Die Computerfamilie

Spectravideo SVI-328

Ein Computer kommt ins Haus

Wie eine Seuche breitet sich momentan das "Home-Computer"-Fieber aus. Da stellen sich doch die Fragen, was ist ein Computer überhaupt und was bezeichnet man als "Home-Computer" (engl.: Heimcomputer)?

Sehen wir uns zuerst den Computer allgemein an. Die Funktion dieses Gerätes ist trotz seiner vielfältigen Anwendungsmöglichkeiten sehr einfach umrissen. Alles, was der Computer macht, ist im wesentlichen das Aufnehmen von Informationen, das Verarbeiten dieser Daten und die Ausgabe. So komisch das klingt, aber unser Taschenrechner ist auch ein Computer, und an ihm können wir genauer sehen, wie Computer arbeiten. Dazu wollen wir eine einfache Addition betrachten.

Wir müssen in unseren Taschenrechner zuerst drei Informationen eingeben: Die erste Zahl, das Pluszeichen, die zweite Zahl. Unser Gerät braucht ja die beiden Werte, um zu wissen, mit welchen Zahlen es hantieren soll. Ebenso ist das Pluszeichen notwendig, um eine Addition durchführen zu können (sonst kann der Rechner ja nicht erkennen, welche Rechenoperation er erledigen soll, es könnte ja auch eine Multiplikation gewünscht sein!). Im Inneren bearbeitet unser Gerät dann die Informationen, in unserem Fall führt es eine Addition durch, und hat danach das Ergebnis vorliegen. Zuletzt zeigt unser kleines "Kastl" dieses Ergebnis noch an. Bei einem Taschenrechner wird dies auf einer sogenannten Sieben-Segment-Anzeige gemacht (Bild 1.1). Dabei wird ein Zeichen mit sieben "Strichen" dargestellt. Die Ausgabe kann aber auch über viele andere Wege erfolgen, zum Beispiel über einen Bildschirm oder ganz einfach nur als Spannungspegel. Wir sehen, das Funktionsprinzip eines Computers ist ganz einfach.

Bild 1.1

Ebenso einfach ist die Handhabung des Computers. Wir werden im Laufe der dreizehn Folgen bemerken, daß es keine große Schwierigkeit ist, einen Home-Computer zu bedienen. Bleibt nur noch die Frage offen, welchen Unterschied hat ein Heimcomputer gegenüber anderen Computern und aus welchen Bestandteilen ist er zusammengesetzt?

Der Heimcomputer

Noch vor wenigen Jahren wäre es undenkbar gewesen, einen eigenen Computer für zu Hause zu besitzen, oder gar einen tragbaren zu verwenden, denn vor ungefähr 20 Jahren füllten Anlagen mit der Leistung eines Heimcomputers von heute einen oder zwei Räume. Erst durch die Erfindung von Chips (nicht die zum Essen) wurde es ermöglicht, diese Monster kleiner und leistungsfähiger zu machen. Chips sind kleine schwarze "Dinger", die elektronische Bauelemente auf kleinstem Raum kombinieren. So kann man heute mit Leichtig-

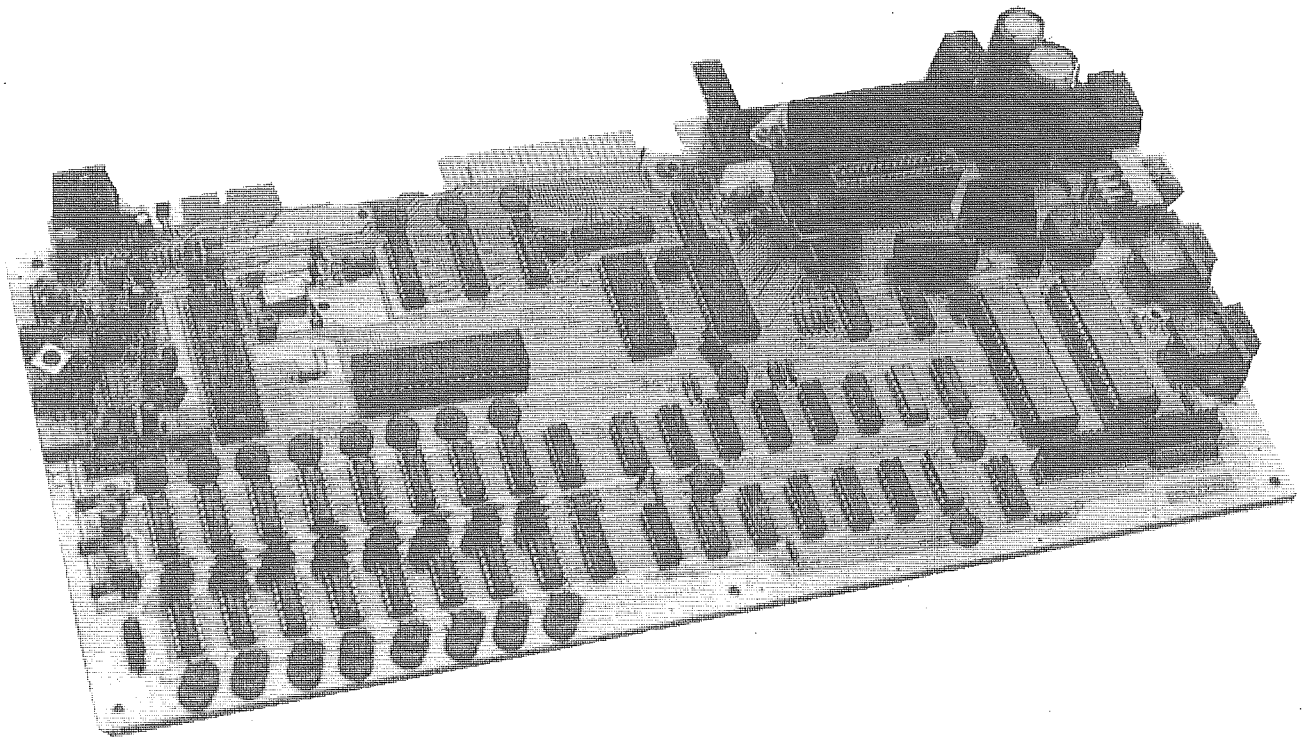


Bild 1.2

keit einige zehntausend solcher Elemente auf 16 Quadratmillimeter integrieren. Mit diesen Methoden schaffte man es auch, den Computer auf ein erträgliches Maß zu verkleinern, die Leistung zu steigern und den Energieverbrauch zu senken. Nun konnte man in großen Stückzahlen billige, kleine, aber leistungsfähige Geräte erzeugen. Die Home-Computer waren geboren!

Welche Elemente beherbergt nun so ein Heimcomputer?

Dazu können wir uns das Innenleben des Spectravideo-Computers SVI-328 im Bild 1.2 ansehen. Wir erkennen einen langen schwarzen "Käfer", der als Mikroprozessor bezeichnet wird. Er ist das Herz unseres Spectravideo (und jedes anderen Home-Computers auch). Beim SVI-328 ist es der Mikroprozessor Z80A. Er ist für fast alle Verarbeitungen von Daten verantwortlich. Natürlich braucht unser "Gehirn", der Mikroprozessor, ein geeignetes Gedächtnis. Dies wird ihm durch weitere Chips gegeben. Man nennt das Gedächtnis eines Computers "Speicher".

ROMs und RAMs

Natürlich gibt es hier mehrere Arten. Aus dem "ROM"(engl. "Read Only Memory"=Nur Lesen möglich) kann man, wie der Name schon sagt, nur Informationen herausholen. Der Benützer eines Computers kann den Inhalt des ROMs nicht verändern. Er bleibt immer gleich und verschwindet auch nicht nach dem Abschalten des Stroms. Gebraucht wird er zum Beispiel für unseren BASIC-Interpreter. Interpreter heißt "Übersetzer". Dieses Ding übersetzt die Sprache BASIC in eine viel einfachere Computersprache, nämlich den Maschinencode. Dieser Maschinencode ist die einzige dem Computer verständliche Sprache. Alle anderen Programmiersprachen werden in den Maschinencode übersetzt, so auch das BASIC.

Die zweite für uns wichtige Speicherart ist das "RAM" (engl. "Random Access Memory" = Speicher mit wahlfreiem Zugriff). Hier dürfen wir sowohl "Schreiben" als auch "Lesen". Beim "Schreiben" wird der Inhalt verändert, beim "Lesen" wird ein Teil des Inhaltes aus dem Speicher geholt und zur Verfügung gestellt, die Informationen bleiben aber weiterhin im RAM enthalten. Nach dem Abschalten des Computers ist der Inhalt jedoch verloren. Diesen Speicher braucht unsereins für das Arbeiten mit dem Computer. Der Prozessor legt nämlich alle Informationen, die er für seine Arbeit braucht, in diesem Speicher ab. Die Programme, die wir später erstellen werden, liegen auch im RAM.

Weiters besitzt unser Computer noch einige Chips, welche die Ein- und Ausgabe regeln. So gibt es den VDP, den "Video-Display-Processor", der die Zeichen am Bildschirm generiert, oder den PSG, der die Töne und Melodien erzeugt. Es gibt dann natürlich auch einen Chip, um die Tastatur abzufragen.

Soweit die "Innereien" unseres Spectravideo! Bleibt nur noch eines zur Funktionsweise eines Computers zu sagen. Alle Vorgänge im Gerät werden mit Strömen und Spannungen verwirklicht. Das Drücken einer Taste erkennt unser "Mikro" (Abkürzung für Mikrocomputer) anhand von Spannungsänderungen, die Verarbeitung von Daten ist auf Änderungen von Spannungen aufgebaut und sogar die Ausgabe auf dem Bildschirm wird bis in den Fernseher mit Strom und Spannung realisiert.

Fassen wir kurz die Baugruppen eines Computers anhand des Spectravideo SV-328 zusammen:

Die Zentraleinheit: Sie sorgt dafür, daß im Computer kein Chaos entsteht. Alle Daten, die ein- oder ausgegeben werden, kontrolliert die Zentraleinheit (auch CPU und bei Home-Computern Mikroprozessor genannt). Kein Datentransfer wird ohne Zustimmung der CPU durchgeführt. Ebenso ist sie für die Verarbeitung der Programme zuständig. Alle Umwandlungen von Werten geschehen in der CPU. In den Spectravideo-Computern ist der Mikroprozessor Z80A eingesetzt.

Die Speicher:

Das ROM: In unserem Computer enthält es den Übersetzer, der die Programmiersprache BASIC in den Maschinencode umwandelt. Der Inhalt ist unveränderbar und kann durch Abschalten des Stromes nicht vernichtet werden.

Das RAM: Hier werden Programme und Daten abgelegt. Der Inhalt kann sehr wohl verändert werden, allerdings ist er nach Abschalten des Computers verloren.

Die Ein-/Ausgabeschnittstellen (auch Interfaces genannt): Sie werden für die Kommunikation zur Außenwelt gebraucht. So wandelt das Video-Interface (oder VDP) die Daten, die vom Prozessor geliefert werden, in Zeichen oder Punkte um. Alle Graphiken werden so erzeugt, daß anhand von Daten die verschiedensten Punkte gesetzt werden. Ebenso gibt es auch ein Interface für die Tastatur. Dieses wandelt von der Tastatur kommende Signale in für den Prozessor verwertbare Daten um.

Programmiersprachen

Es sind nun schon öfters die Worte Programm und Programmiersprache gefallen. Was verbirgt sich dahinter?

Wir alle wissen inzwischen, daß wir dem Computer sagen müssen, was wir tun wollen. Beim Taschenrechner haben wir dies durch Drücken der Taste "+" gemacht. Bei Home-Computern können wir schon mehr Instruktionen geben. Jede Anweisung, die wir geben dürfen, können wir als Wort betrachten. Mehrere Wörter bilden eine Sprache. Wenn man nun diese Wörter so verwendet, daß für den Computer ein sinnvoller "Text" entsteht, dann ist das ein Programm. Unter "Text" ist jedoch kein normaler Text gemeint, sondern ein nur mit Wörtern aus einer Programmiersprache gebildeter Text.

Und hier sind wir schon beim nächsten Problem: Wie umfangreich sind diese Sprachen?

BASIC

Zweifelsohne versteht der Computer kein Deutsch. Sein Wortschatz (auch Befehlssatz genannt) beschränkt sich auf wenige Wörter, der Spectravideo hat im BASIC knapp 200 Instruktionen zur Verfügung. BASIC ist so eine Programmiersprache. Es gibt neben BASIC noch viele andere, uns interessiert jedoch momentan nur diese eine.

BASIC wird aus englischen Wörtern gebildet, die jedoch leicht zweckentfremdet benutzt werden. So heißt zum Beispiel "PRINT" im BASIC, daß etwas am Bildschirm ausgegeben werden soll. PRINT "AS" zeigt die Buchstaben A und S an.

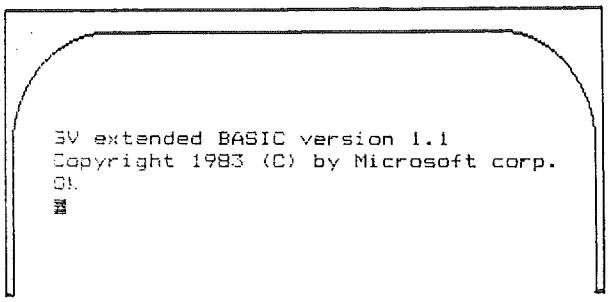
Leider kann der Computer direkt nur eine einzige Sprache verstehen, den Maschinencode. Alle anderen Sprachen werden in diese

primitivste aller Sprachen im Computer übersetzt. Dazu wird übrigens auch der BASIC-Interpreter gebraucht. Der Computer kann nämlich nur zwischen Eins und Null unterscheiden ("Strom", "Kein Strom"). Alle Maschinencodebefehle werden aus 1 und 0 gebildet. So sieht zum Beispiel die Addition von zwei Ziffern so aus: 1000 0000 (im BASIC hingegen: A=A+B). Man sieht hier den "Pferdefuß" vom Maschinencode. "1000 0000" ist bei weitem unverständlicher als A=A+B. Deshalb wurden höhere Sprachen entwickelt, darunter auch BASIC.

Nun haben wir genug theoretisiert. Stürzen wir uns mit der Familie Dischek in das Computer-Vergnügen.

Unser neuer Heimcomputer

Zuerst schalten wir unsere technisches Wunderwerk ein. Es steht einfach folgendes am Bildschirm:



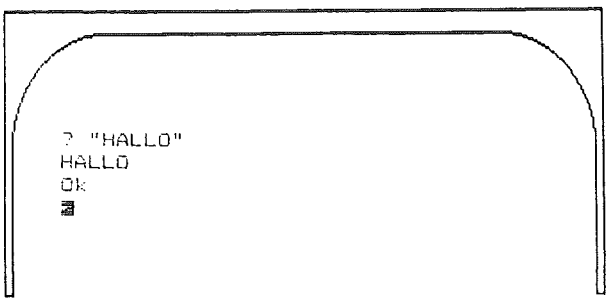
Von Intelligenz ist noch nicht viel zu sehen. Das einzige, was wir erkennen können, ist ein Viereck (wir werden es "Cursor" nennen). Zaghaft drücken wir eine Taste. Der gedrückte Buchstabe erscheint an der Stelle des "Cursors" am Bildschirm und dieser ist um eine Zeichenbreite nach rechts gerückt.

Das hat uns Mut gemacht. Wir geben "HALLO" ein und drücken "ENTER". Doch der Computer will uns anscheinend nicht. Er gibt einen "Syntax Error" aus. In unserer Verzweiflung geben wir ein Fragezeichen ein! Kein "Syntax Error"! Das ist fein! Nun probieren wir "?HALLO" und drücken ENTER.

Es erscheint aber nur eine Null! Mehr als seltsam, wir wollen doch ein HALLO sehen! Wie es in der direkten Rede so üblich ist, stellen wir HALLO in Anführungszeichen:

? "HALLO" 'CR'

(wir kürzen ENTER von nun an immer als 'CR' ab). Es stimmt!



Sollten wir irgendeinen Tippfehler machen. so ist das kein Grund zur Verzweiflung. Mit der DEL-Taste können wir den Buchstaben löschen, der unter dem Cursor steht, mit der Taste darunter (großer Pfeil nach links) wird das Zeichen links neben dem Cursor gelöscht und die Taste INS wird zum Einfügen von Zeichen verwendet. Wenn der Cursor halbhoch ist, befindet sich der Computer im Einfügemodus.

Nun probieren wir:

?4+6 'CR'

Wir sehen 10! Offensichtlich kann der Computer rechnen. Dies prüfen wir mit

?34*3 'CR'

Richtig 102! Wenn wir jedoch

?"4+6" 'CR'

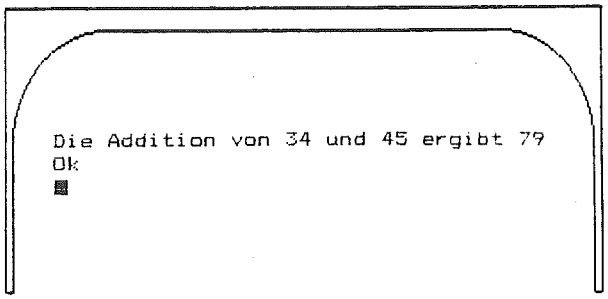
eingeben, dann sehen wir 4+6.

Nach einiger Überlegung kommen wir zu dem Schluß, daß Zahlen ohne Anführungszeichen zum Rechnen verwendet werden. Stehen jedoch zwei '"' vor und hinter einem Text, dann wird dieser, egal ob aus Buchstaben oder Zahlen bestehend, unverändert angezeigt.

Dies bringt uns auf folgende Idee: Wir geben ein:

? "Die Addition von 34 und 45 ergibt";
34+45 'CR'

Nun können wir das Gewünschte sehen:



Gevift, oder?

Den Strichpunkt brauchen wir lediglich dafür, um die beiden Teile, Text und Rechnung, direkt hintereinander anschließen zu lassen. Beim Spectravideo können wir den Strichpunkt auch weglassen.

Die Computerfamilie Spectravideo SVI-328

Ein Nachzipf droht

2. Folge

Wir alle kennen die vielen Video- und Computerspiele, mit denen uns Computerhersteller begeistern. Aber die Home- und Personal-Computer können weit mehr. Es ist nämlich möglich, sie frei zu programmieren. Damit schaffen wir es sogar, unsere eigenen Ideen in Programme umzuwandeln. Von nun an wird unser Spectravideo nur noch das machen, was wir von ihm wollen.

Programmieren bedeutet, daß die einzelnen Befehle einer Programmiersprache (zum Beispiel von BASIC) in ganz speziellen Sinnzusammenhängen zusammengefaßt werden. Wir geben dem Computer Kommandos vor, und er arbeitet sie nacheinander ab. Durch dieses sehr schnelle Verarbeiten von Anweisungen entstehen die tollsten Effekte. Jedes Videospiel, jedes Textverarbeitungssystem ist ein Programm, geschrieben in einer bestimmten Sprache.

Die einzige Schwierigkeit, die für uns besteht, ist zu wissen, welche Anweisungen wir dem Computer wann geben, um die gewünschten Effekte zu erzielen. Dazu brauchen wir logische Überlegungen, die "Kunst zu programmieren". Sie werden aber erkennen, daß diese Logik sehr schnell erlernt werden kann. Alles, was wir zu tun haben, ist einerseits, die wichtigsten Befehlswörter kennenzulernen und uns andererseits ein bißchen mit den Regeln, der Grammatik der Programmiersprache vertraut zu machen.

Der Computer ist ein williger Arbeiter, er tut das, was ihm aufgetragen wird, in Blitzeseile. Enthält ein Befehl aber einen Fehler, dann erkennt er ihn nicht selbsttätig. Es ist somit auch unsere Aufgabe, für den eigentlich "dummen" Computer die Fehler zu beseitigen.

Vorerst werden wir uns aber den ersten "Fremdwörtern" der Sprache BASIC widmen. So lernen wir INPUT, PRINT und LET.

Unser erstes "Programmiererwissen"

Zuerst müssen wir uns klar werden, wie so ein Programm eigentlich in den Computer kommt, und was der Computer damit macht.

Nachdem wir uns ausführlich mit unserem Problem herumgeschlagen haben, und das Programm in groben Zügen feststeht, geben wir es in den Computer ein. Dabei wird das Programm in logische Zeilen gegliedert. Jede dieser Zeilen wird mit ENTER abgeschlossen. Dies ist besonders wichtig! Solange kein ENTER gedrückt ist, hat der Computer den soeben eingegebenen Text nicht in seinem Gedächtnis! Deshalb müssen wir nach jeder Eingabe ENTER drücken. Das eigentliche Erstellen eines Programmes wird später erklärt.

Wenn wir nun die Entwicklung eingeben, dann können wir die Tasten, welche wir schon in der vorigen Folge verwendet haben, auch hier zum Korrigieren benützen. Darunter fallen die Tasten DEL, INS, CLS und die Taste mit dem großen Pfeil nach links.

Unser Computer hat einen Bildschirmeditor. Das heißt, daß wir mit unserem leuchtenden Viereck (auch Cursor genannt) frei beweglich am Bildschirm herumfahren können. Wo immer wir wollen, dürfen wir einen Buchstaben setzen. So wird es uns auch möglich, eine schon eingegebe Zeile zu korrigieren, sofern sie noch am Bildschirm ist (nicht am Fernseher sichtbare Zeilen werden wir später ebenso leicht löschen können, nur brauchen wir dazu dann einen BASIC-Befehl, der uns die Zeile wieder auf den Bildschirm bringt).

Wenn nun das Programm so recht und schlecht im RAM unseres Computers seinen Platz gefunden hat, beginnt der Computer nach einem speziellen Befehl die Programmabarbeitung. Dabei übersetzt er den ersten Befehl des BASIC in Maschinencode und führt den Maschinencodebefehl aus. Danach holt er sich das nächste BASIC-Kommando, übersetzt dieses und läßt den nächsten Maschinencodeteil ablaufen.

Interpreter und Compiler

Die Ausführung und die Übersetzung finden im selben Arbeitsgang statt. Die Einrichtung, die diese Art der Programmabarbeitung möglich macht, nennt man Interpreter. Es gibt auch Compiler. Diese trennen die Ausführung streng von der Übersetzung. Zuerst wird das ganze Programm in Maschinencode umgewandelt. Erst danach, wenn das gesamte BASIC-Programm in Maschinensprache übersetzt wurde, fängt der Computer mit der Abarbeitung des (Maschinencode-)Programms an. Wir werden jedoch mit einem Interpreter arbeiten, der jede Anweisung übersetzt und sie gleich danach ausführt.

BASIC

Sehen wir uns nun die ersten "Rechtschreibregeln" der Programmiersprache BASIC an.

In einem Programm muß jede Zeile eine Zeilennummer haben. Nur mit einer Zahl, welche die Zeile kennzeichnet, kann der Computer später die Zeile wieder identifizieren.

Die Zeile selber besteht einerseits aus Befehlswörtern, wie PRINT, INPUT und anderen. Zusätzlich zu diesen kommen noch Beifügungen dazu, zum Beispiel was ausgegeben werden soll. PRINT "HALLO" zeigt dem Computer an, daß er den Text HALLO auf dem Bildschirm abbilden soll.

Am Ende eines Programms ist es günstig, ein END oder ein STOP zu setzen. Diese beiden Wörter sind BASIC-Befehle, die den Computer dazu veranlassen, mit der Programmabarbeitung aufzuhören. Es ist jedoch nicht zwin-

gend notwendig, eines dieser beiden Kommandos zu verwenden. Der Spectravideo hört am Ende eines Programmes von selber mit der Verarbeitung auf.

Direkt nach dem Einschalten befindet sich der Computer im Eingabemodus. Nun können wir ein Programm eingeben. Wollen wir es ablaufen lassen, müssen wir RUN 'CR' drücken. Ist das Programm gestoppt worden oder zu Ende, befindet sich der Computer wieder im Eingabemodus.

Man bezeichnet diese beiden Betriebsarten des Computers auch als direkten und indirekten Modus.

Der direkte Modus ist der Eingabemodus. Neben der Programmeingabe kann man auch einzelne Befehle im direkten Modus ablaufen lassen. Als Faustregel kann man sich folgendes merken: Steht vor einer BASIC-Zeile eine Zahl, dann ist diese Zeile ein Teil des Programms und kann nur mit RUN 'CR' aktiviert werden. Steht keine Zahl davor, wird nach Eingabe von ENTER der Befehl direkt ausgeführt. Ein Beispiel dieser direkten Abarbeitung haben wir schon in der ersten Folge kennengelernt. Das ? "HALLO" war nichts anderes als ein direkter Befehl (welcher, das werden wir später klären).

Im indirekten Modus läuft das von uns eingegebene Programm ab. Diese Betriebsart wird durch RUN 'CR' aktiviert.

Nun fangen wir aber endgültig an, "Vokabeln" zu lernen. Der erste Befehl ist

INPUT

Mit dieser Anweisung können wir Zahlen oder Zeichen eingeben. Hinter das INPUT kommen ein oder mehrere Buchstaben. Diese Buchstaben bilden einen Variablennamen. Vielleicht werden einige von Ihnen diese "widerlichen" Buchstaben in der höheren Mathematik kennen, für die man jede beliebige Zahl einsetzen darf. Die Professoren in den Schulen haben sie Variablen genannt. Unsere Buchstaben haben ähnliche Funktion. Sie bezeichnen einen "Behälter", in den man kleine Texte oder Zahlen einspeichern kann. Wie das genauer vor sich geht, werden wir noch erfahren. Wichtig ist für uns momentan, daß in so einer Variablen Zahlen drinnen stehen. Wenn hinter dem Namen ein Dollar-Zeichen ($) steht, sind in dem Speicherelement Texte enthalten.

LET

Unsere zweite Anweisung ist LET. Hier können wir den Variablen einen bestimmten Wert zuweisen (Bei INPUT müssen wir den Wert über die Tastatur eingeben).

? oder PRINT

Das dritte neue Kommando ist eigentlich schon altbekannt. Unser Fragezeichen ist nämlich gleichwertig zum Befehl PRINT, der gewisse Zeichen auf den Bildschirm ausgibt. Auch hier dürfen wir wieder Variablen verwenden. Ein kleines Programm soll uns das bisher Gesagte verdeutlichen:

```
10 INPUT A
20 LET X=30
30 PRINT A*X
40 END
```

Wenn wir nun mit RUN 'CR' unser erstes Programm starten, dann erkennen wir ein Fragezeichen am Bildschirm. Dies ist das Anzeichen dafür, daß der Computer beim INPUT-Befehl auf eine Eingabe wartet. Die geben wir ihm mit 3. Sofort zeigt er 90 an. Da wir der Variablen X 30 zugewiesen und in die Variable A 3 eingeben haben, sehen wir das richtige Ergebnis 30*3. Wenn wir nun Zeile 40 löschen, indem wir 40 'CR' drücken und RUN 'CR' eingeben, dann ist der gleiche Effekt wie vorhin zu sehen. Ein Beweis dafür, daß END nicht unbedingt gebraucht wird.

Die Anweisung LET ist übrigens bei den Spectravideo-Computern nicht erforderlich. Man kann Zeile 20 unter Weglassung von LET auch so schreiben:

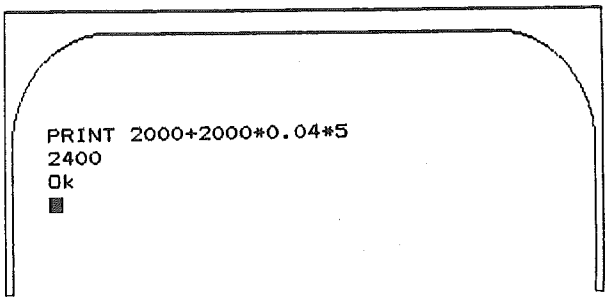
```
20 X=30
```

Das bedeutet, daß der Variablen X der Wert 30 zugewiesen wird.

Unsere ersten Programmierversuche

Zuerst genießen wir noch einmal den direkten Modus, im Bewußtsein, was wir wirklich machen.

Wir geben PRINT 2000+2000*0.04*5 ein, und sehen unser Ergebnis 2400 am Bildschirm. Diese Rechnung wird zum Beispiel für Zinsenberechnungen gebraucht. So wissen wir, um wieviel unser Kapital von 2000 Schilling innerhalb von 5 Jahren bei 4% Zinsen angewachsen ist.



Wenn wir diese Rechnung öfters durchführen wollten, müßten wir sie immer wieder mühsam eintippen. Gott sei Dank hat unser Computer aber die Fähigkeit, diese Programmzeilen zu speichern. Also setzen wir beim zweiten Eingeben eine Zeilennummer davor. Natürlich wird diese Anweisung jetzt nicht sofort ausgeführt, sie ist ja als Teil eines Programmes gedacht.

```
300 PRINT "ENDKAPITAL=";2000*(1+0.04*5);"S"
```

Nun kommen wir aber auf die Idee, daß sich unser Computer auch vorstellen könnte. Daher tippen wir

```
100 PRINT "HALLO! ICH BIN DEIN COMPUTER
CHIPSY!"
```

ein. Wir alle kennen ja den Namen des Computers der Familie Dischek. Nun zeigen wir noch dem Bediener, daß der Computer unsere Hausübung macht. Dazu schreiben wir:

```
200 PRINT "HAUSUEBUNG"
```

LIST

Wenn wir nun das Eingegebene gesammelt sehen wollen, dann tippen wir LIST.

Mit diesem ungemein nützlichen Befehl sehen wir alle Programmzeilen, die wir eingegeben haben in der richtigen Reihenfolge. Der Computer sortiert nämlich die Zeilen nach der Größe der Zeilennummern. Wenn wir also irgendwo zwischen zwei schon vorhandenen Zeilen etwas einfügen wollen, brauchen wir nur eine Zeilennummer zwischen den beiden vorhandenen wählen.

Nach LIST 'CR' sehen wir:

```
10 PRINT "HALLO! ICH BIN DEIN COMPUTER
CHIPSY!"
20 PRINT "HAUSUEBUNG"
30 PRINT "ENDKAPITAL=";2000*(1+0.04*5);
"S"
```

Das END haben wir weggelassen, bei so einfachen Programmen brauchen wir es nicht.

Jetzt kommt der große Moment! Wir lassen unser erstes Programm "laufen", das heißt, wir drücken RUN 'CR'. Innerhalb weniger tausendstel Sekunden wird das Ergebnis auf dem Bildschirm präsentiert:

```
HALLO! ICH BIN DEIN COMPUTER CHIPSY!
HAUSUEBUNG
ENDKAPITAL= 2400 S
Ok
```

Nun weiß der Benützer aber noch nicht, was wir da eigentlich gerechnet haben. Also schreiben wir:

```
250 PRINT "KAPITAL=2000S, ZINS=4%, LAUF
ZEIT=5JAHRE"
```

Nun lassen wir den Computer wieder sein Programm ausführen.

```
HALLO! ICH BIN DEIN COMPUTER CHIPSY!
HAUSUEBUNG
KAPITAL=2000S, ZINS=4%, LAUFZEIT=5JAHRE
ENDKAPITAL= 2400 S
Ok
```

Zum Schluß versuchen wir uns noch mit sogenannten alphanumerischen Variablen. Diese "Dinger" haben ein $-Zeichen hinter ihrem Namen und speichern Zeichenketten.

