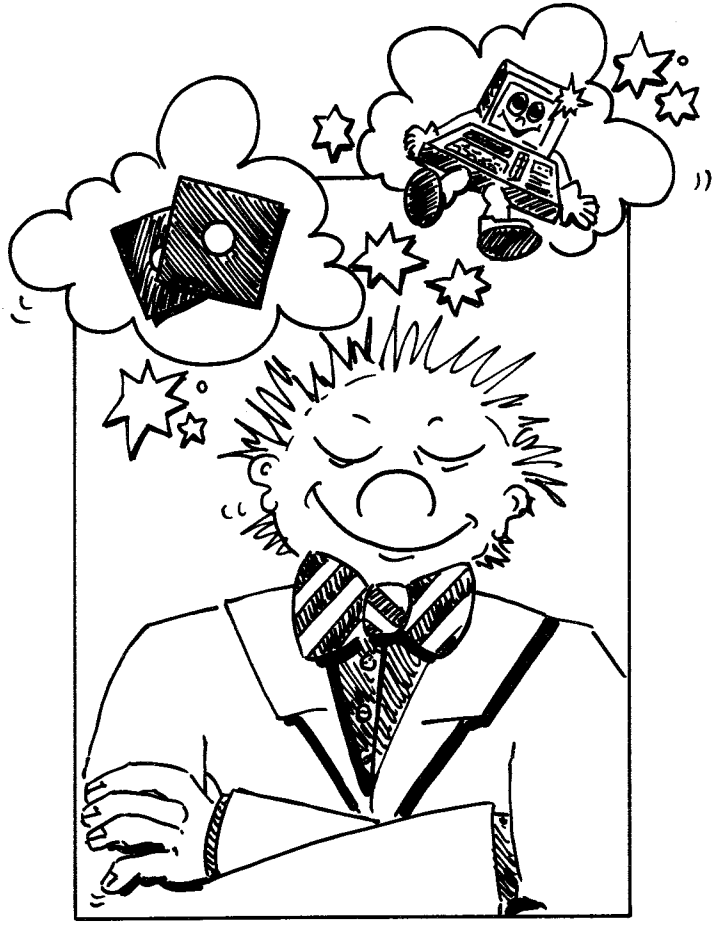
Störungen beim Programmablauf

Das Lernprogramm ist so aufgebaut, daß sämtliche Stufen und Teile interaktiv ablaufen. In der Regel wird es bei diesem Ablauf zu keinerlei Störungen kommen. Sie sollten jedoch grundsätzlich folgende Punkte beachten:

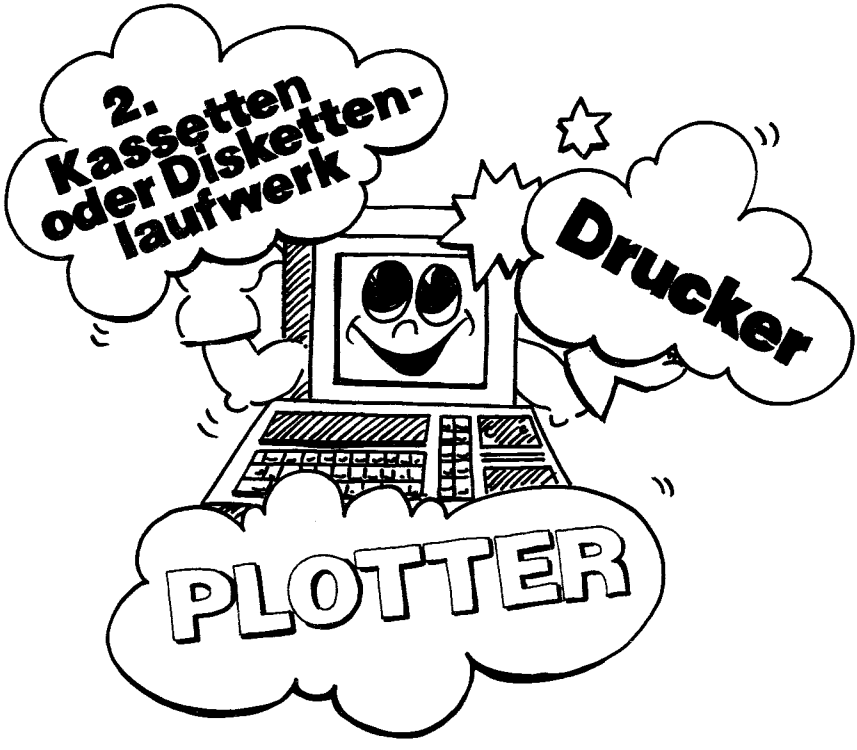
1. Die einzelnen Stufen werden von der Hauptauswahl automatisch geladen.
2. Sollte der Computer in der Hauptauswahl auf die von Ihnen gedrückten Zifferntasten nicht ansprechen, so befindet er sich im SHIFT-Modus, den Sie durch gleichzeitiges Drücken der CTRL und CAPS LOCK Taste ausschalten können. Dies kann auch in anderen Programmteilen (z.B. Menü für die Kurzbeschreibung der Programmiersprachen in Stufe 2, Teil 2, Seite 15) die Ursache für Störungen sein.
3. Vor dem Programmstart sollten Sie den Computer durch gleichzeitiges Drücken der ESC-, SHIFT- und CTRL-Taste in den Einschaltzustand zurückversetzen.
4. Sollten Sie einen Grünmonitor besitzen, dann sorgen Sie bitte durch entsprechende Einstellung der Helligkeit und des Kontrastes für eine Bildeinstellung, bei der Sie alles optimal lesen können.
5. Bei zwei Laufwerken müssen Beide eingeschaltet und aktiviert sein (siehe "Bedienungsanleitung für die Programmdiskette" erster Absatz, vgl. auch "Floppy-Handbuch").
6. Selbst bei einem noch so perfekten Programm kann es durch äußere Einflüsse zu einem Programmabsturz kommen. Der Grund hierfür kann z.B. in einem kurzzeitigen Stromausfall liegen, den Sie meistens gar nicht registrieren können. In solchen und ähnlichen Fällen sollten Sie wie im Punkt 3 beschrieben Ihren Computer zurücksetzen, und das Programm - wie unter "Laden und Starten des Lernprogramms" beschrieben - neu starten.

INHALTSVERZEICHNIS

STUFE 1 ALLES ÜBER MEINEN COMPUTER



Teil 1: Die Geräte



Mit dem Schneider CPC haben Sie einen Computer gekauft, der auf Grund seiner hervorragenden Leistungsmerkmale eine Fülle von Anwendungsmöglichkeiten bietet.
An Ihren Schneider Computer läßt sich eine Vielzahl von zusätzlichen Geräten anschließen. Diese Geräte faßt man unter den Begriff

Peripheriegeräte

zusammen. Als erstes für den Betrieb Ihres Schneider Computers notwendiges Peripheriegerät wird ein Datensichtgerät angeschlossen. Im Gegensatz zu vielen anderen Computern haben Sie beim Kauf Ihres Schneider Computers ein solches Datensichtgerät bereits miterworben. Hierbei handelt es sich entweder um einen Farb-, oder einen Monochrom-

Monitor.

Sollten Sie einen Monochrommonitor besitzen, so ist es in diesem speziellen Fall ein sog. Grünmonitor. Dieser Monitor stellt die verschiedenen Helligkeitsstufen - einem schwarz-weiß Fernsehgerät ähnlich - in verschiedenen Grünstufen dar.
Selbstverständlich haben Sie bei Ihrem Schneider Computer auch die Möglichkeit, ein handelsübliches Farb-, oder schwarz-weiß Fernsehgerät anzuschließen. Hierzu benötigen Sie jedoch eine sog. Schnittstelle, über die Sie aber im Teil 3 dieser Stufe mehr erfahren werden. Sollten Sie ein Fernsehgerät angeschlossen haben, so empfängt dieses über die Antennenbuchse auf

UHF Kanal 36

das vom Computer erzeugte Bild.
Der zu Ihrem Schneider Computer gehörende Monitor besitzt jedoch gegenüber einem Fernsehgerät erhebliche Vorteile:
Es ist allgemein bekannt, daß zu langes Sitzen vor dem Fernsehgerät nicht nur ermüdet, sondern auch für Ihre Augen schädlich ist. Diese negative Eigenschaft wird sowohl durch das Flimmern, als auch durch die relativ ungenaue Zeichenauflösung bei handelsüblichen Fernsehgeräten verursacht. Sollten Sie also längere Zeit mit dem Computer arbeiten, so ist es auf jeden Fall ratsam, den mitgelieferten Monitor zu benutzen. Ein Monitor sorgt nämlich durch höhere Zeichenauflösung und höhere Bildfrequenzen für ein schärferes und somit augenfreundlicheres Bild.
Welches Datensichtgerät Sie nun auch benutzen mögen; es handelt sich hierbei für die weitere Arbeit mit dem Computer um Ihre wichtigste Informationsquelle. Bei dem Schneider Computerkurs können Sie von dem Bildschirm wie aus einem Buch lesen, vor- oder zurückblättern sowie jederzeit zum Inhaltsverzeichnis zurückspringen.

Ihr Schneider Computer kann, falls Sie einen Farbmonitor oder ein Farbfernsehgerät angeschlossen haben, 27 verschiedene Farben auf dem Bildschirm erzeugen.
In der Hauptauswahl befindet sich unter 8. der Menüpunkt "Einstellungen von Farben". In diesem Programmabschnitt haben Sie die Möglichkeit, die Farbmöglichkeiten Ihres Computers in allen Variationen durchzuprobieren. Anschließend sollten Sie aber wieder die Grundfarben einstellen, weil hierdurch auch beim weiteren Programmverlauf eine ausgewogene Farbzusammenstellung gewährleistet wird. Mit dem Schneider Computerkurs haben Sie neben diesem Buch auch ein

Computerprogramm

erworben. Dieses Programm befindet sich auf einem

Datenträger.

Wenn Sie Ihren Computer ausschalten, so werden alle von Ihnen erzeugten Daten oder Programme im

Arbeitsspeicher

Ihres Computers gelöscht, weil die Stromversorgung unterbrochen wird.
Alle Daten, die man später wieder benötigt, müssen also auf stromunabhängige Datenträger übertragen werden.
Ihr Schneider Computer verwendet als Datenträger

Kassetten

oder

Disketten.

Für den Kassettenbetrieb steht Ihnen im Schneider CPC 464 ein im Computergehäuse miteingebautes

Kassettenlaufwerk

zur Verfügung. Wie bei allen anderen Audiokassetten auch, werden vom Computer die Daten bzw. Programme auf das Band überspielt und bei Bedarf wieder vom Band gelesen.

Zum Abspeichern stehen Ihnen zwei verschiedene Schreibgeschwindigkeiten zur Verfügung. Sollten Sie Kassetten minderer Qualität benützen, empfielt es sich wegen der höheren Datensicherheit die langsamere Schreibgeschwindigkeit zu verwenden. Beim Laden von Dateien schaltet Ihr Schneider Computer automatisch auf die richtige Lesegeschwindigkeit um. Nähere Informationen entnehmen Sie bitte Ihrer Bedienungsanleitung.
Der Schneider Computerkurs ist natürlich auch auf Diskette erhältlich. Um dieses Programm von der Diskette in Ihren Computer zu laden, benötigen Sie ein

Diskettenlaufwerk.

Dieses Peripheriegerät wird im Fachjargon auch als

Floppy, Disk Drive oder **Disk Drive Unit**

bezeichnet. Auch ein Diskettenlaufwerk speichert oder liest Daten bzw. Programme, ähnlich einem Kassettenlaufwerk. Der Unterschied zum Kassettenlaufwerk besteht in einer weitaus höheren Datensicherheit, sowie einer höheren Geschwindigkeit beim Abspeichern oder Laden von Daten bzw. Programmen. Ebenfalls entfällt hierbei das lästige Hin- und Herspulen der Kassette, um das benötigte Programm aufzufinden. Man spricht bei dem Auffinden bestimmter Daten oder Programme auf einem Datenspeicher auch von der sog. Zugriffszeit, welche bei der Diskette erheblich bequemer und schneller abläuft.
Das externe Diskettenlaufwerk Ihres Schneider Computers muß, ebenso wie ein evt. vorhandener Matrixdrucker, über eine Schnittstelle an Ihren Computer angeschlossen werden. Die Schneider Diskettenlaufwerke bearbeiten

Dreizoll - Disketten

Double Sided - Double Density,

auf Deutsch: auf zwei Seiten mit doppelter Schreibdichte, wobei Sie durch Umdrehen der Diskette die zweite Seite beschreiben können.

All diese zuletzt erwähnten Daten sollten Sie beim Kauf von

Leerdisketten

Ihrem Fachhändler mitteilen, falls dieser nicht wissen sollte, welche Disketten Ihr Laufwerk verarbeitet.

Bevor Sie auf neu erworbene Leerdisketten zum ersten Mal Daten oder Programme speichern, müssen Sie die Disketten

formatieren.

Da verschiedene Diskettenlaufwerke Dreizoll - Disketten bearbeiten, jedoch das

Format der Daten

von Computer zu Computer verschieden auf Disketten ausgegeben wird, werden Disketten in der Regel

unformatiert

von den Herstellern geliefert. Das Formatieren einer Diskette geschieht durch ein speziell hierfür vorgesehenes Programm, welches sich auf der Systemdiskette befindet, die Ihnen beim Kauf Ihres Diskettenlaufwerkes mitgegeben wurde. Sie werden beim Durcharbeiten des Schneider Computerkurses erste Programmbeispiele kennenlernen und vielleicht anschließend Ihr erstes eigenes Programm ausprobieren. Damit Sie von Anfang an Ihr "Erstlingswerk" abspeichern können, haben wir Ihnen die Vorgehensweise beim Arbeiten mit Kassetten bzw. Disketten in der diesem Buch beigefügten Broschüre genauestens beschrieben.
Zum "klassischen Computersystem" gehört neben Computer, Monitor und Diskettenlaufwerk auch noch ein

Drucker.

Obwohl man in Verbindung mit dem Computer oft von dem sog. papierlosen Büro spricht, ist diese Aussage nur bedingt richtig.
Auch beim Arbeiten mit dem Computer gibt es viele Arten von Computerergebnisse, die man gerne "schwarz auf weiß" hätte. Bei der Inventur des Lagerbestandes zum Beispiel wird wohl niemand auf den Gedanken kommen, den Computer mit ins Lager zu nehmen, um die vom Computer angezeigten Sollwerte mit dem tatsächlichen Lagerbestand zu vergleichen.

Ebensowenig praktikabel wäre wohl der Versuch, auf einer Vorstandsversammlung die neuesten Umsatzzahlen auf Diskette präsentieren zu wollen. Ich glaube, auch in diesem Fall dürfte niemand ernsthaft an der Notwendigkeit einer Liste oder Tabelle -auch wenn vom Computer ausgedruckt - zweifeln. Je nach den an den Ausdruck gestellten Anforderungen sollte deshalb zu gegebener Zeit ein passender Drucker als zusätzliches Peripheriegerät an den Computer angeschlossen werden.
Zum Ausdruck der vom Computer erstellten Ergebnisse stehen uns eine Vielzahl verschiedener Drucker zur Verfügung.
Hierbei lassen sich grundsätzlich drei verschiedene Systeme unterscheiden:

Typenraddrucker

sind eine Art elektronische Schreibmaschine ohne Tastatur. Von allen Möglichkeiten einen Text auszudrucken besitzen Sie das schönste Schriftbild, welches - normalen Typenradschreibmaschinen ähnlich - lediglich von der Art des eingelegten Typenrades abhängig ist.