NEW

Vorerst lernen wir aber noch den Befehl NEW kennen. Damit können wir Programme aus dem Speicher löschen. Einfach NEW 'CR' eingeben, und unsere Entwicklung ist weg. Mit LIST überprüfen wir, ob wirklich nichts mehr im Speicher steht:

```
LIST 'CR'
Ok
```

Es stimmt, nichts ist mehr drinnen! Nun haben wir die Möglichkeit, ein neues Programm zu entwerfen. Wir erstellen folgendes Programm:

```
10 PRINT "WIE HEISSEN SIE";
20 INPUT NAME$
30 PRINT "SIE SIND EIN SPECTRAVIDEO-FAN,
";NAME$
```

Sehen wir uns das Programm an. In der ersten Zeile wird der Text "WIE HEISSEN SIE" ausgegeben. Der Strichpunkt sorgt dafür, daß die nächste Ausgabe gleich am Ende der ersten fortsetzt. So wird das Fragezeichen des INPUT-Befehls direkt angehängt. Nach dem Start tippen wir einen Namen ein. Durch das ENTER erscheint flugs der nächste Satz, für den die Zeile 30 verantwortlich ist. In dieser Anweisung haben wir zuerst direkten Text zwischen den beiden Anführungszeichen. Das Semikolon kennen wir inzwischen. NAME$ ist die Variable, in welcher unser Name gespeichert ist. Daher wird auch unser eingegebenes Wort angezeigt. Sehen wir uns nun den Effekt in "Natura" an. Man sieht zuerst:

```
WIE HEISSEN SIE?
```

Nun geben wir unseren Namen ein. Nach ENTER haben wir folgendes am Bildschirm:

```
WIE HEISSEN SIE? HANS
SIE SIND EIN SPECTRAVIDEO-FAN, HANS
Ok
```

Natürlich beherrschen unsere SVI-Computer auch Groß- und Kleinschreibung. Wir haben aus Gründen der Übersichtlichkeit jedoch bisher alle Eingaben in Großbuchstaben geschrieben. BASIC-Befehle versteht unser Computer sowohl in Groß- als auch in Kleinschrift. Bei Strings (Text) erfolgt die Ausgabe entsprechend der Eingabe in Groß- und Kleinbuchstaben.

Auch die deutschen Umlaute sind kein Problem. Mit wenigen Programmzeilen hat man einen deutschen Zeichensatz.

Soweit unsere Experimente mit den ersten Befehlen in der Programmiersprache BASIC!

Eine kleine Bemerkung noch: Ab jetzt wird im Text das ENTER (oder 'CR') teilweise weggelassen. Der Programmierer muß aber weiterhin die Taste nach jeder Eingabe betätigen!

Alles wird teurer

3. Folge

Der eigentliche Zweck von Personal-Computern ist die Datenverwaltung, das Verarbeiten von Zahlen und Tabellen. Wir alle hätten gerne elektronische Heinzelmännchen, die uns das lästige Kassabuch- oder Inventarlistenführen erleichtern oder ersparen.

Nun, ganz erspart kann uns das Erstellen solcher Listen nicht werden. Wir können aber sehr wohl mit unserem Spectravideo-Computer die schönsten Hilfsprogramme, auch "Software-Werkzeuge" genannt, zu Hilfe nehmen, um solche Probleme zu lösen.

Mit den entsprechenden Programmen haben wir zum Beispiel die Möglichkeit, die Hausfrau bei ihrer Arbeit zu unterstützen. Sie braucht keine umständlichen Rechnungen mehr zu bewerkstelligen, wie viel Salz oder Rindfleisch gebraucht wird. Einfach das Rezept der Speise und die Anzahl der hungrigen Personen eingeben, in Windeseile präsentiert uns unser Helfer sein Ergebnis. Ebenso gibt es Werkzeuge, welche bei der Verarbeitung von Geldausgaben sehr hilfreich sein können.

Tools, Helfer im Speicher

Unter "Tools" verstehen wir Programmwerkzeuge. Dies sind Programme, welche den Bediener in einer bestimmten Hinsicht unterstützen. Wir werden so ein Werkzeug kennenlernen, das Tabellenkalkulationsprogramm CalcStar.

Diese Programme haben aber etwas besonderes an sich. Im wesentlichen laufen ja die diversen Lohnverrechnungen, Tabellenkalkulationen und so weiter immer nach dem gleichen Schema ab. Es sind nur geringe Unterschiede, die ein normales Programm nicht für alles verwendungsfähig machen. Die Tools hingegen kann man für seine Zwecke ohne Kenntnis einer Programmiersprache anpassen. Dabei werden natürlich keine BASIC-Befehle verwendet, sondern Kommandos einer höheren Ebene.

Diese Tools werden meistens auf Disketten oder Kassetten geliefert, manchmal gibt es sie auch in ROMs zu kaufen. Sie erinnern sich doch hoffentlich noch, ROM ist Read Only Memory, ein Nur-Lese-Speicher!

Tabellenkalkulation

Wir werden uns in dieser Folge mit einem "Tool" für Tabellenkalkulation "herumschlagen". Man kann mit dieser Art von Werkzeugen Zahlen und Daten in Tabellen verarbeiten und mit ihnen Rechenoperationen durchführen. Solche Programme werden manchmal auch Arbeitsblattprogramme (engl. Spreadsheed) genannt. Man kann mit ihnen Kalkulationen erstellen, Angebote schreiben, betriebliche Abschreibungen errechnen und anderes mehr. Auch ein Kassabuch läßt sich damit führen, unsere Familie Dischek führt damit ihr Haushaltsbuch.

CalcStar

Das Programm, das uns zur Verfügung steht, heißt CalcStar und läuft unter dem Betriebssystem CP/M. Wir wissen ja, ein großer Vorteil unseres Computers ist, daß das weitverbreitete Betriebssystem CP/M auf dem Spectravideo eingesetzt ist. CalcStar wird für CP/M auf Diskette geliefert.

Sehen wir uns einmal die Funktion dieses CalcStar an. Wie schon erwähnt, verarbeitet es Zahlen und Text in Zeilen- und Spaltenform, wie es bei Tabellen üblich ist.

Nach der Initialisierung sieht man den Abschnitt eines großen Datenblattes. Man kann natürlich nicht das ganze Blatt auf einmal zeigen, das würde den Umfang des Fernsehschirms sprengen. Statt dessen greift man zu der Technik, nur einen Teil zu zeigen. Man nennt diesen Ausschnitt auch "Fenster".

Wir können in diesem Datenblatt eine Einteilung erkennen. Das Programm kann 255 Zeilen und 127 Spalten verarbeiten. Nun wird ähnlich dem Schachfeld eine Adressierung der einzelnen Reihen vorgenommen. Dazu werden die Zeilen durch Zahlen und die Spalten durch Buchstaben von A bis DW gekennzeichnet. Nun kann man durch eine Zahl und einen Buchstaben jedes beliebige Feld auf dem Blatt ansprechen. Selbstverständlich ist die Breite dieser Felder variabel. Man kann durch Veränderung der Anzahl von Spalten die Breiten ändern. Man nennt diese kleinen Felder auch Zellen. Bild 3.1 zeigt Ihnen den Bildschirm, wie ihn CalcStar präsentiert.

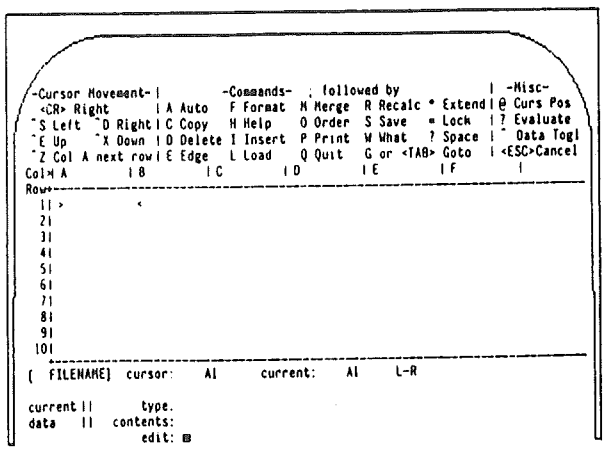


Bild 3.1

Nun darf man dieses Fenster natürlich über das ganze Blatt umherbewegen. Es ist ebenso möglich, in jede Zelle etwas hineinzuschreiben. Dabei ist es egal, ob sie Text, Zahlen oder sogar Formeln eintragen. Mit einem Cursor können Sie von Feld zu Feld springen

	A	B	C	D	E	F	G	H	I	J	K	L
1	**********											
2	* HAUSHALTSBUCH *											
3	**********											
4	----------											
5	JAHR :	1984										
6	MONAT:	6										
7	----------											
8	DATUM	EINKAUF	PREIS	SPALTE	BETRIEB	NAHRUNG	KLEIDER	KFZ	WOHNUNG	FREIZEIT	BERUF	DIVERSES
9	==========											
10	020684	Fleisch	224.00	2		224.00						
11	030684	Mantel/M	1600.00	3			1600.00					
12	040684	Vase	333.00	5					333.00			
13	050684	Meinl	34.00	2		34.00						
14	060684	Kino	123.00	6						123.00		
15	070684	Museum	50.00	6						50.00		
16	080684	Benzin	356.00	4				356.00				
17	090684	Miete	3000.00	1	3000.00							
18	100684	Hausvers	350.00	1	350.00							
19	110684	Meinl	234.00	2		234.00						
20	120684	Tasche	126.00	1	126.00							
21	130684	Strom	1000.00	1	1000.00							
22	140684	Telefon	876.00	1	876.00							
23	150684	Wein	899.00	2		899.00						
24	160684	Bücher	233.00	6						233.00		
25	170684	Bücher	455.00	7							455.00	
26	----------											
27			9893.00		5352.00	1391.00	1600.00	356.00	333.00	406.00	455.00	
28			==========									

Bild 3.2

und Ihre Eintragungen machen. Sie können aber mit bestimmten Befehlen auch größere Sprünge auf Ihrem Arbeitsblatt machen.

Was ist nun das Prinzip von CALCSTAR?

Wir können in die einzelnen Zellen Werte oder Formeln oder Text hineinschreiben. Das Programm kann dann Beziehungen zwischen den Zellen herstellen. Wenn wir zum Beispiel in A3 den Wert 3 einschreiben, in S4 2*A3 und in F6 2*S4, dann ersetzt unser Tool sofort die beiden Formeln durch den richtigen Wert. In S4 steht dann 6 und in F6 12. So einfach ist das.

Wenn wir nun unsere Tabelle erstellen wollen, können wir in die verschiedensten Stellen Texte schreiben, welche die Funktion von Erklärungen übernehmen. Weiters legen wir die Zellen fest, in die unsere Eingaben erfolgen sollen. Zu guter Letzt genehmigen wir noch dem Programm die Zellen, die CalcStar dann anhand unserer Eingaben ausrechnen soll (darin liegt eine der bedeutensten Hilfen dieses Programms). Man kennzeichnet diese Zellen mit Formeln. Der Computer setzt dann nur noch in diese Formeln ein.

Auch die Familie Dischek hat inzwischen das Computerfieber gepackt. Sie haben vor, ihren Haushalt zu "computerisieren". Dabei soll ihnen unser Werkzeug helfen.

Das Haushaltsbuch

Was soll das Haushaltsbuch der Familie Dischek beinhalten?

Vorerst ist klar, daß man die Ausgaben der einzelnen Bereiche kennen will. Dazu kommt noch ein Datum. Die Ausgaben werden in acht Gruppen eingeteilt. Diese Gruppen werden aus den einzelnen Spalten gebildet. Die Familie möchte folgende Unterteilungen in ihrem Programm haben:

1) Betrieb
2) Nahrung
3) Kleider
4) KFZ
5) Wohnung
6) Freizeit
7) Beruf
8) Verschiedenes

Nach jedem Monatsende sind alle Spalten ausgefüllt. Nun beginnt die Überprüfung. Der Computer gibt die Gesamtausgaben und die der Teilbereiche von einem Monat aus und stellt die Ausgaben der vorigen Monate zum Vergleich. So wird es deutlich ersichtlich, ob man mehr oder weniger verbraucht hat. Ein interessanter Beitrag zum Sparen.

Wir werden sehen, daß es halb so schlimm ist, mit diesem Tool zu arbeiten. Werfen wir nun einen Blick auf das Haushaltsbuch der Familie in Bild 3.2.

Am oberen Rand erkennt man zuerst die einzelnen Textzeilen. Man braucht sie vor allem dazu, daß der Benützer weiß, wovon überhaupt die Rede ist. Darunter können wir die Zellen erkennen, in die wir eingeben müssen. Die ersten vier Spalten müssen von uns ausgefüllt werden. Das Datum, die Ware, den Preis und die Spalte, in welche die Ausgabe gehört, müssen wir feststellen. Den Rest macht das Programm anhand unserer Formeln.

Sehen wir uns die erste Ausgabe vom 2.Juni 1984 an. Wir geben das Datum an und den Artikel. Danach wird der Preis eingegeben. Durch Bekanntgabe der Spaltennummer erkennt CalcStar, in welche Spalte eingetragen werden muß. In unserem Fall ist es die zweite Spalte. Diese Entscheidung wird durch bedingte Verzeigungen erzeugt.

Der Computer frägt bei diesen Bedingungen, ob ein bestimmter Ausdruck gleich einem anderen ist. Im BASIC werden wir dafür "IF...THEN" kennenlernen. Im CalcStar sieht das so aus:

```
+D10=2:+C10:0
```

Der Computer überprüft, ob die Zelle D10 den Wert 2 enthält. Ist dies der Fall, wird der Wert der Zelle C10 in das Feld geschrieben, welches die dargestellte Bedingung enthält. Hat D10 einen Wert ungleich 2, wird Null in die Zelle geschrieben (die Null am Ende ist dafür verantwortlich).

Selbstverständlich kann man auch andere Vergleichsoperatoren für die Bedingung verwenden. Viele werden vielleicht noch aus der Schulzeit die drei Operatoren kennen, aus denen sich alle anderen zusammensetzen:

```
A1 > A2   A1 muß größer als A2 sein
A1 < A2   A1 muß kleiner als A2 sein
A1 = A2   A1 muß gleich A2 sein
```

Nun kann man jeden Operator mit jedem verbinden. "<>" ergibt zum Beispiel ungleich. Wenn nämlich ein Wert entweder kleiner oder größer als ein anderer sein muß, dann ist er auf jeden Fall ungleich (beziehungsweise: nur die Zahl 2 mit dem gleichen Wert wie die Zahl 1 kann die Bedingung nicht erfüllen).

Neben den Vergleichen gibt es noch genug andere Funktionen. So ist es auch möglich, sämtliche arithmetische Rechenoperatoren zu verwenden. Doch nicht nur die allgemeinen, "normalen" Rechenoperatoren dürfen gebraucht werden. CALCSTAR stellt zusätzlich zu den genannten noch einige Sonderfunktionen zur Verfügung.

Man kann zum Beispiel Summen über Zeilen oder Spalten hinweg berechnen. So erstellt +SUM(A1>A20) die Summe von Zelle A1, A2 und so weiter bis A20. Ebenso ist es möglich, ganz leicht Minimums- oder Maximumsberechnungen durchzuführen.

Übrigens ist die Eingabe von Ziffern bei den Spectravideo-Computern SVI-328 und SVI-728 eine einfache Sache, denn es gibt ein separates Zifferneingabefeld auf der rechten Seite der Tastatur und die Eingabe kann wie auf einer Rechenmaschine erfolgen. Auch die Tasten für die Rechenoperationen und eine zweite ENTER-Taste befinden sich in dem separaten Zehnerblock.

CalcStar stellt ein Programmwerkzeug dar, mit dem sich eine große Zahl von Aufgaben, insbesondere solche, die sich im wirtschaftlichen Leben stellen, bewerkstelligen läßt, ohne daß man sich erst lange mit Programmiersprachen beschäftigen muß. Die Einarbeitung erfordert nur kurze Zeit und schon kann man das Tool einsetzen.

Welcher Schüler kennt nicht das tagelange "Strebern" von Vokabeln einer Fremdsprache, das staubtrockene Eintrichtern von unverständlichen, unaussprechbaren Buchstabenansammlungen. Der Computer kann diese "fade" Arbeit lustiger und abwechslungsreicher gestalten. Doch nicht nur für die Schule ist dieses Programm zu gebrauchen. Familien, die in ferne Länder reisen, werden gut daran tun, die jeweilige Fremdsprache zu erlernen. Der Spectravideo hilft dabei.

Wir werden ein Programm erstellen, welches den Benützer ein Vokabel nach dem anderen abfrägt. Dabei werden wir erkennen, daß wir mit ganz einfachen Mitteln ein sehr nützliches Programm erstellen können. Zusätzlich lernen wir noch einige neue Befehle des BASIC kennen. Neben der wichtigen IF...THEN-Anweisung finden wir auch GOTO in dieser Folge. Vorerst jedoch beschäftigen wir uns ausführlich mit den Variablen und was es mit ihnen auf sich hat.

Notizblock im Computer

Klären wir zunächst einmal den Begriff der Variablen allgemein. Bis jetzt haben wir den Nutzen dieser Einrichtung ja nur wage umrissen.

Die Variable

Man kann sich die Variable in der Programmiersprache BASIC als Behälter vorstellen. In diesem Behälter können entweder Zahlen oder Zeichen untergebracht sein. Man kann mit diesen Variablen aber auch genauso hantieren, wie mit absoluten Zahlen. 4+3 könnte ebenso A+B heißen, wenn A den Wert 4 und B den Wert 3 hätte. Der große Vorteil von Variablen ist jedoch, daß ihr Inhalt, wie der Name schon sagt, variabel ist. Wir werden die Variablen manchmal als Behälter und manchmal als Zahlen ansehen (wenn wir wissen, welche Zahl in der Variablen steht). Durch diese Unterscheidung wird es uns erleichtert, die Variablen zu verstehen.

Variablennamen

Selbstverständlich braucht unsere Variable auch einen Namen. Wie sollten wir sie sonst von anderen unterscheiden. Das BASIC läßt hier wieder sehr viel Spielraum für die Wahl des Namens.

Grundsätzlich darf ein Variablenname beliebig lang sein. Zur Unterscheidung werden jedoch nur die ersten zwei Buchstaben verwendet. Demnach sind die beiden Variablen DER und DEI für den Computer ein und die selbe, nämlich die Variable mit dem Namen DE. Der Vorteil, daß man mehr Buchstaben verwenden darf, als vom Computer verarbeitet werden, liegt einzig und allein in der Verständlichkeit der Namen. Unsere erste alphanumerische Variable hieß NAME$. NAME ist sehr deutlich. Der Betrachter des Programmlistings (man nennt das mit LIST auf den Bildschirm geholte Programm ein Listing) weiß sofort, daß hier ein Name in die Variable gespeichert wird.

Weiters muß das erste Zeichen ein Buchstabe sein. Alle weiteren Zeichen sind beliebig. Ausgenommen sind lediglich "!", "#", "$" und "%". Es können ohne weiteres auch Zahlen im Namen vorkommen. Die vier genannten Zeichen dürfen insofern nicht verwendet werden, weil sie festlegen, welche Type die Variable hat. Im Namen haben sie nichts verloren.

Aufpassen muß man nur, daß der Variablenname kein Befehlswort aus dem BASIC enthält. WORT ist ein unzulässiger Name, weil OR eine BASIC-Funktion ist. Wohl aber kann man WOT verwenden.

Wir haben von verschiedenen Typen gesprochen, was wurde damit gemeint?

Numerische Variable

In diese Behälter dürfen nur Zahlen eingefüllt werden. Man kann diese Variablen auch zu Berechnungen verwenden.

Es gibt hier drei Unterteilungen:

Die Integer-Variable: Sie wird durch das Prozentzeichen (%) gekennzeichnet. Nur ganzzahlige Werte im Bereich zwischen -32768 und 32767 dürfen eingespeichert werden. Die Integervariable verbraucht im Speicher den wenigsten Platz, dafür ist sie auch die ungenaueste Type.

Die einfachgenaue Variable: Sie wird durch das Rufzeichen (!) gekennzeichnet. Man darf Dezimalzahlen aus dem ganzen Zahlenbereich des Computers einspeichern. Die Gesamtanzahl der Stellen einer Zahl ist jedoch auf 6 beschränkt.

Die doppeltgenaue Variable: Bei den Spectravideo-Computern wird sie automatisch voreingestellt. Setzt man daher kein Zeichen hinter den Namen, wird eine doppeltgenaue Variable angenommen. Gekennzeichnet wird sie durch ein #-Zeichen. Diese Type verbraucht den meisten Speicherplatz, sie ist jedoch die Genaueste. 14 Stellen einer Zahl aus dem gesamten Zahlenbereich des Computers können eingespeichert werden.

String (=Text)-Variable

Es dürfen in diesen Typ Zeichen jeder Art eingegeben werden. Bis zu 255 Zeichen faßt eine dieser Variablen. Man kann mit diesen Behältern jedoch keine Rechenoperationen durchführen. Selbst wenn nur Ziffernzeichen in einer dieser Variablen zu finden sind, können keine Multiplikationen oder Divisionen durchgeführt werden.

Das einzige, was erlaubt ist, nennt sich Verkettung. Man kann zwei Variablen aneinanderstückeln. "A$=A$+B$" bedeutet, daß an die Variable A$ die Variable B$ angehängt wurde.

Außerdem kann man noch alphanumerische Variablen für Vergleiche in der "IF...THEN"-Anweisung gebrauchen.

Gekennzeichnet wird diese Art durch das altbekannte Dollar-Zeichen ($).

Die oben genannten Typenkennzeichen werden immer am Schluß des Variablenamens angehängt:

```
NAME$   NAME ist der Name
        $ das Zeichen für Stringvariablen
```

Jetzt stürzen wir uns aber endgültig auf die nächsten BASIC-Anweisungen.

GOTO

Wie wir schon wissen, arbeitet der Computer die einzelnen Programmzeilen von der kleinsten bis zur größten nacheinander ab. Doch was, wenn wir einige überspringen oder wiederholen wollen?

Dazu können wir GOTO verwenden! Das Befehlswort setzt sich aus den Englisch-Vokabeln GO und TO zusammen, was übersetzt soviel wie "gehe zu" heißt. Nach dem Befehlswort schreiben wir die Zeilennummer, die angesprungen werden soll. "Gehe zu 240" kann man den Befehl "GOTO 240" im Deutschen übersetzen. Uns ist schon längst klar, daß der Computer nun mit so einem Sprungbefehl an jede beliebige Stelle des Programms springen kann. Man nennt "GOTO" auch einen "unbedingten Sprung"-Befehl, weil der Interpreter beim Abarbeiten eines solchen Befehles auf jeden Fall springt. Wir erzeugen nun ein Programm, welches in eine numerische Variable eine Zahl einspeichert, diese mit 4 multipliziert und das Ergebnis anzeigt. Danach wird das Programm automatisch wiederholt, weil wieder an den Anfang zurückgesprungen wird.

```
10 INPUT A
20 LET B=A*4
30 PRINT B
40 GOTO 10
```

Zum Befehl LET: Wir dürfen ohne weiteres das Befehlswort weglassen. So ist es möglich, statt "LET B=A*4" nur "B=A*4" zu schreiben.

```
10 INPUT A
20 B=A*4
30 PRINT B
40 GOTO 10
```

IF...THEN...ELSE

GOTO ist ein unbedingter Sprung, es gibt aber auch die Möglichkeit, zuerst zu fragen, ob man einen bestimmten Befehl wirklich ausführen soll. Man stellt dazu eine Bedingung und es wird zum Beispiel nur dann gesprungen, wenn diese erfüllt ist. Realisieren kann man diese Bedingung durch IF...THEN.

Hierzu erweitern wir unser Programm von vorhin mit einer Zeile 25:

```
25 IF B=8 THEN PRINT "FALSCHE EINGABE":
GOTO 10
```

Wenn der Computer bei der Abarbeitung des Programms zur Zeile 25 kommt, wird zuerst geprüft, ob die Variable B den Wert 8 enthält. Wenn nicht, dann wird die nächste Zeile bearbeitet. Ist die Bedingung erfüllt, dann werden die Befehle hinter THEN ausgeführt. Zuerst wird "FALSCHE EINGABE" ausgegeben und dann zum Anfang zurückgesprungen.

Der IF...THEN-Befehl teilt sich in folgende Teile:

IF Bedingung THEN Anweisung(en) ELSE Anweisung(en)

In der Bedingung können folgende Vergleichsoperatoren verwendet werden:

> < >= <= = <>

Wenn nun die Bedingung erfüllt ist, dann wird der THEN-Zweig mit seinen Anweisungen ausgeführt. Wenn der Vergleich nicht positiv ausgefallen ist, dann wird der ELSE-Zweig abgearbeitet. Leider hat der Computer, den die Familie Dischek besitzt, kein ELSE, der Spectravideo hat es. Deshalb werden wir das ELSE nicht verwenden. Beim Spectravideo darf man es ja auch weglassen. Wenn kein ELSE in der Anweisung vorkommt, wird die nächste Zeile abgearbeitet.

Eine Feinheit gibt es noch zu sagen: Wenn das GOTO in einer IF...THEN-Anweisung direkt hinter dem THEN steht, kann das Befehlswort weggelassen werden. Statt IF A=B THEN GOTO 3 können wir auch IF A=B THEN 3 schreiben.

Unser Vokabelprogramm

Genug theoretisiert! Jetzt werfen wir uns in das harte Programmiererleben. Gefragt wurde nach einem Vokabelprüfprogramm. Wir zeigen zuerst, wie es sich meldet, und dann, wie wir es erstellen. In Bild 4.1 sehen wir die Meldung:

```
DIESES PROGRAMM PRUEFT DEINE
KENNTNISSE ITALIENISCHER VOKABELN.

BEIM START ERHAELTST DU DIE FRAGE,
OB DU ITALIENISCH ODER DEUTSCH
GEPRUEFT WERDEN MOECHTEST.

NACH DEINER WAHL, FALSCH ODER RICHTIG,
WIRD DIR DIE RICHTIGE ANTWORT ANGEZEIGT.

DAS PROGRAMM MISCHT ALLE DATEN,
BEVOR ES EINE AUSWAHL TRIFFT.

ZUM BEGINN DRUECKE 'ENTER' ■
```

Bild 4.1

```
ES SIND DERZEIT 67 WORTE GESPEICHERT



DATEN GEMISCHT

DEUTSCHE ODER ITALIENISCHE WOERTER
('D' ODER 'I') ? ■
```

Bild 4.2

Man erfährt, was das Programm macht und ob man Deutsch / Italienisch oder Italienisch / Deutsch abgeprüft werden möchte. Nun müssen wir uns entscheiden (Bild 4.2). Nach unserer Eingabe "mischt" der Computer die Vokabeln und holt sich eines ganz zufällig heraus. Nun muß man das richtige Äquivalent der anderen Sprache eingeben. Wenn es richtig ist, quittiert der Computer die Eingabe mit einem Ok, stimmt das Wort nicht, zeigt der Computer das richtige Wort an. Zusätzlich bildet der Computer auf jeden Fall die Zahl der noch zu fragenden Vokabeln ab.

Das richtige Programmieren

Wir werden nun das Struktogramm kennenlernen. Diese Einrichtung gehört zu der Gruppe der Programmablaufpläne und hilft dem Bediener, sein Programm effizienter und schneller zu programmieren. Es stellt nämlich die Funktionsgruppen eines Programms vereinfacht als Blöcke dar. Wir werden später noch genauer den Aufbau von Struktogrammen erläutern. In Bild 4.3 sehen wir das Struktogramm für unseren "Übersetzer". Zuerst wird der Benützer begrüßt. Wir kennen ja schon die Anfangsmeldung. Danach erklärt sich das Programm von selbst durch einige Sätze, uns auch schon bekannt! Nachdem wir unsere Entscheidung getroffen haben, werden die Vokabeln gemischt. Der eigentliche Hauptteil des Programms ist in zwei Schleifen untergebracht. Je nachdem, ob wir D/I oder I/D gewählt haben, wird eine der beiden Schleifen vom Rahmenprogramm aktiviert.

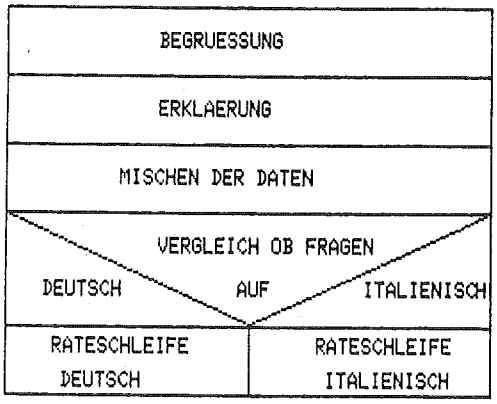


Bild 4.3

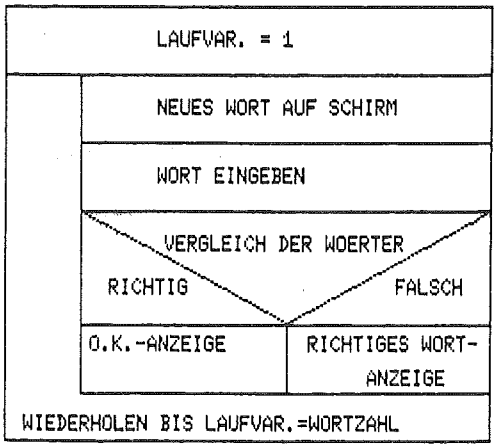


Bild 4.4

Die Rateschleifen gleichen sich im Struktogramm vollkommen (Bild 4.4), nur im BASIC ergeben sich leichte Unterschiede. Sehen wir uns das Gerüst an:

Zuerst wird ein Index auf Eins gesetzt, der die schon abgefragten Vokabeln zählt. Danach zeigt der Computer dem Bediener ein Vokabel an. Der Benützer gibt seine Antwort ein. Nun kommt wieder ein Vergleich: Wenn die Eingabe gleich dem zur Ausgabe gehörigen Wort ist, dann wird Ok ausgegeben. Andernfalls sehen wir das richtige Wort auf dem Bildschirm. Der Index wird um Eins erhöht, und das Ganze fängt wieder bei der Ausgabe eines Vokabels an, bis der Index gleich einer Paarzahl ist. Diese Paarzahl gibt an, wie viele Vokabel abzufragen sind.

Zusätzlich zu dieser Übersicht in Form eines Struktogramms sehen wir uns noch einige Spezialitäten aus dem Programm an.

Die Vokabeln sind paarweise, immer Deutsch-Italienisch gemischt, abgelegt. Zu diesem Zweck wurden DATA-Zeilen eingespeichert.

READ, DATA

Die Anweisung DATA reserviert Daten für den Computer im Programm. Mit READ können dann diese Daten in Variablen eingespeichert und somit aktiviert werden. Vorher sind sie für den Ablauf des Programms unbrauchbar. Diese Art des Datenbereitstellens wird immer dann gemacht, wenn man einige Informationen bei jedem Programmablauf braucht. Die Daten von Variablen verschwinden, wenn man das Programm stoppt und neu startet. Die DATA-Informationen gehen nicht verloren.

```
10000 DATA WARTEN,ASPETTARE,ZENTRUM,CENTRO
10010 DATA AMPEL,IL SEMIFORIO,AUSSTEIGEN,
SCENDERE,SAGEN,DIRE,TRINKGLAS,IL BICCIERE
10020 DATA WOHNZIMMER,IL SODDIORNO,PFANNE,
LA TEGLIA,REGEN,LA POGGIA
10030 DATA HONIG,IL MIELE,KENNEN,CONOSCERE,
ZUKUNFT,IL FUTURO
.
.
.
10200 DATA Z,Z:REM LETZTES WORT
```

Weiters sind die beiden Rateschleifen deutlich erkennbar. So ist die Rateschleife deutsch von 6044 bis 7700 und die italienische von 7800 bis 8200 abgelegt.

Sehen wir uns nun stellvertretend für die beiden Schleifen die deutsche an.

Das Kommando INPUT A$ in Zeile 7500 sorgt dafür, daß unser eingegebenes Wort in die Variable A$ abgelegt wird.

Der Vergleich zwischen den beiden Vokabeln, dem von uns vorgeschlagenen und dem vorgesehenen, wird in Zeile 7556 realisiert. Dabei ist A$ unser eingegebenes Wort, E$(X) enthält das richtige Vokabel.

Im IF...THEN-Befehl steht dann auch die Meldung, daß die beiden Wörter gleich sind. Wenn A$=E$(X), dann wird der THEN-Zweig ausgeführt, daß heißt, die Wörter sind gleich.

Wenn das Vokabel falsch geraten wurde, wird die Zeile 7600 bearbeitet, welche für die Falschmeldungen zuständig ist.

Am Spectravideo hätte man diese Problemstellung mit ELSE lösen können, der Computer der Familie Dischek muß sich leider mit

dieser Variante helfen. Gekoppelt sieht das nun so aus:

```
.
.
.
7556 IF A$=E$(X) THEN PRINT "O.K.";:FOR E=1
TO 550:NEXT E:NEXT T:GOTO 300
7600 ......:PRINT E$(X):....
.
.
.
```

Bei 7556 wird auf Richtigkeit des Vokabels überprüft, wenn für die Eingabe der Vergleich positiv ausfällt, arbeitet der Computer den THEN-Zweig ab, andernfalls setzt er mit 7600 fort.

```
1  REM
2  REM  DEUTSCH-ITALIENISCH VOKABELTEST
3  REM  ================================
5  DIM D(700),D$(700),E$(700)
50 REM
60 REM       BESCHREIBUNG
70 REM       ============
82 CLS
85 PRINTTAB(5) "DIESES PROGRAMM PRUEFT DEINE
"
87 PRINT"     KENNTNISSE ITALIENISCHER VOKAB
ELN." : PRINT
88 PRINT"BEIM START ERHAELTST DU DIE FRAGE,"
: PRINT "OB DU ITALIENISCH ODER DEUTSCH "
89 PRINT "GEPRUEFT WERDEN MOECHTEST."
91 PRINT:PRINT "NACH DEINER WAHL, FALSCH ODE
R RICHTIG,"
92 PRINT "WIRD DIR DIE RICHTIGE ANTWORT ANGE
ZEIGT"
93 PRINT:PRINT "DAS PROGRAMM MISCHT ALLE DAT
EN,": PRINT "BEVOR ES EINE AUSWAHL TRIFFT."
94 PRINT : INPUT "ZUM BEGINN DRUECKE 'ENTER'
";A$
300 REM
350 REM       BEGINN
360 REM       ======
370 REM
400 CLS : LOCATE 7,12: PRINT "MISCHEN DER DA
TEN":::
800 REM
990 REM ANZAHL DER WORTPAARE
998 REM ====================
999 REM
1000 FOR X = 1 TO 5000
1100 READ D$,E$
1105 IF D$ = "Z" THEN 1310
1111 D$(X) = D$:E$(X) =E$
1300 NEXT X
1310 X = X - 1: RESTORE
1320 REM
1340 CLS:LOCATE 1,4: PRINT "ES SIND DERZEIT"
;X;"WORTE GESPEICHERT": PRINT:PRINT: Q = X
2500 REM
3000 REM UMORDNEN DER WORTE
3010 REM ==================
3020 REM
5000 FOR F=1 TO Q
5010 D(F) = F
5020 NEXT F
5030 FOR K = 1 TO INT ((9/10) * Q)+2
5040 R = INT ( RND (-TIME) *Q) + 1
5045 S = INT ( RND (-TIME) *Q) + 1
5050 SWAP D(R),D(S)
5080 NEXT K
5090 REM
5095 REM
6000 LOCATE 11,14: PRINT "DATEN GEMISCHT":PR
INT:PRINT
6030 PRINT" DEUTSCHE ODER ITALIENISCHE WOERT
ER": INPUT" ('D' ODER 'I') ";C$
6035 IF C$ = "D" OR C$ ="I"THEN GOTO 6040ELS
E GOTO 6030
6040 IF C$ = "I" THEN 7800
6044 REM
6045 REM RATESCHLEIFE DEUTSCH
6046 REM ====================
6047 REM
6050 FOR T = 1 TO Q
6070 X = D(T)
6090 B = Q -T
7500 CLS : LOCATE 0,14:PRINTB;" REST" :LOCAT
E 11,3: PRINT D$(X): LOCATE 14,12 :PRINT "->
 ";: INPUT A$
7550 REM
7555 REM VERGLEICH ***************
7556 IF A$ = E$(X) THEN PRINT "O.K.";:FOR E
= 1 TO 550:NEXT E: NEXT T: GOTO 300
7600 LOCATE 14,16 : PRINT "                ":L
OCATE 14,16: PRINT E$(X): FOR E = 1 TO 550:
NEXT E
7700 NEXT T: GOTO 300
7800 REM
7801 REM RATESCHLEIFE ITALIENISCH
7802 REM ========================
7803 REM
7809 FOR T = 1 TO Q: X = D (T)
7810 B = Q - T
7900 CLS : LOCATE 0,15: PRINT B;" REST": LOC
ATE 11,3: PRINT E$(X): LOCATE 14,12 :PRINT "
-> ";: INPUT A$
7950 REM
7955 REM VERGLEICH ************
7956 IF A$ = D$(X) THEN PRINT " RICHTIG";: F
OR E = 1 TO 550: NEXT E: NEXT T:GOTO 300
8000 LOCATE 14,16: PRINT "                ":LO
CATE 14,16 : PRINT D$(X): FOR E = 1 TO 550 :
 NEXT E
8100 NEXT T: GOTO 300
8200 END
10000 DATA WARTEN,ASPETTARE,ZENTRUM,CENTRO
10010 DATA AMPEL,IL SEMAFORIO,AUSSTEIGEN,SCE
NDERE,SAGEN,DIRE,TRINKGLAS,IL BICCHIERE
10020 DATA WOHNZIMMER,IL SODDIORNO,PFANNE,LA
 TEGLIA,REGEN,LA POGGIA
10030 DATA HONIG,IL MIELE,KENNEN,CONOSCERE,Z
UKUNFT,IL FUTURO
10040 DATA HOTEL,L'ALBERGO,DANN,ALLORA,VORWA
ERTS,AVANTI,WO,DOVE;UND,E
10050 DATA ABER,MA,NACHHER,POI,DAME,LA SIGNO
RA,HERR,IL SIGNORE
10060 DATA STRASSE,LA VIA,NAH,VICCINO,BIS,FI
NO
10070 DATA NACH RECHTS,A DESTRA,NACH LINKS,
A SINISTRA,DANKE,GRAZIE
10080 DATA NICHT,NON,BITTE,PER FAVORE,ENTSCH
ULDIGEN SIE,SCUSI
10090 DATA JA,SI,WIR SIND,SIAMO,GEPAECK,I BA
GAGLI,TASCHE,LA BORSA
10100 DATA ZIMMER,CAMERA,BETT,LETTO,AUTO,LA
MACCHINA
10110 DATA VIEL,MOLTO,REISEPASS,IL PASSAPORT
O,PLATZ,LA PIAZZA,IMMER,SEMPRE
10120 DATA BAD,I SERVICI,ALLES,TUTTO,ICH,IO,
SIE,LEI,MIT,CON
10130 DATA NEIN,NO,ICH HABE,HO,ER HAT,HA,ICH
 BIN, SONO
10140 DATA VORBESTELLT,PRENOTATO,ZWEI,DUE,BA
NK,LA BANCA,GUT,BUONO
10150 DATA WARM,CALDO,RECHNUNG,IL CONTO,SPEI
SEEIS,IL GELATO
10200 DATA Z,Z : REM  LETZTES WORT
```

Wäre es nicht all zu schön, einen Totozwölfer zu tippen? Doch die Chance ist nur zu gering, dies richtig zu bewerkstelligen. Es sei denn, man hat einen Computer zur Verfügung, der uns ein "Tip"-Programm erstellt.

Wir müssen lediglich einen Algorithmus dafür finden, wie wir unseren Totoschein am Besten ausfüllen könnten. Gute Totospieler haben nämlich bewußt oder unbewußt immer ein bestimmtes System, nachdem sie ihre Scheinchen anzeichnen. Der Computer könnte viel gründlicher nach den jeweiligen Systemen vorgehen und die Chance eines "Zwölfers" erhöhen.

Das Struktogramm

Um nun unser Programm effizient und schnell zu erstellen, werden wir uns wieder mit dem Struktogramm beschäftigen. Doch wir wissen noch nicht einmal, welche Symbole für Struktogramme zugelassen sind! Deshalb sehen wir uns jetzt einmal alle Varianten dieses Ablaufplanes an.

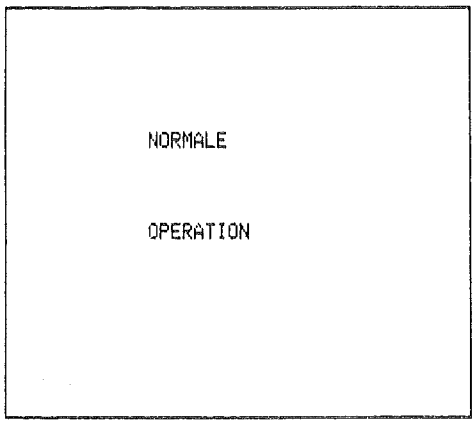


Bild 5.1

Zuerst widmen wir uns dem einfachen Rechteck. Wir brauchen es, um ganz normale Operationen zu kennzeichnen (Bild 5.1). Unter ganz normal verstehen wir alles das, was nicht ausdrücklich durch andere Symbole dargestellt wird.

Als nächstes finden wir das Vergleichsymbol in Bild 5.2. Im verkehrt stehenden Dreieck steht die Bedingung. Ist sie erfüllt, wird der "JA"-Zweig ausgeführt, ist sie nicht positiv, wird der "NEIN"-Zweig bearbeitet. Die Zweige selber bestehen dann wieder aus einer Ansammlung einzelner Symbole (weitere Operationen).

Weiters finden sich noch zwei Arten von Schleifen in unserem Repertoire. Die eine wird solange ausgeführt, solange eine Bedingung gilt, die andere jedoch hört erst dann auf, wenn die Bedingung erfüllt ist. In Bild 5.3 sehen wir die erste der beiden. Oberhalb des Schleifenkörpers ist die Bedingung angegeben, darunter liegen die Anweisungen,

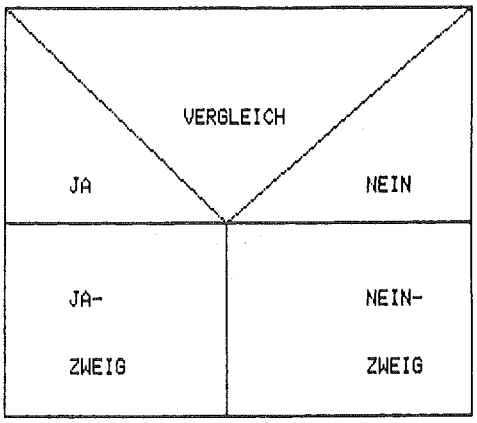


Bild 5.2

welche solange wiederholt werden sollen, bis der Vergleich negativ ausfällt.

In Bild 5.4 ist das Gegenteil zu sehen. Die Bedingung findet sich im Schleifenkörper wieder. Die Schleife selber wird nur dann ausgeführt, wenn der Vergleich negativ bleibt.

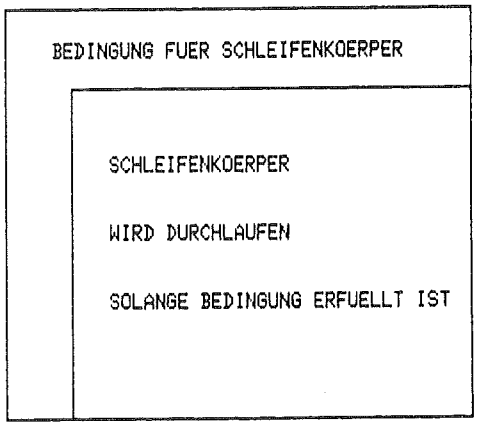


Bild 5.3

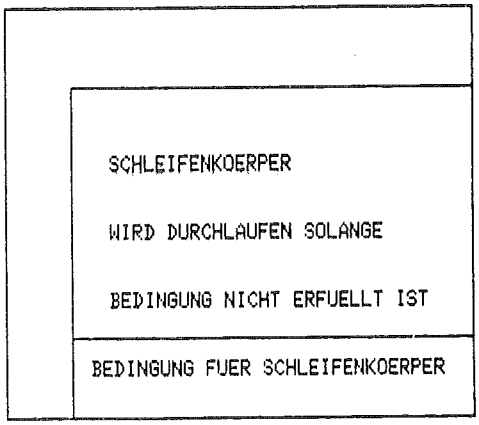


Bild 5.4

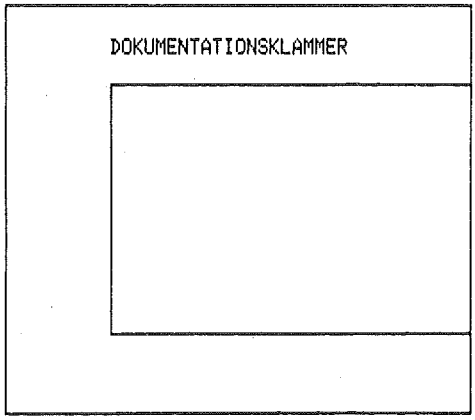


Bild 5.5

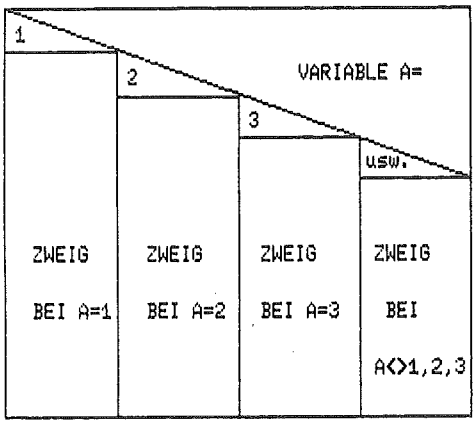


Bild 5.6

Zuletzt gibt es noch die Dokumentationsklammer (Bild 5.5) und die CASE-Anweisung (Bild 5.6).

Die Klammer brauchen wir, um irgendwelche erklärenden Texte im Struktogramm einzufügen. Die CASE-Anweisung prüft eine Variable auf ihren Wert. Dann führt sie einen der darunter angegeben Zweige aus. Wenn die Variable den Wert 1 hat, wird der Zweig 1 bearbeitet, bei der Zahl 2 der Zweig 2 und so weiter.

Wenn man mehrere Anweisungen hintereinander anführt, dann spricht man von einer Sequenz (Bild 5.7).

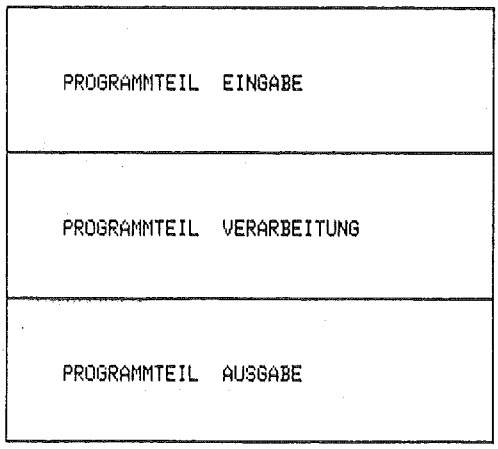


Bild 5.7

Nach der Erklärung der Struktogrammsymbole widmen wir uns wieder einigen BASIC-Befehlen. Zuerst sehen wir uns FOR...NEXT an, danach kommt DIM an die Reihe.

FOR. . .NEXT

Dies ist neben den schon genannten Befehlen eine der wichtigsten Anweisungen im BASIC überhaupt. Sie wird gebraucht, um eine Schleife zu erzeugen. Wenn wir einen Programmteil mehrmals hintereinander ablaufen lassen wollen, dann stellen wir ihn zwischen den FOR- und den NEXT-Befehl, zum Beispiel

```
FOR I=1 TO 3  STEP 1
```

Hinter dem Befehlswort FOR befindet sich die Laufvariable des Befehls, die angibt, wie oft noch durchlaufen werden muß, beziehungsweise wie oft schon durchlaufen wurde. Danach folgen der Anfangswert für I und der Endwert für I. Nach jedem Schleifendurchgang wird I um eins erhöht. Wenn ein STEP mit einer Zahl ungleich eins angegeben ist, so wird immer der Wert hinter STEP dazuaddiert. Das STEP ist nicht notwendig, wenn um eins erhöht werden soll.

Wenn der Computer beim Abarbeiten eines Programms auf einen FOR...TO-Befehl trifft, wird die Laufvariable auf den Anfangswert gesetzt. Danach werden solange Befehle ausgeführt, bis der Interpreter auf ein NEXT trifft. Daraufhin springt der Computer wieder zum FOR...TO-Befehl und erhöht die Laufvariable um den Wert von STEP (wenn STEP einen negativen Wert hat, wird die Variable erniedrigt). Dies geht solange vor sich, bis die Laufvariable den Endwert erreicht hat. Dann wird das NEXT ignoriert und die folgenden Programmteile werden verarbeitet.

Man darf Schleifen auch verschachteln. Dabei muß man aber immer achten, daß sich die Teile nicht überkreuzen. Eine Schleife muß die andere völlig umgreifen. In Bild 5.8 ist ein falsches Beispiel und in Bild 5.9 ein richtiges angegeben.

falsch

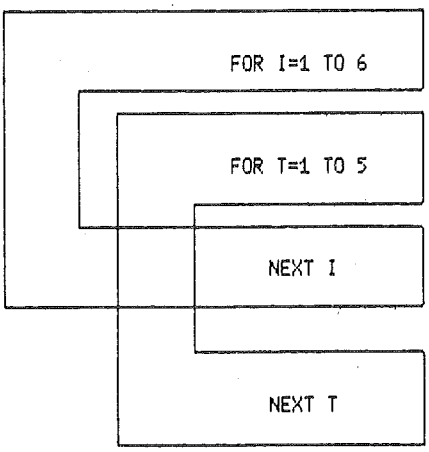


Bild 5.8

richtig

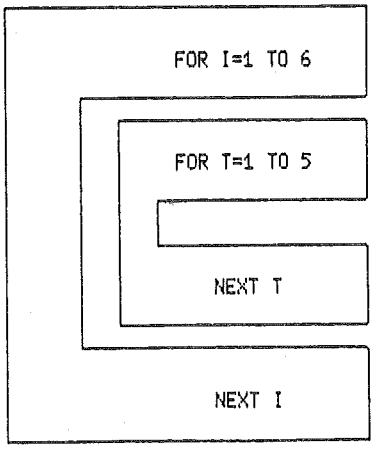


Bild 5.9

DIM

Die nächste Anweisung ist DIM. Wir brauchen sie, um Variablenfelder zu dimensionieren. Da wirft sich nun die Frage auf, was sind diese Felder überhaupt?

Alles über Felder

Wir haben im BASIC die Möglichkeit, mehrere Variablen mit einem Namen "herzustellen". Diese Variablen unterscheiden sich dann nur mehr durch eine Zahl in Klammern hinter dem Namen. Der große Vorteil dieser Felder ist, daß man mit einem einzigen Befehl auf alle Teile dieses Feldes zugreifen kann. Man braucht nur die Zahl in der Klammer zu ändern. Bei INPUT A können wir das A nicht auf ein B ändern, bei diesem Kommando wird immer nur die eine Variable A geladen. Doch bei INPUT A (I) können weit mehr Zellen angesprochen werden. Wir müssen nur die Variable I jeweils ändern. Hier liegt der Nutzen der Felder.

Wir haben außerdem noch die Möglichkeit, Felder nicht nur in einer Linie zu dimensionieren. Man kann auch zwei-, drei- und mehrfach dimensionieren. Wir brauchen nur die Klammer mit mehr Zahlen erweitern. Man kann sich dabei die einzelnen Dimensionen wie die Raumdimensionen vorstellen. Demnach ergeben einfache Felder eine Linie von Laden (die Variablen sind als Laden gemeint), zweifach dimensionierte werden als Fläche angesehen und die dreifachen kann man sich als Raum vorstellen. Die drei Darstellungen in Bild 5.10 sollen dies verdeutlichen. Weitere Dimensionen können bekanntlich nicht dargestellt werden, oder haben Sie schon in vier Dimensionen gesehen oder gedacht?

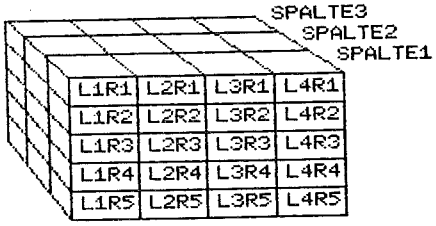
einfach

L1	L2	L3	L4

zweifach

L1R1	L2R1	L3R1	L4R1
L1R2	L2R2	L3R2	L4R2
L1R3	L2R3	L3R3	L4R3
L1R4	L2R4	L3R4	L4R4
L1R5	L2R5	L3R5	L4R5

dreifach



dimensioniert

Bild 5.10

Felder bis 10 Zellen kann man ohne DIM verwenden, ab dann muß mit dieser Anweisung das Feld reserviert werden.

DIM ist nur dafür verantwortlich, daß für eine bestimmte Variable ein Feld mit einer bestimmten Anzahl von Zellen definiert werden kann. Das Format von DIM:

DIM Variable (Anzahl), Variable (Anzahl),...

Nun werfen wir uns wie jedesmal wieder in den harten praktischen Programmiereralltag. Wir werden das Totoprogramm mit den dazugehörigen Struktogrammen erstellen.

Das Totoprogramm

Wir haben in unserem Programm das Ergebnis des jeweils vorigen Totozwölfers eingespeichert. Zusätzlich liegen noch fünf weitere Ergebnisse der einzelnen Mannschaften vor. Die Tordifferenz wird bei jedem Spiel selbstverständlich auch angegeben. Wie bei richtigen Scheinen vermerken auch wir ein Unentschieden mit einem "X", einen Sieg mit einer "1" und eine Niederlage mit einer "2".

Im Verlauf des Programmes legt der Computer die Stärken der einzelnen Mannschaften fest. Für jede erkennbare Niederlage werden 10 Punkte von einem "Stärkekonto" abgezogen, bei einem Sieg 10 addiert. Bei einem Unentschieden bleibt der Wert jedoch gleich. Wenn eine Mannschaft mehr als 2 Tore Differenz herausgeholt hat, wird das Konto mit 20 % erhöht. Spielt ein Team auf seinem Heimatplatz, bekommt es 10 % Erhöhung der Punktezahl.

Wie struktuieren wir nun das Programm?

Dazu unterteilen wir das Struktogramm. Zuerst zeigen wir nur ein Baumdiagramm, welches schematisch die einzelnen Programmteile umreißt (Bild 5.11).

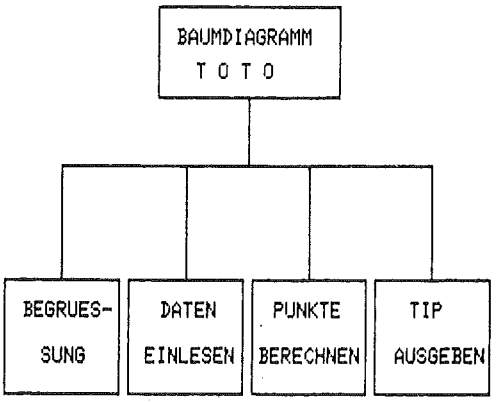


Bild 5.11

Nach der Begrüßung liest der Computer Daten ein. Danach berechnet der Interpreter das "Stärkekonto" einer Mannschaft. Zum Schluß gibt er seinen Vorschlag für den Tip ab.

Sehen wir uns nun die einzelnen Programmteile näher an.

Zuerst kommt das "Daten einlesen" an die Reihe. Die Begrüßung ist nur eine Ansammlung von einigen einfachen Befehlen, wir können daher die Erklärung guten Gewissens beiseite schieben.

In Bild 5.12 sehen wir das Struktogramm für "Daten einlesen".

Nachdem zwei Variablen, PAARE und PLAYZAHL, definiert wurden, fängt der Computer in einer Schleife an, die letzten Resultate samt Tordifferenz einzulesen. Hierbei werden einige Schleifen verschachtelt. Die Schleife, welche die einzelnen Tips verarbeitet, umrahmt eine weitere, die für die beiden Teams pro Spiel verantwortlich ist. Pro Tip wird nun pro Manschaft die Stärke ermittelt. Die dritte Schleife ist dann endgültig für die Stärke eines Teams zuständig.

Das nächste Kapitel lautet "Punkte berechnen" (Bild 5.13).

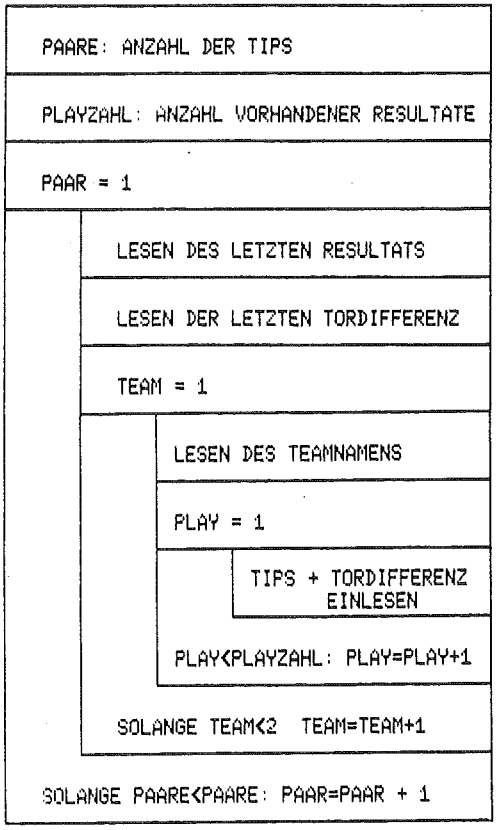


Bild 5.12

Nun wird pro Tip die Stärke der beiden Gegner verglichen. Das Unterprogramm "PUNKTEBERECHNUNG" (Bild 5.14) sorgt dabei dafür, daß die einzelnen Daten, die in Teil 1 eingelesen wurden, nun auf das "Stärkekonto" gebucht werden. Dabei werden alle oben genannten Faktoren berücksichtigt.

Der letzte Teil nennt sich "Tip ausgeben" (Bild 5.15).

Nach der Berechnung der einzelnen Tips muß nur noch das Ergebnis in einer ordentlichen Form an den Mann gebracht werden. Dafür ist dieser Teil zuständig. Wieder haben wir eine Schleife im Spiel. Für jeden Tip werden nun die Mannschaftsnamen, die errechnete Punktezahl, welche die Stärke der Mannschaft angibt, und das Ergebnis der computerinternen Beratung ausgegeben.

Interessant ist noch die Entscheidung, ob es ein Unentschieden werden könnte oder nicht. Der Computer orientiert sich hier

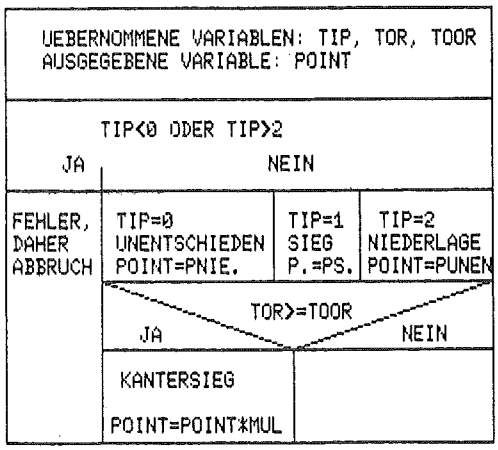


Bild 5.14

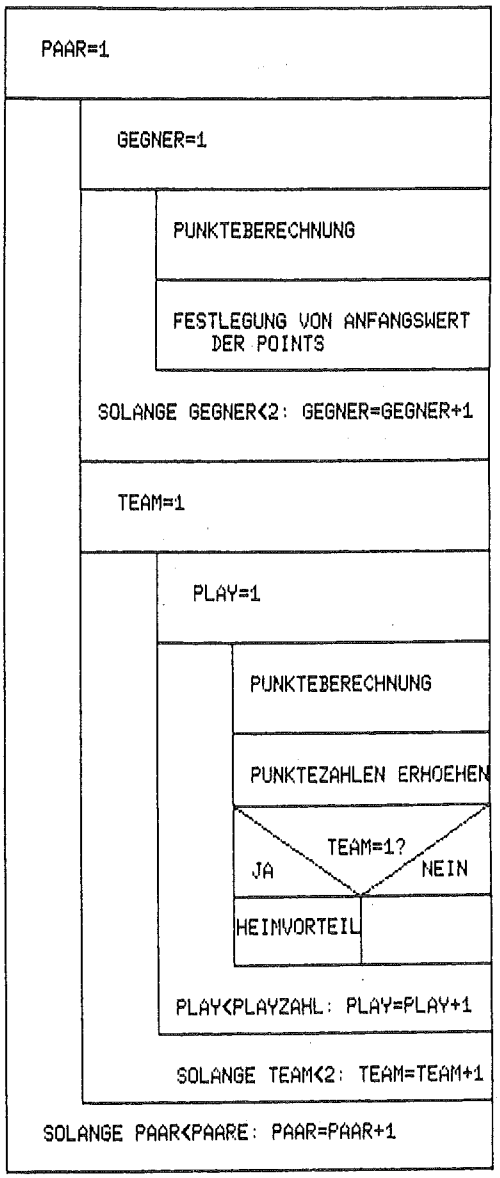


Bild 5.13

nicht daran, ob die Punktedifferenz Null ist, sondern ob eine bestimmte Schwelle unterschritten ist. Der Programmierer hat nämlich von der Überlegung auszugehen, daß eine Abfrage auf Gleichheit der Punkte zu genau ist. Schon Mannschaften, die mit den anderen Teams halbwegs gleich sind, können gegen sie unentschieden spielen.