Eine andere Möglichkeit des Ausdrucks bieten die

Matrixdrucker.

Sie haben ein sehr "computerspezifisches" Schriftbild, da jeder Buchstabe aus einzelnen Punkten zusammengesetzt ist. Dieses System hat aber den Vorteil, daß mit dem selben Druckkopf, bei entsprechender Programmierung des Druckers, alle nur erdenklichen Schriftarten - einschließlich der Schreibschrift - praktikabel sind. Dieses System ermöglicht aber auch den Ausdruck von Bildern oder Grafiken, welche aus einem recht groben Punktraster entstehen. Sie sind den in Zeitungen abgedruckten Photographien sehr ähnlich, besitzen jedoch eine etwas niedrigere Auflösung.
Der Matrixdrucker läßt sich über einen Programmbefehl in den Near-Letter-Quality-Modus umschalten. Dies bedeutet, daß die Buchstaben und Zeichen zwar wesentlich langsamer aber dafür nahezu in Schreibmaschinenqualität gedruckt werden.
Der Matrixdrucker stellt also ein sehr vielseitiges System dar, und kann wohl auch wegen seiner relativ hohen Schreibgeschwindigkeit als ein sehr universell einsetzbares Gerät betrachtet werden. Für den Einsteiger sind sie jedoch auch deshalb sehr gut geeignet, weil Matrixdrucker bereits für Preise zu haben sind, die deutlich unter den Preisen einer elektronischen Schreibmaschine liegen.

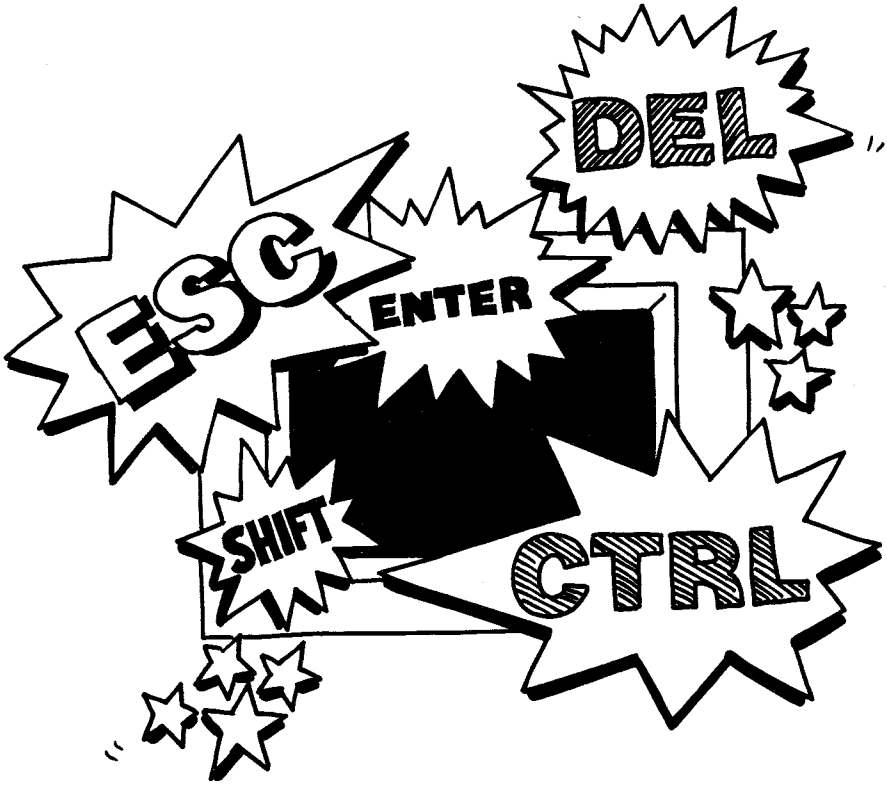
Die wohl interessanteste Möglichkeit, Computerausdrucke zu erhalten, ist zweifelsohne die Benutzung eines

Plotters.

Der Plotter führt - ähnlich der menschlichen Hand - einen Zeichenstift übers Papier, wobei der Zeichenstift nach links und rechts und das Papier nach oben und unten verschoben werden. Wie fein die Auflösung der so gezeichneten Kurven zum Beispiel wird, ist sehr von der Art und Qualität des Plotters abhängig.

Für welches Gerät Sie sich also entscheiden, hängt einerseits von den gewünschten Anwendungenden, andererseits aber auch von Ihren preislichen Vorstellungen ab.

Teil 2: Die Tastatur



Durch Drucker und Bildschirm sind uns bereits zwei Möglichkeiten bekannt, wie der Computer uns seine Ergebnisse mitteilen kann. Das Instrument, mit dem wir dem Computer unsere Informationen mitteilen, ist die

Tastatur.

Dieses Bindeglied zwischen Mensch und Computer ist so wichtig, daß wir ihm den ganzen zweiten Teil gewidmet haben.
Die Tastatur besteht aus verschiedenen Funktionsgruppen, die größtenteils durch ihre Farbe bzw. durch ihre Position im Tastenfeld gekennzeichnet sind. Sie ist in drei Blöcke aufgeteilt.

1. Ganz rechts, neben dem Kassetten- bzw. Diskettenteil, befindet sich der Zehnerblock zur Zifferneingabe, der auch als Funktionstastengruppe verwendet werden kann. Obwohl es sich hierbei um zwölf Tasten handelt, werden sie als Zehnerblock bezeichnet, weil sie unter anderem die Ziffern Null bis Neun präsentieren. Die zwei restlichen Tasten sind nach dem Einschalten des Computers als Punkt- und ENTER- Taste definiert. Diese Art der Vorbelegung bezeichnet man im Fachjargon auch als Defaultwert. Sämtliche zwölf Tasten können Sie jedoch auch mit einem beliebigen neuen Text mit jeweils bis zu 32 Zeichen Länge belegen.

2. Direkt über dem Zehnerblock befinden sich die vier Tasten zur Cursorsteuerung. Der Cursor ist das helle Rechteck, welches Ihre momentane Schreibposition anzeigt. Inmitten dieser vier Cursorsteuertasten befindet sich die sog. COPY-Taste, welche Sie zum Editieren (d.h. verändern) Ihrer Programme benötigen. Zur Benutzung dieser COPY-Taste lesen Sie bitte in Ihrem Handbuch den Teil mit der Überschrift Copycursor-Methode.

3. Links von dem Zehnerblock befindet sich die eigentliche Schreibmaschinentastatur. Sie ist nach der amerikanischen Norm aufgebaut:
- Es sind keine Umlaute vorhanden
- Y und Z sind vertauscht im Vergleich zur deutschen Norm
- Die Sonderzeichen (wie z.B. *, ?, = usw.) befinden sich nicht an den von der deutschen Norm her gewohnten Plätzen.

4. Unter diesen Schreibmaschinentasten befinden sich auch einige Sondertasten, welche uns jedoch von der Schreibmaschine her bereits bekannt sind:
- Ganz unten in der gewohnten länglichen Form befindet sich die Leertaste
- Am linken und rechten Rand der vorletzen Zeile sind die sog.

SHIFT- Tasten.

Dies sind die der Schreibmaschine entsprechenden Umschalttasten, mit denen wir z.B. auf Großbuchstaben umschalten können.

- Oberhalb der linken SHIFT- Taste befindet sich die

CAPS LOCK- Taste.

Diese Taste hat eine ähnliche Funktion wie die SHIFT- Taste, braucht jedoch nur einmal gedrückt zu werden, um die Shiftfunktion auf Dauer zu arretieren. Wird sie nochmals gedrückt, so kehrt Ihr Computer sie wieder in den Normalmodus zurück. Im Gegensatz zur SHIFT-Taste jedoch, werden nur die Buchstaben als Großbuchstaben ausgegeben. Diese Umschaltung erstreckt sich also nicht auf die Nummerntasten. Wollen Sie also die sich auf den Nummerntasten befindenden Sonderzeichen erreichen, so müssen Sie trotz gedrückter CAPS-LOCK-Taste mit der SHIFT-Taste auf die obere Zeichenreihe umschalten oder die Taste ESC gedrückt halten und dann die CAPS-LOCK-Taste betätigen.

- Ebenfalls von der Schreibmaschine her bekannt ist uns die

TAB- Taste.

Sie bewegt unsere Schreibmarke (Cursor) zur nächsten eingestellten Tabulatorposition.

5. Neben diesen Sondertasten gibt es noch eine Anzahl von Computerspezifischen Tasten, die wir auf keiner Schreibmaschinentastatur finden werden:

- Rechts neben der Leertaste befindet sich die

CTRL- Taste.

Ähnlich der SHIFT- Taste, welche die Tastatur sozusagen in eine zweite Ebene umschaltet, schaltet die CTRL- Taste in eine dritte Ebene um.

- Über der rechten SHIFT-Taste liegt die wohl wichtigste Taste: Die

ENTER- Taste.

Sie sagt dem Computer, daß Ihre Eingabe beendet ist und er Sie ausführen soll. Die ENTER- Taste muß nach jeder Anweisung bzw. nach jeder neuen oder geänderten Programmzeile gedrückt werden, da sonst der Befehl nicht ausgeführt bzw. die Programmzeile nicht oder an falscher Stelle gespeichert wird.

- Im rechten oberen Eck des Tastaturfeldes, also über der Entertaste, liegen die Tasten

CLR und DEL.

Sowohl die CLR (Clear)- Taste als auch die DEL (Delete)- Taste dienen zum Löschen bereits eingegebener Zeichen.
Die Delete- Taste wird benutzt, um am Bildschirm das Zeichen links vom Cursor zu löschen.
Die Clear- Taste CLR löscht das Zeichen innerhalb des Cursors, und verschiebt den rechts vom Cursor stehenden Zeilenrest um ein Zeichen.

- Gegenüber im linken oberen Eck des Tastaturfeldes befindet sich die

ESC- Taste.

ESC ist die Abkürzung für das englische Wort escape, was auf Deutsch übersetzt fliehen bzw. verlassen bedeutet. Wenn Sie diese Taste einmal drücken, unterbricht der Computer seine laufende Funktion, arbeitet jedoch weiter, sobald eine andere Taste gedrückt wird. Beim zweimaligen Drücken der ESC- Taste verläßt der Computer seine Funktion, gibt die Meldung "Break" (deutsch: unterbrochen) aus und erwartet neue Anweisungen von Ihnen. Die ESC- Taste in Verbindung mit der CTRL- und einer der SHIFT- Tasten zusammengedrückt verursacht einen sog. System-reset. Diese "Killer"- Kombination versetzt Ihren Computer in den Anfangszustand, den er sonst nur nach dem Einschalten einnimmt. Es werden also sämtliche Informationen, die Sie in der Zwischenzeit Ihrem Computer eingegeben haben, gelöscht, weshalb bei dieser Tastenkombination äußerste Vorsicht angebracht ist!

Teil 3: Die Begriffe

Wie rechnet der Computer eigentlich?

Der Computer kennt nur zwei "Zustände". Wenn wir uns seinen Speicher als viele kleine Kästchen vorstellen, so kann jedes dieser Kästchen entweder gefüllt oder leer sein. Ein solches Kästchen nennt man

BIT.

Ein Bit ist entweder leer (0) oder voll (1). Eine weitere Möglichkeit gibt es nicht!
Man kann sich gut vorstellen, daß so ein Kästchen durch einen kleinen Strom gefüllt bzw. geleert wird. Durch eine Folge solcher Ströme entsteht auch eine Folge von "Entleerungen" bzw. "Füllungen". Diese Vorgänge lassen sich durchaus mit einer Folge von EIN (1)-/AUS (0)-Schaltern vergleichen. Ein Schalter kann aber nur entweder ein- oder ausgeschaltet sein; weitere Möglichkeiten gibt es auch hier nicht.
Hat man jedoch zwei Schalter nebeneinander, so gibt es bereits vier verschiedene Kombinationsmöglichkeiten:

EIN	EIN
EIN	AUS
AUS	EIN
AUS	AUS

Im Computer werden , bildlich gesehen, acht Schalter pro Schaltvorgang gleichzeitig betätigt. Das ergibt 256 verschiedene Möglichkeiten, acht mal 1BIT (Ein bzw. Aus) zu schalten. Unser Alphabet hat 26 Buchstaben, unser Zahlenalphabet 10 Ziffern. Selbst die ganzen Sonderzeichen zusammen belegen nicht diese gesamten 256 Möglichkeiten.
Einen Schaltvorgang, bestehend aus 8 BITS, die man

1 BYTE nennt.

Jedes BYTE (acht Bit) entspricht genau einem Zeichen. (2 KB = Kilobyte entsprechen etwa einer DIN A4 Seite mit Text) Wie aber entsteht aus den Zeichen ein sinnvoller Text?
Die Hauptarbeit verrichtet die **CPU**, was Central Processing Unit oder Zentrale Verarbeitungseinheit heißt. Sie kann mit einem Briefträger verglichen werden. Die CPU empfängt von verschiedenen Stellen Meldungen, die sie auswertet (sortiert) und dann an diverse Geräte weitergibt. Die CPU alleine kann zwar im Prinzip schon "rechnen", aber sie weiß weder, was sie rechnen soll noch kann sie uns ihre Ergebnisse mitteilen. Dazu werden Tastatur und Bildschirm benötigt.

Auch kann sich unser Rechner noch nichts merken. Er braucht dazu ein "Notizbuch"(Speicher). Dieser Speicher, in dem der Computer schreiben und lesen kann, nennt man

RAM

oder Random Access Memory (Schreib- und Lesespeicher).
Der Speicherplatz des RAM (in Kilobytes) entscheidet darüber, wieviel Platz für Daten und Programme vorhanden ist.
Nun muß aber die CPU auch noch "wissen", wie sie die von uns über die Tastatur eingegebenen Anweisungen verarbeiten soll. Entweder erteilen wir ihr darüber Informationen oder diese sind irgendwo im Computer vorhanden.
Tatsächlich gibt es im Computer einen solchen Speicher mit immer vorhandenen Informationen (auch bei ausgeschaltetem Gerät). In diesem Speicher kann die CPU "nachschlagen" wie in einem "Lexikon". Dieses Nachschlagewerk, aus dem der Computer z.B. sein Wissen über mathematische Formeln und BASIC- Befehle bezieht, nennt man

ROM

oder Read Only Memory (nur Lesespeicher). Darin befinden sich alle Befehle und alle dem Computer "bekannten" Funktionen, kurz alles, was der Computer weiß. Man spricht auch vom

Betriebssystem.

Die Tastatur, die Zentrale Verarbeitungseinheit (CPU), die beiden Speicherbereiche (RAM = Schreib- und Lesespeicher, ROM = Nur-Lesespeicher) sowie eine Kassetten- bzw. Diskettenstation befinden sich im Computergehäuse. Alle anderen Geräte wie z.B. Monitor, Drucker oder zusätzliche Kassetten- bzw. Diskettenstationen werden als Peripheriegeräte angeschlossen. Um ein Peripheriegerät anschließen zu können, benötigt man eine

Schnittstelle (Interface).