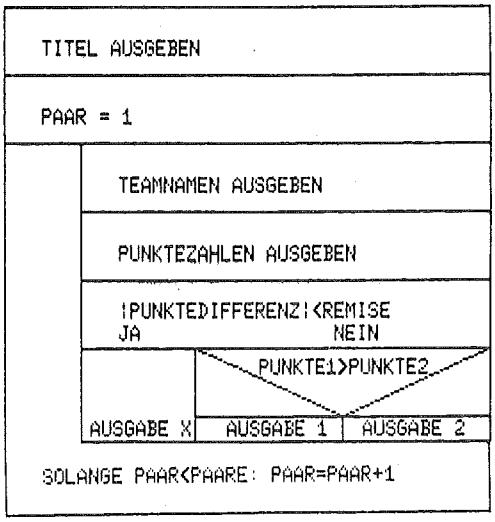


Bild 5.15

Deshalb wird eine Schranke angenommen, welche ungefähr die Grenze der Chance angibt, ein Unentschieden zu erzielen. Nach dieser Schranke wird verglichen, wenn das Unentschieden ermittelt werden soll.

Zuletzt sehen wir uns noch an, wie sich der Computer nach Start des Programms meldet:

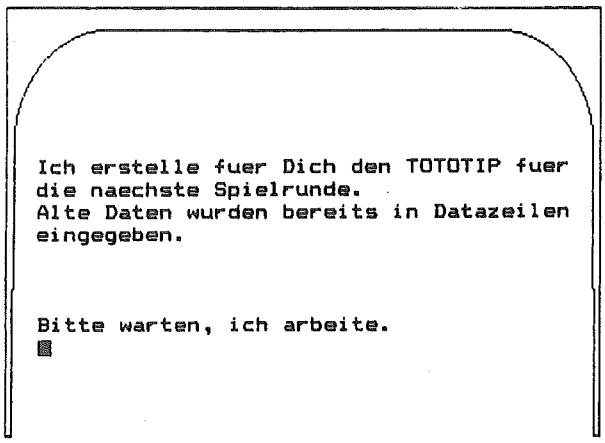


Bild 5.16

Viel Spaß beim Gewinnen!

```
10 REM PROGRAMM ZUM ERSTELLEN DES
11 REM ==========================
20 REM         TOTOTIPS
21 REM         ========
22 REM
23 REM
100 CLS : PRINT"Ich erstelle fuer Dich d
en TOTOTIP fuer die naechste Spielrunde.
110 PRINT "Alte Daten wurden bereits in
Datazeilen eingegeben.
120 PRINT:PRINT:PRINT:PRINT "Bitte warte
n,ich arbeite.
130 REM
140 REM      Variable
150 REM      ========
155 REM Variable die durch das Unterprog
ramm 'Punktberechnung verwendet werden.
160 PK=0:REM Punktezahl eines Spiels
170 TIP=0:REM Tip = 1- Sieg / Tip = 0- U
nentschieden / Tip = 2- Niederlage
180 GOAL=0:REM Tordifferenz des Spiels
190 TR=3:REM Wenn mehr als 3 Tore gescho
ssen werden,wird mit Multiplikator MUL m
ultipliziert
210 MUL=1.2 :REM Multiplikator
220 SIEG=10 :REM Punktezahl fuer Sieg
230 PUNENT=0:REM Punktezahl fuer Unentsc
hieden
240 PNIEDER=-10:REM Punktezahl fuer Nied
erlage
250 HEIMV=1.1:REM Heimvorteil 10% der Ge
samtpunkte
260 REM
270 PR=-1:REM Printerbetrieb
280 REM
290 REM
1010 REM    Daten einlesen
1020 REM    ==============
1200 READ PN,SP :REM PN'PAARUNGEN' SP'SP
IELZAHL'
1210 DATA 12:REM PAARUNGEN
1220 DATA 5 :REM SPIELZAHL (ohne das let
zte Spiel der 2 Kontrahenten)
1240 DIM MA$(PN,2) :REM 2 Mannschaften
1250 DIM GOAL(PN,2,SP):REM Tordifferenz
1260 DIM TIP(PN,2,SP):REM Ergebnis eines
 Spiels
1270 REM 1:Sieg, x:Unentschieden, 2:Nied
erlage
1280 DIM LP(PN),LT(PN):REM LP'LETZTTIP'
LT'LETZTTOR'
1290 DIM PE(PN,2) :REM PE'PUNKTE'
1300 MP=PS*(1+SP)*MUL:REM MP'MAX.PUNKTEZ
AHL' SP'SPIELZAHL'
1310 UNENT=MP/5 :REM Punkteschwelle fuer
 Unentschieden
1315 REM
1316 REM Lesen der Ergebnisse aus den le
tzten Spielen
1317 REM
1320 FOR PA=1 TO PN :REM PN'PAARUNGEN'
1330    READ LP$:LP(PA)=VAL(LP$)
1340    READ LT(PA)
1350    FOR MA=1 TO 2
1360       READ MA$(PA,MA)
1370       FOR SL=1 TO SP
1380           READ TIP$:TIP(PA,MA,SL)=V
AL(TIP$)
1390           READ GOAL(PA,MA,SL)
1400       NEXT SL
1410    NEXT MA
1420 NEXT PA
1430 REM
1440 REM Eingabedaten fuer die letzten S
piele
1450 REM
1460 REM 1.Paarung (Tip 1)
1470 REM Mannschaft1 gewinnt immer mit 1
 Tor Differenz
1480 REM Mannschaft2 verliert immer mit
1 Tor Differenz
1490 DATA 1,1:REM Voriges Spiel beider M
annschaften
1500 DATA m01-1,1,2,1,2,1,2,1,2,1,2
1510 DATA m01-2,2,2,2,2,2,2,2,2,2,2
1520 REM
1530 REM 2. Paarung (Tip 1)
1535 DATA 1,1
1540 DATAm02-1,1,3,1,3,1,3,1,3,1,3
1550 DATAm02-2,2,3,2,3,2,3,2,3,2,3
1560 REM
1570 REM 3. Paarung (Tip x)
1590 DATA 1.5,0
1600 DATAm03-1,x,0,x,0,x,0,x,0,x,0
1610 DATAm03-2,x,0,x,0,x,0,x,0,x,0
1620 REM
1630 REM 4. Paarung (Tip 1)
1640 DATA 2,3
1650 REM Mannschaft1 verliert zwar letzt
es Spiel gegen 2,
1660 REM gewinnt aber sonst immer mit 2
Toren Differenz
1670 DATAm04-1,1,2,1,2,1,2,1,2,1,2
1680 REM Mannschaft2 verliert immer
1690 DATAm04-2,2,1,2,1,2,1,2,1,2,1
1700 REM
1710 REM 5. Paarung (Tip 2)
1720 REM Hier wird die Unentschieden _
        schranke getstet
1730 DATA 2,2
1740 DATAm05-1,x,0,x,0,x,0,x,0,x,0
1750 DATAm05-2,x,0,x,0,x,0,x,0,1,3
1760 REM
1770 REM 6.Paarung (Tip x)
1780 REM Testen des Heimvorteils,beide
 gleich
1782 REM Heimvorteil allein ist fuer
 Sieg-Tip
1783 REM nicht ausreichend
1790 DATA x  ,0
1800 DATAm06-1,1,2,1,2,1,2,1,2,1,2
1810 DATAm06-2,1,2,1,2,1,2,1,2,1,2
1820 REM
1830 REM 7.Paarung (Tip x)
1840 REM Testen des Unterschiedes bei ho
hen Siegen
1850 REM Mannschaft 1 hat Heimvorteil, d
afuer hat Mannschaft 2 das letzte Spiel
gewonnen
1860 DATA 2,2
```

```
1870 DATAm07-1,1,3,1,3,1,3,1,3,1,3
1880 DATAm07-2,1,3,1,3,1,3,1,3,1,3
1890 REM
1900 REM 8.Paarung wie 3.Paarung(Tip x)
1910 DATA x  ,0
1920 DATAm08-1,x,0,x,0,x,0,x,0,x,0
1930 DATAm08-2,x,0,x,0,x,0,x,0,x,0
1940 REM
1950 REM 9.Paarung wie 1.Paarung(Tip 1)
1960 REM Mannschaft1 gewinnt immer mit 1
 Tor Differenz
1970 REM Mannschaft2 verliert immer mit
 1 Tor Differenz
1980 DATA 1,1: REM Voriges Spiel beider
Mannschaften
1990 DATAm09-1,1,2,1,2,1,2,1,2,1,2
2000 DATAm09-2,2,2,2,2,2,2,2,2,2,2
2010 REM
2020 REM 10. Paarung wie 1.Paarung
                                (Tip1)
2030 REM Mannschaft1 gewinnt immer mit 1
 Tor Differenz
2040 REM Mannschaft2 verliert immer mit
1Tor Differenz
2050 DATA 1,1:REM Voriges Spiel beider M
annschaften
2060 DATA m10-1,1,2,1,2,1,2,1,2,1,2
2070 DATA m10-2,2,2,2,2,2,2,2,2,2,2
2080 REM
2090 REM 11. Paarung wie 1.Paarung(Tip1)
2100 REM Mannschaft1 gewinnt immer mit 1
Tor Differenz
2110 REM Mannschaft2 verliert immer mit
1Tor Differenz
2120 DATA 1,1: REM Voriges Spiel beider
Mannschaften
2130 DATA m11-1,1,2,1,2,1,2,1,2,1,2
2140 DATA m11-2,2,2,2,2,2,2,2,2,2,2
2150 REM
2160 REM 12. Paarung wie 5.Paarung(Tip2)
2170 REM Hier wird die Unentschiedenschr
anke getestet
2180 DATA 2,2
2190 DATA m12-1,x,0,x,0,x,0,x,0,x,0
2200 DATA m12-2,x,0,x,0,x,0,x,0,1,3
2210 REM
5030 REM  PUNKTEBERECHNUNG
5040 REM  ================
5050 FOR PA=1 TO PN
5060    TIP=LP(PA):GOAL=LT(PA)
5070    GOSUB 12000
5080    FOR GEG=1 TO 2 :REM GEG'GEGNER
5090     IF GEG=2 THEN PK=PK*(-1)
5100     PE(PA,GEG)=PK:REM PE'PUNKTE'
5110    NEXT GEG
5120    FOR MA=1 TO 2
5130      FOR SL=1 TO SP:REM SL'SPIEL
                          SP'SPIELZAHL'
5140          TIP=TIP(PA,MA,SP)
5150          GOAL=GOAL(PA,MA,S)
5160          GOSUB 12000
5170          PE(PA,MA)=PE(PA,MA)+PUNKT
5180       NEXT SL
5190       PE(PA,1)=PE(PA,1)*HEIMV
5200    NEXT MA
5210 NEXT PA
10000 REM
10010 REM    Ausgabe
10020 REM    =======
10030 CLS
10040 PRINT"Der Toto-Tip lautet:"
10060 PRINT"---------------------"
10080 PRINT"Mannschaft 1:Mannschaft 2: P
unkte  TIP"
10100 PRINT"-----------------------------
-----------"
10120 REM
10130 FOR PA =1 TO PN
10140 PRINTTAB(3)   : PRINT MA$(PA,1),
10150 PRINT":  ";MA$(PA,2);
10160 PRINT USING"########";PE(PA,1);
10170 PRINT ":";
10180 PRINT USING "####";PE(PA,2);
10190 IF ABS (PE(PA,1)-PE(PA,2))<PUNENT
THEN PRINT "  X": GOTO 10210
10200 IF PE (PA,1)> PE(PA,2) THEN PRINT
     " :1" ELSE PRINT " :2"
10210 NEXT PA
10220 PRINT"-----------------------------
-----------"
10230 END
12000 REM
12010 REM Punkteberechnung Unterpr.
12020 REM
12030 REM Eingabevariable TIP (0,1,2)
12040 REM                 GOAL=TORDIFF.
12050 REM Ausgangsvariable:
                      Punkt=Punktezahl
12060 IF TIP <0 OR TIP>2 THEN STOP
12070 IF TIP =2 THEN PK=PNIEDER
12080 IF TIP =1 THEN PK=PSIEG
12090 IF TIP =0 THEN PK=PUNENT
12100 IF GOAL>=TR THEN PK=PK*MUL
12110 RETURN
```

Der Streß schlägt sich auf den Magen 6. Folge

Selbst im Gesundheitswesen werden Computer heutzutage schon verwendet. Denken wir nur an die vielen computergesteuerten Geräte in Spitälern. Die moderne Ultraschalldiagnose und die Computertomographie wären ohne modernste Mikroelektronik nicht denkbar.

Doch selbst im trauten Eigenheim kann man seinen Home-Computer in den Dienst der Gesundheit stellen. Wir können zwar keinen Computertomographen nachbauen, aber unser Gerät hat die Fähigkeit, unsere Nährstoffzufuhr überwachen zu können. Alles was wir dazu brauchen, ist wieder nur ein kleines Programm!

Werden wir konkret! Wir wollen den Joulegehalt unserer Speisen, die wir zu uns nehmen, überprüfen, und feststellen, ob wir zu viel oder zu wenig essen.

Damit wird es möglich, eine kostenlose Kontrolle für unser Gewicht zu bekommen. Wenn wir 20000 Joule zu uns nehmen, aber nur 5000 verbrauchen würden, dann wären wir auf kurz oder lang "etwas" übergewichtig, ist doch logisch, oder?

Vorher müssen wir uns jedoch eine neue Art des Speichermediums ansehen.

Diskettenlaufwerke

Wir haben schon die Speicher RAM und ROM kennengelernt. Doch der eine Speicher verliert seinen Inhalt nach Abschalten des Stromes und der andere kann seinen Inhalt nicht ändern. Wenn wir nun ein Medium hätten, in das wir sehr wohl etwas einspeichern könnten, aber das uns nach Abschalten des Stromes nicht verloren geht, wäre uns sehr geholfen.

Es gibt so eine Art des Speichers, die Disketten. Da diese Platten nicht innerhalb des Computers, sondern meistens außerhalb angebracht sind, spricht man von einem externen Speichermedium.

Das Prinzip von Disketten ist sehr einfach. Man nehme eine runde Scheibe, beschichte sie mit einem Magnetfilm und lasse im Betrieb die Daten ähnlich wie bei einem Kassettenrekorder per Tonkopf auf die Magnetschicht schreiben.

Sehen wir uns eine Diskette an. Die meisten Home-Computer verwenden derzeit 5.25 Zoll-Disketten, auch Floppy-Disks genannt. Neben diesen gibt es die Minidisks (3.5 Zoll) und die 8,5 Zoll-Disketten. Von den diversen Harddisks sehen wir derzeit einmal ab.

Floppy-Disks heißen die Disketten deshalb, weil sie biegsam sind. So eine Scheibe besteht nämlich aus einer Kunststoffplatte, die mit einer Magnetschicht umhüllt ist. Weil diese Einrichtung alleine zu vielen rauhen Umweltfaktoren ausgesetzt wäre, wird sie zum Schutz noch in eine Kunststoffhülle mit einer Filzinnenseite eingebettet. In dieser Hülle befinden sich übrigens einige Löcher. In diesen kann man die innen liegende Scheibe erkennen (Bild 6.1).

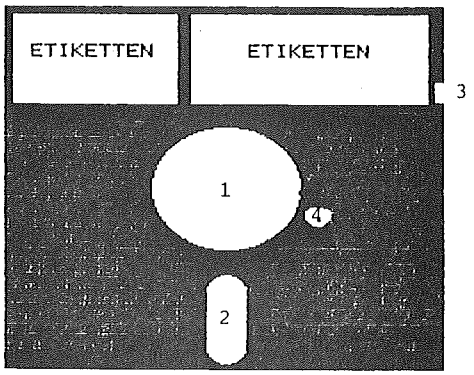


Bild 6.1

Man braucht in der Mitte (1) ein großes Loch, um die Scheibe drehen zu können. Der Lesekopf muß die Platte berühren (2). Ebenso gibt es noch ähnlich den Kassettenrekordern einen Schreibschutz (3). Zum Schluß findet sich noch das sogenannte "Indexloch" (4). Der Computer muß ja feststellen, wo der "Anfang" der Diskette ist. Wieso wir von einem "Anfang" auf einer runden Scheibe sprechen, wird später erklärt. Tatsache ist, daß ein kleines Loch in der Platte diesen Anfang signalisiert.

Sehen wir uns nun die Einteilung so einer Scheibe an. Die Disk ist bei uns in 40 Spuren eingeteilt. Jede Spur (auch Track genannt) ist dann noch in 17 Sektoren mit je 256 Bytes Speicherkapazität unterteilt. Wie man sich diese Tracks vorstellen kann, zeigt Bild 6.2 (Sektor 0 ist im Übrigen der Anfang der Diskette).

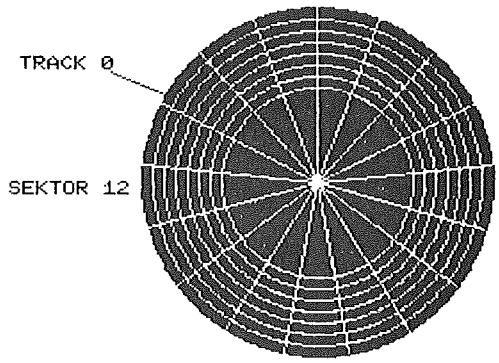


Bild 6.2

Der Spectravideo hat wie viele andere Computer auch ein spezielles Disk-BASIC, welches die Benutzung von Disketten ermöglicht. Dieses BASIC-"System" ist auf den Spuren 0,1 und 2 untergebracht. Diese drei Tracks bleiben im Normalfall immer für das System reserviert. Selbst wenn am Anfang kein System auf der Diskette ist, bleiben die Tracks frei. Man kann daher nachträglich jederzeit das BASIC-Diskettenbetriebssystem dazuladen, ohne den Inhalt zu zerstören.

Wie bei jeder ordentlichen Bibliothek, so gibt es auch bei unserer Software-"Bibliothek" ein "Inhaltsverzeichnis", Directory genannt. Es befindet sich auf Track 20 und zeigt dem Computer, wo welches Programm auf der Diskette zu finden ist. Wir können im BASIC jederzeit mit "FILES" abfragen, welche Programme auf der Disk gespeichert sind.

Kommen wir nun kurz zu der Peripherie der Disk. Im praktischen Gebrauch ist natürlich ein Gerät notwendig, welches die Floppy bearbeitet. Wir nennen es Diskettenlaufwerk. Es beinhaltet einen Motor zum Drehen der Floppy, Lese- und Schreibköpfe und die übliche Elektronik zum Absenden der Daten zum Computer. Beim Spectravideo verwaltet ein externer Floppy-Controller den Datentransfer zwischen Computer und Laufwerk. Dieser Controller ist ein eigener kleiner Prozessor, der es schafft, daß 256000 Zeichen pro Sekunde übertragen werden können. Wenn wir jetzt schon mit Zahlen herumwerfen, dann können wir auch gleich die Speicherkapazität der Disketten angeben: Man darf ungefähr 160000 Zeichen auf so einer SVI-Disk ablegen, neuerdings sogar 320000.

Neben der Disk gibt es noch die gute alte Kassette als Datenspeicher. Hier werden Daten einfach in dem Kassettenrekorder "angenehme" Signale umgewandelt und per Schnittstelle ausgegeben. Die Disk ist bei uns aber bedeutend schneller, der Rekorder hat nämlich nur eine Übertragungsrate von 1800 Zeichen pro Sekunde (Die Übertragungsrate wird auch Baud, "Bood" gesprochen, genannt).

Was wir hier über Floppy-Disketten gehört haben, gilt für die BASIC-Disketten von Spectravideo. Andere Firmen haben andere Einteilungen.

Sogar Spectravideo selber hat noch ein anderes Aufzeichnungsverfahren. Unter dem Betriebssystem CP/M, welches wir schon in Folge 3 kennengelernt haben, sind ein paar Details anders. Wir wollen hier jedoch nicht näher darauf eingehen.

Files = Dateien

Selbstverständlich darf man seine Daten und Informationen nicht ohne Rahmen auf die Floppy "hinausschleudern". Das Chaos, welches innerhalb kürzester Zeit entstehen würde, wäre furchterregend. Wir schicken unsere Daten alle in Form von Dateien (=Files) aus dem Computer hinaus. Programme werden vom Betriebssystem selber in die richtige Form gebracht, um "ordentlich" abgespeichert zu werden. Bei sonstigen Daten, welche wir übertragen wollen, müssen wir uns den Rahmen schon selber basteln.

Es gibt zwei Arten von Dateien. Die RANDOM-Datei und die sequentielle Datei. Bei der ersten kann man jederzeit jede Information aus der Datei abfragen oder überschreiben. Bei der sequentiellen Datei wird eine Information nach der anderen abgearbeitet. Kein Überspringen von Daten ist möglich. Wir werden uns mit den einfacheren sequentiellen Files beschäftigen.

Hierfür haben wir unsere eigenen BASIC-Kommandos. Zuerst müssen wir den Transferkanal zum Laufwerk öffnen und angeben, wie unsere Datei heißt. Dies machen wir mit dem OPEN-Befehl. Sehen wir uns seine Form an:

OPEN Dateiname (FOR Modus) AS Dateinummer

Der Dateiname beinhaltet die Laufwerknummer. Beim SVI können zwei Floppy-Laufwerke angeschlossen werden (1 und 2). Danach folgt ein Doppelpunkt und ein sechs Zeichen langer Name.

"FOR Modus" darf weggelassen werden. Für Modus können folgende Werte stehen:

INPUT: Datei wird zum Einlesen geöffnet
OUTPUT: Datei wird zum Auslesen geöffnet
APPEND: Mit diesem Anhang kann man eine Datei verlängern, wir brauchen diesen Modus jedoch nicht.

Zum Schluß wird die Dateinummer angegeben. Wir brauchen sie später zur Übertragung von Daten.

Der nächste Schritt ist das Bearbeiten von Dateien. Nach dem Öffnen des Kanals dürfen wir mit "PRINT # Dateinummer" und "INPUT # Dateinummer" Daten schreiben und lesen.

Zuletzt schließen wir unsere Bearbeitung mit "CLOSE Dateinummer". Damit ist das File geschlossen. Wir müssen ein File immer zuerst schließen, bevor wir es neu eröffnen.

Zusätzlich zu diesen Kommandos gibt es noch weitere Befehle zur Bearbeitung von Dateien, die wir hier aber nicht anführen.

Nun stürzen wir uns wieder ins Programmieren! Das Joule-Programm wartet schon.

Programm Joule

Das Wichtigste im Programm sind natürlich die einzelnen Joule-Werte der Speisen und der verrichteten Arbeiten. Hier wurde folgende Einteilung getroffen: Pro Programmzeile wird eine Information aufgeschrieben. Zuerst steht eine Zahl, welche kennzeichnet, ob es sich um eine Speise handelt, oder ob eine Arbeit angegeben wurde. Unsereins weiß zwar, daß zum Beispiel "Schinken" eine Speise und keine Arbeit ist, der Computer braucht zur Unterscheidung jedoch eine Zahl (1=Speise, 0=Aktivität). Danach folgt der Name der Speise (oder Aktivität) und vier weitere Werte. Diese geben die Mahlzeiten an, in denen die Speisen vorkommen. Schweinsbraten wird man eher selten zum Frühstück essen!

Beispiel für Speise: 1,BROT,250,4,4,4,4

Beispiel für Aktivität: 0,SCHLAFEN,65

Nach RUN folgt natürlich wieder die unvermeidliche Vorstellung des Programms. Man darf seinen Namen eingeben und kann nun zwischen zwei Betriebsarten wählen:

1) In der ersten Betriebsart ist es möglich, seine Speisen und seine Arbeiten zu ändern. Wenn man zum Beispiel beim Frühstück von Marmelade auf Butter umsteigt, dann ändert man seine Joule-Tabelle entsprechend um. Nach getanem Wechsel speichert der Computer die neuen Daten auf die Floppy.

2) Nun kann man seine Tagesbilanz ermitteln. Man gibt die Menge der Speisen ein, die man zu sich genommen hat. Danach folgen die Aktivitäten. Zum Schluß wird einem präsentiert, ob man "ungesund gelebt" hat oder so weitermachen darf.

Sehen wir uns nun die Strukturierung des Programms an. In Bild 6.3 sehen wir das Struktogramm für die einzelnen Programmteile. Zuerst kommt die Begrüßung an die Reihe, danach wird die Betriebsart gewählt. Es folgen die beiden Hauptteile "Ändern der Jouletabelle" und "Ermitteln der Tagesbilanz".

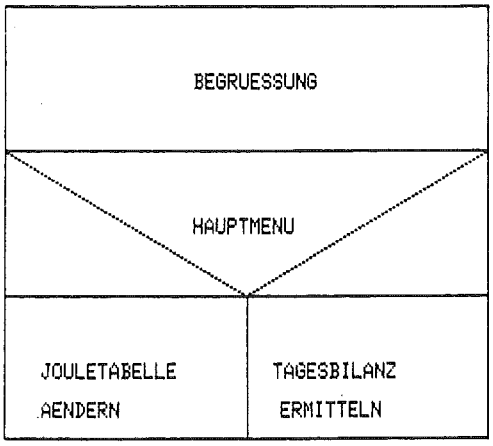


Bild 6.3

Jetzt werden wir die beiden Teile etwas näher beleuchten:

In Bild 6.4 können wir das Struktogramm für den ersten Teil erkennen.

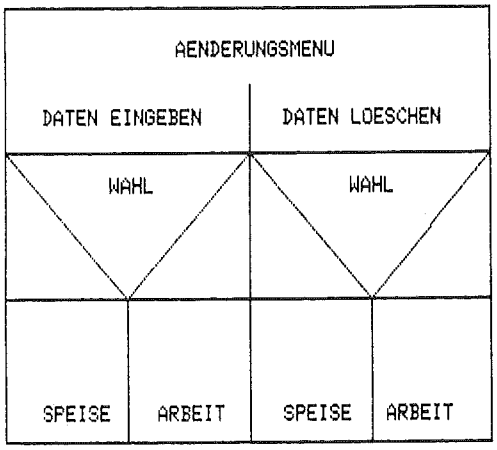


Bild 6.4

Wir haben die Wahl: Wollen wir Daten ändern, oder sie löschen? Je nachdem welchen Zweig wir wählen, arbeitet der Computer den linken oder den rechten Zweig des Struktogramms ab. Beim linken Teil dürfen wir zwischen Arbeit und Speise wählen, danach wird solange eingegeben, bis man aus diesem Zweig aussteigt. Ebenso funktioniert der rechte Teil. Der einzige Unterschied zum linken Zweig besteht darin, daß keine Daten eingegeben, sondern welche gelöscht werden.

Am Schluß werden alle Informationen wieder auf die Disk gebannt.

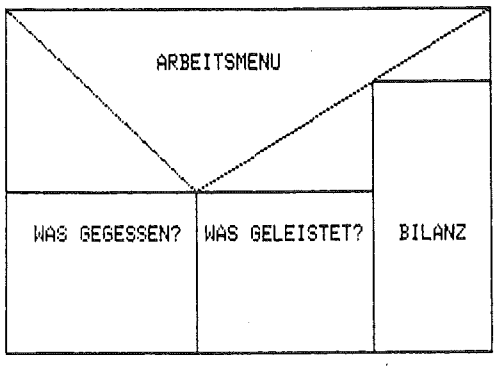


Bild 6.5

Etwas anders geht es im Modus "Ermitteln der Tagesbilanz" zu. Es gibt zwar auch hier die Frage nach Arbeit und Speise, doch wird hier lediglich nach der Menge der geleisteten Aktivität und dem verbrauchten Essen gefragt. Anhand dieser neuesten Informationen bildet dann der Computer die neueste Tagesbilanz. Das zugehörige Struktogramm zeigt Bild 6.5.

Zum Schluß sehen wir uns noch den Bildschirm in drei Phasen des Programms an.

Bild 6.6 zeigt uns den Anfang des Programms, Bild 6.7 bringt das Hauptmenü, Bild 6.8 zeigt einen Ausschnitt aus der Betriebsart 2.

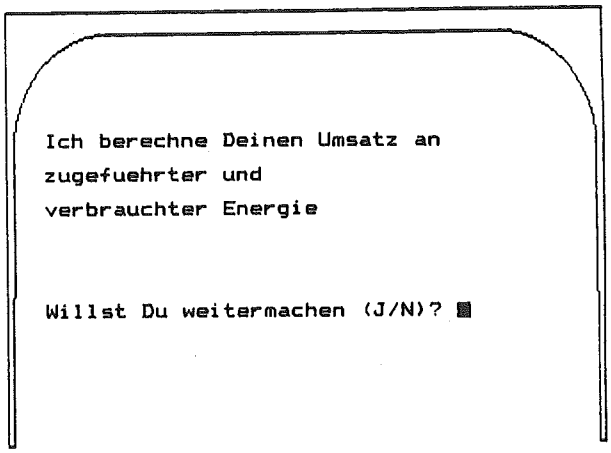


Bild 6.6

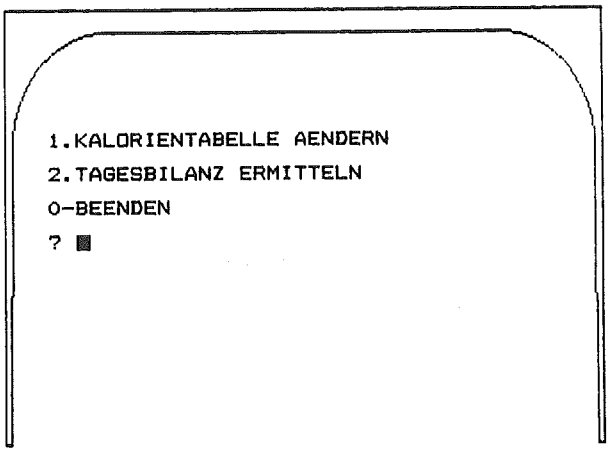


Bild 6.7

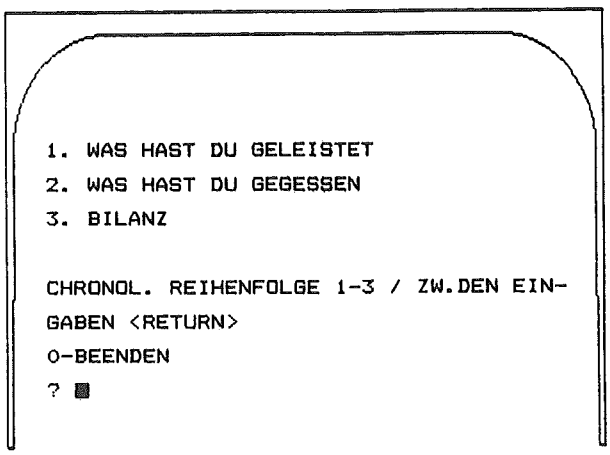


Bild 6.8

```
100 REM
110 REM ***** JOULE *****
115 REM
130 BH=24
140 CLS :LOCATE 0,4: PRINT"   Ich berechne d
einen Umsatz an          zugefuehrter und ve
rbrauchter"
150 PRINT "   Energie":LOCATE 0,10
160 PRINTTAB(3):INPUT "Willst Du weitermache
n (J/N)";T$
170 IF T$="N" OR T$="n" THEN END
180 IF T$="J" OR T$="j" THEN 510 ELSE GOTO 1
60
510 REM DEFINITIONEN
520 REM ============
530 ON ERROR GOTO 10920:REM Fehlerbehandlung
540 CR$=CHR$(13)
550 SZ=100 :REM SZ'SPEISEZAHL'
560 DIM SP$(SZ):DIM ENZ(SZ)
570 A=50
580 DIM A$(A):DIM EA(A) :REM EA'ENERGIEAB'
600 MA=6:DIM KAT(MA,SZ):DIM MA$(MA):REM
   MA'MAHLZEIT',KAT'KATEGORIE'
610 MA$(1)="Fruehstueck"
620 MA$(2)="Jause"
630 MA$(3)="Mittagessen"
640 MA$(4)="Kaffeejause"
650 MA$(5)="Abendessen"
660 MA$(6)="Kleine Suende zwischendurch"
670 PRINTTAB(3) :INPUT"Wie ist dein Name ";Z
$
671 CLS:PRINT:PRINT:PRINT:PRINT"        BITTE
 WARTEN"
680 GOSUB10050:BN$=Z$:REM VERWANDLUNG IN GRO
SZSCHRIFT
1000 REM
1010 REM EINLESEN DER KALORIENTABELLE
1020 REM ============================
1030 GOSUB 20510
1040 REM
1050 GOSUB 1110:REM AUSFUEHRUNG DES HAUPTPRO
GRAMMS
1060 END
1070 REM
1080 REM ** 2 HAUPTFUNKTIONEN **
1090 REM    1.KALORIENTABELLE AENDERN
1100 REM    2.KALORIENBILANZ ERMITTELN
1110 REM
1120 PRINT
1130 CLS:PRINT "1.KALORIENTABELLE AENDERN"
1140 PRINT "2.TAGESBILANZ ERMITTELN"
1150 REM
1160 TM=2:GOSUB10410:
1170 ON T GOSUB 2050,5050
1175 GOTO 1080
1180 RETURN
2000 REM
2010 REM KALORIENTABELLE AENDERN
2020 REM ========================
2030 REM
2040 PRINT
2050 PRINT"1. DATEN EINGEBEN"
2070 PRINT"2. DATEN SPEICHERN"
2080 TM=3:GOSUB10410:REM TASTE HOLEN ODER BE
ENDEN
2090 ON T GOSUB 2110,20320
2100 RETURN
2110 REM
2120 REM DATENEINGABE
2130 REM
2140 PRINT
2150 PRINT "1. SPEISE"
2160 PRINT "2. ARBEIT"
2170 TM=2:GOSUB 10410:REM TASTE HOLEN ODER B
EENDEN
2180 ON T GOSUB 2230,2420
2190 RETURN
2200 REM
2210 REM NEUE SPEISEN
2220 REM
2230 SZ=SZ+1
2240 PRINT:PRINT "BEZEICHNUNG (BEENDEN MIT "
E");";:INPUTX$
2250 IF X$="e" ORX$="E"THEN2360ELSE SP$(SZ)=
X$
2260 INPUT "KALORIE";ENZ(SZ)
2270 FOR MA=1TO6
2280    PRINT "Wie haeufig iszt Du ";SP$
    (SZ);" als ";:PRINTMA$(MA)
2290    PRINT "5. TAEGLICH"
2300    PRINT "4. HAEUFIG"
2310    PRINT "3. SELTEN"
2320    PRINT "2. SEHR SELTEN"
2330    PRINT "1. NIE"
2340    TM=5:GOSUB 10360:KAT(MA,SZ)=T-1: REM
T'TASTE"
2350 NEXT MA
2360 RETURN
2370 REM
2380 REM
2390 REM NEUE ARBEIT
2400 REM
2410 AZ=AZ+1 : REM AZ'ARBEITSZAHL'
2420 PRINT:PRINT "BEZEICHNUNG (BEENDEN MIT '
e');":INPUTX$
2430 IFX$="E"OR X$='e' THEN2450 ELSE A$(AZ)=
X$
2440 INPUT "KALORIEN/STUNDE:";ENZ(AZ)
2450 RETURN
2460 REM
5000 REM
5010 REM
5020 REM  TAGESBILANZ ERMITTELN
5030 REM  ====================
5040 REM
5050 REM
5060 PRINT
5070 PRINT "1. WAS HAST DU GELEISTET ";BN$
          :REM BN'BNAME'
5080 PRINT "2. WAS HAST DU GEGESSEN  ";BN$
5090 PRINT "3. BILANZ "
5092 PRINT:PRINT "CHRONOL. REIHENFOLGE 1-3 /
 ZW.DEN EIN- GABEN <RETURN>"
5095 TM=3:GOSUB10410:REM HOLEN EINER GUELTIG
EN TASTE ODER BEENDEN
5100 ON T GOSUB 5520,5150,6000
5110 IFT$="E" THEN T$="": RETURN
5115 GOTO 5050
5125 REM
5130 REM WAS WURDE GEGESSEN ?
5140 REM
5150 FOR E=1TO 6
5160    ON E GOSUB 5180,5190,5200,5210,5220,5
230
5170    GOTO 5250
5180    MA=1:RETURN
5190    MA=2:RETURN
5200    MA=3:RETURN
5210    MA=1:RETURN
5220    MA=2:RETURN
5230    MA=4:RETURN
5240    REM
5250    REM END VON ON
5260    GOSUB 10850:REM BILDSCHIRM LOESCHEN
5270    SP=0
5280    IF SP>=SZ GOTO 5460
5290    ZE=1:Z=1 :REM ZE'ZEILE'
5300    GOSUB 10850
5310    PRINT MA$(E)
5320    PRINT "=================="
5330    IF ZE >BH-5 OR SP>=SZ THEN 5370 :REM
BH'BILDH'
5335    SP=SP+1
5340    IF KAT(MA,SP) =0 THEN 5350
5343    PRINTSP;". ";SP$(SP)
5344    ZE=ZE+1
5350    REM
5365    GOTO 5330
5370    REM
5380    REM
5390    REM BILDSCHIRMENDE ODER LETZTE KATEGO
RIE ERREICHT
5400    PRINT "NR<- / MENGE(in100g)<- /0<-":R
EM NAECHSTES BLATT
5410    AK=1 ;REM AK'AKTION'
5420    IFAK=0 AND DA=0 GOTO5450:REM DA'DAUER
'
5430    GOSUB 10630:REM HOLE ZAHL ODER ENDE-M
ELDUNG
5440    BZ=BZ+ENZ(AK)*DA:REM BZ'BILANZZU'
5445    GOTO5420
5450    GOTO 5280
5460    REM
5470 NEXT E
5480 RETURN
5490 REM
5500 REM WAS WURDE GELEISTET ?
5510 REM
```

```
5520 A=1
5530 IF A >AZ GOTO 5720
5540 GOSUB 10870:REM BILDSCHIRM LOESCHEN
5550 ZE=1:Z=1
5560 PRINT "WELCHE ARBEIT HAST DU HEUTE AUSG
EFUEHRT " BN$"?"
5570 PRINT"--------------------------------
-----"
5580 IF ZE >BH-5 OR A>AZ GOTO 5620
5590 PRINT A". ";A$(A)
5600 ZE=ZE+1:A=A+1
5610 GOTO5580
5620 REM
5630 REM BILDSCHIRMENDE ODER LETZTE ARBEIT E
RREICHT
5640 PRINT "NR<- / DAUER(H)<-     0<- / 0<- "
:REM NAECHSTES BLATT
5650 AK=1
5660 IF AK=0 AND DA=0 GOTO5710
5670 GOSUB10630:REM HOLE ZAHL ODER ENDE-MELD
UNG
5680 BA=BA+ EA(AK)*DA : REM BA'BILANZAB'
5690 BT=BT+DA: REM BT'BILANZABT'
5700 GOTO5660
5710 GOTO5530
5720 RETURN
6000 REM
6010 REM BILANZ
6020 REM
6030 GOSUB 10850 :
     REM  BILDSCHIRM LOESCHEN
6040 PRINT
6050 PRINT BN$",Deine heutige Kalorienbilanz
":PRINT" ist folgende :"
6060 PRINT"--------------------------------
-----"
6070 PRINT"Aktivitaetszeit        :"BT" Stund
en"
6080 PRINT"daher Ruhezeit         :"24-BT" St
unden": REM 24-BILANZABT
6090 PRINT
6100 PRINT"geleistete Arbeit      :"BA"     kc
al"
6110 BR=(24-BT)*65 : REM BR'BILANZABR'
6120 PRINT"Ruheverbrauch          :"BR" kcal"
6130 PRINT"                        ---------
-"
6140 BG=BA+BR : REM BG'BILANZABG'
6150 PRINT"Gesamtverbrauch        :"BG" kcal"
6160 PRINT
6170 PRINT"Nahrungsaufnahme       :"BZ"     kc
al"
6180 PRINT
6190 IF BZ>BG THEN PRINT "DU hast heute scho
n genug gegessen "BN$" !"
6200 IF BZ<BG THEN PRINT "DU kannst heute no
ch"BG-BZ"kcal essen":PRINT BN$", GUTEN APPET
IT ! ! ! "
6210 PRINT"--------------------------------
-----"
6220 RETURN
10000 REM
10010 REM
10020 REM UNTERPROGRAMME
10030 REM ==============
10040 REM
10050 REM UMWANDLUNG IN GROSZBUCHSTABEN
10060 F=0:X$="":REM HILFSVARIABLE F'FEHLER'
10070 ZL=LEN(Z$) :REM ZL'ZLEN'
10080 REM
10090 FOR I=1TOZL
10100     ST$=MID$(Z$,I,1) :REM ST'STELLE'
10110     ST=ASC(ST$)
10120     IFST<65 OR ST>123 THEN F=1:GOTO 101
80
10130     IFST>96AND ST<123THENST=ST-32
   : REM VERWANDLE IN GROSZBUCHST
10140 X$=X$+CHR$(ST)
10150 REM
10160 NEXT
10170 Z$=X$
10180 RETURN
10190 REM
10200 REM
10210 REM HOLEN DES ASCII-WERTES EINER TASTE
10220 T$=INKEY$:T=VAL(T$):RETURN:REM T'TASTE
10230 REM
10240 REM
10250 REM HOLEN TASTE UND WARTEN
10260 T$=""
10270 IF T$<>"" GOTO10300
10280 GOSUB 10220
10290 GOTO10270
10300 RETURN
10310 REM
10320 REM
10330 REM WARTEN AUF RICHTIGE TASTE
10340 T=0
10350 IF T<>0 OR T<TM GOTO 10380
10360 GOSUB10250
10370 GOTO10350
10380 RETURN
10390 REM
10400 REM WARTEN AUF TASTE, RUECKKEHR,
     WENN<TMAX,ODER WENN 'e'
10410 PRINT "0-BEENDEN "
10420 INPUT T:IFT=0THENEND
10430 RETURN
10490 REM
10610 REM
10620 REM HOLE SPEISE/ARBEIT, SCHREIBE
     '/', HOLE MENGE/DAUER
10630 GOSUB10710:AK=Z:REM HOLE SPEISE/ARBEIT
10640 PRINT"/";
10650 GOSUB 10710:DA=Z:REM HOLE MENGE/DAUER
10660 PRINT ":";
10670 RETURN
10680 REM
10690 REM
10700 REM HOLE ZAHL BEENDE MIT CR,
           SCHREIBE MIT
10710 Z=0:Z$="":CR$=CHR$(13)
10720 T$=""
10730 IF T$=CR$ GOTO10820
10740 T=&H30
10750 IF T<&H30 OR T>&H39 GOTO 10810
10760 Z$=Z$+T$: PRINT T$;
10770 T$=""
10780 IFT$<>""GOTO10796
10790 T$=INKEY$
10795 GOTO10780
10796 T=ASC(T$)
10800 GOTO10750
10810 GOTO10730
10820 REM
10830 Z=VAL(Z$)
10840 RETURN
10850 REM
10860 REM
10870 REM BILDSCHIRM LOESCHEN
10880 FOR I=1 TO BH:PRINT:NEXTI
10890 RETURN
10900 REM
10910 REM FEHLERBEHANDLUNG
10920 IF ERR=53 THEN PRINT" ES GIBT NOCH KEI
NE KALORIEN-DATEI":CALL 9690:CALL 8590
   :RESUME NEXT
10930 PRINT "FEHLERCODE:";ERR:PRINT "FEHLER
IN ZEILE:";ERL
10940 END
20000 REM
20010 REM
20020 REM BEHANDLUNG DER KAL.TABELLE
20030 REM ==========================
20040 REM
20050 REM LESEN VON DISK
20060 REM
20070 OPEN "I",#1,"KALTAB"
20080 SZ=0:AZ=0
20090 IF EOF(1)<>0 GOTO20280
20100     INPUT #1,SP
20110     IF SP<>1 THEN 20220
20120     REM
20130     REM SPEISE
20140     INPUT #1,SP$(SZ)
20150     INPUT #1,ENZ(SZ)
20160     FOR I=1TO4
20170        INPUT #1,KAT(I,SZ)
20180     NEXT I
20190     SZ=SZ+1:GOTO 20260
20200     REM
20210     REM ARBEIT
20220     INPUT #1,A$(AZ)
20230     INPUT #1,EA(AZ)
20240     AZ=AZ+1
20260 REM END IF
20270 GOTO 20090
20280 RETURN
```

```
20290 REM
20300 REM KALORIENTABELLE IN FILE
     'KALTAB' SPEICHERN
20310 REM
20320 OPEN "1:KALTAB"FOR OUTPUT AS#1
20330 FOR SP=1 TO SZ
20340   PRINT#1,1,SP$(SP),ENZ(SP)
20350   REM
20360   FOR MA=1 TO 4
20370    PRINT#1,KAT(MA,SP)
20380   REM
20390   NEXT MA
20400  REM
20410 NEXT SP
20420 FOR A=1 TO AZ
20430   PRINT #1,0,A$(A),EA(A)
20440   REM
20450 NEXT A
20460 CLOSE1
20470 RETURN
20480 REM
20490 REM EINLESEN DER KALORIENTABELLE
     AUS DEN DATA-STATEMENTS
AM ENDE DES PROGRAMMS
20500 REM
20510 SZ=1::AZ=1
20520 READSP:IF SP=-1 THEN 1050: REM
  EINSTIEG IN DAS HAUPTPROGRAMM
20530 IF SP<>1THEN 20610
20540   READ SP$(SZ)
20550   READ ENZ(SZ)
20560   FOR MA=1 TO 4
20570     READ KAT(MA,SZ)
20580   NEXT MA
20590   SZ=SZ+1
20600 GOTO20520
20610 IFSP<>0 THEN PRINT "FORMATFEHLER":
     STOP
20620 READ A$(AZ)
20630 READ EA(AZ)
20640 AZ=AZ+1
20650 GOTO20520
20660 RETURN
20670 REM
20680 REM
20690 REM SPEISENVERZEICHNIS
     TAETIGKEITSVERZEICHNIS
20700 REM ======================
20710 REM
20720 REM BROT, MEHL, TEIGWAREN
20730 REM
20740 DATA 1,BROT,250,4,4,4,4
20750 DATA 1,VOLLKORNBROT,234,4,3,1,0
20760 DATA 1,ZWIEBACK,420,3,2,0,2
20770 DATA 1,CORN-FLAKES,360,4,2,1,0
20780 DATA 1,NUDELN,380,0,3,4,0
20790 REM
20800 REM SUESZIGKEITEN
20810 REM
20820 DATA 1,HONIG,320,3,0,0,3
20830 DATA 1,BONBONS,400,0,2,0,4
20840 DATA 1,MARZIPAN,495,0,0,0,44
20845 DATA 1,MARMELADE,274,4,0,0,0
20850 DATA 1,SCHOKOLADE,550,0,2,0,4
20860 DATA 1,ZUCKER,400,1,2,0,0
20870 DATA 1,WUERFELZUCKER,20,4,1,1,4
20880 REM
20890 REM GEMUESE
20895 REM
20900 DATA 1,KARFIOL,32,0,2,3,0
20910 DATA 1,BOHNEN,38,0,2,3,0
20940 DATA 1,ERBSEN,60,0,2,3,0
20950 DATA 1,KOHL,65,0,2,3,0
20960 DATA 1,GURKE,8,0,2,3,0
20970 DATA 1,SALAT,16,0,4,4,0
20980 DATA 1,PORREE,36,0,2,3,0
20990 DATA 1,PAPRIKA,38,0,4,3,0
21000 DATA 1,TOMATE,26,0,4,3,0
21010 DATA 1,KARTOFFEL ged.,166,0,3,4,0
21020 DATA 1,KARTOFFEL ger.,211,0,3,4,0
21030 DATA 1,LINSEN,341,0,2,3,0
21040 DATA 1,BOHNEN,32,0,2,3,0
21050 REM
21060 REM FRUECHTE/SAEFTE
21070 REM
21080 DATA 1,ANANAS,,0,3,2,3
21090 DATA 1,APFEL,50,0,2,2,4
21100 DATA 1,APFELSAFT,70,0,4,4,4
21110 DATA 1,ORANGENSAFT,45,0,2,2,4
21120 DATA 1,ORANGE,45,0,2,2,4
21130 DATA 1,BANANE,90,0,2,2,4
21140 DATA 1,PFIRSICH,55,0,2,2,4
21150 DATA 1,TRAUBEN,75,0,2,2,4
21160 REM
21170 REM MILCH,MILCHGETRAENKE
21180 REM
21190 DATA 1,VOLLMILCH,69,4,2,2,3
21200 DATA 1,MAGERMILCH,38,4,2,2,3
21210 DATA 1,BUTTERMILCH,36,4,2,2,4
21220 DATA 1,KAKAO,100,4,0,0,2
21230 DATA 1,JOGHURT,56,4,2,0,4
21240 REM
21250 REM KAESE,EIER
21260 REM
21270 DATA 1,CAMEMBERT,285,2,4,0,0
21280 DATA 1,EDAMER,384,2,4,0,0
21290 DATA 1,1EI,87,4,3,2,0
21300 DATA 1,SPIEGELEI,218,4,3,2,0
21310 DATA 1,RUEHREI,198,4,2,1,0
21320 DATA 1,OMLETT,285,1,3,4,0
21330 REM
21340 REM FLEISCH
21350 REM
21360 DATA 1,RINDFLEISCH gekocht mager,182,0
,2,4,0
21370 DATA 1,RINDFLEISCH gekocht fett,339,0,
2,4,0
21380 DATA 1,RINDFLEISCH gebraten,300,0,2,4,
0
21390 DATA 1,KALBSSCHNITZEL,160,0,2,4,0
21400 DATA 1,SCHWEINEFLEISCH mager,143,0,2,4
,0
21410 DATA 1,SCHWEINEFLEISCH fett,389,0,2,4,
0
21420 DATA 1,LEBER gebraten,178,0,2,4,0
21430 REM
21440 REM WURST,SCHINKEN
21450 REM
21460 DATA 1,SCHINKEN,335,4,3,1,0
21470 DATA 1,SPECK,532,4,3,1,0
21480 DATA 1,METWURST,460,2,4,1,0
21490 DATA 1,SALAMI,420,2,4,1,0
21510 REM
21520 REM FISCH
21530 REM
21540 DATA 1,KABELJAU,70,0,2,4,0
21550 REM
21560 REM GETRAENKE
21570 REM
21580 DATA 1,BIER,66,1,4,4,4
21590 DATA 1,BIER dkl,120,0,4,4,4
21600 DATA 1,COLA,45,2,4,4,4
21610 DATA 1,SCHNAPS,266,0,2,2,4
21620 DATA 1,COGNAC,336,0,2,2,4
21630 DATA 1,LIKOER,250,0,2,2,4
21640 DATA 1,SEKT,100,0,0,0,4
21650 REM
21660 REM ARBEITEN, VERBRAUCHTE
     KALORIEN
21670 DATA 0,SCHLAFEN,65
21680 DATA 0,LIEGEN,77
21690 DATA 0,SITZEN,100
21700 DATA 0,NAEHEN,115
21710 DATA 0,STRICKEN,118
21720 DATA 0,LESEN ODER SCHREIBEN,120
21730 DATA 0,SINGEN,122
21740 DATA 0,STAUBWISCHEN,130
21750 DATA 0,BUEGELN,144
21760 DATA 0,SPUELEN,144
21770 DATA 0,MASCHINESCHREIBEN,146
21780 DATA 0,ANSTREICHEN,160
21790 DATA 0,FUSZBODENWISCHEN,169
21800 DATA 0,GYMNASTIK,170
21810 DATA 0,NORMAL GEHEN,200
21820 DATA 0,HANTELN,290
21830 DATA 0,SCHNELL GEHEN,300
21840 DATA 0,RADFAHREN,410
21850 DATA 0,SCHWIMMEN,500
21860 DATA 0,TANZEN,500
21870 DATA 0,LAUFEN,570
21880 DATA 0,SCHILAUFEN,600
21890 DATA 0,RUDERN,750
21900 DATA 0,HOLZHACKEN,800
21910 DATA 0,RINGKAMPF,980
21920 DATA 0,TREPPENSTEIGEN,1100
21930 DATA 0,RADRENNEN FAHREN,1300
21940 REM
21950 DATA -1:REM ENDE DER KALORIENTABELLE
```

Diesmal lernen wir wieder eine sehr wichtige Aufgabe der Computertechnik kennen, die Planung. Viele Unternehmungen im Laufe der letzten Jahre wären ohne geeignete Planung durch die Mikroelektronik überhaupt nicht zustande gekommen. Niemand kann vollkommener die einzelnen Fälle einer Unternehmung durchspielen und "Was wäre, wenn"-Überlegungen anstellen als der Computer.

Auch unser Personal-Computer, der Spectravideo kann uns bei Planungen behilflich sein. Wir wollen ihn vor allem in Bereichen von größeren Investitionen einsetzen, beim Autokauf etwa. Deshalb entwerfen wir am Ende dieser Folge ein Programm, mit dem wir uns die wichtigsten Daten einiger Automobile vor Augen halten können.

Vorerst widmen wir uns jedoch einigen neuen BASIC-Anweisungen.

DATA, READ, RESTORE

In unseren Programmen kamen bis jetzt sehr viele "DATA"-Zeilen vor. Dabei haben wir die Befehlsgruppe DATA, READ, RESTORE eher stiefmütterlich behandelt. Wir wollen die ausführliche Erklärung nun nachholen.

Wie wir wissen, dürfen alle möglichen Informationen in DATA-Zeilen stehen. Es ist egal, ob numerische oder alphanumerische Werte vorkommen, ob sie getrennt oder gemischt sind. Sehen wir uns aber sicherheitshalber die einzelnen Fälle und ihre Tücken an:

Numerische Daten müssen ohne Anführungszeichen geschrieben werden. Sie werden durch Beistriche getrennt.

Alphanumerische Daten dürfen auch ohne Anführungszeichen angeschrieben werden, sie werden ebenso durch Beistriche getrennt. Es gelten hier jedoch Ausnahmen:

Wenn eine Zeichenkette einen Beistrich beinhaltet, muß das Anführungszeichen gesetzt werden. Der Computer würde sonst aus der einen Information zwei machen (der Beistrich gilt ja als Trennstrich).

Wenn eine Zeichenkette (=String) nur aus Zahlen und Zahlenzeichen besteht (also auch ".","+" und so weiter), dann sieht der Computer diese Kette als Zahl an, wenn kein Anführungszeichen steht. Eine Fehlermeldung wäre die unvermeidliche Folge.

Wir müssen bei der Verwendung dieser Zeilen immer peinlichst darauf achten, daß die richtigen Typen richtig ausgelesen werden. Wenn man eine Zeichenkette als Zahl lesen will oder umgekehrt, dann meldet sich der Interpreter unwirsch mit einer Fehlermeldung.

Wie kann man nun die einzelnen Werte aus den DATA-Anweisungen "herauskitzeln"?

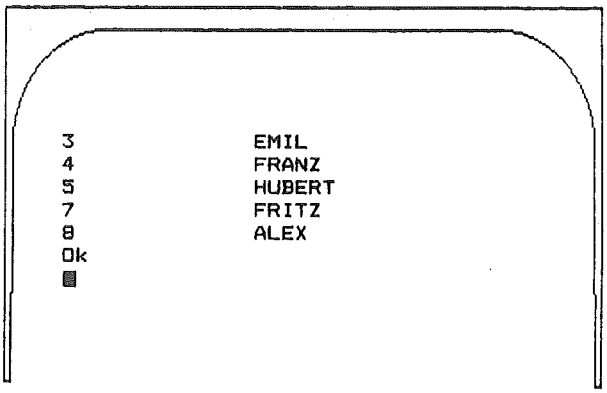
Dies geht mit READ. Wir schreiben hinter READ den Variablennamen, in welchen die Information abgespeichert werden soll. Bei der Ausführung des Befehls READ wird dann die Zeichenkette oder die Zahl in die jeweilige Variable eingeordnet.

Der Computer fängt dabei mit der ersten Information der ersten DATA-Zeile an. Danach arbeitet er der Reihe nach einen Wert nach dem anderen ab. Die Daten in den einzelnen DATA-Kommandos sieht der Computer als einen Strom von Information an. Es ist also egal, ob man seine Daten zum Beispiel in zwei oder in fünf Zeilen aufschreibt.

Sehen wir uns nun ein Beispiel dazu an:

```
10 FOR I=1 TO 5
20 READ A,B$
30 PRINT A,B$
40 NEXT I
50 DATA 3,EMIL,4,FRANZ,5,HUBERT,7,FRITZ
60 DATA 8,ALEX
```

Nach RUN sehen wir folgendes am Bildschirm:



In der Schleife "FOR I=1 TO 5" holt sich der Computer fünf mal je eine Zahl und einen String (=Zeichenkette) aus den DATA-Zeilen. Danach zeigt er sie an. Der Strichpunkt, den wir bis jetzt immer in PRINT-Anweisungen gemacht haben, wurde nun durch einen Beistrich ersetzt. Dies bewirkt einen gehörigen Abstand des Textes von der Zahl. Der Beistrich stellt nämlich die zweite Ausgabe auf die 14. Bildschirmspalte, ein bescheidener Versuch, die Anzeige zu verschönern. Wie wir die Ausgabe noch schöner machen können, zeigen wir etwas später.

Vorerst widmen wir uns aber noch dem dritten Befehl RESTORE. Auch er hat wichtiges zu tun. Wenn wir nämlich in Zeile 45 ein GOTO 10 einbauen würden, dann könnten wir die Ausgabe beliebig oft wiederholen. Doch da machen uns die DATA-Zeilen einen Strich durch die Rechnung. Der Computer hat nämlich einen internen Zeiger, der auf die nächste verwendbare Information in DATA-Zeilen zeigt. Da wir beim ersten Mal schon alle Daten verbraucht haben, ist bei der Wiederholung keine Information mehr da. Der Computer gibt eine Fehlermeldung aus.

Wenn wir den internen DATA-Zeiger nun nach jedem Durchlauf wieder auf den Anfang stellen könnten, würde der Computer diese Schleife beliebig oft wiederholen. Mit

RESTORE schaffen wir dies. Diese Anweisung setzt den Zeiger nämlich auf eine bestimmte Zeilennummer, ab der er dann die nächsten DATA-Zeilen suchen kann. Wenn wir keine Zeilennummer dahinter schreiben, fängt der Zeiger von vorne an.

Wir fügen also zuerst in Zeile 45 ein GOTO 10 ein und vergewissern uns, daß eine Fehlermeldung "OUT of DATA" kommt. Danach setzen wir in Zeile 44 ein RESTORE ein. Die Fehlermeldung verschwindet.

```
10 FOR I=1 TO 5
20 READ A,B$
30 PRINT A,B$
40 NEXTI
44 RESTORE
45 GOTO 10
50 DATA 3,EMIL,4,FRANZ,5,HUBERT,7,FRITZ
60 DATA 8,ALEX
```

Wenn nun DATA-Anweisungen vorkommen, dann wissen wir genauestens Bescheid, wie diese Anweisungen funktionieren und welchen Nutzen sie haben.

TAB, PRINT USING

Als nächstes sehen wir uns die Möglichkeiten an, die Bildschirmausgabe zu formatieren.

Da haben wir zuerst TAB. Diese Anweisung stellt den Cursor an eine bestimmte Stelle am Bildschirm.

Dies probieren wir gleich mit einem kleinen Programm aus.

```
10 FOR I=1 TO 3
20 PRINT TAB(I*3);"HALLO"
30 NEXTI
```

Nach RUN ergibt sich folgendes:

```
   HALLO
      HALLO
         HALLO
Ok
```

Wir sehen, je größer das I*3 wird, desto weiter nach rechts rückt das HALLO.

Weiters haben wir das SPC. Mit dieser Anweisung können wir Zwischenräume beliebiger Größe erzeugen.

```
10 FOR I=1 TO 3
20 PRINT "HALLO";SPC(I*3);"FRANZ"
30 NEXT I
```

Nach RUN sehen wir:

```
HALLO   FRANZ
HALLO      FRANZ
HALLO         FRANZ
Ok
```

Der beste und umfangreichste Befehl zur Bildschirmausgabe kommt nun zum Schluß. Wir werden diesen leider nicht verwenden, da der Computer der Familie Dischek dieses Kommando nicht kennt.