Eine Schnittstelle ist eine Art Übersetzer. Sie übersetzt die Meldung des einen Gerätes (z.B. des Druckers oder der Diskettenstation) in die Sprache des Anderen. Verwendet man Geräte desselben Herstellers, sind in der Regel die Schnittstellen bereits eingebaut oder wurden beim Kauf der Peripherie mitgeliefert wie bei den (zusätzlichen) Schneider Diskettenlaufwerken. Man kann dann die Geräte problemlos miteinander verbinden und betreiben. Beim Anschluß von Geräten anderer Hersteller gilt es, sich auf eine Schnittstelle zu einigen, die sowohl der Computer als auch das angeschlossene Gerät verstehen. Die wichtigsten heute verwendeten, genormten, Schnittstellen sind:

RS 232/V24 Seriell
Centronics Parallel

Es gibt zwei Möglichkeiten, ein Byte (acht Bit) zu senden: Entweder acht Bit gleichzeitig auf acht Leitungen (Parallel) oder ein Bit nach dem anderen auf einer Leitung (Seriell).

Wir haben uns bisher hauptsächlich mit dem Computer, seinen Bestandteilen und den Peripheriegeräten befaßt. Alle diese Dinge lassen sich anfassen und werden unter dem Oberbegriff

Hardware

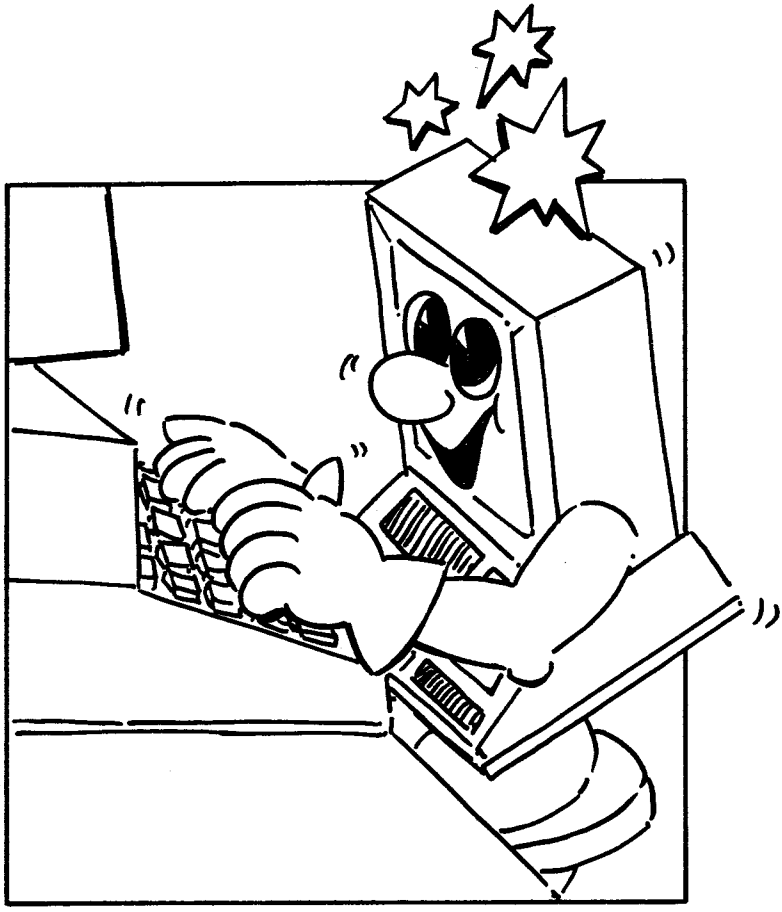
zusammengefaßt.

Die Tatsache aber, daß ein Computer so funktioniert wie wir ihn heute kennen und einsetzen, ist der

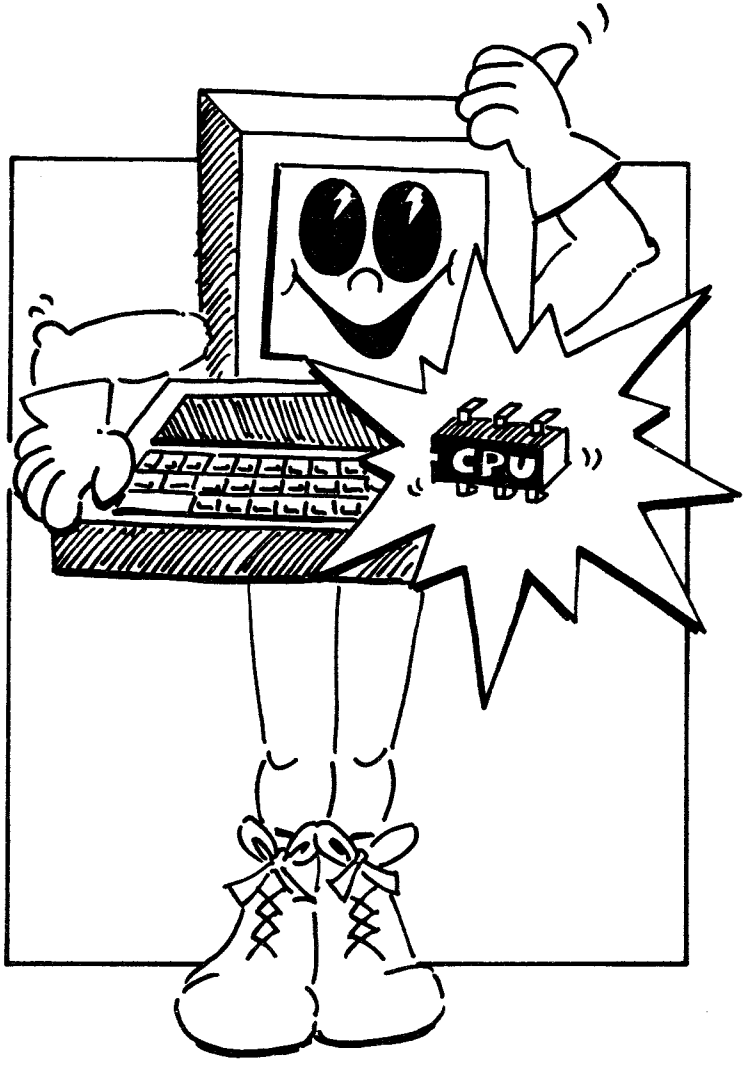
Software

zu verdanken.
Ohne funktionierende Programme - Software ist hierfür der Oberbegriff - ist das Arbeiten mit Computern nicht mehr denkbar. Wir werden noch mehrmals auf dieses wichtige Thema im weiteren Verlauf des Kurses eingehen.

STUFE 2 WIE ARBEITET MEIN COMPUTER



Teil 1: Das Betriebssystem



Wie wir in Stufe 1 erfahren haben, steuert der

Mikroprozessor

- damit ist im allgemeinen die CPU gemeint - alle Vorgänge im Computer. Wie diese Steuervorgänge ablaufen, hängt vom Betriebssystem ab.
In Ihrem Computer-Kurs wird dieses Zusammenspiel zwischen CPU und Betriebssystem anhand mehrerer Bilder verdeutlicht.
Das Betriebssystem setzt sich aus mehreren im ROM (Lesespeicher) gespeicherten Programmen zusammen. Wir wollen auf einige dieser Programme näher eingehen:
Wenn Sie Ihren Computer einschalten, wird - wenn er richtig angeschlossen ist - auf Ihrem Bildschirm ein Bild erzeugt. Dieses Bild besteht aus zahlreichen Einzelpunkten, die in verschiedenen Farben (27 Möglichkeiten pro Punkt) zum Leuchten gebracht werden können. Aus den dunkelblauen Punkten setzt sich der Hintergrund, aus den gelben der Text zusammen. Diesem ersten Bild sind eine Vielzahl von Steuervorgängen vorangegangen, die durch das Einschalten Ihres Computers aktiviert wurden. Die Informationen, wie sich dieses Bild zusammensetzt, hat die CPU im

Monitorprogramm

abgefragt. Dieses Programm sorgt auch dafür, daß beim Drücken von Tasten der

Zeichensatz

(Buchstaben, Ziffern, Grafikzeichen) auf dem Bildschirm richtig dargestellt wird.
Ein weiteres und umfangreicheres Betriebssystemprogramm ist der

Basic-Interpreter.

Dieses Übersetzungsprogramm für die Programmiersprache BASIC prüft, ob die Eingaben in der Sprache richtig erfolgt sind und organisiert die Durchführung des von uns eingegebenen

Programmes.

Sie werden zudem festgestelt haben, daß Ihr Schneider Computer viele Ein- und Ausgänge (die in Ihrem Benutzerhandbuch alle beschrieben sind) aufweist.
Im ROM befinden sich mehrere Programme, die dafür sorgen, daß die Signale, die von diesen Eingängen empfangen werden, von der CPU richtig weiterverarbeitet werden können. Wenn Sie z.B. ein Computerspiel mit

Joysticks

(= Fernbedienungshebel) spielen, muß dafür gesorgt werden, daß Ihre Steuerungen vom Spielprogramm verarbeitet und auf dem Bildschirm richtig ausgeführt werden können.

Wollen Sie von Ihrem Computer erarbeitete Ergebnisse speichern und anschließend auch noch ausdrucken, so sorgen auch hier die Betriebsprogramme dafür, daß die Daten zu den richtigen Ausgängen und von dort zu den richtigen Geräten gelangen.
Je mehr Sie in Zukunft mit Ihrem Computer arbeiten, desto mehr werden Sie "Ihr Betriebssystem" zu schätzen wissen.
Die Alternative zum komfortablem Betriebssystem Ihres Schneider Computers sind eigene

Maschinenprogramme.

Sie müssen dann all das, was ansonsten von den Betriebssystemprogrammen unterstützt wird, selber computergerecht formuliert Ihrem Computer mitteilen. Da der Computer dann aber Ihre Anweisungen nicht in der komfortablen BASIC- Sprache empfangen kann - der Übersetzer wird dann nähmlich ausgeschaltet -, heißt das , daß Sie BIT für BIT selbst steuern müssen. Dies erfordert fundierte Kenntnisse in Logik- und Schaltungstheorie. Für fortgeschrittene Computerfreaks ist dies jedoch ständig eine reizvolle Herausforderung.

Teil 2: Programmiersprachen

Der Stellenwert, welchen die SOFTWARE heute im Computerzeitalter einnimmt, ist den Programmiersprachen zu verdanken. Sie sind das Werkzeug des Programmierers. Sie erlauben ihm klare und eindeutige Anweisungen in

Worten

unter Berücksichtigung der

Grammatik

zu einem

Programm

zusammenzufassen.

In den Anfängen des Computerzeitalters bestand ein Programm (eine definierte zeitliche Folge von Abläufen) aus einer Folge von einzeln geschalteten BITS. Selbst elementarste Vorgänge wie z.B. die Multiplikation zweier Zahlen wurde von Spezialisten auf diese Weise programmiert. Im Laufe der Zeit entstanden eine Sammlung von elementaren Programmen für mathematische Operationen sowie einfache Sortiervorgänge für Zahlen und Worte. Da diese Programme permanent benötigt wurden, hat man sie im Verlauf der Zeit nicht mehr fallweise einprogrammiert, sondern von einem Datenträger geladen und mit Hilfe eines Startwortes als

Unterprogramm

in ein laufendes Programm eingebunden. Letzlich hatte man für alle wichtigen Operationen ein Wort das bei Bedarf ein Unterprogramm aktivieren konnte. Je umfangreicher dieser

Wortschatz

wurde, desto wichtiger wurde es, die Struktur einer Sprache durch die Grammatik festzulegen. Damit man jederzeit in der Sprache arbeiten konnte, hatte man schon bald einen permanent vorhandenen Speicher (ROM) in den Computer integriert. Die Sprachen und deren

Syntax (=Wortschatz und Grammatik)

wurden laufend verbessert, bis sich im Laufe der Zeit einige Sprachen besonders etablieren konnten. Diese Sprachen (BASIC, PASCAL, FORTRAN usw.) sind ohne Anspruch auf endgültige Vollständigkeit in Ihrem Diskettenprogramm aufgeführt.

Ein Werkzeug erfolgreich anzuwenden heißt, es zu kennen und bedienen zu können. Die Schalter des Werkzeuges "Programmiersprache" sind die Befehle (Worte). Wie bei jedem Werkzeug sind auch bei Programmiersprachen nur bestimmte Schaltfolgen (Syntax) erlaubt.
Damit es zu keinen "Fehlschaltungen" kommen kann, müssen unzuläßige Schaltfolgen verhindert und wenn möglich angezeigt werden. Diese

Syntaxprüfung

ist ein wesentlicher Bestandteil einer jeden

höheren Programmiersprache.

Je komfortabler und umfangreicher die Syntaxprüfung einer Programmiersprache ist, desto angenehmer ist für den Programmierer das Arbeiten in dieser Sprache. Die Syntaxprüfung ist aber auch zeitaufwendig. Dies gilt besonders dann, wenn jeder Satz einzeln übersetzt und geprüft wird. Dies ist nämlich die Arbeitsweise vom

Interpreter.

Der Vorteil für den Programmierer ist, daß jeder Fehler sofort angezeigt wird und gleich behoben werden kann.
Es besteht andererseits die Möglichkeit, ein Programm komplett einzugeben, es anschließend zu übersetzen und eine Fehlerliste ausgeben zu lassen. Man vermeidet bei umfangreichen, fehlerfreien Programmen das permanente, satzweise Aktivieren des Prüfprogrammes und spart dadurch, daß die Syntaxprüfung nur einmal erfolgt, viel Zeit. Geübte Programmierer benützen deshalb in der Regel einen

Compiler

- so nennt man dieses Übersetzungsprogramm. Nahezu alle von professionellen Programmierern bevorzugten höheren Programmiersprachen werden compiliert. Für den Anfänger hat diese Form der Übersetzung aufgrund der häufigen Syntaxfehler einen enormen Zeitaufwand bis zur ersten lauffähigen Programmversion zur Folge. Aus diesem Grund bedienen sich Programmiersprachen für Anfänger (wobei BASIC die mit Abstand populärste ist) in der Regel eines Interpreters.
Neben den höheren Programmiersprachen gibt es auch

niedere Programmiersprachen.

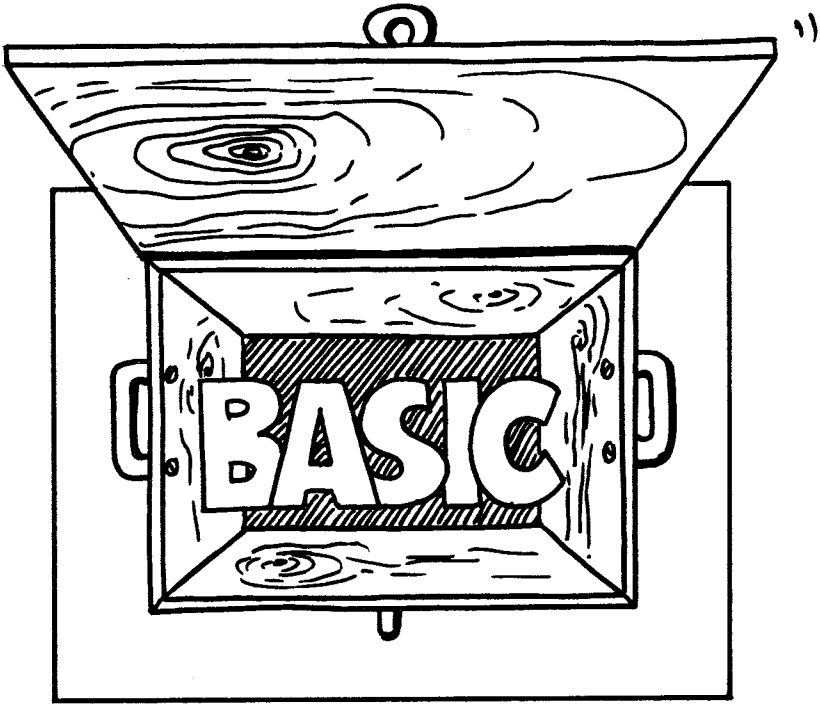
Man nennt sie auch maschinenorientiert, da ihr Befehlssatz (Wortschatz) von der CPU des verwendeten Rechners abhängig ist.

Die am weitesten verbreiteten Arten niederer Programmiersprachen sind die

Assemblersprachen.

Diese Sprachen verfügen über einen der Struktur der CPU angepaßten - im Vergleich zu höheren Programmiersprachen sehr begrenzten - Wortschatz. Dadurch fallen folglich umfangreiche Übersetzungsprogramme sowie Programme zur Syntaxprüfung weg.
Um Programme beschleunigt ausführen zu lassen, werden Assemblersprachen verwendet. Der Aufwand der Programmierung ist wesentlich höher als bei BASIC, PASCAL etc. Die Verarbeitungsgeschwindigkeit ist jedoch um ein Vielfaches schneller.
Bei Grafikprogrammen (z.B. Spielen) lassen sich schnelle Bewegungen auf dem Bildschirm nur durch Assembler- oder andere Maschinenprogramme erreichen. In der Praxis zeigt sich jedoch, daß die Verarbeitungsgeschwindigkeit der BASIC- Sprache Ihres Schneider Computers für die meisten Anwendungen völlig ausreichend ist.

Teil 3: Einblick in BASIC



Beim Einschalten Ihres Schneider Computers erscheint folgende Meldung:

Beim Schneider CPC 464

```
 Schneider 64K Microcomputer  (v1)

 c1984 Amstrad Consumer Electronics plc
 and Locomotive Software Ltd.

 BASIC 1.0

Ready
```

und beim Schneider CPC 664

```
 Schneider 64K Microcomputer  (v2)

 c1984 Amstrad Consumer Electronics plc
 and Locomotive Software Ltd.

 BASIC 1.1

Ready
```

Wenn jetzt eine Buchstaben- oder Zahlentaste gedrückt wird, erscheint unter dem Cursor das gedrückte Zeichen und der Cursor springt ein Feld nach rechts.
Beim Drücken der Leertaste wird natürlich kein Zeichen abgebildet, der Cursor wird jedoch ebenfalls um 1 Feld nach rechts geführt.
Je nach Textmodus springt der Cursor nach einer bestimmten Anzahl abgebildeter Zeichen in die nächste Zeile:

```
Nach 20 abgebildeten Zeichen im Modus 0
     40                              1
     80                              2
```

Nach dem 255.ten Zeichen ist die Eingabe weiterer Zeichen nicht mehr möglich, da dies die maximale Länge einer Befehlszeile (Programmzeile) ist.
Bis jetzt hat der Computer die von uns eingegebenen Zeichen kommentarlos dargestellt, jedoch noch keine Anweisung (Befehl) erhalten.
Befehle werden erteilt durch Eingabe des entsprechenden Befehlswortes und durch darauffolgendes Drücken der Taste ENTER. Auch wenn der Computer nach der Eingabe verschiedener Zeichen durch Drücken der Taste ENTER die Zeichenreihe mitgeteilt bekommt, kann er nur dann die Anweisung ausführen, wenn er die Mitteilung "verstanden" hat: Es muß ihm ein Wort aus dem ihm bekannten BASIC- Wortschatz mitgeteilt worden sein.
Kennt er die eingegebenen Worte oder Zeichen nicht, meldet er sich mit einer Fehlermeldung (z.B. BAD NAME, BAD LINE NUMBER, hauptsächlich SYNTAX ERROR).

Auf einigen Tasten sehen Sie nur einen Buchstaben, auf anderen steht über dem Buchstaben, der Zahl oder dem Zeichen ein weiteres Zeichen. Diese Zeichen stellt man durch gleichzeitiges Drücken einer der beiden SHIFT- Tasten und der entsprechenden Taste auf dem Bildschirm dar.
Es ist bei Ihrem Schneider Computer nicht erforderlich, BASIC-Befehle (Worte) entweder einheitlich groß oder einheitlich klein zu schreiben. Auch eine Kombination von Groß- und Kleinbuchstaben wird richtig erkannt (interpretiert). Ausschlaggebend ist es jedenfalls nicht, da sämtliche BASIC- Befehle bei der Auflistung des Programms ausschließlich in Großschrift dargestellt werden.

In der Programmiersprache BASIC kann man dem Computer Befehle einzeln mitteilen oder in einem Programm (Folge von mehreren Anweisungen) ausführen lassen. Im ersten Fall spricht man von

Direktmodus.

Der Computer akzeptiert grundsätzlich nur eindeutige Anweisungen. Fehler in Befehlsfolgen oder Befehlsworten werden vom Computer nur dann angezeigt, wenn sie die Darstellung des Ergebnisses unmöglich machen.

Führen die Eingaben zu einem darstellbaren Ergebnis, kann der Computer keinen Fehler feststellen, weil er die Qualität des Programmes natürlich nicht prüfen kann. Das gilt immmer dann, wenn die Befehlsworte syntaktisch richtig sind, aber im Programmverlauf logische Fehler sind, die die Syntaxprüfung nicht erkennen kann. Ist zum Beispiel in einer mathematischen Formel ein Fehler, so wird die Formel vom Computer trotzdem berechnet, wenn dieser Fehler lediglich eine veränderte, aber berechenbare Formel zur Folge hat. Das Ergebnis ist für den Anwender dann falsch und in der Regel wertlos.

Der Computer kann nicht mitdenken!

Die Bedeutung von BASIC- Befehlen lernen Sie am besten kennen, indem Sie sie einfach ausprobieren. Sie haben dazu in Stufe 2 und Stufe 3 Ihres Schneider Computerkurses mehrmals Gelegenheit. Dabei verhindert das Programm Falscheingaben, so daß vom Computer auch keine Fehlermeldungen angezeigt werden können. Sie werden aber vom Programm auf grobe Fehler (z.B. Teilung durch Null) aufmerksam gemacht.
Nachdem Sie diese Übungen durchexerziert haben, empfehlen wir Ihnen, einmal ohne die Unterstützung des Lehrprogrammes die folgenden Seiten dieses Buches praktisch an Ihrem Schneider Computer durchzuarbeiten.

Tippen Sie dazu einfach den gängigsten BASIC- Befehl ein:

PRINT

und vergessen Sie nicht, anschließend die ENTER- Taste zu drücken.
Geben Sie anstatt des richtigen Befehls

PRINNT

ein (und drücken anschließend ENTER), antwortet der Computer sofort mit der Fehlermeldung

SYNTAX ERROR

Der Cursor springt eine Zeile weiter und der Computer meldet sich wieder mit

READY

was soviel bedeutet, daß der Computer Ihren Befehl ausgeführt hat und zur Aufnahme neuer Befehle bereit ist.

PRINT

bedeutet soviel wie: Drucke, stelle dar, zeige. Sie sollten dem Computer aber auch sagen, was er zeigen soll.

Sie tippen ein:		Ihr Computer zeigt:
a) **PRINT 5**	**<ENTER>**	**5**
b) **PRINT 5+2**	**<ENTER>**	**7**
c) **PRINT 5+2*4**	**<ENTER>**	**13**
d) **PRINT (5+2)*4**	**<ENTER>**	**28**
e) **PRINT 4*2^3-32**	**<ENTER>**	**0**
f) **PRINT (5*2)^3-32**	**<ENTER>**	**968**

Aus den oben genannten Beispielen erhalten wir bereits einige wichtige Aussagen über die Syntax unseres Computers bei der Behandlung von Zahlen.
Die mathematischen Zeichen werden folgendermaßen dargestellt:

^	**(Potenz)**
/	**(Division)**
*****	**(Multiplikation)**
+	**(Addition)**
-	**(Subtraktion)**

wobei sog. Prioritäten berücksichtigt werden müssen. Mit Prioritäten ist hierbei nichts anderes als eine Rangfolge gemeint. Die höchste Priorität besitzt das Potenzieren, die niedrigste die Subtraktion. Sie können dies deutlich am Beispiel c) erkennen. Diese Rangfolge können Sie jedoch durch das Setzen von Klammern beliebig verändern (siehe d). Welche unterschiedlichen Ergebnisse durch Klammersetzung erzielt werden, können Sie bei den letzten beiden Beispielen erkennen.

Sollten zwei mathematische Operationen von gleicher Priorität aufeinandertreffen, so erfolgt die Bearbeitung durch den Computer in Leserichtung (von links nach rechts). Versuchen Sie, sich die Auswirkung der Prioritäten auf die Abarbeitungsreihenfolge des Computers an dieser Rechnung zu verdeutlichen:

```
((2 + 3) * 20) + 5 - 2^3 + 2 * 11 + 7 + 36 / 3
                     !
((2 + 3) * 20) + 5 -  8  + 2 * 11 + 7 + 36 / 3
                        !
((2 + 3) * 20) + 5 -  8  +   22   + 7 + 36 / 3
                                    !
((2 + 3) * 20) + 5 -  8  +   22   + 7 +   12
   !
(  5    * 20) + 5 -  8  +   22   + 7 +   12
     !
   100    + 5 -  8  +   22   + 7 +   12
          !
   105    - 8  +   22   + 7 +   12
          !
    97      +   22   + 7 +   12
            !
   119          + 7 +   12
                !
   126      +   12
            !
   138
```

Ebenso wichtig ist bei der Eingabe von Zahlen die korrekte Schreibweise der Zeichen "Null" und "Komma".

0 (Null)
O (großes "o")
. (Dezimalpunkt; Bitte kein Komma verwenden!)

Da - wie allgemein bekannt - ein Computer nicht nur mit Zahlen, sondern auch mit Texten umgehen kann, versuchen wir die Zahlen durch einen Text zu ersetzen.

Tippen Sie ein:
PRINT ERWIN **<ENTER>**

Der Computer zeigt 0 an! Sie haben vom Computer verlangt, Erwin zu zeigen, ohne dem Computer jemals mitgeteilt zu haben, wer oder was Erwin ist. Wir sollten uns als erstes darüber klar werden, was der Computer mit der angezeigten 0 meint.
Dazu müssen wir uns über das Thema

VARIABLE

Gedanken machen. Variable sind Zahlen, denen jeweils ein bestimmter Name zugeordnet worden ist. Diesen Namen braucht man, um später die entsprechenden Werte wiederzufinden. Variable lassen sich mit Schubladen vergleichen, deren Inhalt aus Zahlenwerten besteht.

Der Computer hat also durch den Befehl

PRINT ERWIN

in seinem Speicher (vergleichbar mit einer Wand voller Schubladen) die Schublade ERWIN vergebens gesucht:
Er hat nur leere unbeschriftete Schubladen gefunden und darum 0 angezeigt!

Sie tippen ein:		Ihr Computer zeigt:
ERWIN = 15	**<ENTER>**	
PRINT ERWIN	**<ENTER>**	**15**

Diesen Vorgang kann man sich folgendermaßen verdeutlichen:
ERWIN = 15 stellt keine Gleichung dar, sondern eine sog.

VARIABLENZUWEISUNG.

Bitte versuchen Sie, eine mathematische Gleichung und die Variablenzuweisung auseinanderzuzuhalten. Verwechseln Sie diese beiden Sachen nicht! Während eine Gleichung aussagt, daß die beiden Teile links und rechts vom Gleichheitszeichen identisch sind, wird bei einer Variablenzuweisung dem Namen links vom Gleichheitszeichen der Zahlenwert rechts vom Gleichheitszeichen zugewiesen.
Der wohl wichtigste Unterschied besteht darin, daß eine mathematische Gleichung jederzeit umkehrbar ist, ohne ihre Gültigkeit zu verlieren. Eine Variablenzuweisung dagegen ist nur in einer Form gültig, so daß die beiden Seiten links und rechts vom Gleichheitszeichen nicht beliebig vertauscht werden können.

Mit unserem Schubladenbeispiel verglichen, würde eine Gleichung bedeuten, daß der Inhalt zweier Schubladen völlig identisch ist. Die Variablenzuweisung hingegen stellt das Auffüllen einer Schublade mit einem bestimmten Inhalt dar.
Bei unserem letzten Beispiel ist Ihr Computer folgendermaßen vorgegangen:
Er hat sich eine der freien Schubladen genommen (er sucht sie sich selbst aus), sie mit dem Etikett ERWIN beschriftet und die Zahl 15 hineingelegt.
Durch den Befehl PRINT ERWIN haben Sie ihn dazu veranlaßt, diejenige Schublade zu suchen, die mit dem Namen ERWIN bezeichnet ist, und Ihnen den Inhalt auf dem Bildschirm zu zeigen.

Sie tippen ein:		Ihr Computer zeigt:
ERWIN = 16	**<ENTER>**	
PRINT ERWIN	**<ENTER>**	**16**

Sie haben als erste Anweisung Ihrem Computer ERWIN = 16 eingegeben, worauf er die Schublade mit dem Namen ERWIN gesucht hat. Hätte eine Schublade mit diesem Namen noch nicht existiert, so hätte er eine neue, noch leere, Schublade mit diesem Namen beschriftet. Da von unserer letzten Variablenzuweisung her eine Schublade mit diesem Namen bereits existiert, ist der Computer in diesem Falle einen etwas anderen Weg gegangen:

Der Computer hat eine Schublade mit dem Namen ERWIN gefunden, in der jedoch von unserer letzten Variablenzuweisung her noch die Zahl 15 gelegen ist. Diese 15 hat er aus der Schublade entfernt, und - unserer letzten Variablenzuweisung entsprechend - die Zahl 16 hineingelegt. Der Befehl PRINT ERWIN hat ihn wieder dazu veranlaßt, den Inhalt der Schublade mit dem Namen ERWIN auf dem Bildschirm anzuzeigen. Da unsere Schublade inzwischen jedoch einen neuen Wert beinhaltet, wurde natürlich dieser neue Wert (16) auf dem Bildschirm angezeigt.

Sie tippen ein:		Ihr Computer zeigt:
NEW	**<ENTER>**	**Ready**
PRINT ERWIN	**<ENTER>**	**0**

Der Computer zeigt wieder 0 an! Der Befehl

NEW

bedeutet demnach: ALLES NEU.
Es werden also alle Speicherinhalte und Programme gelöscht. Mit unserem Schubladenbeispiel verglichen bedeutet dies, daß sowohl alle Schubladen ausgeleert werden, als auch sämtliche Namen von den Etiketten auf den Schubladen gelöscht werden.

Sie tippen ein:		Ihr Computer zeigt:
NEW	**<ENTER>**	**Ready**
KILOMETER = 640	**<ENTER>**	**Ready**

WICHTIG: Bitte denken Sie daran, daß das Komma bei Dezimalzahlen beim Computer immer als "." (Punkt) geschrieben wird.