Es ist dies der PRINT USING-Befehl. Mit ihm können wir die einzelnen Bildschirmzeilen in ein festes Schema pressen und so Tabellen anlegen. Wir definieren für die einzelnen Freiräume am Bildschirm, ob eine Zahl oder ein Zeichen dort hingesetzt werden soll. Für diese Definitionen haben wir einige Zeichen zur Verfügung. Fangen wir mit den alphanumerischen Reservierungen an:

Hier haben wir hauptsächlich das "\"-Zeichen. Wir setzen nun zwei dieser Zeichen und füllen beliebig viele Blanks zwischen sie. Nach der Formel "Spaces + 2 = reservierte Plätze" haben wir immer die Möglichkeit, zu berechnen, wie viele Blanks programmiert werden müssen. Um nur ein Zeichen eines Strings abzubilden, nehmen wir das "!" als Reservierung. Ein kleines Beispiel soll dies verdeutlichen:

```
10 INPUT A$
20 PRINT USING "\      \!";A$;"Hallo"
30 GOTO 10
```

Mit diesem kleinen Programm erkennen wir, daß die Stringvariable A$ in das Schema von sechs Zeichen gepreßt wird. Hat sie zu wenig Zeichen, werden Freiräume gebildet, ist sie zu lang, wird sie abgeschnitten. Nach einem Blank Freiraum wird danach das "H" von "Hallo" angezeigt. Die Zeichen zwischen den Anführungszeichen hinter dem Befehlswort nennt man auch Formatierstring (in unserem Fall \   \!).

Sehen wir uns nun die numerischen Reservierungszeichen an: Hier gibt es bedeutend mehr.

Das wichtigste ist das "#"-Zeichen. Es steht für eine Ziffernstelle. "###" ist demnach die Reservierung für eine dreistellige Zahl.

Der Punkt "." kennzeichnet den Dezimalpunkt. Das Komma der Zahl steht immer auf dem Platz des angegebenen Punktes.

Das Plus-(+) und das Minuszeichen(-) reservieren eigene Plätze für das Vorzeichen. Dabei gilt: + vor der Zahl zeigt ein Plus vor einer positiven und ein Minus vor einer negativen Zahl an, - vor der Zahl zeigt einen Leerraum vor der positiven und ein Minus vor der negativen Zahl. + hinter der Zahl, zeigt ein Blank hinter einer positiven und ein Minus hinter einer negativen Zahl. Egal ob positiv oder negativ, bei einem - hinter der Zahl wird immer ein Minus hingeschrieben.

Das "^"-Zeichen sorgt für die Exponentenschreibweise. Hinter der Zahl müssen einfach vier "^" angehängt werden, dann wird diese Zahl immer in dieser Schreibweise angegeben.

Das "*" und das "$" haben die Aufgabe, die Freiräume bei einer zu großen Reservierung einerseits mit Sternen aufzufüllen, oder andererseits am Anfang der Reservierung ein "$"-Zeichen hinzusetzen.

Auch für die numerischen Zeichen gibt es ein Beispiel:

```
10 INPUT A
20 PRINT USING "-###.##^^^^";A
30 GOTO 10
```

Selbstverständlich kann man jedes Zeichen mit jedem kombinieren und in einen Format-

string packen. Man darf auch unterschiedliche Typen mischen, nur muß man achtgeben, daß immer nur ein numerisches Reservierungszeichen für Zahlen und ein alphanumerisches für Zeichenketten steht.

Zu guter letzt hat man sogar die Möglichkeit, in den Formatstring Text zu schreiben. Man kann zum Beispiel "HALLO" direkt hinter das " " programmieren.

Es würde den Rahmen unserer Folge sprengen, wenn wir nun versuchen würden, sämtliche Zeichen mit Beispielen zu demonstrieren. Wir setzen uns lieber mit dem nächsten Programm, dem KFZ-Programm auseinander!

Programm KFZ

Wir wollen uns von einigen Fahrzeugen die Kilometergelder auflisten lassen. Dazu brauchen wir natürlich die Daten über Benzinverbrauch, Steuer und Versicherung. Außerdem werden noch der Hubraum, der Neupreis, der Wertverlust, die Leistung in PS, der Oktanbedarf und der Preis der Reifen in die Berechnung miteinbezogen.

Da es ein unmögliches Unterfangen ist, sämtliche auf dem Markt befindliche Automobile und Motorräder in dem Programm zu vereinigen, wurden nur wenige ausgewählt. Es kann jedoch selbstverständlich jedes beliebige Kraftfahrzeug nachträglich eingegliedert werden.

Nach dem Starten des Programms werden für alle Autos sämtliche Daten in die uns schon bekannten Variablenfelder eingespeichert. Nun kann man in Windeseile auf jede beliebige Information zugreifen.

Der Computer frägt nun nach den eigenen Daten des Fahrzeugs. Wir müssen nun die in einem Jahr gefahrenen Kilometer, die Nutzungsdauer in Jahren, den Anteil der verschiedenen Straßentypen und die Art der Versicherung angeben.

Anhand dieser Daten rechnet der Computer nun das Kilometergeld für die eingespeicherten Fahrzeuge aus. Man kann dann die Kosten vergleichen und sehen, ob das eigene Auto wirtschaftlich ist oder nicht.

In Bild 7.1 sehen wir das Struktogramm für unsere Entwicklung.

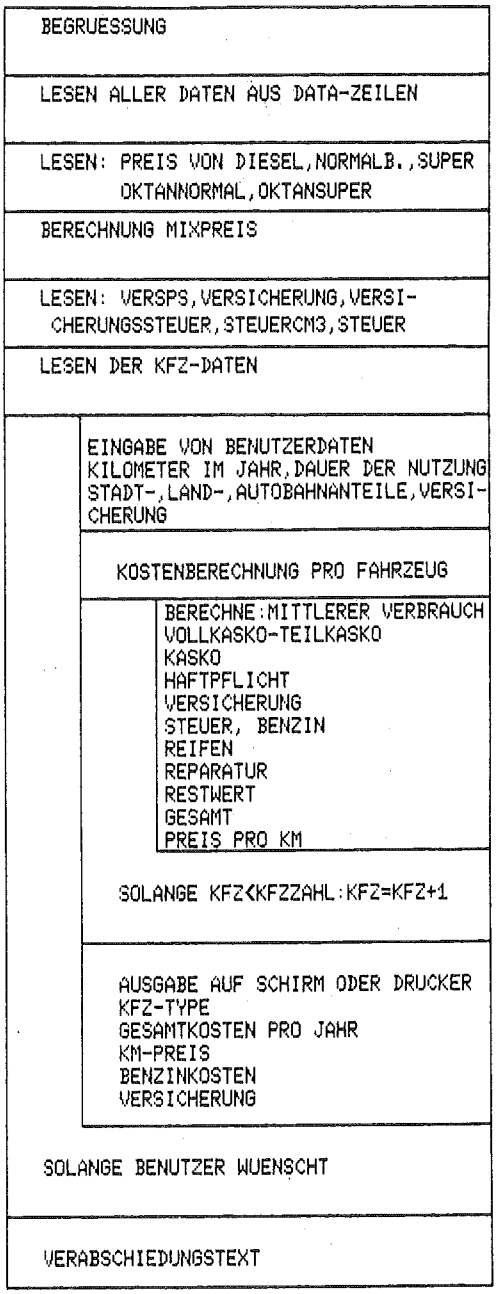


Bild 7.1

Nach der allseits bekannten Vorstellung des Programmes liest der Computer zuerst einmal die Daten für Benzinpreise, Oktanzahlen, Dieselpreise und so weiter heraus.

Nun berechnet der Interpreter den Preis des verwendeten Treibstoffes. Erst danach widmet er sich der Versicherung. Nun werden alle Daten, welche die Versicherung und die Steuer betreffen, ausgelesen. Erst zum Schluß folgen die Kraftfahrzeugdaten für jedes Automobil.

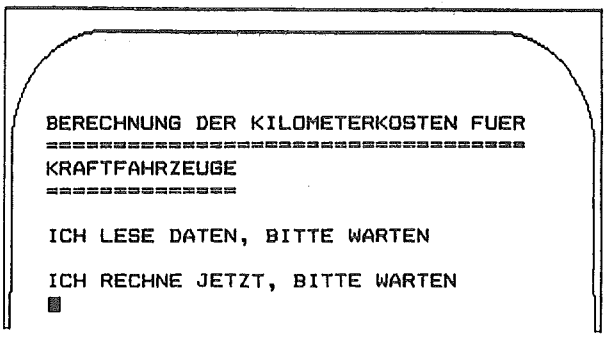
Nachdem alles eingelesen wurde, darf der Bediener die Daten seiner Reisen eingeben. Wie viele Kilometer er im Jahr fährt, welche Nutzungsdauer sein Auto hat, welche Versicherung er abgeschlossen hat und wie hoch der Anteil an Stadt- und Landstraßen, sowie an Autobahnen ist.

Nun hat der Computer eine Unmenge von Daten und kann mit seinen Berechnungen beginnen. Dazu werden für jedes Fahrzeug alle vorgesehenen Parameter festgestellt. Dies sind der Verbrauch im Durchschnitt, die Versicherung, die Steuer und einige andere.

Nachdem der Computer mit den Berechnungen fertig ist, werden alle Daten in einer Schleife auf den Bildschirm geworfen. Nun wird dem Bediener das schon oben erwähnte Ergebnis präsentiert und er kann entscheiden, ob er wirtschaftlich gefahren ist oder nicht.

Danach stellt der Computer den Benützer vor die Wahl, ob er noch einen Vergleich starten will. Wenn ja, dann läuft alles von der Eingabe der Daten an noch einmal ab.

Zuletzt zeigen wir das Anfangsbild am Schirm des Fernsehers:



```
10 REM  BERECHNUNG DER KILOMETERKOSTEN
20 REM  FUER KRAFTFAHRZEUGE
2000 CLS
2010 PRINT "BERECHNUNG DER KILOMETERKOSTEN F
UER": PRINT "==============================
====": PRINT "KRAFTFAHRZEUGE"
2020 PRINT"==============="
2030 PRINT
2040 PRINT"ICH LESE DATEN, BITTE WARTEN"
2060 REM LESEN DER DATEN
2070 REM ================
2080 REM BENZINPREISE
2090 READ DP,NP,SP
2100 DATA 11,10.6,11.1
2110 MP=(NP+SP)/2
2130 REM OKTANZAHLEN
2140 READ OL,OS
2150 DATA 87,97
2170 REM VERSICHERUNGSSUMMEN
2180 READ VK,VS
2190 DATA 8,8.5
2200 VS=1+(VS/100)
2210 FOR SE=1TO VK
2220     READ VP(SE),VG(SE)
2230     VG(SE)=VG(SE)*VS
2240 NEXT SE
2250 REM PS-ZAHL,VERSICHERUNGSSUMME
2260 DATA 16,1439
2270 DATA 20,1805
2280 DATA 34,2450
2290 DATA 50,3392
2300 DATA 70,4168
2310 DATA 90,5197
2320 DATA 120,5590
2330 DATA 200,6810
2350 REM KFZ-STEUER
2360 READ SK
2370 DIM SC3(SK),S(SK)
2380 FOR SE=1TO SK
2390   READ SC3(SE),S(SE)
2400   S(SE)=S(SE)*12
2410 NEXT SE
2420 DATA 10
2430 REM CM3,MONATLICHE STEUER
2440 DATA 1000,60
2450 DATA 1250,90
2460 DATA 1500,120
2470 DATA 1750,180
2480 DATA 2000,225
2490 DATA 2500,360
2500 DATA 3000,450
2510 DATA 3500,600
2520 DATA 4000,750
2530 DATA 10000,1050
2550 REM KFZ-DATEN
2560 READ KZ
2570 DIM KT$(KZ),NP(KZ),WM(KZ),KT(KZ)
2580 DIM KC3(KZ),KP(KZ),SV(KZ),LV(KZ)
2590 DIM AV(KZ),OZ(KZ),GV(KZ)
2600 DIM V(KZ),VO(KZ),TK(KZ),HP(KZ)
2610 DIM KS(KZ),BK(KZ),R(KZ),RR(KZ),G(KZ),RW
(KZ),KM(KZ)
2620 FOR K=1 TOKZ
2630   READ KT$(K),NP(K),WM(K),KT(K)
2650   READ KC3(K),KP(K),SV(K),LV(K)
2670   READ AV(K),OZ(K),R(K)
2690   NEXT K
2700 REM AUFBAU DER DATAZEILEN FUER KFZ
2710 REM KFZ-NAME, NEUPREIS, WERTMINDERUNG
2720 REM KASKOTARIF, HUBRAUM, LEISTUNG
2730 REM STADTVERBRAUCH, LANDVERBRAUCH
2740 REM AUTOBAHNVERBRAUCH, OKTAN, REIFEN
2750 DATA 8
2760 DATA PRAERIE,147900,8,7.2,1488,70,7.8,6
.2,8.7,98,1000
2770 DATA MERCEDES240D,282216,5,5.1,2399,72,
9.5,7.2,9.9,0,1000
2780 DATA SUBARU,151300,9,6.9,1595,70,9.5,7.
1,10,90,1000
2790 DATA R4GTL,93300,10,6.6,1108,34,6.3,5.4
,6.5,98,1000
2800 DATA VW-POLO,111340,5,6.3,1043,40,7.6,5
.6,7.5,91,1000
2810 DATA PORSCHE-TARGA,890000,10,4.5,3229,3
00,15.5,9.7,11.8,98,1000
2820 DATA HONDA-ACCORD,143800,8,6.6,1598,90,
9.1,5.7,7.8,91,1000
2830 DATA CX-25-DIESEL,233540,10,5.7,2499,75
,8.9,6.1,8.1,0,1000
3000 REM EINGABE DER DATEN
3010 REM BZW. STANDARDDATEN
3060 REM
3070 JK=20000:ND=10:ST=50:L=20:A=30:V=3:GOTO
3180
3080 INPUT"GEFAHRENE KILOMETER/JAHR  :";JK
3090 INPUT"NUTZUNGSDAUER IN JAHREN :";ND
3100 INPUT"STADTANTEIL (%)           :";ST
3110 INPUT "LANDSTRASZENANTEIL (%)    :";L
3120 INPUT "AUTOBAHNANTEIL (%)        :";A
3130 PRINT "VERSICHERUNG (ZAHL EINGEBEN)"
3140 PRINT "          1. HAFTPFLICHT"
3150 PRINT "          2. TEILKASKO"
3160 PRINT "          3. VOLLKASKO"
3170 INPUT V
3180 PRINT
3190 PRINT"ICH RECHNE JETZT, BITTE WARTEN"
4000 REM BERECHNUNG DER KOSTEN
4010 REM ====================
4030 FOR K=1TO KZ
4040 IF T THEN PRINT:PRINTKZ$(K)
4060 V(K)=(ST*SV(K)+L*LV(K)+A*AV(K))/100
4070 IF T THEN PRINT "VRBRAUCH:"V(K)
4100 VO(K)=(NP(K)*KT(K))/100
4110 IF TTHEN PRINT "VOLLKASKO:":PRINTVO(K)
4120 TK(K)=VO(K)/2
4130 IF T THEN PRINT "TEILKASKO:"TK(K)
4160 SE=0
4170 IF KP(K)<VP(SE) OR SE>VKGOTO4190
4180 SE=SE+1:GOTO4170
4190 HP(K)=VG(SE)
4200 IF TEST THEN PRINT"HAFTPFLICHT:"HP(K)
4240 SE=0
4250 IFKC3(K)<SC3(SE) OR SE>SKGOTO4270
4260 SE=SE+1:GOTO4250
4270 KS(K)=S(SE)
4280 IF TEST THEN PRINT "KFZSTEUER:"KS(K)
4320 OM=(OL+OS)/2
4330 O1=(OL+OM)/2
4340 O2=(OM+OS)/2
4350 IFOZ(K)=0 THEN BP=DP ELSE IF OZ(K)<O1 T
HENBP=NPELSE IFOZ(K)<O2THENBP=MPELSEBP=SP
4360 BK(K)=JK*V(K)*BP/100
4370 IFTEST THEN PRINT "BENZINKOSTEN:"BK(K)
4430 RR(K)=VO(K)*.25
4440 IFT     THEN PRINT "REPARATUR:"RR(K)
4470 RW(K)=NP(K)-(NP(K)*ND*WM(K)/100)
4480 IFTEST THEN PRINT"RESTWERT:"RW(K)
4510 ON V GOSUB 4550,4560,4570
4520 GV(K)=GV(K)+HP(K)
4530 IFTEST THEN PRINT" GESVERSICHERUNG:"GV(
K)
4540 GOTO4600
4550 GV(K)=0:RETURN
4560 GV(K)=TK(K):RETURN
4570 GV(K)=VO(K):RETURN
4600 G(K)=((KS(K)+GV(K)+BK(K) +R  (K)+RR(K))*
ND+NP(K)-RW(K))/ND
4610 IF TEST THEN PRINT "GESAMT:"G(K)
4620 REM
4630 REM
4640 KM(K)=G(K)/JK
4650 IF T THEN PRINT " KMPREIS:"KM(K)
4660 NEXT K
4670 PRINT"BERECHNUNG BEENDET"
4680 PRINT:PRINT
5000 REM AUSGABE DER BERECHNETEN DATEN
5010 REM ============================
5020 CLS:PRINT"KFZ-TYP ";"S/JAHR  ";"KM-PREI
S ";"BENZIN ";"VERS."
5040 PRINT"-------------------------------
------"
5080 K=0
5090 FORY=3TO 19 STEP 2
5095  K=K+1:IF K>KZ THEN GOTO 5170
5100  LOCATE 0,Y :A$=KT$(K)
5110  L=INT(G(K))
5120  M= INT(KM(K)*100)/100
5130  N=INT(BK(K))
5140  O=INT(GV(K)):PRINTA$
5150  PRINT"       ";L;" ";M;" ";N;" ";O
5160 NEXTY
5170 PRINT "NOCH EINE BESCHREIBUNG (J/N)":IN
PUT W$ :IFW$="J"ORW$="j"THEN3080
5190 PRINT
5200 PRINT"HABEN SIE GUT GEWAEHLT?"
5210 PRINT"VIELLEICHT SOLLTEN SIE DOCH LIEBE
R SPAREN"
5220 END
```

In dieser Folge schneiden wir einen weiteren sehr wichtigen Bereich der Computertechnik an, die Verwaltung von Lagerbeständen. Anlaß gibt uns der Vater in der Familie Dischek, der in einer großen Firma arbeitet. Die Firma stellt auf EDV um und Herr Dischek nimmt dies zum Anlaß auch zu Hause einiges auf EDV umzustellen.

Wir werden ebenso wie die Familie Dischek mit unserem Spectravideo eine kleine Verwaltung von Lagerbeständen "aufziehen". Zu diesem Zweck wollen wir unsere Schallplattensammlung einmal geordnet im Computer haben.

Bevor wir uns aber über das Programm DataStar stürzen und mit ihm unsere Sammlung sortieren, nehmen wir ein paar "Nachhilfelektionen" in Sachen "Datenbank".

Dateiorganisation

Wir haben Dateien ja schon kennengelernt, nämlich als wir unser Kalorienprogramm erstellt hatten. Nun beschäftigen wir uns aber etwas länger mit den Files.

Wenn wir mit Daten in Files arbeiten, dann wollen wir bestimmte Forderungen erfüllt sehen. Einerseits möchten wir Files anlegen und löschen können. Andererseits wollen wir aber auch schon vorhandene Dateien mit Informationen erweitern, Informationen löschen oder Informationen ändern.

Als zusätzliche Wünsche haben wir das Ausdrucken von Dateien, das Suchen von bestimmten Daten und das Sortieren von Daten in einer Datei.

Wir konnten im BASIC sehr leicht Dateien anlegen und löschen, aber zum Sortieren oder Suchen von Informationen brauchen wir längere Programme. Es liegt daher nahe, ein eigenes Software-Werkzeug zu verwenden, welches uns alle Wünsche erfüllt. Natürlich haben schon viele Leute vor uns diesen Wunsch gehabt, und so ist es nicht verwunderlich, daß wir in unserem Fall wieder nur auf ein schon vorhandenes Programmwerkzeug zurückzugreifen brauchen. DataStar heißt das Programm, welches unter CP/M läuft.

Zum Verwalten von Daten und Dateien gibt es genug Programme. Diese haben natürlich auch einen Namen. Sie heißen Datenbanksysteme.

DataStar hat aber noch andere Vorteile. Man spricht bei diesem Tool von einem Datenbanksystem mit Benutzerinterface. Wir können die Datenbank auf unsere Bedürfnisse abstimmen und über den Bildschirm alles notwendige veranlassen, um für jeden Spezialfall gewappnet zu sein.

Dabei ist es sehr günstig, daß uns das Programm jeden Schritt, den wir zur Anpassung machen müssen, mittels Menütechnik genau angibt. So sind keine Programmierkenntnisse notwendig, um das Programm "abzustimmen".

Klären wir nun den Begriff der Datei:

Files, Records, Felder

Wir sagten schon, daß in einer Datei (oder File) Daten in einem Rahmen ordnungsgemäß gesammelt untergebracht sind.

Wir können eine Software-Datei ohne weiteres mit einer Kartei vergleichen, mit einigen Zetteln, die nicht kunterbunt am Tisch herumliegen, sondern ordnungsgemäß in einer Schachtel einsortiert sind (Bild 8.1).

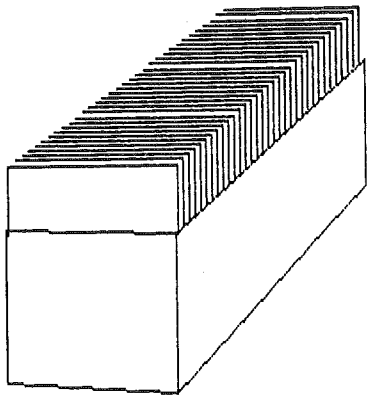


Bild 8.1

Aber es bleibt natürlich nicht nur bei der Bezeichnung Datei alleine, das File wird in viele kleinere Teile unterteilt. So finden wir ein File in einzelne Records aufgegliedert. Bleiben wir bei unserem Beispiel mit der Kartei. Angenommen, wir sammeln die Titeln von Büchern in ihr, dann ist jedes Blatt ein Record (oder Datensatz).

Aber diese Datensätze sind weiter unterteilt. Wir haben ja auch auf unserem Zettel, den Buchtitel, den Autor, eventuell die Seitenzahl und so weiter stehen. Diese Abschnitte nennt man Felder. Bild 8.2 zeigt uns die einzelnen Felder unserer Karteikarte.

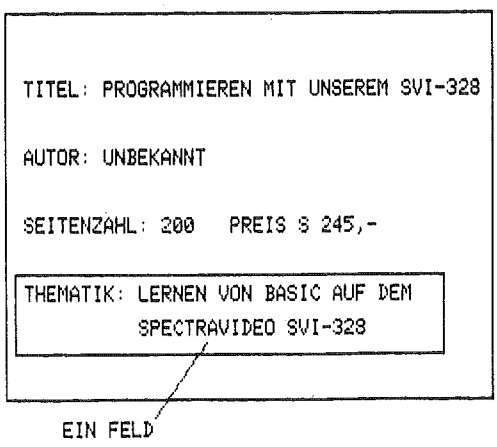


Bild 8.2

Die einzelnen Felder bestehen ihrerseits aus Zeichen, so wie der Buchtitel aus Buchstaben, Zahlen und so weiter besteht. Diese Zeichen sind die kleinsten Einheiten der Files.

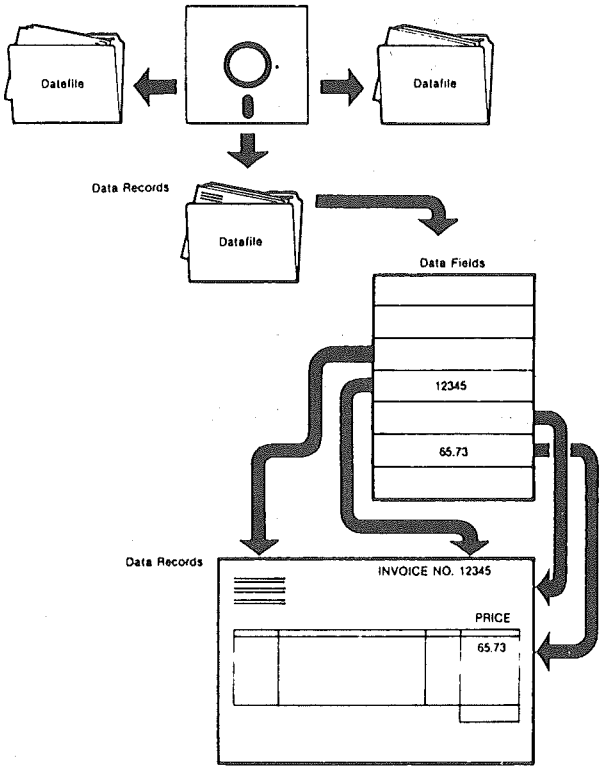


Bild 8.3

Die Darstellung in Bild 8.3 verdeutlicht uns nochmals den Zusammenhang zwischen Dateien, Datensätzen und Feldern. Auf einer Diskette befinden sich mehrere Dateien (Datafiles). Jede Datei beinhaltet eine Anzahl von Datensätzen (Data Records). Jeder Record besteht aus Datenfeldern (Data Fields). Solche Felder können zum Beispiel sein: Rechnungsnummer, Preis usw.

Damit haben wir aber noch nicht genug definiert. Wir müssen noch den Begriff Index im Gebiet der Dateistrukturen erklären. Um eine raschere Suche nach bestimmten Datensätzen zu ermöglichen, wird eine sogenannte Indexdatei angelegt. In dieser werden die Daten aus den Indexfeldern (auch Schlüsselfelder genannt) untergebracht. Bei jeder Eintragung findet sich ein Index (Zeiger) auf den tatsächlichen Datensatz. So läßt sich die Suche nach bestimmten Records wesentlich rascher erledigen. Dies ist besonders bei längeren Dateien wünschenswert.

Sehen wir uns nun das Programm DataStar an.

DataStar

Das Datenbanksystem DataStar besteht aus zwei Teilen: FormGen und DataStar. FormGen ist der Programmteil, der dazu verwendet wird, die Struktur einer Datei zu bestimmen. Als Vorbereitungsarbeit für jede Datei muß eine Maske erstellt und abgespeichert werden. Erst jetzt können mit DataStar Eintragungen in die Datei vorgenommen werden. Das Zusammenwirken von FormGen und DataStar ist in Bild 8.4 illustriert.

Unsere Schallplattensammlung

Mit Hilfe von FormGen schafft es die Familie Dischek sehr leicht, ihre Schallplattensammlung auf der Diskette zu verewigen. Das Programm ist nämlich bedienergeführt, das heißt, daß der Bediener keinerlei Programmiererfahrung haben muß. Alles, was der Benützer zu machen hat, wird im Programm erklärt.

Bevor wir unsere Datei anlegen, müssen wir uns jedoch zuerst einmal der Form unserer Records bewußt werden. Die Familie Dischek hat dies getan und ein Record folgendermaßen eingeteilt:

Schallplattentitel:	20 Stellen
Interpret:	20 Stellen
Komponist:	20 Stellen
Fortlaufende Nummer der Schallplatte:	6 Stellen
Anschaffungsdatum:	12 Stellen
Kategorie:	10 Stellen
Bemerkung:	20 Stellen
Inhalt Seite 1:	4 Zeilen je 60 Stellen
Inhalt Seite 2:	4 Zeilen je 60 Stellen

Weiters wurde noch festgelegt, daß nach den Feldern "Schallplattentitel", "Interpret" und "Komponist" sortiert werden soll und daß auf diese Felder direkt zugegriffen werden kann.

Nach Aufruf des Programms FormGen können die entsprechenden Funktionen aus einem Hilfsmenü ausgewählt werden. Nach Angabe des Dateinamens erscheint am Bildschirm ein leeres Dateiblatt. In dieses können nun zwei verschiedene Arten von Informationen eingegeben werden:

Anleitungs- und Kommentartexte
Felder, die im Record später Daten aufnehmen sollen

Die Anleitungen und Kommentare werden beim späteren Arbeiten mit der Datei am Bildschirm dargestellt. Sie sind der Text der Maske, in die Daten eingetragen werden. Gute Information erleichtert dem Bediener das Ausfüllen der Datei.

Durch Verwendung des Unterstreichungszeichens, das im Kommentartext nicht verwendet werden darf, können Felder erstellt werden, die beim Arbeiten mit der Datei die eigentliche Information pro Record aufnehmen sollen. Bild 8.5 zeigt das Beispiel einer Adreßkartei.

Felder können mit Attributen versehen und als Schlüsselfelder definiert werden. Schlüsselfelder werden in der Maske mit "*" angezeigt. Nach Abschluß der Definitionsarbeit ist die Datei zur Eingabe von Daten mittels DataStar bereit und wir können mit

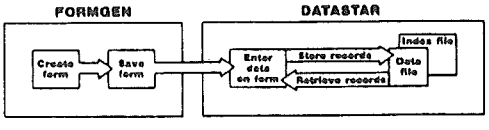


Bild 8.4

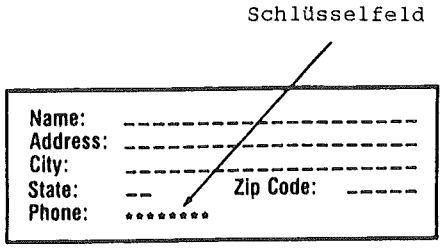


Bild 8.5

der Funktion "Data Entry" die Schallplattensammlung der Computerfamilie eingeben. Einen Ausschnitt zeigt Bild 8.6.

DataStar enthält Funktionen, um den Inhalt von Records am Drucker auszugeben, sowie einfache Listen auszudrucken. Mit Hilfe des Programms ReportStar können darüber hinaus komplex aufgebaute Listen mit Informationen erstellt werden, die aus DataStar-Dateien stammen. ReportStar erlaubt es, mittels komfortabler Editierfunktionen auch Ausdrucke zu erstellen, die Berechnungen enthalten. Diese "Berichte" können darüber hinaus auch noch mit WordStar editiert werden.

Die Zusammenarbeit der Programmwerkzeuge DataStar und ReportStar zeigt Bild 8.7. Mit FormGen wird ein sogenannter Form Definition File(.DEF) erzeugt. Mit DataStar wird dann der Datenfile(.DTA) und der zugehörige Indexfile(.NDX) erzeugt. Mit ReportStar kann man nun den Report Specification File(.RPT) als Formular für den auszugebenden Textfile erstellen. Der Textfile kann dann entweder direkt am Drucker ausgegeben werden oder aber auf der Diskette als Textfile(.PRN) gespeichert werden. In letzterem Fall könnte man diesen File auch mit WordStar editieren und erst später ausdrucken. In Klammer wurden jeweils die sogenannten Extensions

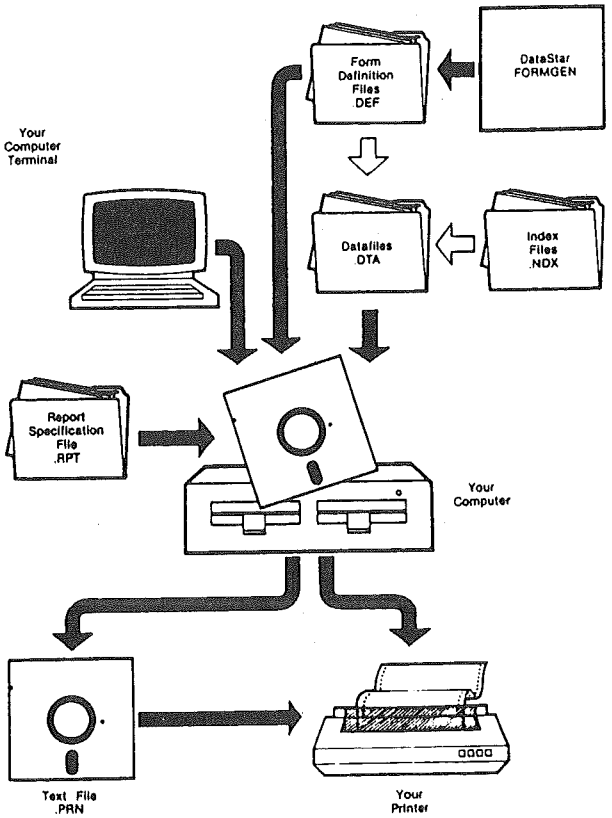


Bild 8.7

(Dateikennung) angegeben, mit deren Hilfe man die einzelnen Files unterscheiden kann.