LITER = 48	**<ENTER>**	
PRINT LITER / KILOMETER * 100	**<ENTER>**	**7.5**

Die letzte vom Computer angezeigte Zahl ist der Verbrauch eines Kraftfahrzeugs pro 100 Kilometer bei gefahrenen 640 Kilometern und getankten 48 Litern.
Die von uns verwendeten Variablennamen haben zwar den Vorteil, daß sofort klar wird, welche Zahlenwerte sie repräsentieren. Sowohl vom Speicherplatzverbrauch, als auch von der Geschwindigkeit beim Eintippen her bietet sich jedoch die Verwendung möglichst kurzer Variablennamen an.

Sie tippen ein:		Ihr Computer zeigt:
K = 700	**<ENTER>**	**Ready**
L = 57.75	**<ENTER>**	**Ready**
V = L / K * 100	**<ENTER>**	**Ready**
PRINT V	**<ENTER>**	**8.25**

Bisher haben wir dem Computer "Schritt für Schritt" mitgeteilt, was er für uns tun soll. Wir haben also in dem bereits erwähnten DIREKTMODUS gearbeitet. Die Wirtschaftlichkeit von Computersystemen jedoch liegt darin begründet, daß Sie sich wiederholende Aufgabenstellungen in hoher Geschwindigkeit immer wieder mit neuen Eingabedaten durchrechnen können.

Dieses Ziel werden wir durch die Eingabe von Befehlen im Direktmodus nicht erreichen können:
Wir müssen hierfür die Befehle in Form eines

PROGRAMMES

eingeben. Ein Programm besteht aus mehreren Befehlen, die in einer vorgegebenen Reihenfolge ausgeführt werden. In BASIC arbeitet man mit sog.

PROGRAMMZEILEN.

Jede Programmzeile hat am Anfang eine Nummer. Die Programmzeilen werden in der Reihenfolge der aufsteigenden Nummern aufgeführt.

Bitte tippen Sie ein:

```
1 K = 325                    <ENTER>
2 L = 41.11                  <ENTER>
3 V = L / K * 100            <ENTER>
4 PRINT V                    <ENTER>
```

Sie werden schon bemerkt haben, daß im Programmodus - im Gegensatz zum Direktmodus - nach der Eingabe einer Programmzeile die Antwort "Ready" vom Computer nicht mehr angezeigt wird!

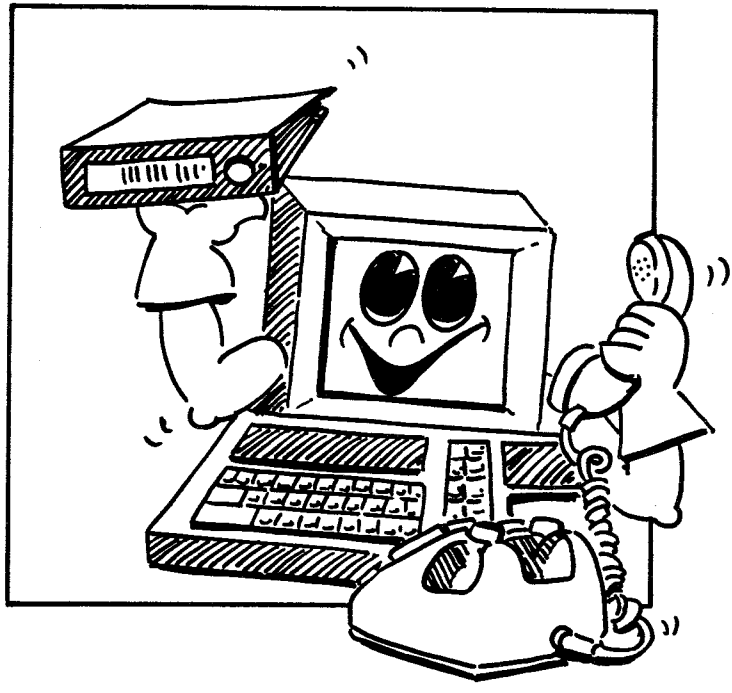
Der Start des Programmes erfolgt durch die Eingabe von

RUN.

Bitte tippen Sie ein:

```
RUN                          <ENTER>
```

STUFE 3 ICH LASSE MEINEN COMPUTER ARBEITEN!



Teil 1: Aufbau eines Programmes

Zunächst halten Sie sich bitte vor Augen, daß ein Computer ganz präzise Befehle verlangt, und zwar in einer Sprache, die er versteht. Dafür arbeitet er dann auch genau und fehlerfrei. Als Mensch ist man zwar üblicherweise etwas "schlampiger", doch Sie haben sich in der Zwischenzeit bestimmt schon angewöhnt, Befehle sorgfältig einzugeben, denn dies ist eine grundsätzliche Voraussetzung fürs Weiterkommen.
Daß sich aber Ungenauigkeit (z.B. ein falsch gesetztes Komma) auch im alltäglichen Leben verheerend auswirken kann, wird durch eine - aus den Pionierzeiten Amerikas - mündlich überlieferte Anekdote sehr gut verdeutlicht:
Ein Rechtsanwalt stellte zu dieser Zeit an den Gouverneur des entsprechenden Staates ein Begnadigungsgesuch, betreffend eines seiner Mandanten. Dieser Mann saß seit Wochen im staatlichen Gefängnis und sollte innerhalb der nächsten 24 Stunden gehängt werden. Ihm wurde vorgeworfen, die Schwester der Telegraphenbeamtin aus Eifersucht getötet zu haben. Es konnte doch niemand mit Gewissheit nachweisen, daß er tatsächlich der Täter war. Einige Stunden vor der Vollstreckung des Urteils erreichte das langersehnte Telegramm des Gouverneurs den kleinen Ort des Schauspiels. Der Inhalt lautete:

BEGNADIGT IHN NICHT, HÄNGEN.

Ob sich die Telegraphenbeamtin - welche übrigens der Überzeugung war, daß der verurteilte Mann auch wirklich der Täter sei - verhört oder nur vertippt hat, konnte nie endgültig geklärt werden. Der Text des vom Gouverneur tatsächlich abgesandten Antwortschreibens aber hat folgendermaßen gelautet:

BEGNADIGT IHN, NICHT HÄNGEN.

Nachdem Sie nun gesehen haben, wie wichtig selbst ein Komma sein kann, begeben wir uns mit diesem frisch gestärkten Bewußtsein für Genauigkeit wieder in die Untiefen der Programmiersprache BASIC.

Wie ist ein BASIC-Programm aufgebaut?

Bitte legen Sie neben sich irgend ein gedrucktes BASIC-Programm. Auf den ersten Blick besteht es nur aus einem unübersichtlichen Haufen verschiedenster Befehle.
Auffallend sind zunächst nur die Zahlen am Anfang jeder Programmzeile. Sie heißen ihrer Funktion entsprechend

ZEILENNUMMERN.

Sie geben die Reihenfolge an, in der die Befehle ausgeführt werden sollen. Der Computer beginnt mit der Zeile, die die kleinste Zeilennummer aufweist. Diese Befehle werden vom Computer

interpretiert und dann ausgeführt. Danach wird die Zeile mit der nächst höheren Zeilennummer abgearbeitet usw. Dabei ist es unwichtig, in welcher zeitlichen Reihenfolge die Zeilennummern eingegeben worden sind. Der Computer ordnet die Zeilen in aufsteigender Reihenfolge und führt danach erst die Befehle aus, wenn das Programm mit RUN gestartet wird.

Damit können wir sagen: Ein Programm ist eine Anzahl von Befehlen, die in einer vorbestimmten Reihenfolge ausgeführt werden.
Es dürfen hinter einer Zeilennummer auch mehrere Befehle stehen. Diese werden dann nacheinander von links nach rechts ausgeführt. Die Reihenfolge ist also wiederum festgelegt.
Zeilennummern sind aber auch Markierungen. Ein BASIC-Befehl, den wir in Teil 2 noch kennenlernen werden, kann im Klartext lauten: "Spring zur Zeile 100 und arbeite dort weiter. Vergiß, wo Du hergekommen bist!" In BASIC schreibt man dafür einfach "GOTO 100".

Aufbau eines Programmes in anderen Programmiersprachen

Ein langes BASIC-Programm wirkt unübersichtlich, wenn es nicht gegliedert ist. In den meisten BASIC Dialekten hat man aber auch keine Gelegenheit, die Programme zu strukturieren. Das Schneider BASIC stellt hier eine rühmliche Ausnahme dar:
Befehle wie IF THEN ... ELSE ... , oder die sogenannten DO-LOOPS ermöglichen es, ständiges Herumspringen im Programm durch GOTO-Befehle zu vermeiden, und dadurch das Programmlisting übersichtlicher zu gestalten. Trotzdem können die meisten komplexeren BASIC-Programme meist nur von demjenigen verstanden werden, der sie geschrieben hat. Es gibt deshalb andere Programmiersprachen, welche eine Gliederung ähnlich einer Hierarchie geradezu erzwingen.

PASCAL

ist so eine Sprache und wohl die Extremste in dieser Hinsicht. Zeilennummern werden in PASCAL und verwandten Sprachen nicht benötigt, sind aber möglich (das gilt für die meisten von der Industrie verwendeten Sprachen, z.B. ALGOL, Stammvater von PASCAL). In solchen Sprachen werden die Programme "einfach " von vorne nach hinten abgearbeitet. Mit

BASIC

ist man aber als Nicht-Profi gut beraten. Diese Sprache ist einfach zu erlernen und ermöglicht schnelle Erfolge. Die Handhabung ist also denkbar einfach. Erst wenn man andere Sprachen probiert hat, lernt man dies zu schätzen. Sie brauchen sich der Programmiersprache BASIC aber nicht zu schämen, denn es gibt auch viele Profis, die fast ausschließlich in BASIC programmieren.

Teil 2: BASIC-Grundbefehle

Ein uns bereits bekannter und häufig angewandter Befehl ist

PRINT,

was soviel bedeutet wie "schreibe"! Mit PRINT kann man Worte oder Zahlen auf den Bildschirm schreiben. Will man einen Text schreiben, so muß er in Anführungszeichen ("") gesetzt werden.

PRINT "Nur Mut." ist so ein Schreibbefehl.

Er kann in einem Programm verwendet werden, wenn man ihm eine Zeilennummer voranstellt.
Es läßt sich aber auch der Inhalt einer Variablen mit diesem Befehl ausgeben. Z.B.: PRINT A oder PRINT A$.
Variable in ihrer ganzen Bedeutung begriffen zu haben, ist ein großer Schritt vorwärts, meist aber auch ein schwieriger. Deshalb soll hier nochmals auf eine andere Art versucht werden, Variable zu beschreiben:
Angenommen, Sie haben ein Programm, das von Ihnen eine Zahl erfragt, diese Zahl verdoppelt und das Ergebnis hinschreibt, dann könnte es etwa so ablaufen:

1. Befehl:Zahl erfragen
2. Befehl:Zahl verdoppeln
3. Befehl:Ergebnis hinschreiben.

Bleiben wir beim 1. Befehl. Die Zahl, die erfragt wurde, muß vom Computer irgendwo gespeichert werden. Er braucht sie ja noch für den nächsten Befehl. Deshalb gibt man der Zahl einen Namen, z.B. "ZA" oder "A" oder "ZAHL" oder sonst etwas. Nehmen wir "A".

"A" ist nun eine Variable.

Der 2. Befehl könnte nun lauten: Verdopple "A". Für den Computer ist dieses "A" eine Zahl. Also kann er "A" verdoppeln. Das Ergebnis wird mit dem 3. Befehl ausgegeben. Anschliessend beginnt man wieder von vorne. Eine neue, andere Zahl wird erfragt. Sie bekommt wieder den Namen "A". Das "A" steht jetzt für diese neue Zahl; die Zahl von vorhin ist vergessen. Dieses neue "A" wird wiederum verdoppelt und ausgegeben.
Das läßt sich mit vielen Zahlenwerten wiederholen, wobei jedesmal "A" für eben diesen Zahlenwert steht, also variabel ist. Deshalb wird "A" als Variable bezeichnet.

Textvariable müssen von Zahlenvariablen unterschieden werden können. Man kennzeichnet sie deswegen mit einem "$". Also sind etwa A$, AD$, BR$ usw. nur Textvariable.

Folgendes Programm leistet dasselbe wie unser Übungsbeispiel im Lehrprogramm:

```
1 A$="Nur Mut"
2 B$="Es wird schon klappen"
10 PRINT A$
20 PRINT B$
```

(Daß Sie nach jeder Zeile **"ENTER"** drücken müssen, wissen Sie ja bereits!)

Geben Sie auch dieses Beispiel einmal in ihren Computer ein und überzeugen Sie sich, daß es läuft.

Nun kommen wir zu ein paar Befehlen, die meist außerhalb eines Programmes direkt eingegeben werden. Sie gehören zu den meistverwendeten und wichtigsten Befehlen. Man bezeichnet Sie als sogenannte Systembefehle, weil sie grundsätzlich nicht in Programmen vorkommen sollen, sondern Befehle zur **Steuerung des Programmablaufes** sind.

RUN

Dieser Befehl startet das BASIC-Programm, das sich gerade im Speicher befindet. Sie verwenden diesen Befehl auch, um das Lehrprogramm zu starten. Ist kein Programm im Speicher, so meldet sich der Computer sofort wieder mit "READY". Eine Fehlermeldung wird nicht ausgegeben.
Tippen Sie irgendein Programm ein, löschen Sie dann den Bildschirm (mit dem Befehl **"CLS"**) und geben dann den Befehl

LIST

ein. Das ganze eingetippte Programm erscheint wieder auf dem Bildschirm. Der Computer hat es also gespeichert, auch wenn es vom Bildschirm gelöscht wurde. Für Variable gilt dasselbe. Sie werden ebenfalls gespeichert, bis sie durch einen Befehl gelöscht werden.
Merke:
Der "RUN"-Befehl löscht alle Variablen, bevor das Programm gestartet wird.

NEW

Dieser Befehl (NEU) hat nur die Funktion des Löschens. Er löscht sowohl alle Variablen als auch das im Computer gespeicherte BASIC-Programm. Der Speicher ist nach Eingabe von "NEW" völlig leer.

DELETE

Auch mit diesem Befehl kann der Programmspeicher gelöscht werden. Der Unterschied zum NEW-Befehl besteht darin, daß nur bestimmte Teile des Programmspeichers gelöscht werden können.
Haben Sie also folgendes Programm im Speicher:

```
10 PRINT "Zeile 10"
20 PRINT "Zeile 20"
30 PRINT "Zeile 30"
40 PRINT "Zeile 40"
50 PRINT "Zeile 50"
```

dann können Sie die Zeilen 20 bis einschließlich 40 mit dem Befehl DELETE 20-40 löschen. Im Speicher befinden sich nach dem Ausführen dieses Befehls (bitte ENTER nicht vergessen) nur noch die Zeilen 10 und 50:

Sie tippen ein:	Ihr Computer zeigt:
LIST	**10 PRINT "Zeile 10"** **50 PRINT "Zeile 50"** **Ready**

Selbstverständlich wäre es auch möglich gewesen, das gesamte Programm durch DELETE 10-50 zu löschen. In diesem Fall ist jedoch der NEW-Befehl einfacher.

Teil 3: Daten Intern

Alle Eingaben an den Computer werden als

Daten

bezeichnet. Die Einzahl von Daten ist ein Datum. Natürlich handelt es sich sowohl bei den Ausgaben, die wir vom Computer erhalten, als auch bei den zu verarbeitenden Texten und Zahlen um Daten.
Wie eben erwähnt, treten Daten grundsätzlich in zwei verschiedenen Formen auf:
Entweder als

numerische Daten

oder als

alphanumerische Daten.

Numerische Daten bestehen aus den Zahlen 0 bis 9 (einschließlich mathematischer Sonderzeichen), alphanumerische Daten setzen sich aus den Zahlen 0 bis 9, den Buchstaben des Alphabetes A bis Z und den Sonderzeichen zusammen.
Sowohl die numerischen, als auch die alphanumerischen Daten können als Konstanten, oder als Variable vorkommen.

Konstanten

sind unveränderbare Daten. Sie werden einmal eingegeben, und verändern sich im weiteren Programmverlauf nicht mehr.

Variable

dagegen - auch Platzhalter genannt - machen genau das, was der deutsche Name auch andeutet: Sie halten einen Platz frei, den Sie beliebig belegen können.
Der Befehl PRINT"1985" weist den Computer an, die Konstante "1985" auf dem Bildschirm auszugeben. Geben Sie stattdessen PRINT Jahr ein, so ist "Jahr" der Name einer numerischen Variable, die den ihr zugewiesenen - jederzeit veränderbaren - Wert enthält. Weitere Beispiele finden Sie auf Ihrer Programmdiskette in Stufe 3 Teil 3.

Während die Konstanten eine einheitliche Gruppe darstellen und nicht weiter unterteilt werden müssen, unterscheidet man bei den Variablen drei verschiedene Formen voneinander:
Da wären zuerst die uns bestens bekannten Textvariablen, auch

Stringvariable

genannt. Sie enthalten alphanumerische Daten.
Um numerische Daten unterzubringen, stehen uns zwei Möglichkeiten zur Verfügung: Die

Fließkommavariable

ist die am häufigsten verwendete Art der Zahlendarstellung. Hierbei handelt es sich um Zahlen mit einer beliebigen Anzahl von Dezimalstellen hinter dem Komma. Im Computerjargon werden sie auch "floating point" Variable genannt, was im Grunde genommen nur die englische Form des deutschen Ausdrucks darstellt.
Die zweite Art numerischer Variablen ist die

Ganzzahlvariable.

Wie der Name schon sagt, kann sie nur ganze Zahlen beinhalten. Ebenfalls vom Englischen abgeleitet ist der Fachausdruck hierfür: "Integervariable".
Damit Ihr Computer weiß, welchen Variablentyp Sie verwenden wollen, hat man sich auf folgende Unterscheidungsmerkmale geeinigt:
Die Textvariable wird durch das "$"-Zeichen kenntlich gemacht. Bei Ganzzahlvariablen benutzt man das "%"-Zeichen. Ist ein Variablenname nicht besonders gekennzeichnet, so handelt es sich automatisch um eine Fließkommavariable.
Denken Sie bitte daran, daß diese Unterscheidungsmerkmale nie für sich allein stehen können, sondern an den eigentlichen Variablennamen angehängt werden müssen. Durch diese verschiedenen Kennzeichnungen bedingt handelt es sich dann auch um drei voneinander vollkommen unabhängige Variable.
Wenn Sie eine Schublade z.B. mit dem Namen "Werkzeug" versehen, so wählt der Computer eine noch unbenutzte Schublade aus und beschriftet sie mit diesem Namen. Erzeugen Sie jetzt eine weitere Variable, welcher Sie den Namen "Werkzeug%" geben, so wird nicht etwa das Etikett der ersteren Schublade umgeschrieben, sondern eine neue - noch leere - Schublade mit diesem Namen versehen.
Wie Sie sehen, können Sie mit ein und demselben Namen drei voneinander unabhängige Schubladen beschriften. Es muß sich hierbei jedoch auch um drei verschiedene Schubladenarten handeln.
Versehen Sie dagegen eine Schublade nacheinander mit zwei verschiedenen Inhalten, so werden Sie feststellen, daß die Schublade jeweils nur über den zuletzt eingegebenen Inhalt verfügt: Wenn Sie nämlich einer bereits gefüllten Schublade einen neuen Inhalt zuweisen, so wird erst der alte Inhalt gelöscht und die Schublade anschließend mit dem neuen Inhalt versehen.
Den Vorgang, durch den Sie eine Schublade mit einem Inhalt versehen, nennt man

Variablenzuweisung.

Wollen Sie also z.B. eine Schublade mit dem Inhalt "150" und dem Namen "Zahl", so tippen Sie in Ihren Computer "Zahl=150".

Dies sieht zwar aus wie eine Gleichung, darf mit derselben aber unter keinen Umständen verwechselt werden, da im Gegensatz zu einer Gleichung die Variablenzuweisung nicht umkehrbar ist:
Die Variablenzuweisung "150=Zahl" würde nämlich bedeuten, daß im Gegensatz zum oben genannten Beispiel diesmal die Variable mit dem Namen "150" den Inhalt "Zahl" zugewiesen bekommt. Es ist offensichtlich, daß es sich hierbei um zwei verschiedene Fälle handelt. Ganz abgesehen von dem eben Gesagten, handelt es sich um ein rein theoretisches Beispiel, da Ihr Computer den Variablennamen "150" nicht akzeptieren würde.
Ein grundsätzlicher Unterschied zwischen alphanumerischen Textvariablen und numerischen Fließkomma- bzw. Ganzzahlvariablen besteht darin, daß Sie mit Textvariablen nicht rechnen können. Numerische Variable wiederum können nicht manipuliert werden. Eine Art der Variablenmanipulation stellt z.B. die sog. Stringaddition dar. Ein Beispiel dafür finden Sie auf Ihrer Programmdiskette in Stufe 3 Teil 3.
Je nach Problemstellung kann es jedoch sinnvoll sein, einerseits mit numerischen Variablen zu rechnen und andererseits alphanumerische Variable zu manipulieren. Damit Sie je nach Bedarf zwischen diesen beiden Formen wählen können, verfügt Ihr Schneider Computer über zwei Funktionen, mit denen Sie die jeweils notwendige Umwandlung vornehmen können:

VAL(Textausdruck)

wandelt den in Klammern stehenden Textausdruck in einen numerischen Ausdruck.

STR$(numerischer Ausdruck)

wandelt den in Klammern stehenden numerischen Ausdruck in einen Textausdruck.
Das folgende Beispiel zeigt Ihnen, wie Sie mit Hilfe dieser Funktionen Textvariable in numerische Variable und umgekehrt wandeln können:

Zuerst initialisieren wir zwei Variable, d.h. wir beschriften zwei Schubladen und versehen sie mit einem Inhalt:

```
a=19                <ENTER>
Ready
b=85                <ENTER>
Ready
```

Als nächstes wandeln wir die Fließkomma- (also numerische) Variable in Textvariable:

```
a$=STR$(a)          <ENTER>
Ready
b$=STR$(b)          <ENTER>
Ready
```

Jetzt vollziehen wir eine Stringaddition. Wie bereits oben erwähnt, handelt es sich hierbei nicht um eine mathematische Operation, sondern um eine Variablenmanipulation. Der Unterschied ist so offensichtlich, daß sich jedes weitere Wort erübrigt.

```
c$=a$+b$            <ENTER>
Ready
PRINT c$            <ENTER>
1985
Ready
```

Wollen wir nun zu 1985 die Zahl 17 dazuaddieren, so geht das nur, wenn wir die Textvariable c$ vorher in eine numerische Variable umwandeln:

```
PRINT c$+17         <ENTER>
Type mismatch
Ready
PRINT VAL(c$)+17    <ENTER>
2002
Ready
```

Wenn Sie den Wert von c$ für weitere Berechnungen noch öfters brauchen sollten, so empfiehlt sich folgende Vorgehensweise:

```
c=VAL(c$)           <ENTER>
Ready
PRINT c+17          <ENTER>
2002
Ready
```

STUFE 4 NOCH MEHR ARBEIT FÜR MEINEN COMPUTER

Teil 1: Weitere BASIC-Befehle

Im Verlauf dieses Buches ist uns schon ein paar Mal der Befehl

GOTO

begegnet. Er leitet sich ab von dem englischen "go to", was wortwörtlich übersetzt soviel heißt wie "geh nach ...!". Dieser Befehl wird vor allem in Programmen verwendet, kann jedoch auch im Direktmodus eingegeben werden.

Tippen Sie ein:

```
10 PRINT "Zeile 10"
20 GOTO 40
30 PRINT "Zeile 30"
40 PRINT "Zeile 40"
50 PRINT "Zeile 50"
```

Wenn Sie das Programm starten, können Sie folgendes beobachten: Der Schreibbefehl in Zeile 30 wird offensichtlich nicht ausgeführt. Der Grund dafür ist, daß der Computer in Zeile 20 den Befehl erhalten hat, nach Zeile 40 zu springen und dort weiterzuarbeiten. Es ist sehr wichtig, daß Sie diesen Befehl verstehen, da er sehr grundlegend ist.

Zum Beenden bzw. Anhalten des Programmes stehen Ihnen die Befehle

END und STOP

zur Verfügung. Diese Befehle werden ausschließlich innerhalb eines Programmes verwendet.
Der END-Befehl teilt dem Computer mit, daß das Programm beendet sei. Er bricht daraufhin die Bearbeitung des laufenden Programmes ab, unabhängig davon ob noch weitere Programmzeilen folgen oder nicht. Den Abbruch der Bearbeitung erkennen Sie an der Rückmeldung des Computers durch "Ready". Er teilt hierdurch mit, daß alles von ihm Verlangte ordnungsgemäß abgearbeitet und beendet ist.
Wenn von Ihrem Programmaufbau der END-Befehl in der letzten Programmzeile (also am Ende Ihres Programmes) stehen würde, können Sie ihn genauso gut weglassen. Er wird jedoch unumgänglich, sobald Sie mit Unterprogrammen arbeiten. Weitere Erläuterungen hierzu entnehmen Sie bitte Stufe 4 Teil 2, bei der Erklärung der Unterprogrammbefehle GOSUB und RETURN. (Seite 59)

Auch beim STOP-Befehl wird die Bearbeitung des laufenden Programmes abgebrochen; im Gegensatz zum END-Befehl jedoch nur vorläufig. Wurde im Programm ein STOP-Befehl angetroffen, so meldet sich Ihr Computer mit "Break in ..." und "Ready", wobei in der ersten Meldung anstatt der Punkte diejenige Zeilennummer steht, in welcher der STOP-Befehl angetroffen worden ist. Wenn Sie mit der Programmbearbeitung fortfahren wollen, benutzen Sie den

CONT

Befehl. "CONT" ist die Abkürzung, die Ihr Computer für das englische Verb "continue" benutzt. Auf Deutsch bedeutet das "fahre fort!". Sie sehen also, daß sich Ihr Computer im Gegensatz zum END-Befehl nur vorläufig aus der Programmbearbeitung zurückzieht. Wenn Sie eine sogenannte

Mehrbefehlszeile

erstellen wollen, so müssen Sie die einzelnen Befehle durch Doppelpunkte voneinander trennen. Das folgende Beispiel leistet dasselbe, wie das bereits behandelte Programm, welches wir bereits in Stufe 3 Teil 2 behandelt haben:

```
10 A$="Nur Mut":B$="Es wird schon klappen":PRINT A$:PRINT B$
```

Sie finden dieses Programm (auf vier Programmzeilen verteilt) auch in Ihrem Handbuch, wenn Sie auf Seite 46 zurückblättern.
Zwei weitere, fast unentbehrliche, Befehle erblicken wir in dem folgenden Programm:

```
10 INPUT"Mögen Sie Computer (j/n) ";Y$
15 PRINT
20 IF Y$="j" THEN 40
30 IF Y$="n" THEN 50
35 PRINT"Unzuläßige Eingabe"
37 PRINT:RUN
40 PRINT"Das freut mich!"
45 END
50 PRINT"Schade."
```

Dieses Beispiel entspricht nicht dem Beispiel im Lehrprogramm.
Der PRINT-Befehl in Zeile 15 bewirkt eine Leerzeile, was die Übersichtlichkeit bei der Ausgabe verbessert. In Zeile 37 wird zunächst wieder eine Leerzeile auf den Bildschirm gebracht und anschließend das Programm neu gestartet.
Anstelle von RUN könnte man hier auch GOTO 10 einsetzen, denn auch dieser Befehl würde bewirken, daß das Programm von vorne beginnt. RUN löscht jedoch zusätzlich alle Variablen.
Nach Zeile 50 haben wir den END-Befehl weggelassen, weil keine weiteren Befehle folgen. Der Computer beendet in diesem Fall das Programm von selbst, auch ohne diese Anweisung.

IF ... THEN ...

Dieser Befehl (wenn ... dann ..) ist durchaus wörtlich zu nehmen. Zeile 30 lautet ins Deutsche übersetzt etwa so: Wenn "j" (für ja) eingegeben wurde, dann gehe zur Zeile 40. Sonst mache mit dem nächsten Befehl (also Zeile 35) weiter.

Der

INPUT

Befehl ermöglicht es, einen Text ("ja","nein","j","Chaos"...) in die Variable Y$ einzulesen. Würde hinter dem INPUT-Befehl die numerische Variable Y stehen, so könnten in diese Variable selbstverständlich nur Zahlenwerte eingelesen werden.
Starten Sie dieses Programm mehrmals und testen Sie es mit verschiedenen Eingaben! Wenn Sie es verstanden haben, kennen Sie die Art und Weise, wie ein Computer seine "Entscheidungen" trifft.
Mit dem bisher Gelernten lassen sich bereits recht gute Programme schreiben. Es gibt aber noch weitere, sehr praktische Befehle:

Der

FOR ... NEXT ...

Befehl wird verwendet, wenn der Computer ein und dieselbe Sache mehrmals ausführen soll:

```
10 Zahl=7
20 FOR i=1 TO Zahl
30 PRINT i "auf einen Streich!"
40 NEXT i
```

Statt 7 darf die Variable "Zahl" natürlich auch irgendeine andere Zahl sein. Der Computer wird den Text entsprechend oft ausgeben. Vor dem Text sehen Sie noch ein "i". Dieses "i" sagt dem Computer, daß er den augenblicklichen Wert der Variablen i schreiben soll. Wenn Sie das Programm starten, sehen Sie, daß sich der Wert von i jedesmal um eins erhöht. Sobald der Wert "Zahl" (=Endwert der Schleifenvariable (-1!) = Anzahl der Schleifendurchläufe = Wert (Inhalt) der Variable mit dem Namen "Zahl") erreicht ist, bricht das Programm die Wiederholungen ab. Den Abschnitt von Zeile 10 bis 30 nennt man eine

Programm-Schleife.

Die Variable "i" nennt man eine

Schleifen-, bzw. Laufvariable.

1 ist der Startwert, der **Anfangswert** der Laufvariablen. Der Wert (Inhalt) der Variablen "Zahl" ist der entsprechende **Endwert**, der Höchstwert der Laufvariablen. Probieren Sie verschiedene Anfangs- und Endwerte aus und machen Sie ein paar Probeläufe. Um sich die

Funktion einer FOR ... NEXT ... - Schleife genau vor Augen zu führen, können Sie sich das Beispiel auf Ihrer Programmdiskette in Stufe 4 Teil 1 Seite 18 anschauen.
Eine andere Art von Programmschleifen stellt Ihnen der

WHILE ... WEND

Befehl zur Verfügung. Ähnlich der FOR ... NEXT ... - Schleife wiederholt sich auch hier der "Schleifeninhalt" solange, bis die Abbruchbedingung dieser Schleife erfüllt ist. Die Abbruchbedingung einer FOR ... NEXT ... - Schleife ist dann erreicht, wenn die Laufvariable den vorgegebenen Schleifen-Endwert erreicht hat. Die Abbruchbedingung einer WHILE ... WEND - Schleife dagegen kann verschieden definiert werden. Immer handelt es sich hierbei jedoch um einen sogenannten **logischen Ausdruck.** Ein solcher logischer Ausdruck ist gekennzeichnet durch eine der folgenden **mathematischen Relationen:**

```
... <  ...      (... kleiner ...)
... <= ...      (... kleiner oder gleich ...)
... <> ...      (... ungleich ...)
... =  ...      (... ist gleich ...)
... >= ...      (... größer oder gleich ...)
... >  ...      (... größer ...)
```

Auch die Funktionsweise dieser Schleife können Sie Ihrer Programmdiskette in Stufe 4 Teil 1 Seiten 19 ff. entnehmen.