Mit Hilfe von DataStar und ReportStar können nahezu alle Aufgaben einer Datenbank, die vom Datenvolumen her gesehen mit Hilfe von Personalcomputern sinnvoll lösbar sind, rasch und effizient und ohne Programmierkenntnisse bewältigt werden.

INTERPRET	BEATLES	KOMPONIST BEATLES
TITEL	1966-1974	
NUMMER	999999	KATEGORIE BEAT
DATUM	1980	BEMERKUNG DOPPEL-LP
SEITE 1	YESTERDAY	TICKET TO RIDE
	HEY JUDE	HELP
	SOMETHING	WHY DON'T WE DO IT IN THE ROAD
	YELLOW SUBMARINE	YEAH,YEAH,YEAH
SEITE 2	HELTER SKELTER	LET IT BE
	LITTLE PIGGIES	ALL YOU NEED IS LOVE
	MISTER UNIVERSE	

INTERPRET	TALKING HEADS	KOMPONIST DAVID BYRNS
TITEL	FEAR OF MUSIC	
NUMMER	111111	KATEGORIE NEW WAVE
DATUM	1980	BEMERKUNG
SEITE 1	I ZIMBRA	MIND
	PAPER	CITIES
	LIFE DURING WARTIME	MEMORIES CAN'T WAIT
	AIR	
SEITE 2	HEAVEN	ANIMALS
	ELECTRIC DREAMS	DRUGS

Bild 8.6

Wer kennt nicht die hektische Rennatmosphäre bei Formel 1-Rennen, das Zittern der Zuschauer vor dem Bildschirm, wenn unsere Schiasse die schneebedeckten Hänge hinabstürzen. All diese sportlichen Ereignisse wären ohne Computer undenkbar. Die computergesteuerte Zeitmessung der Teilnehmer garantiert optimale unbestechliche Zeiten.

Natürlich findet auch die Familie Dischek einen Anwendungsbereich für ihren Computer in Sachen Sport. Ziel des diesmaligen Programms soll es sein zu entscheiden, welche Fußballspieler für die Nationalmannschaft am Besten sind. In der Fernsehfamilie gibt es nämlich die Streitfrage, ob die Stürmer von Vaters Lieblingsklub, oder die Stürmer vom Lieblingsklub des Sohnes besser in das Österreich-Team passen. Was liegt hier näher, als den unbestechlichen Computer auswählen zu lassen?

Vorher lernen wir jedoch die Anweisungen INKEY$ und INPUT$ kennen. Zusätzlich sehen wir eine sehr nützliche Einrichtung im BASIC an, nämlich das Unterprogramm.

INKEY$, INPUT$

Zur Eingabe von Zeichen kennen wir schon INPUT. Allerdings müssen wir bei dieser Art der Eingabe zum Abschluß immer ENTER drücken. Außerdem erscheint ein Fragezeichen am Bildschirm, wenn der Interpreter auf so einen INPUT-Befehl trifft. Im Übrigen kann man das INPUT-Kommando nicht in der hochauflösenden Graphik verwenden. Also müssen wir nach einem weiteren Eingabebefehl im Bedienerhandbuch stöbern. Da treffen wir auf die Anweisung INKEY$. Mit ihr sind wir in der Lage, die Tastatur auf ein Zeichen abzufragen.

Wenn der Computer in einem Programm auf so einen Befehl trifft, dann wird die Tastatur abgefragt, ob eine Taste gedrückt ist. Wenn ja, wird in die vor dem Befehlswort stehende Variable das Zeichen der gedrückten Taste eingespeichert (kann immer nur ein Zeichen sein!). Wenn keine Taste betätigt wurde, wird der Leerstring eingespeichert. Dieser Leerstring enthält kein einziges Zeichen, nicht einmal ein Blank. Er hat daher die Länge Null, und wird im BASIC durch zwei Anführungszeichen ohne Zwischenraum gekennzeichnet: "".

Wenn der Computer nun warten soll, bis eine Taste gedrückt ist, dann müssen wir mit einer IF..,THEN-Anweisung abfragen, ob in der Variablen der Leerstring steht. Wenn dem so ist, wird wieder zu INKEY$ zurückgesprungen (da noch keine Taste gedrückt worden ist, der Computer aber solange warten soll, bis eine betätigt wurde). Dies geht solange vor sich, bis ein Zeichen in der Variablen abgespeichert ist, dann läßt der Vergleichsbefehl hinter INKEY$ den Computer weitere Befehle abarbeiten.

Sehen wir uns nun den Ausschnitt eines Programms an, welches diesen Befehl verwendet.

```
10 PRINT "TEST
20 PRINT " 1) TEST WIEDERHOLEN
30 PRINT " 2) TEST ABSPEICHERN
40 PRINT " 3) TEST AUSWERTEN
50 PRINT " GEWUENSCHTE ZAHL EINGEBEN
60 A$=INKEY$
70 IF A$=""THEN 60
80 ....
```

Dieses Programm soll aus einem größeren herausgenommen sein und ein Menü darstellen. Wichtig ist für uns nur die Funktion des INKEY$. Wir sehen die IF...THEN-Anweisung, welche immer wieder zur Zeile 60 zurückspringt, wenn die Variable A$ leer ist. Erst nach dem Drücken einer Taste wird Zeile 80 verarbeitet.

INKEY$ wird in der Regel dazu verwendet, um während eines langen Programmablaufes, zum Beispiel während einer Berechnung immer wieder nur kurz die Tastatur abzufragen. Wenn eine Taste gedrückt worden ist, wird dann die lange Berechnung abgebrochen. Da dieser Befehl nicht auf eine Eingabe wartet (man muß dies mit IF...THEN erreichen), ist das INKEY$ prädestiniert dazu zu fragen, ob irgend ein Vorgang abgebrochen werden soll.

Wenn man definiert auf eine Eingabe warten will, aber kein ENTER drücken möchte, nimmt man INPUT$ (nicht INPUT!). Hier kann man eine bestimmte Anzahl von Zeichen vorbestimmen, die eingegeben werden soll. Wenn man A$=INPUT$(3) schreibt, wartet der Computer, bis eben 3 Zeichen eingegeben wurden. Danach setzt er automatisch das Programm fort. Wenn wir A$=INPUT$(1) programmieren, dann können wir ein Zeichen eingeben, ohne die Warteschleife mit INKEY$ zu verwenden. Das Programm von vorhin lautet demnach verbessert:

```
10 PRINT "TEST
20 PRINT " 1) TEST WIEDERHOLEN
30 PRINT " 2) TEST ABSPEICHERN
40 PRINT " 3) TEST AUSWERTEN
50 PRINT " GEWUENSCHTE ZAHL EINGEBEN
60 A$=INPUT$(1)
80 .....
```

Zusätzlich gibt es beim MSX-BASIC noch das LINEINPUT. Es funktioniert im wesentlichen wie das INPUT, nur das keine numerischen Variablen "gefüllt" werden dürfen. Der Vorteil bei dieser Anweisung ist, daß kein Fragezeichen am Bildschirm auftaucht.

Bei INKEY$ und INPUT$ muß man im Gegensatz zu INPUT und LINEINPUT auch die Anzeige der Eingabe mit PRINT bewerkstelligen. Die beiden zuletzt Genannten sorgen von selber für die Ausgabe der Zeichen auf den Bildschirm. Wir erweitern daher unser Programm von vorhin mit Zeile 70 wie folgt:

```
10 PRINT "TEST"
20 PRINT " 1) TEST WIEDERHOLEN
30 PRINT " 2) TEST ABSPEICHERN
40 PRINT " 3) TEST AUSWERTEN
50 PRINT " GEWUENSCHTE ZAHL EINGEBEN
60 A$=INPUT$(1)
70 PRINT A$
80 .....
```

Einen Pferdefuß hat das Ganze noch:

Bei INKEY$ und INPUT$ dürfen nur alphanumerische Variable verwendet werden. Wenn wir aber eine Zahl aus der Eingabe ermitteln wollen, müssen wir uns etwas anderes überlegen. In unserem Fall wird ja nach einer Zahl gefragt. A$ ist aber eine alphanumerische Variable. Es gibt daher einen ungemein nützlichen Befehl, der sich VAL nennt. Mit diesem kann man alphanumerische Strings in reine Zahlen umwandeln. Kommt ein Zeichen im String vor, welches nicht als Zahlenzeichen erkannt werden kann, wird eine Null ausgegeben. Stehen vor dem nicht zu brauchenden Zeichen Ziffern, so werden diese Ziffern als eine Zahl angesehen und als Ergebnis der Umwandlung betrachtet.

In der vollständigen Fassung muß das Programm nun so lauten:

```
10 PRINT "TEST"
20 PRINT " 1) TEST WIEDERHOLEN
30 PRINT " 2) TEST ABSPEICHERN
40 PRINT " 3) TEST AUSWERTEN
50 PRINT " GEWUENSCHTE TASTE DRUECKEN
60 A$=INPUT$(1)
70 PRINT A$:B=VAL(A$)
80 ....
```

Nach dieser zugegeben ausführlichen Beschreibung der verschiedenen Eingabearten im MSX-BASIC widmen wir uns nun dem zweiten wichtigen Kapitel in dieser Folge, dem Unterprogramm (auch oft Subroutine genannt).

Unterprogramme

Es gibt in vielen Fällen Programme, die gewisse Teile sehr oft in unveränderter Form brauchen. Es kann zum Beispiel eine Anzeigeneinheit mit einigen Zeilen Umfang sechs oder sieben mal gebraucht werden. Wenn man nun sehr oft immer das Gleiche programmieren muß, nur an verschiedenen Stellen, dann ist dies eigentlich eine Speichervergeudung. Es muß doch eine Möglichkeit geben, diese oft verwendete Einheit nur einmal zu programmieren, und dann von jeder beliebigen Stelle anzuspringen. Wir dürfen das nicht mit GOTO machen, denn wenn wir einmal abgesprungen sind, dann weiß der Interpreter nicht mehr, von woher er gekommen ist. Nun ist es aber notwendig zu wissen, wo die Absprungzeile steht, denn wir müssen nach der Anzeigeneinheit wieder dorthin zurückkehren.

Ebenso können wir am Ende der Einheit kein GOTO hinstellen, denn GOTO springt immer zu einer bestimmten Stelle. Der Rücksprung zum Hauptprogramm muß aber in Abhängigkeit zur Absprungstelle erfolgen. Wir brauchen neue Befehle!

Die bekommen wir mit GOSUB und RETURN. GOSUB springt zu einer bestimmten Zeile, merkt sich aber, von wo es weggesprungen ist. RETURN springt zum nächstfolgenden Befehl hinter der Absprungstelle zurück. Jetzt können wir unseren kleinen Programmteil eliminieren, am Ende des nun entstandenen kleinen Programms setzen wir ein RETURN. Dies garantiert uns den Rücksprung in das Mutterprogramm.

Im Hauptprogramm setzen wir wiederum überall dort, wo bisher die Einheit gestanden hat, das GOSUB. In Bild 9.1 und 9.2 sehen wir den Unterschied der beiden Programmierarten, einmal mit und einmal ohne eliminierten Programmteil. Dieser Programmteil nennt sich übrigens Unterprogramm.

Wir sehen, daß unser Unterprogramm nur durch GOSUB anspringbar ist, durch keinen anderen Befehl. Im BASIC können wir das

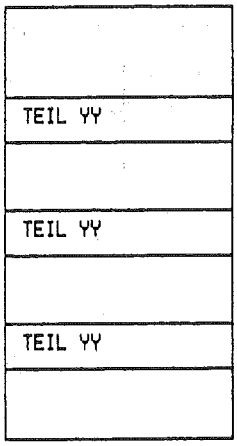


Bild 9.1

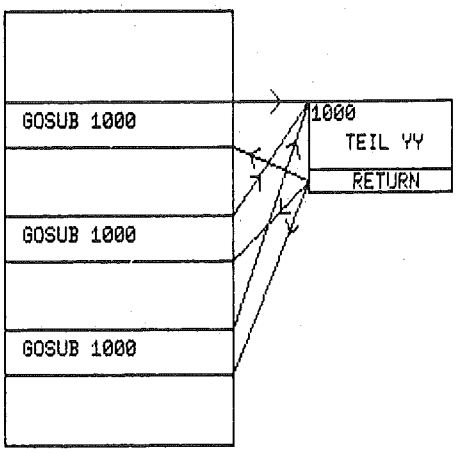


Bild 9.2

Unterprogramm aber nicht so deutlich unterscheiden, wie in unserem Bild. Daher müssen wir bei der Verwendung von Subroutines IMMER darauf achten, daß das Hauptprogamm mit END deutlich abgesichert ist. Kein GOTO darf vom Hauptprogramm in das Unterprogramm zeigen.

Es ist übrigens immer ratsam, das Unterprogramm durch hohe Zeilennummern vom Hauptprogramm abzutrennen. Ein Beispiel für die Kombination von Unterprogrammen wird nun gegeben:

```
10 REM HAUPTPROGRAMM
20 ....
40 GOSUB 1000
50 ....
90 GOSUB 1000
100 ....
....
200 END
1000 REM SUBROUTINE
1010 ....
1020 .....
1040 RETURN
```

In unserem Beispiel wird das Unterprogramm zweimal angesprungen. Man kann hier auch sehen, daß hinter GOSUB eine Zeilennummer steht, ähnlich dem GOTO. Diese Zeilennummer kennzeichnet den Anfang des Unterprogramms.

Man kann Subroutines auch verschachteln. Von einem Unterprogramm kann in ein weiteres gesprungen werden. Bild 9.3 zeigt diesen Effekt. Beim Spectravideo wird die Zahl der Verschachtelungen nur durch die Speicherkapazität begrenzt.

Wie zu Erwarten war, bietet auch das RETURN einige "Gustostückerln". Es gibt den Fall, daß man mit GOSUB in ein Unterprogramm ein-

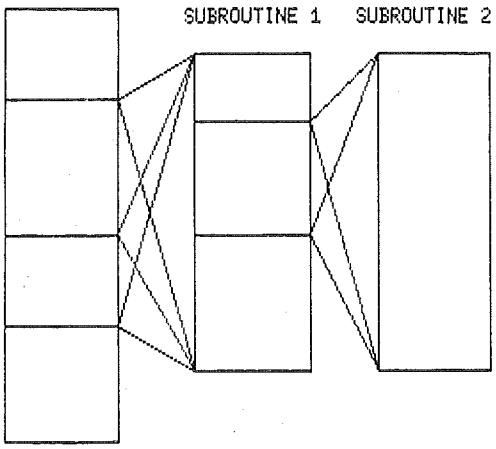


Bild 9.3

steigt, aber anhand irgendwelcher Vergleiche einmal nicht zur Absprungstelle im Hauptprogramm sondern an eine ganz bestimmte definierte Stelle im Hauptprogramm zurückspringen möchte. Mit GOTO darf man das natürlich nicht lösen. Man darf aber hinter RETURN eine Zahl schreiben. Zu dieser Zeilennummer wird dann zurückgesprungen.

Man muß immer mit einem RETURN aus dem Unterprogramm zurückkehren, sonst merkt sich der Computer weiterhin die Absprungstelle im Hauptprogramm. Dies kann zu Fehlfunktionen im Programm führen.

Wenn ein Unterprogramm ohne GOSUB aktiviert wird, bekommt der Bediener eine Fehlermeldung präsentiert, wenn der Interpreter auf das RETURN stößt.

Nach dieser Theorie stürzen wir uns wieder in den harten Programmieralltag. Wir wollen den besten Stürmer ermitteln.

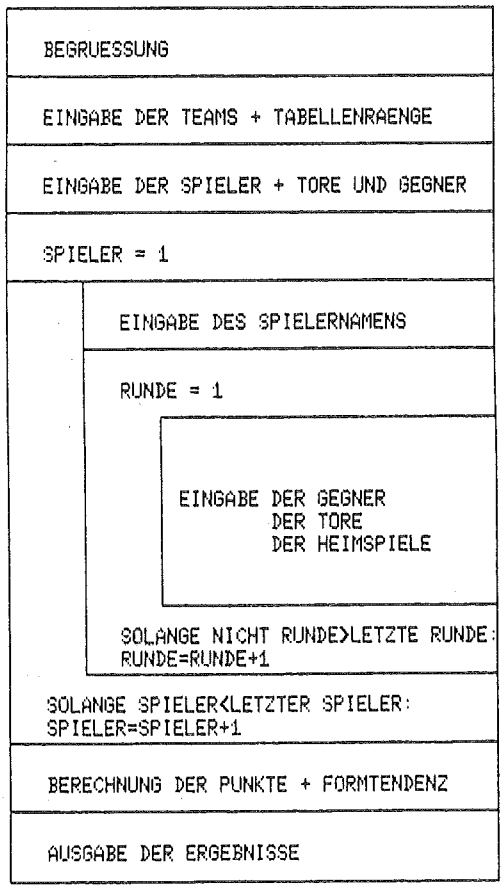


Bild 9.4

Programm Tor

Welche Kriterien brauchen wir für diese Entscheidung?

Hauptsächlich werden wir die geschossenen Tore bewerten, allerdings verwerten wir auch, ob diese Tore bei einem Heimspiel geschossen wurden. Wenn ein Tor bei einem Auswärtsmatch gefallen ist, wird das Tor mit zusätzlichen 10 % honoriert. Weiters wird die Stärke der gegnerischen Mannschaft in die Bewertung miteinbezogen. Eine sehr schwache Mannschaft läßt bekanntlich mehr Tore zu als der Tabellenerste.

Zum Schluß wird der Trend der letzten Spiele bestimmt. Ein Stürmer bekommt einen 20 prozentigen Zuschlag, wenn er in den letzten Spielen mehr Tore schießen konnte als vorher.

Am Anfang des Programms wird die Tabelle nach dem letzten eingegeben Stand präsentiert, sollte sie nicht stimmen, kann man die Änderungen eingeben. Danach wählt man die Anzahl der zu testenden Spieler. Nun gibt man den Namen, die Mannschaft, und die Anzahl der geschossenen Tore ein. Ebenso muß man angeben, ob die Tore bei einem Heimspiel erzielt wurden.

Nachdem alle Spieler mit Daten vermerkt wurden, stellt der Computer seine Berechnungen an und präsentiert das Ergebnis.

Sehen wir uns nun das Struktogramm des Programms an (Bild 9.4).

Wir haben es hier wieder mit einem relativ einfachen Programm zu tun. Nach der schon bekannten Begrüßung werden die Tabellenränge der Mannschaften berichtigt. Danach sorgt der Computer für die Eingabe der einzelnen Daten der zu testenden Stürmer. Dies wird mit einer Schleife erledigt. Für jeden Spieler wird pro Match die Anzahl der Tore eingegeben. Wir können hier zwei ineinandergeschachtelte Schleifen erkennen. Nun beginnt der Computer mit der Berechnung und Auswertung der Daten. Hierzu wird für jeden Stürmer die Stärke anhand eines Punktekontos berechnet. Weiters wird in der gleichen Schleife zum Berechnen der Stärke auch die Formtendenz des Spielers ermittelt. Zum Schluß gibt der Computer seine Weisheiten von sich. Dies geschieht im Programmteil "Ausgabe".

Wie sich das Programm am Bildschirm meldet, zeigen wir noch in Bild 9.5.

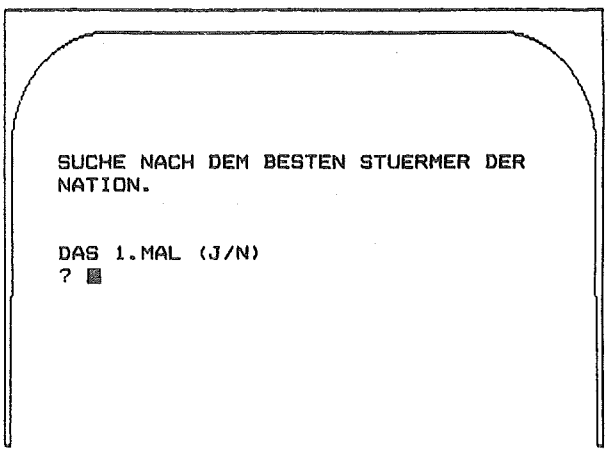


Bild 9.5

```
10 REM *** T O R ***
100 REM
110 CLS:READ MZAHL
120 DIM MANNSCHAFT$(MZAHL),PATZ(MZAHL)
132 SZAHL=10:RZAHL=12:DIM TREG(SZAHL),PU
NKTEG(SZAHL),TALD(SZAHL)
134 DIM E(SZAHL,RZAHL):DIM UNKTE(SZAHL,R
ZAHL):DIM HEIM(SZAHL,RZAHL)
150 FOR I=1 TO MZAHL:READ MANNSCHAFT$(I)
:NEXTI
160 DATA 4,AMANN,BMANN,CMANN,DMANN
200 PRINT "SUCHE NACH DEM BESTEN STUERME
R DER":PRINT"NATION"
210 PRINT:PRINT "DAS 1.MAL (J/N)":INPUT
TASTE$:PRINT:PRINT
220 IF TASTE$="N" OR TASTE$="n" THEN GOS
UB 4000 :GOTO 230 ELSEIF TASTE$="j" OR T
ASTE$="J" THEN230
221 GOTO 210
230 GOSUB 5000:
240 PRINT:PRINT "STIMMEN DIE PLATZZIFFER
N (J/N)":INPUT TASTE$:PRINT:PRINT
250 IF TASTE$="N" OR TASTE$="n" THEN GOS
UB 3000:GOSUB4500:GOTO999 ELSE IF TASTE$
 ="j" OR TASTE$="J"THEN 999
260 GOTO 240
999 REM
1000 PRINT "BITTE JETZT EINGABE"
1010 PRINT "DER SPIELER"
1020 PRINT "IHRER GEGNER"
1030 PRINT "SOWIE IHRER TORE"
1050 PRINT "WIEVIELE SPIELER SOLLEN VERG
LICHEN ":PRINT"WERDEN":INPUT SZAHL
1052 IF SZAHL<1 OR SZAHL>10 THEN 1050
1055 PRINT"WIEVIELE RUNDEN SOLLEN VERGLI
CHEN ":PRINT"WERDEN":INPUT RZAHL
1060 IF RZAHL <1 OR SZAHL> 10 THEN 1055
1080 FOR SPIELER=1 TO SZAHL
1090 PRINT:PRINT "NAME DES"SPIELER". SPI
ELERS:";:INPUT SPIELER$(SPIELER)
1091 IF SPIELER$(SPIELER)="" THEN1090
1110 PRINT:PRINT " IN WELCHER MANNSCHAFT
 SPIELT ";SPIELER$(SPIELER);" ?"
1115 GOSUB 5500
1120 INPUT EIGENMANN(SPIELER)
1130 FOR NDE=1TO RZAHL
1135 PRINT
1170 PRINT "GEGNER :";NDE;" RUNDE:"
1175 GOSUB 5500
1180 INPUT GEGNER(SPIELER,NDE):IF GEGNER
(SPIELER,NDE)<1 ORGEGNER(SPIELER,NDE)>MZ
AHLTHEN 1180
1181 IF GEGNER(SPIELER,NDE)=EIGENMANN(SP
IELER) THEN PRINT SPIELER$(SPIELER);" KA
NN DOCH NICHT GEGEN SICH SELBST SPIELEN
! ":GOTO 1170
1182 PL(SPIELER,NDE)=PATZ(EIGENMANN(SPIE
LER))-PATZ(GEGNER(SPIELER,NDE))
1183 RLPLATZ=PL(SPIELER,NDE)/MZAHL
1184 REM
1185 IF RLPLATZ<0 THEN RLPLATZ=1/(1+ABS(
RLPLATZ)) ELSE RLPLATZ=1+RLPLATZ
1190 PRINT"WIEVIELE TORE HAT ";SPIELER$(
SPIELER);" GEGEN ";MANNSCHAFT$(GEGNER(SP
IELER,NDE));:PRINT" GESCHOSSEN ";
1220 INPUT E(SPIELER,NDE)
1225 TREG(SPIELER)=TREG(SPIELER)+E(SPIEL
ER,  NDE)
1226 UNKTE(SPIELER,  NDE)= E(SPIELER,  N
DE)*RLPLATZ
1230 PRINT "HEIMSPIEL.......0"
1240 PRINT "AUSWAERTSSPIEL.1"
1245 INPUT HEIM(SPIELER,  NDE)
1250 IF HEIM(SPIELER,  NDE)<0 OR HEIM(SP
IELER,  NDE)>1 THEN PRINT "NUR 0 ODER 1
ALS EINGABE MOEGLICH!":GOTO 1245
1260 IF HEIM(SPIELER,  NDE)=1 THEN UNKTE
(SPIELER,  NDE)=UNKTE(SPIELER,  NDE)*1.1
1270 REM
1275  PUNKTEG(SPIELER)= UNKTE*(SPIELER)+
 UNKTE(SPIELER,  NDE)
1280 NEXT   NDE
1290 REM
1500 IFRZAHL=1THEN TALD(SPIELER)=0:
     GOTO 1610
1510 HALBZEIT=INT(RZAHL/2)
1512 REM
1515 RE  =0
1520 FOR NDE=1 TO HALBZEIT
1530 RE =RE + UNKTE(SPIELER,  NDE)
1540 NEXT NDE
1545 RE = RE /HALBZEIT
1547 REM
1550 TE =0
1560 FOR NDE=HALBZEIT+1 TO RZAHL
1570 TE=TE + UNKTE(SPIELER,  NDE)
1580 NEXT NDE
1581 TE =TE /(RZAHL-HALBZEIT)
1583 REM
1585 IF TE - RE <.1 THEN TALD (SPIELER)=
0
1590 IF TE > RE THEN TA (SPIELER)=1 ELSE
 TA (SPIELER)=-1
1610 IF TA (SPIELER)=1 THEN PUNKTEG(SPIE
LER)=PUNKTEG(SPIELER)*1.2
1640 NEXT SPIELER
2000 REM AUSGABE
2005 PRINT "PUNKTEVERGLEICH"
2010 FOR SPIELER=1TO SZAHL
2100 PRINT
2200 PRINT SPIELER$(SPIELER);" VON ";MAN
NSCHAFT$(EIGENMANN(SPIELER))" HAT IN DEN
 LETZTEN "RZAHL"RUNDEN"TREG(SPIELER)" TO
RE GESCHOSSEN UND DABEI "PUNKTEG(SPIELER
)"PUNKTE BEI ";
2210  IF TALD (SPIELER)=0 THEN PRINT "GL
EICHBLEIBENDER"; ELSE IF TALD  (SPIELER)
=1 THEN PRINT"STEIGENDER";ELSE PRINT "FA
LLENDER";
2220 PRINT " FORM ERREICHT.":PRINT
2250 PRINT "RUNDE/GEGNER/PLDIFF./HEIM-AU
S/TORE/PKTE"
2260 FOR NDE=1TO RZAHL
2270  PRINT NDE "/"MANNSCHAFT$(GEGNER(SP
IELER,NDE))"/";PL(SPIELER,NDE)"/";
2280   IF HEIM(SPIELER,NDE)=0 THEN PRINT
 "H";ELSE PRINT"A";
2290 PRINT"/"E(SPIELER,NDE)"/"UNKTE(SPIE
LER,NDE)
2300 NEXT NDE
2900 NEXT SPIELER
2990 END
3000 REM
3030 PRINT:PRINT "EINGABE DES TABELLENPL
ATZES JEDER MANNSCHAFT"
3035 PRINT
3040 FOR I=1TO MZAHL
3050 PRINT"PLATZ VON ";MANNSCHAFT$(I);:I
NPUT ZAHL
3060 IF ZAHL<>0 THEN PATZ(I)=ZAHL
3070 NEXT I
3075 PRINT
3080 RETURN
4000 REM
4030 OPEN "1:TORPLATZ"FOR INPUT AS#1
4035 MZAHL=0
4040 IF EOF(1) THEN 4070
4050 MZAHL=MZAHL+1
4060 INPUT #1,MANNSCHAFT$(MZAHL)
4065 INPUT #1,PATZ(MZAHL):GOTO4040
4070 CLOSE 1
4080 RETURN
4500 REM
4530 OPEN"1:TORPLATZ"FOR OUTPUT AS #1
4540 FOR I=1 TO MZAHL
4550 PRINT#1,MANNSCHAFT$(I);PATZ(I)
4560 NEXT
4565 CLOSE#1
4570 RETURN
5000 REM
5040 PRINT "MANNSCHAFT","TABELLENRANG"
5050 PRINT "------------------------"
5060 FOR I=1 TO MZAHL
5070 PRINT MANNSCHAFT$(I),PATZ(I)
5080 NEXT I
5090 RETURN
5500 REM
5540 PRINT:PRINT"NUMMER","MANNSCHAFT"
5550 PRINT"------------------------"
5560 FORI=1 TO MZAHL
5570 PRINT I,MANNSCHAFT$(I)
5580 NEXT I
9000 FOR I=1 TO BILDH:PRINT:NEXT I:RETUR
N
10000 DATA 0
```

Die Funktion von Datenbanken haben wir schon in Folge 8 kennengelernt. Nun will auch Mutter Dischek ihre Rezeptbücher mit DataStar computerisieren. Grund genug für uns, ihr über die Schulter zu schauen.

Dabei widmen wir uns aber auch noch einmal genauer dem Programm DataStar und seinem Anhängsel FormGen.

Vorher jedoch sehen wir uns einmal unsere Anschlüsse beim Spectravideo an.

Spectravideo Peripherie

Jeder Computer braucht Verbindungen zur Außenwelt. Diese werden ihm durch Interfaces (Schnittstellen) ermöglicht. Auch unser Computer hat einige davon. Für den Benützer werden solche Verbindungen aber lediglich am Computergehäuse durch die einzelnen Anschlüsse ersichtlich. Schauen wir sie uns näher an:

Am wichtigsten ist der Anschluß zu einem Sichtgerät, also zum Fernseher oder Monitor. Je nachdem, ob Fernseher oder nicht, wird ein Modulator in den Weg des Bildschirmsignals geschalten. Nach diesem Modulator hat das Signal den Charakter eines Antennensignals. Unser Fernseher wird überlistet, er glaubt, von der Antenne Zeichen zu bekommen. Sie werden jedoch vom Modulator geschickt.