Eine ebenfalls sehr nützliche Funktion ist

INKEY$.

Sie ist mit dem INPUT-Befehl insofern verwandt, als dem Benutzer auch hier die Möglichkeit gegeben wird, vom Programm zu verarbeitende Daten mittels der Tastatur einzugeben. Soviel zu den Gemeinsamkeiten. Die gesammte Bandbreite der Unterschiede anzuführen, würde den Rahmen dieses Kurses jedoch sprengen. Zum Verständnis aber ist es ausreichend, auf die Auswirkungen dieser Funktion näher einzugehen:
Diese Funktion fragt 50 mal in der Sekunde die Tastatur Ihres Computers ab und speichert das Ergebnis dieser Abfrage in INKEY$. (Es ist zwar etwas verwirrend, daß die Funktion den selben Namen trägt, wie die Variable, in der das Ergebnis der Tastaturabfrage abgespeichert wird. Versuchen Sie es nicht zu verstehen, nehmen Sie es einfach hin!). Da jedoch eine 50-tel Sekunde später bereits die nächste Tastaturabfrage erfolgt, wird das alte Ergebnis sofort wieder überschrieben. Um dies zu umgehen, weisen wir in Zeile 20 der Variablen Y$ den Wert von INKEY$ solange zu, bis eine Taste gedrückt wird, INKEY$ also nicht mehr gleich "" (also leer) ist.

Wie Sie sehen, kann mit dieser Funktion jeweils nur ein Zeichen eingelesen werden. Dafür braucht der Benutzer nach der Eingabe auch nicht mehr die ENTER-Taste zu drücken, da die gedrückte Taste vom Computer sofort ausgewertet wird.
Im Folgenden sehen Sie ein Beispielprogramm, welches sich dieser Funktion bedient. In dieser oder ähnlicher Form werden Sie es in jedem BASIC-Spielprogramm finden können.

```
10 PRINT"Programm beenden? (j/n)"
20 y$=INKEY$:IF y$="" THEN 20
30 IF y$="j" THEN END
40 GOTO 10
```

In Zeile 10 wird ein Text auf dem Bildschirm ausgegeben, der die Aufforderungenthält,die"j"- oder "n"-Taste zu drücken.In Zeile 20 wird geprüft, ob eine Taste gedrückt worden ist. Hierzu wird zuerst das Ergebnis der Tastaturabfrage in Y$ gespeichert, und anschließend geprüft, ob Y$="" (also leer) ist. Wenn dies zutreffen sollte, so geht das Programm an den Anfang von Zeile 20 zurück und führt die Prüfung erneut durch. Das geht so lange, bis eine Taste gedrückt wird.
Danach wird in Zeile 30 verglichen, ob das Ergebnis der Tastaturabfrage "j" ist oder nicht. Fällt der Vergleich positiv aus (wurde also die "j"-Taste gedrückt), so wird der Befehl END ausgeführt und damit das Programm beendet. Bei jeder anderen Taste wäre die Bedingung y$="j" nicht erfüllt und das Programm würde weiter gehen zu Zeile 40, wo es neu gestartet wird.
Die Auswertung der Tastaturabfrage erfolgt also in Zeile 30 und kann nach Belieben verändert werden.
Die Zeile 20 dagegen bewirkt ein "Warten" des Programmes und ist somit eine sehr wichtige und häufig gebrauchte Funktion. Außerdem ist sie für den Benutzer bequemer als der INPUT-Befehl, denn er muß nur eine einzige Taste drücken, um dem Computer seine Entscheidung mitzuteilen.

Teil 2: Strukturierte Programme

Bereits im vorhergehenden Teil haben wir das Thema

Unterprogramme

angeschnitten. Sie sind ein wichtiges Hilfsmittel, um BASIC-Programme zu strukturieren und das Programm dadurch übersichtlicher zu gestalten.
Erinnern Sie sich noch an GOTO? Wir haben gesagt, daß dieser Befehl den Computer zu der angegebenen Zeile schickt, mit der weiter arbeitet und vergißt, wo er hergekommen ist.

Auch der Befehl

GOSUB

bewirkt, daß Ihr Computer zu einer neuen Zeile springt und dort weiterarbeitet. Diesmal merkt er sich jedoch die Stelle im Programm, von der er weggesprungen ist. Trifft er dann im weiteren Programmverlauf (nämlich im Unterprogramm, wo er sich jetzt befindet) ein

RETURN,

so kehrt er an diejenige Stelle zurück, von der er weggesprungen ist und macht mit dem nächsten Befehl weiter. Das könnte so aussehen:

```
10 PRINT"Ich bin in Zeile 10"
20 GOSUB 80
30 PRINT"Zeile 30"
40 GOSUB 80
50 PRINT"Zeile 50"
60 END
80 PRINT"Grüße aus dem Unterprogramm!"
90 RETURN
```

Ungewöhnlicherweise haben wir diesmal mitten im Programm einen END-Befehl. Würden wir aber die Zeile 60 löschen, erhielten wir vom Computer die Fehlermeldung "Unexpected RETURN in 90".
Der Grund hierfür ist leicht verständlich: Wie Sie wissen, orientiert sich Ihr Computer beim Abarbeiten eines BASIC-Programmes an den Zeilennummern. Ohne die Zeile 60 würde er nach dem Ausführen von Zeile 50 zur Zeile 80 und anschließend zur Zeile 90 springen. Hier würde er die Anweisung erhalten, aus einem Unterprogramm zurückzukehren, obwohl er sich gar nicht in einem Solchen befindet!
Bitte versuchen Sie - bevor Sie umblättern - herauszufinden, in welcher Reihenfolge Ihr Computer das oben stehende Programm abarbeitet!

Die Lösung lautet:
10 - 20 - **80** - **90** - 30 - 40 - **80** - **90** - 50 - 60

Unterprogramme benutzt man natürlich nicht nur, um BASIC-Programme zu strukturieren:
Das Unterprogramm in unserem Beispiel leistet zwar nicht sehr viel. Aber nehmen wir an, das Programm wäre viel länger und müßte an 100 im Programm verstreuten, Stellen einige im Unterprogramm zusammengefaßte Befehle ausführen. Dann ist so ein Unterprogramm bereits nützlich, weil man nur "GOSUB 80" an all diesen Stellen einfügt, anstatt die im Unterprogramm enthaltenen Befehle ständig wiederholen zu müssen.
Je länger ein Programm ist, desto wichtiger ist es, sich solcher Unterprogramme zu bedienen. Das verbessert nicht nur die Übersichtlichkeit, sondern spart auch Speicherplatz!
Zur Strukturierung von BASIC-Programmen können Sie aber auch die Befehle FOR ... NEXT ... und WHILE ... WEND verwenden, die Sie bereits im vorhergehenden Teil 1 gelernt haben. Die folgenden Gegenüberstellungen sind zur Darstellung der oben genannten Vorteile besser geeignet als viele Worte.

Programmbeispiel: **Speicherplatzbedarf:**

```
10 FOR i=1 TO 9                                  I
20 PRINT i"-ter Schleifendurchgang"              I
30 NEXT i                                        I 110 Bytes
40 PRINT"i="i", also Schleife beendet!"          I
50 END                                           I

10 i=1                                           I
20 IF i<=9 THEN 30 ELSE 60                       I
30 PRINT i"-ter Schleifendurchgang"              I 145 Bytes
40 i=i+1                                         I also ca.
50 GOTO 20                                       I 32% mehr!
60 PRINT"i="i", also Schleife beendet!"          I
70 END                                           I


10 a=0                                           I
20 WHILE a<=9                                    I
30 PRINT"a ist "a", also kleiner als 9"          I
40 a=a+1                                         I 137 Bytes
50 WEND                                          I
60 PRINT"a ist"a"damit größer als 9!"            I
70 END                                           I

10 a=0                                           I
20 IF a<=9 THEN 30 ELSE 60                       I
30 PRINT"a ist "a", also kleiner als 9"          I 154 Bytes
40 a=a+1                                         I also ca.
50 GOTO 20                                       I 12,5% mehr!
60 PRINT"a ist"a"damit größer als 9!"            I
70 END                                           I
```

Eine weitere Möglichkeit zur strukturierten Programmierung können Sie den obigen Beispielen entnehmen. Es handelt sich hierbei um den erweiterten IF ... THEN ... - Befehl.

IF ... THEN ... ELSE ...

kann genauso wortwörtlich übersetzt werden, wie wir es bereits bei dem IF ... THEN ... - Befehl getan haben. Es heißt in etwa soviel wie "wenn ... dann ... sonst ...". Der IF ... THEN ... Teil funktioniert genauso wie der einfache IF ... THEN ... - Befehl, den wir bereits in Stufe 4 Teil 1 besprochen haben.
Die Befehle hinter ELSE werden dann ausgeführt, wenn der logische Ausdruck (zwischen IF und THEN) nicht erfüllt ist. In Zeile 20 des 2. und 4. Programmbeispiels von der vorhergehenden Seite wird dieser Befehl zum simulieren von Programmschleifen verwendet. Aufgrund des eben Gelernten wissen wir auch, was dieser Befehl dort bewirkt: Solange die Variable i bzw. a kleiner oder gleich neun ist, wird der Befehl hinter THEN ausgeführt, also zur Programmzeile 30 gesprungen. Sobald der logische Ausdruck (a<=9) nicht mehr erfüllt wird, - a also größer als 9 ist - wird der Befehl hinter ELSE ausgeführt, also zur Programmzeile 60 gesprungen.

REM

ist ein sogenannter nicht ausführbarer Befehl. Er bewirkt im Programm nichts und dient lediglich der besseren Übersicht. REM ist die Abkürzung des englischen Wortes remark, was auf deutsch übersetzt Bemerkung heißt. Dieser Befehl kann ohne weiteren Zusatz hinter einer Zeilennummer stehen, was im Programmlisting eine Art Leerzeile bewirkt.
Die andere Möglichkeit ist, den REM-Befehl mit einem beliebigen Text zu versehen. Dies gibt Ihnen die Möglichkeit, Ihre Programme zu kommentieren, da alles, was hinter dem REM-Befehl steht, vom Computer bei der Programmausführung ignoriert wird. Zu beachten ist hierbei, daß einem REM-Befehl nicht innerhalb derselben Zeile ein BASIC-Befehl folgt, der ausgeführt werden sollte. Aus dem eben Gesagten können Sie nämlich ersehen, daß auch er ignoriert werden würde.
Beispiel:

```
10 REM Benzinverbrauch
15 REM
20 INPUT "Gefahrene Kilometer ";KM
30 INPUT "Verbrauch in Litern ";LI
35 REM
40 V=LI/KM*100
45 REM
50 PRINT "Verbrauch je 100 KM:";V:REM Ergebnis
```

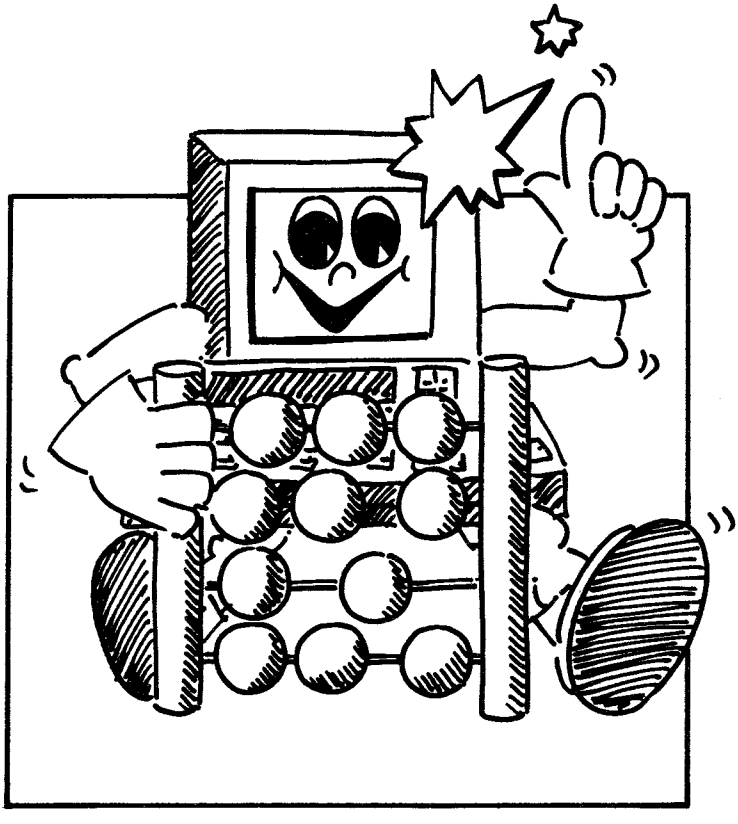
Benutzen Sie den REM-Befehl häufig, wenn Sie ein langes Programm entwickeln! Es kann sonst passieren, daß das Programm für andere Programmierer - und nach einiger Zeit auch für Sie! - nicht mehr durchschaubar ist.

In Zeile 10 des obigen Programmes benutzen wir den REM-Befehl für die Programmüberschrift. Auch Sie sollten darauf achten, in solchen Fällen Namen zu gebrauchen, die einerseits auf den Programminhalt hinweisen, andererseits aber eventuell sogar mit dem Namen identisch sind, unter dem Sie das entsprechende Programm abgespeichert haben.
Die Zeilen 15, 35 und 45 haben wir eingefügt, um das Programm in drei Teile zu trennen, nach denen auch Sie jedes Programm gliedern sollten:

Eingabe
Verarbeitung
Ausgabe

Der Einfachheit halber nennt man diese Unterteilung auch das EVA-Prinzip.
Der REM-Befehl in Zeile 50 ist nicht unbedingt erforderlich. Wir haben ihn jedoch eingefügt, um Ihnen zu zeigen, daß es durchaus möglich ist, in einer Zeile auszuführende BASIC-Befehle und Kommentare zu kombinieren. Achten Sie aber unbedingt auf die Reihenfolge: Zuerst die BASIC-Befehle, und erst dann den Kommentar. Wenn Sie den REM- und den PRINT-Befehl vertauschen, dann hätte das zur Folge, daß auch der PRINT-Befehl nicht mehr ausgeführt werden würde.

STUFE 5 PROGRAMMIEREN IN BASIC



Teil 1: Mit dem Computer rechnen

Gerade im wissenschaftlichen Sprachgebrauch wird der Computer häufig als

Rechner

bezeichnet. In technisch-wissenschaftlichen Bereichen wurde mit Computern fast ausschließlich gerechnet, so daß letztendlich Rechengenauigkeit und -geschwindigkeit heute einen extrem hohen Entwicklungsstand haben. Auch in der Wirtschaft gibt es große Zahlenmengen zu verarbeiten. Die "schnellen" Ergebnisse, die heute Computer permanent liefern, ermöglichen schnelle Entscheidungen. Die Zeitersparnis schafft Freiräume für Kreativität.

Bereits der heute nahezu von jedermann verwendete Taschenrechner ist ein kleiner Computer, der diesem Anspruch gerecht wird. Kein Computer ist jedoch in der Lage, von sich aus unsere Rechenaufgaben anzupacken. Wir selbst müssen die Aufgaben für ihn formulieren und ihm in einer Form präsentieren (z.B. BASIC-Programm), die er verarbeiten kann.
Das auf Ihrer Programmdiskette im Lehrgang auftretende Beispiel für die Berechnung des Benzinverbrauchs ist für jemanden, der gelegentlich einen Sonntagsausflug macht, mehr ein netter Gag als ein echter Nutzen. Wer jedoch tagtäglich im Auto große Strecken zurücklegt, kann durch ähnliche Programme die Wirtschaftlichkeit seines Fahrzeuges permanent im Auge behalten. Das kleine Verbrauchsprogramm ist dabei schon der erste Fortschritt bei der Verbrauchsermittlung:
Man kann immerhin die Liter- und Kilometerwerte nach dem Eintippen durch erneutes Drücken der ENTER-Taste sofort ermitteln und erspart sich somit das permanente Durchrechnen der Formel (V=L/K*100).
Es ist nur ein kleiner Mehraufwand im Programm und man erhält auf einen weiteren Knopfdruck hin z.B. den Durchschnittsverbrauch des Kraftfahrzeugs einer fahrtenreichen Woche. Führt man diese Überlegung weiter und fügt man noch einen Programmteil für einen Listenausdruck hinzu, so kann dieses elementare Programm für den Verwalter eines Fuhrparks mit mehreren Fahrzeugen nicht nur eine wesentliche Zeitersparnis, sondern auch eine erhebliche Planungshilfe aufgrund exakter Daten für die Fuhrparkkosten sein.
Die aufgrund dieser Tatsache ersparten Kosten rechtfertigen über einen Zeitraum von z. B. einem Jahr bereits die Anschaffung eines kleinen Computersystems nur für diese Aufgabe.
Ein ähnliches Beispiel läßt sich für ein Programm zur Preiskalkulation aufzeigen. Es gibt viele Kaufleute, die eine Vielzahl von Kleinteilen lagern und in dicken Ordnern auf Preislisten erfaßt haben. Bei Preisänderungen hat dies zur Folge, daß die gesamten Listen überarbeitet werden müssen. Mit dieser Arbeit können Mitarbeiter je nach Zahl der Artikel Stunden oder Tage beschäftigt sein.

In einen "4-Funktionen Taschenrechner" muß jedesmal der Einkaufspreis und die Formel für die Berechnung des Verkaufspreises eingetippt werden. Das Resultat wird dann in die neu angelegte Preisliste eingetragen. Mit einem Schneidercomputer und einem kleinen Kalkulationsprogramm, das ähnlich aufgebaut ist wie das vorher besprochene Benzinverbrauchsprogramm, verkürzt sich dieser Vorgang auf das Eintippen des Einkaufspreises und das Drücken der ENTER-Taste. Verwendet man einen Drucker, so läßt sich dieses Programm ohne großen Aufwand dahingehend ergänzen, daß in ein fertiges Formular die passenden Preisänderungen vom Drucker automatisch eingefügt werden.
Diese einzelne Anwendung kann bereits für zahlreiche Einzelhändler eine deutliche Zeitersparnis bringen.
Immer wenn es gilt, große Mengen von Zahlen oder Werte mit immer wiederkehrenden Formeln berechnen zu lassen, ist der Computer ein gerngesehener Helfer.
Der Computer ist kein Mathematiker, hat aber die Eigenschaft, daß er enorm schnell und im Rahmen der vom Entwickler zugelassenen Rechengenauigkeit fehlerfrei rechnen kann. Der Benutzer muß unter Berücksichtigung der BASIC-Syntax dem Computer die Formeln für seine mathematischen Zielsetzungen mitteilen.
Die Mathematik-Kenntnisse des Anwenders sollten dem Einsatz dieser Formeln entsprechen. Es kann also durchaus sinnvoll sein, sich mit der Mathematik zu beschäftigen, wenn die Erinnerungen an dieses Fach aus der Schulzeit nicht mehr ganz greifbar sind.

Teil 2: Mit dem Computer gestalten

Wir haben im vorigen Kapitel gesehen, wie mit Hilfe eines guten Programmes eine Vielzahl von Werten ohne großen Aufwand ermittelt werden können. Viele Menschen arbeiten jedoch ungern mit Listen, Tabellen oder sonstigen umfangreichen Zahlensammlungen. Populäre Zahlenergebnisse werden in den Medien seit langem in Form von anschaulichen Grafiken dargestellt. Wir alle kennen bei Wahlen die sog.

Tortendiagramme,

bei denen die verschieden großen - farblich unterschiedlichen - "Tortenstücke" die einzelnen prozentualen Ergebnisse der Parteien verdeutlichen.
Ähnlich oft werden

Balkendiagramme

verwendet. Hierbei lassen sich negative Ergebnisse von positiven sehr anschaulich dadurch unterschieden, daß man sie unter- bzw. oberhalb einer Null-Linie in Form von verschieden langen Balken darstellt. In der Regel werden daneben auch andersfarbige Balken, die Vergleichswerte verdeutlichen sollen, mit eingebaut. Besonders bei prozentualen Abweichungen ist diese Grafikdarstellung sehr verbreitet.
Solche Grafik jedoch selbst zu programmieren, erfordert viel mehr Aufwand als das Programmieren einfacher Berechnungen. Auch sind Kenntnisse im Umgang mit mathematischen Funktionen und geometrischen Formeln erforderlich.
Computergrafik wird aber auch in anderen Bereichen zum Gestalten oder Verdeutlichen von Figuren verwendet. Über diesen Weg sollten Sie auch den Einstieg in die Grafik-Programmierung in BASIC suchen.
Am einfachsten gestaltet man Bilder durch die Verwendung des

Print-Befehls

und ausgewählter Grafikzeichen, die zwischen die dem PRINT-Befehl folgenden Anführungszeichen ("") geschrieben werden.
Sie werden bereits nach kurzer Zeit einen Blick dafür bekommen, wie man durch Einfügen von Leerzeichen die Grafiksymbole an die entsprechende Stelle auf dem Bildschirm "eindirigieren" kann.
Durch Verwendung der leistungsfähigen Grafiksymbole Ihres Schneider Computers ist es aber auch in BASIC möglich, Bilder und Diagramme mit hoher

Auflösung

zu erstellen.

Die Auflösung sagt aus, wieviel Punkte auf Ihrem Bildschirmen vom Computer einzeln "angesprochen" werden können. Ihr Schneider Computer vermag maximal 200 Zeilen mit jeweils 640 Punkten zu erstellen, was einer Auflösung von 640 x 200 bzw. 128 000 Bildschirmpunkten entspricht. Hierbei handelt es sich allerdings nicht um Textzeilen (das sind diejenigen Zeilen, die Ihr Computer erzeugt, wenn Texte auf den Bildschirm geschrieben werden), sondern um sog. Bildschirmzeilen (die der schmalsten Linie entsprechen, die Sie mit Ihrem Schneider Computer auf dem Bildschirm erzeugen können). Eine solche Textzeile entsteht nämlich aus acht Bildschirmzeilen, was am Beispiel des Buchstabens "A" folgendermaßen aussieht:

```
1. Bildschirmzeile:     - - - * * - - -
2. Bildschirmzeile:     - - * * * * - -
3. Bildschirmzeile:     - * * - - * * -
4. Bildschirmzeile:     - * * - - * * -
5. Bildschirmzeile:     - * * * * * * -
6. Bildschirmzeile:     - * * - - * * -
7. Bildschirmzeile:     - * * - - * * -
8. Bildschirmzeile:     - - - - - - - -
```

Abschließend läßt sich zu dem Kapitel "Mit dem Computer gestalten" feststellen, daß Computergrafik wohl die variationsreichste und damit auch interessanteste Form der Computeranwendung ist. Letztendlich entscheidet aber die Kreativität des Benutzers über die Qualität der mit dem Computer produzierten Ergebnisse.

Teil 3: Mit dem Computer Daten verwalten

Datenverwaltung ist wohl die populärste Form der Computeranwendung. Selbst im privaten Bereich haben sich bei vielen Menschen bis heute schon soviel Daten (Tel.-Nrn., Adressen, Zahlen, Termine usw.) angesammelt, daß es erforderlich wäre, diese Daten organisiert zu verwalten, will man vermeiden, den Überblick zu verlieren.
Vor der Computerzeit war das

Papier

der beliebteste

Datenträger.

Karteikästen, Bücher, Blöcke, Hefte oder Ringordner sind dabei heute noch beliebte Ordnungssysteme. Eines der wichtigsten Kriterien bei jeder Form der Datenverwaltung war und ist immer noch die

Zugriffszeit.

Man macht ja schließlich organisierte Datenverwaltung, um all das schnellstmöglich zu finden, was man nicht im Kopf behalten kann. Ein zweites, nahezu genauso wichtiges, Kriterium ist die Möglichkeit der

Datenverknüpfung.

Damit ist z.B. das Anfertigen von Listen, das Berechnen von in verschiedenen Datensätzen enthaltenen Werten oder auch das Sortieren von Daten gemeint. Bei Datenverknüpfung erweist sich der Datenträger Papier als wenig flexibel. Besonders dann, wenn sich die Daten auf verschiedenen Blättern befinden.
Betrachten wir als Beispiel für einen Datensatz eine Adreßdatei. In der Regel enthält diese folgende

Felder:

```
Feld  1:   NAME
Feld  2:   VORNAME
Feld  3:   STRASSE
Feld  4:   HAUSNUMMER
Feld  5:   POSTFACH
Feld  6:   POSTLEITZAHL
Feld  7:   ORT
Feld  8:   POSTBEZIRK
Feld  9:   ORTSVORWAHL
Feld 10:   TELEFONNUMMER
```

Selbstverständlich lassen sich einige Felder zusammenfassen: z.B. ein großes Feld für Postleitzahl, Ort und Postbezirk (8000 München 2). Muß man aber z.B. nach Postleitzahlen sortieren oder nach der Häufigkeit der verschiedenen vorhandenen Postleitzahlen auswerten, ist es erforderlich, daß man auf Postleitzahlen als eigenständiges Feld zugreifen kann.
Bei dem Datenträger Papier ist es bei jeder Form der Verknüpfungen bzw. Auswertungen notwendig, die Inhalte der einzelnen Felder auf ein Neues zu übertragen. Es ist auch beim Zugriff immer erforderlich, eine ganze "Karteikarte" (entspricht 1 Datensatz) in die Hand zu nehmen.
Der große Vorteil des Computers liegt darin, daß er sowohl einen kompletten Datensatz, als auch einzelne Felder anzeigen oder verarbeiten kann. Bei guten Datenverarbeitungsprogrammen lassen sich zudem weitere Felder hinzufügen. So könnte unser Adreßdatensatz durch folgende Felder ergänzt werden:

```
Feld 11:   GEBURTSDATUM
Feld 12:   ALTER
Feld 13:   GRÖSSE
Feld 14:   HOBBIES    usw.
```

In einem gut organisierten Programm wird dabei das Alter automatisch aufgrund einer fest eingegebenen Formel aus dem Geburtsdatum berechnet. Der Computer kann dann (wenn es vom Programm vorgesehen ist) z.B. bei einer Altersgruppe die Durchschnittsgrösse aus den einzelnen Feldern berechnen. Dazu müssen entsprechende

Suchkriterien

herausgestellt werden, wie zum Beispiel:
Alter 24-26, Postleitzahlbezirk 7000-8000.

Bei Datenfeldern wird zwischen zwei Arten unterschieden:

-Numerische Felder
(enthalten Zahlenwerte aller Art)

-Stringfelder
(enthalten Worte, Buchstaben und Texte)

Numerische Felder können sowohl berechnet als auch sortiert werden.
Stringfelder können nur sortiert werden.

Dabei läßt sich zudem die Häufigkeit bestimmter Zahlenreihen feststellen, oder Worte können miteinander verknüpft werden.

Dies alles klingt in der Theorie sehr schön; wie aber programmiert man das in BASIC?

Bis zur Programmierung einer eigenen Datenverarbeitung ist es für den Einsteiger ein relativ weiter Weg.
Trotzdem sind selbst umfangreiche

Datenbanken

unwesentlich anders aufgebaut, als kleine Verwaltungsprogramme. Damit der Computer Daten verwalten kann, müssen alle diese Daten gespeichert sein. Die Speicherung kann auf Diskette bzw. Kassette erfolgen. Von diesen Datenträgern werden sämtliche Daten, die ja alle einen Variablennamen haben, durch ein Programm in den Arbeitsspeicher geladen, je nach Programm verarbeitet und aktualisiert wieder auf Datenträger abgespeichert.
Auf Ihrer Programmdiskette ist ein erstes Beispiel für Datenverwaltung (Telefonliste) aufgeführt. Je größer Ihre BASIC-Kenntnisse werden, desto leichter wird aus diesem kleinen Programm einmal Ihre persönliche Datenbank werden.
Dabei sollte Ihr individueller Bedarf die Entscheidung herbeiführen, ob Sie Daten durch eigene Programme verwalten oder fertige Software (Stufe 6) anwenden wollen.
Zudem können Sie anhand zahlreicher vorhandener Literatur sich einen Überblick über die Einsatzvielfalt Ihres Schneidercomputers verschaffen.

PROGRAMM-BEISPIELE

Programm 1

Dieses Programm berechnet den Benzinverbrauch je 100 km.

```
10 INPUT"GEFAHRENE KILOMETER";KM
20 INPUT"GETANKTE LITER";LI
30 V=LI/KM*100
40 PRINT"VERBRAUCH:";V
```

Programm 2

Dieses Programm berechnet die Hypothenuse eines Dreiecks.

```
10 INPUT"KATHETE 1";A
20 INPUT"KATHETE 2";B
30 C=SQR(A^2 + B^2)
40 PRINT"HYPOTHENUSE:";C
```

Programm 3

Dieses Programm berechnet den Nettopreis und den Steuerbetrag aus dem Bruttopreis und dem Steuersatz.

```
10 INPUT"BRUTTOPREIS";B
20 INPUT"STEUERSATZ IN %",S
30 N=B/(1+S/100)
40 PRINT"NETTOPREIS";N
50 M=B-N
60 PRINT"STEUER:";M
```

Programm 4

Dieses Programm sucht Name und Telefonnummer aus einem "Verzeichnis".

```
10   INPUT "WELCHER NAME";NA$
20   FOR I=1 TO 5
30   READ N$,V$,T$
40   IF N$=NA$ THEN GOTO 80
50   NEXT I
60   PRINT"NAME NICHT GEFUNDEN"
65   RESTORE
70   GOTO 10
80   PRINT"NAME","VORNAME","TEL."
90   PRINT N$,V$,T$
100  END
110  DATA MUELLER,PETER, 099/342345
120  DATA HUBER,FRANZ,    071/45234
130  DATA ABELE,PAULINE,0221/455566
140  DATA GRANZ,MARIA,    0854/1234
150  DATA MEIER,KLAUS, 04711/125589
```