Beim Spectravideo-Computer ist der Modulator eingebaut und es gibt zwei Ausgänge, einen für TV-Anschluß und einen für Monitoranschluß.

Gleich daneben ist der Kassettenrecorder beheimatet. Mit der siebenpoligen Steckerleiste werden alle Signale zur Steuerung und Datenübertragung verschickt oder eingelesen.

Wenn wir nun auf die rechte Seite des Gerätes wandern, erkennen wir neben dem Netzstecker und dem Ein-/Ausschalter auch die beiden Joystick-Anschlüsse.

Mit den 9 Steckerkontakten werden wieder alle nötigen Signale samt Stromversorgung an den Joystick gebracht. Man kann neben Joysticks übrigens auch Paddles und das SVI-Graphik-Tablett an diesen Anschlüssen an den Computer hängen.

Unsere Blicke werden nun auf die Oberseite des Gerätes zum kleinen Modulschacht gelenkt. Hier finden bei passionierten Videospielern die Spectravideo-Module(Cartridges) ihren Platz. Wenn so ein Modul im SVI-Computer steckt, dann wird es übrigens ähnlich dem RAM und dem ROM als Speicherbank angesehen. Module sind immer Programme in ROM-Form.

Aus diesem Grund werden hier auch schon mehr Signale ausgegeben, als bei den vorherigen Anschlüssen. Für den Computer sind eingesteckte Module Teile seiner selbst. Deshalb dürfen sie auch auf die internen Busse zugreifen (Datenbus, Adreßbus und Kontrollbus). Ebenso werden noch verschiedene weitere Kontrollsignale für Module übermittelt.

Ganz umfangreich wird die Pinbelegung beim Erweiterungsport (Expanderport). Hier finden sage und schreibe 50 Pins Platz zum Datenübertragen.

Ein Wort zur Funktion des Erweiterungsports:

Wir brauchen es, um einen der Expander anzuschließen. In den Expander werden andererseits wieder sämtliche Erweiterungen gesteckt.

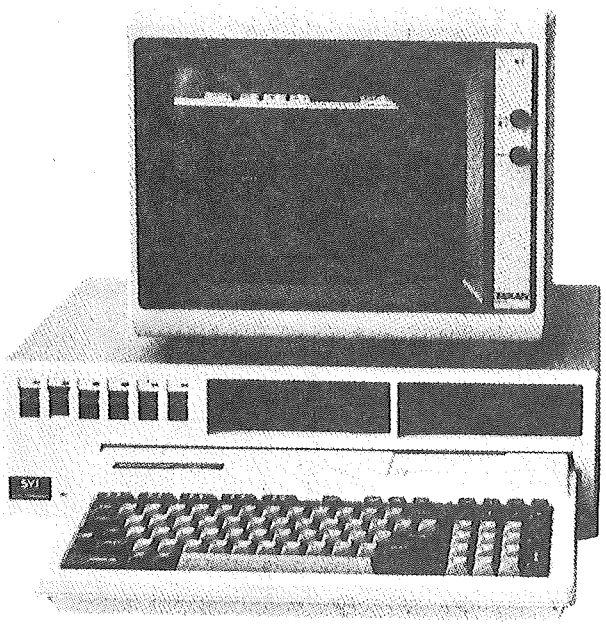


Bild 10.1

So wälzen sich über diesen Port nicht nur das gesamte Bussystem, sondern auch die verschiedensten Signale für diverse Adapter. So erzeugt Spectravideo einen COLECO-Spiele-Adapter, der einige Signale mit dem SVI-Prozessor austauscht. Daß auch die Spannungsversorgung über diesen Port weitergereicht wird, dürfte klar sein (die Expander haben aber ein eigenes Netzteil).

Im Expander selber wird dann der Erweiterungsport vervielfältigt. So kann man gleichzeitig den Floppy-Disk-Controller, die Drucker-Karte und viele andere anschließen.

Welche Erweiterungen stehen uns nun zur Verfügung?

Zum Einen kennen wir den Floppy-Controller. Er verarbeitet die Signale von zwei Diskettenlaufwerken. Der Controller muß immer zwischen der Floppy und dem Computer hängen,

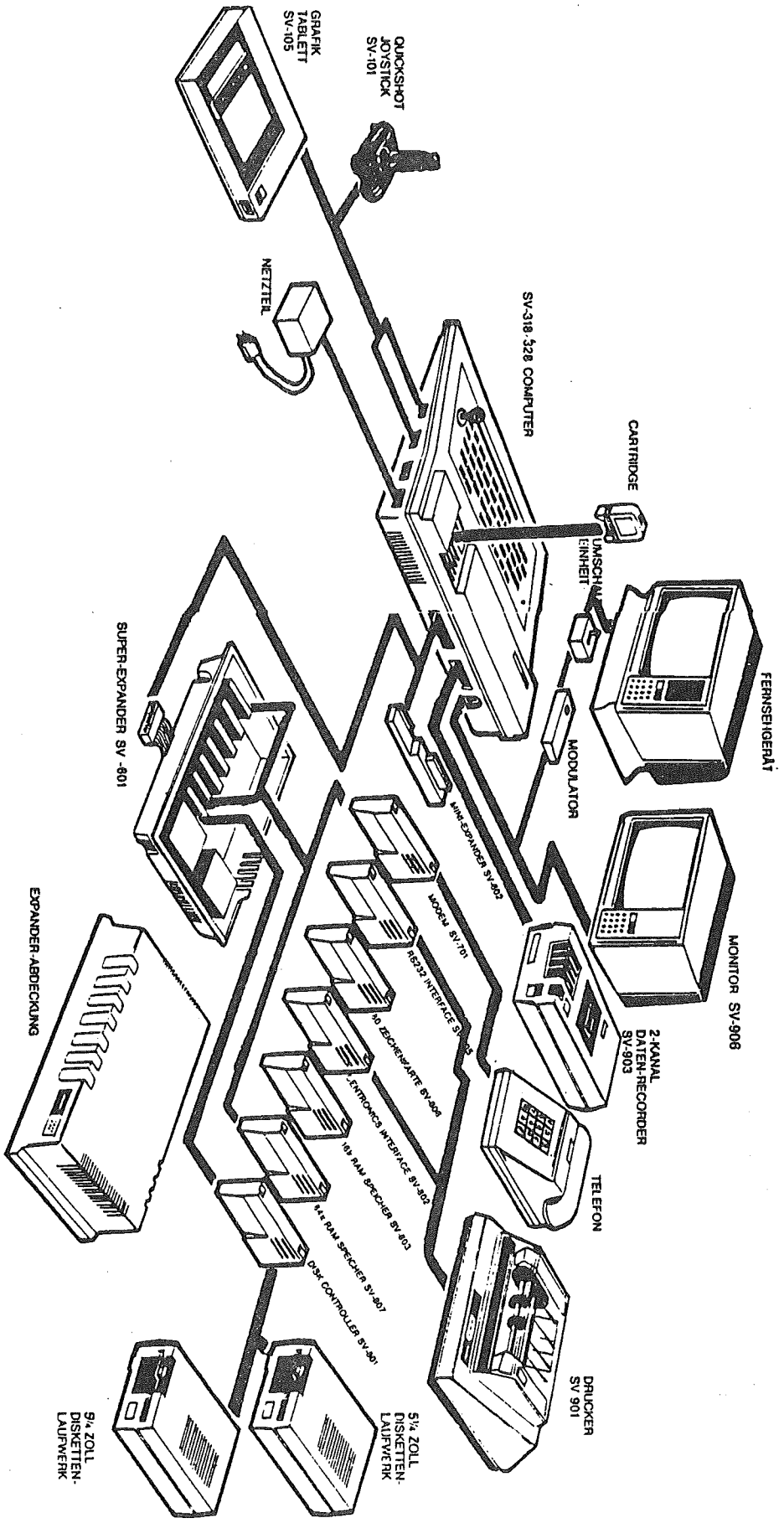

GRAFIK TABLETT SV-105
QUICKSHOT JOYSTICK SV-101
NETZTEIL
SV-318 328 COMPUTER
CARTRIDGE
UMSCHALT EINHEIT
FERNSEHGERÄT
MODULATOR
MONITOR SV-906
SUPER-EXPANDER SV-601
MINI EXPANDER SV-402
MODEM SV-701
RS232 INTERFACE SV-504
80 ZEICHENKARTE SV-908
CENTRONICS INTERFACE SV-302
16k RAM SPEICHER SV-403
64k RAM SPEICHER SV-807
DISK CONTROLLER SV-801
2-KANAL DATEN-RECORDER SV-903
TELEFON
DRUCKER SV 901
EXPANDER-ABDECKUNG
5¼ ZOLL DISKETTEN-LAUFWERK
5¼ ZOLL DISKETTEN-LAUFWERK

Bild 10.2

weil er die gesamte Steuerung der Daten übernimmt (in Zusammenarbeit mit dem Prozessor).

Weiters gibt es die Drucker-Karte. Sie formt die Signale so um, daß ein Centronics-Interface daraus wird. Dieses Interface ist nach einer bestimmten Norm entwickelt, nach der fast jeder Drucker angeschlossen werden kann.

Als nächstes gibt es die 64 KByte-Speichererweiterung. Sie vervollständigt die Banken im Computer, sodaß mehr Speicher zur Verfügung steht. Mittels DIP-Schaltern kann man jeweils 32 KByte-Bereiche schalten.

Unter einer Bank versteht man beim Spectravideo 64 KBytes, welche auf einmal angesprochen werden können. Eine Darstellung der Speicherbelegung finden Sie in Bild 10.3.

Zuletzt wollen wir uns noch kurz der 80-Zeichenkarte widmen. Von Haus aus kann der SVI ja keine 80 Zeichen am Bildschirm darstellen. Deshalb gibt es einen eigenen Prozessor mit einem eigenen Speicher, der die Regelung für 80 Zeichen übernimmt. Das ist das wesentliche Prinzip dieser Karte. Nun darf man natürlich das Bildschirmsignal nicht am normalen Anschluß abnehmen, sondern man muß den Anschluß der 80 Zeichenkarte verwenden. Diese 80 Zeichen-Karte ist auch mit deutschem Zeichensatz (Umlaute) erhältlich.

Bild 10.2 gibt einen Überblick über die umfangreichen Erweiterungsmöglichkeiten der Spectravideo-Computer. Neben dem in der Abbildung dargestellten Expander SVI-601 gibt es auch die Serie der Super-Expander SVI-605 mit eingebauten Laufwerken (wie im Foto Bild 10.1 abgebildet).

DataStar

Kommen wir nun wieder zu unserer Dateiverwaltung zurück.

Wir hatten schon einmal mit unserem Software-Tool DataStar gearbeitet. Da erkannten wir die grundsätzliche Funktion dieses Programms. Nun wollen wir etwas genauer auf unser Werkzeug eingehen.

Zuerst stellt sich uns die Wahl, ob wir neue Dateien erstellen oder bestehende verändern wollen.

Wenn wir uns für erstere Funktion entscheiden, müssen wir zuerst die Struktur der Datei erstellen.

Nun sehen wir uns einmal die Erstellung der neuen Datei an.

Wir rufen das Programm FormGen auf. Zuerst geben wir den Namen der neuen Datei ein, danach wird ein leeres Bildschirmblatt präsentiert. Nun können wir die Maske für unsere Datei erstellen.

Jetzt müssen wir die Felder definieren, die wir später einmal beschreiben wollen. Dazu geben wir bei jedem Feld den Namen ein, und die Art. Ein Feld wird durch Unterstreichung dargestellt:

Das Zeichen "_" ist daher in den Kommentaren und Texten auf dem Maskenformular nicht zulässig.

Wenn wir den Cursor in ein so markiertes Feld stellen, erhalten wir am oberen Rand

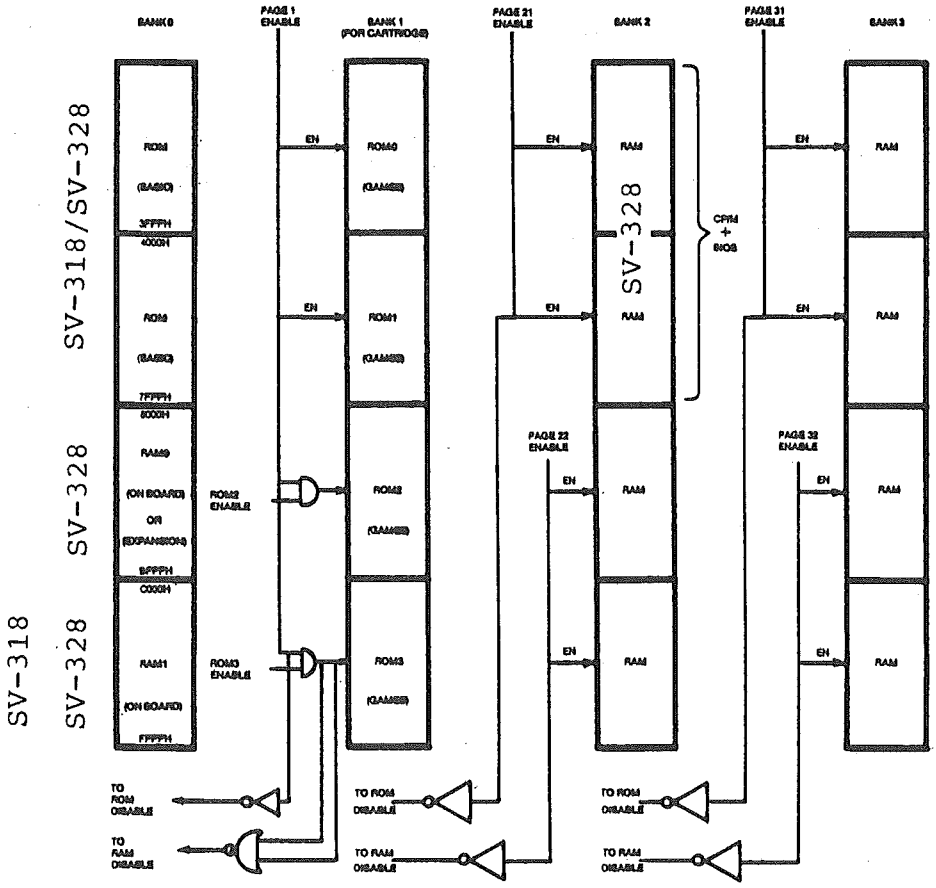


Bild 10.3

des Bildschirms verschiedene Informationen über dieses Feld (Länge, Cursorposition im Feld). Jetzt können wir weitere Angaben zum Feld machen, zum Beispiel ob das Feld unbedingt ausgefüllt werden muß oder nicht, ob Ziffern oder Buchstaben oder beides zulässig sind. Wir werden dabei durch ein Menü von Frage zu Frage geführt. Das Programm nummeriert natürlich auch die einzelnen Felder.

DataStar überprüft selbstverständlich die Eingaben auf ihre Richtigkeit. So kann es nie vorkommen, daß ein formaler Fehler später die Bearbeitung eines Files behindert.

Wir müssen nun noch das Arbeiten mit bereits bestehenden Dateien erklären.

Wir rufen DataStar auf und geben den Namen der Datei an. Wir brauchen keine Strukturen mehr bestimmen und können gleich mit dem Ändern von Daten beginnen.

Doch nicht nur diese Funktionen bietet uns das Programm. Wir dürfen auch neue Datensätze erstellen.

In dieser Funktion bekommen wir die mit FormGen definierte Maske auf den Bildschirm. Nun dürfen wir einen Datensatz ausfüllen. Nach Abschluß können wir das nächste Record bearbeiten.

Weiters ist es auch möglich, schon bestehende Records zu suchen.

Wir haben für diese Funktion einige Hilfen. Von Folge 8 her wissen wir ja, daß wir Indices für die Suche von Datensätzen bestimmen können. Ebenso kann man nach den Nummern der Records suchen. Die letzte Variante wird mit Feldern durchgeführt. Wir können den Inhalt von Feldern zur Suche verwenden, ebenso nur Teile davon. Der Computer listet danach automatisch alle Records auf, die in den entsprechenden Feldern den gesuchten Begriff enthalten.

Es kann natürlich auch einmal passieren, daß man Teile einer Datei wegwerfen möchte. Dies geht mit DataStar ohne weiteres. Das Löschen von Datensätzen ist ebenso einfach, wie das Löschen von ganzen Dateien.

Es kann sehr nützlich sein, nur die Eintragungen einer Datei zu vernichten, die Struktur jedoch weiter zu erhalten. Dies geht ebenso, wie das vollständige Löschen der Dateien samt Datenstruktur. Dabei verschwindet auch der Name im Inhaltsverzeichnis der Diskette.

Das pure Gegenteil zum Löschen kann auch bewerkstelligt werden, das Kopieren auf andere Disketten.

Beim Arbeiten mit Disketten soll man es sich zur Gewohnheit machen, Sicherheitskopien anzulegen. Wenn die Arbeitsdiskette einmal unter widrigen Umständen "ex" geht, hat man immer noch Kopien zur Verfügung.

Doch nicht nur zur Sicherheit werden Kopien angelegt. Ein simples Vervielfältigen, um auch anderen Bedienern die Daten zugänglich zu machen, kann auch schon der Grund zur Kopie sein. Daher ist die Möglichkeit zum Kopieren einer Datei vorhanden.

Neben diesen Funktionen hat DataStar noch viele andere auf Lager, wie zum Beispiel das Sortieren von Dateien, das Erstellen von Listen und so weiter.

Letzteres erledigt das Software-Tool ReportStar. Mit dem Programmteil REDIT kann man das Ausgabeformat der gewünschten Listen erstellen. Aber nicht nur die gewünschte Form des Ausdrucks kann man so festlegen, man kann auch während des Ausdrucks erst Werte errechnen lassen und diese ausgeben. Nicht zuletzt kann man solche Listen vor dem endgültigen Druck noch mit WordStar editieren und damit weiter verschönern.

Das Rezeptbuch

Kommen wir nun wieder zur Familie Dischek und ihrem "Rezeptbuch" zurück. Wir werden jetzt die Daten aufzählen, die unsere Datei bestimmen sollen:

Ein Record soll 17 Felder umfassen: Dabei gelten für die einzelnen Felder folgende Bestimmungen:

Feld 1: Hier wird mit 25 Zeichen der Name der Speise eingetragen.

Feld 2: Kategorie (z.B. Vorspeise).

Feld 3: Hier sind 15 Zeichen für die Zutaten der Speise reserviert.

Feld 4 - 10: wie Feld 3.

Feld 11 - 16: Zubereitung.

Feld 17: Das letzte Feld beinhaltet das Datum, wann das Gericht zuletzt gekocht wurde.

Mit dieser Einteilung haben wir den ersten Schritt schon geschafft. Wir rufen nun FormGen auf und erstellen eine Maske für unsere Felder. Gleichzeitig geben wir alle Definitionen ein. Wenn wir mit der Einteilung zufrieden sind, speichern wir die Maske ab. Nun brauchen wir lediglich ein Rezeptbuch nach dem anderen in den Computer zu übertragen.

Es ist nun die Aufgabe unserer Hausfrau Mutter Dischek, das Rezeptbuch so umfangreich wie möglich gestalten.

Ebenso ist der interessierte Leser eingeladen, seine eigene Rezeptdatei zu erzeugen. Es würde aber den Rahmen dieser Folge sprengen, wenn wir nun anfangen würden, über die besten Vor- oder Nachspeisen zu berichten.

Nach so vielen Folgen, welche die Anwendungen des Computers in professionellen Gebieten gezeigt haben, wollen wir uns diesmal wieder den Gebrauch als Spielcomputer im trauten Heim ansehen. Dadurch kann nämlich unser Spectravideo eine Party interessanter gestalten. Viele Leute reizt die Spannung bei Computerspielen. Um unseren Computer im Mittelpunkt stehen zu lassen, werden wir ein "Hangman"-Programm erstellen.

Bevor wir dies jedoch tun, widmen wir uns den Stringmanipulationen (ein schreckliches Wort!) und den umfangreichen Graphikmöglichkeiten.

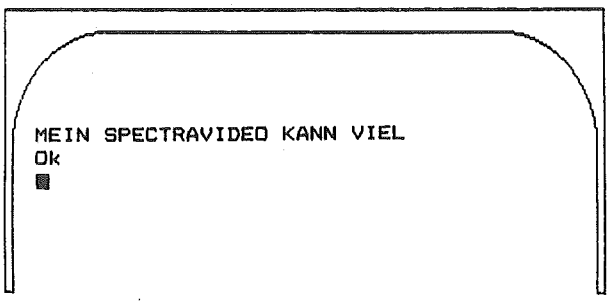
Strings=Zeichenketten

Sehen wir uns zuerst die verschiedenen Methoden an, mit denen wir unsere Zeichenketten "verstümmeln" können.

Die einfachste aller Manipulationen ist die Verkettung, von der wir schon kurz gesprochen haben (in Folge 1). Dabei werden zwei alphanumerische Variablen durch ein Pluszeichen verbunden. Der Effekt ist einfach zu verstehen, die zweite Variable wird einfach an die erste angehängt. Ein Beispiel wartet natürlich auch schon auf uns:

```
10 A$="MEIN "
20 B$="SPECTRAVIDEO "
30 C$="KANN VIEL!"
40 D$=A$+B$
50 C$=D$+C$
60 PRINT C$
70 END
```

Nach RUN erkennen wir:



Dieses geistreiche Beispiel hat hoffentlich die Funktionsweise von Verkettungen klar gemacht. Viel interessanter ist die Möglichkeit, einen Teil eines Strings zu eliminieren, oder ihn sogar auszuwechseln. Dafür hat der Spectravideo ebenso seine Anweisungen.

LEFT$, RIGHT$, MID$

Nun sehen wir die Funktionsweise deutlich: Bei den beiden Anweisungen wird nach dem Befehlswort zuerst der String geschrieben, von dem der Teil genommen werden soll und danach erst, wie viele Buchstaben. Bei LEFT$ wird von links nach rechts gezählt, bei RIGHT$ logischerweise umgekehrt. In unserem vorigen Beispiel haben wir übrigens wieder Verkettungen gezeigt. Man sieht, daß diese Art der Manipulation eine der wichtigsten ist.

Hart im Konkurrenzkampf an Bedeutung liegt sie noch mit MID$, der vierten wichtigen Funktion. Mit ihr werden nämlich gleich zwei Fliegen mit einer Klappe geschlagen, beziehungsweise zwei verschiedene Arbeiten mit einem Befehlswort durchgeführt. Je nachdem, ob man es als Befehl oder als Funktion verwendet, ergeben sich Unterschiede. Wir sehen uns zuerst die Funktion MID$ an.

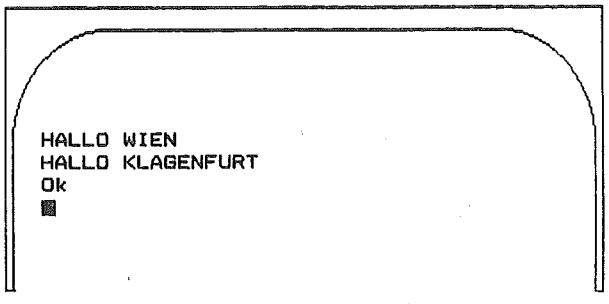
Man kann nicht nur von Rechts und von links weg Teile aus einer Zeichenkette kopieren, sondern auch einen Mittelteil. Dabei ergibt sich wie von selbst auch gleich das Aussehen dieser Anweisung:

MID$(Variable,Anfangswert,Zahl der Zeichen)

Wenn wir den Teil von rechts eliminieren wollen, nehmen wir RIGHT$, ist die Trennung des linken Teils gewünscht, verwenden wir LEFT$. "right" heißt auf englisch "rechts" und "left" "links". Das Dollarzeichen ist das Kennzeichen, daß mit Strings manipuliert wird. So einfach sind die beiden Funktionen zu merken. Mit Beispielen lassen sich solche Anweisungen meistens am besten erklären, deshalb soll wieder unser Computer zeigen, wie LEFT$ und RIGHT$ arbeiten:

```
10 A$="HALLO "
20 B$="WIENKLAGENFURT"
30 C$=A$+LEFT$(B$,4)
40 D$=A$+RIGHT$(B$,10)
50 PRINT C$:PRINT D$
```

Nach dem dezenten RUN ist folgendes ersichtlich:



Die verwendete Variable ist uns bekannt, sie gibt immer an, von welchem String kopiert werden soll. Auch die "Anzahl der Zeichen" kennen wir schon von LEFT$ und RIGHT$ her. Bleibt nur noch die "Anfangszahl". Es ist nicht schwer zu erraten, welche Aufgabe dieser Wert hat. Er markiert das erste Zeichen im String, welches kopiert werden soll. Ab dann werden "Anzahl" Zeichen mit herausgenommen. Ein Beispiel steht uns auch wieder zur Verfügung:

```
10 A$="HALLO "
20 B$="FRITZFRANZFREDDY"
30 C$=A$+MID$(B$,6,5)
40 PRINT C$
50 END
```

RUN tippen:

```
HALLO FRANZ
Ok
```

Bis jetzt wurden durch die Anweisungen die ursprünglichen Zeichenketten nicht ausdrücklich verändert. Nun ändert sich dieser Sachverhalt aber. Die zweite Anweisung MID$ ersetzt nämlich einen bestimmten Teil eines Strings durch einen anderen. Das Format sieht fast gleich mit unserer ersten Anweisung aus:

MID$(Variable,Anfangswert,Zahl der Zeichen) =Ausdruck

Der in der Klammer angegebene Teil der Zeichenkette "Variable" wird durch Ausdruck ersetzt. Beispiele haben wir auch wieder auf Lager:

Wir ändern das vorige Programm geringfügig:

```
10 A$="HALLO"
20 B$=FRITZFRANZFREDDY"
30 MID$(B$,6,5)=A$
40 PRINT B$
50 END
```

RUN:

```
FRITZHALLOFREDDY
Ok
```

Doch damit nicht genug, der SV mit seinem MSX-BASIC kennt weitere Anweisungen zur Zeichenkettenveränderung:

Man kann mit LEN die Länge eines Strings bestimmen:

```
10 INPUT A$
20 B=LEN(A$)
30 PRINT"DIE LAENGE VON "A$" IST"B
40 GOTO 10
```

RUN:

```
? EMIL
DIE LAENGE VON EMIL IST 4
?
```

Mit diesem Programm können wir beliebig lange Texte eingeben, der Computer wird uns immer die richtige Länge sagen.

Die letzte Funktion, die wir untersuchen wollen, heißt INSTR. Mit ihr können wir einen bestimmten Ausdruck in einer Zeichenkette suchen.

```
10 A$="FRITZFRANZFREDDY"
20 B=INSTR(A$,"FREDDY")
30 PRINT "FREDDY KOMMT IN A$ AN DER STELLE"
B"VOR"
40 END
```

RUN:

```
FREDDY KOMMT IN A$ AN DER STELLE 11 VOR
Ok
```

Wir können leider nicht sämtliche Sonderfälle der Anweisungen aufzählen und sagen, wie sie sich dabei verhalten. Es ist nun die Aufgabe des interessierten Lesers, sich mit den Demonstrationsprogrammen und dem Bedienerhandbuch vor den Computer zu setzen und zu versuchen, die Ausnahmen selber zu erzeugen. Der Spectravideo zeigt dann seine Reaktionen.

Wir wollen uns lieber in ein anderes Gebiet stürzen, die Graphikfähigkeiten der SVI-Computer.

Spectravideo-Graphik

Neben dem Textmodus, in dem wir bis jetzt gearbeitet haben, gibt es auch noch zwei Graphik-Modi. Die hochauflösende Graphik kann 256*192 Points, die niederauflösende Graphik 64*48 Punkte adressieren. Die Koordinaten werden aber in beiden Fällen gleich bestimmt. Der einzige Unterschied besteht darin, daß im niederauflösenden Modus die Punkte größer sind. Wir werden uns vor allem im hochauflösenden Modus aufhalten.

Einschalten kann man die verschiedenen Bildschirme mit SCREEN:

SCREEN 0: Textmodus
SCREEN 1: hochauflösende Graphik
SCREEN 2: niederauflösende Graphik

Die Koordinaten werden wie in Bild 11.1 ersichtlich in X- und Y-Teile zerlegt. X kennzeichnet die waagrechte Position (Wert steigt von links nach rechts an), Y die senkrechte (Wert steigt von oben nach unten an). Durch die Angabe von einem X- und einem Y-Wert läßt sich jeder beliebige Bildschirmpunkt innerhalb des Koordinatenfeldes adressieren. Es gibt allerdings einen Rand am Schirm, der nicht erreicht werden kann, man darf ihn nur einfärben, ansonsten gibt es keine Möglichkeit, dort Graphik anzusprechen.

Farben hat der SVI daher natürlich auch zu bieten. Mit dem Befehl COLOR lassen sich sechzehn verschiedene Farben generieren. Dabei wird hinter das Befehlswort ein be-

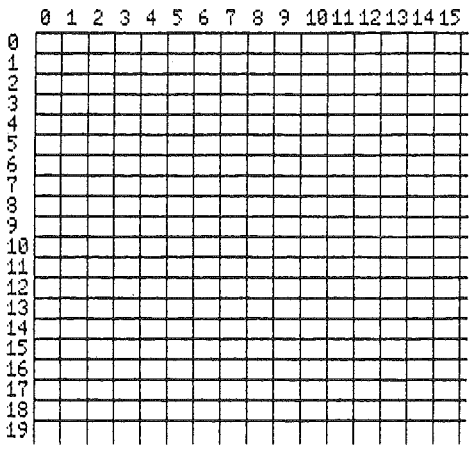


Bild 11.1

stimmter Farbcode gesetzt. Der erste Code gibt die Vordergrundfarbe an, der zweite die Hintergrundfarbe und der dritte sorgt für die Färbung des Randes, der nicht angesteuert werden kann. Die sechzehn Farbcodes werden nun hier angegeben:

0 transparent	8 mittelrot
1 schwarz	9 hellrot
2 mittelgrün	10 dunkelgelb
3 hellgrün	11 hellgelb
4 dunkelblau	12 dunkelgrün
5 hellblau	13 magenta
6 dunkelrot	14 grau
7 cyan	15 weiß

Das Format von COLOR:

```
COLOR Vordergrundfarbe,Hintergrundf.,Randf.
```

Kommen wir nun zu den einzelnen Graphikbefehlen:

Die beiden einfachsten Anweisungen sind PSET und PRESET. Mit der ersten wird ein Punkt am Bildschirm gesetzt, mit der zweiten ein Punkt gelöscht.

```
PSET (X,Y)
PRESET (X,Y)
```

LINE: mit dieser Anweisung können wir Linien zeichnen. Dazu geben wir die Koordinaten zweier Punkte an, die dann verbunden werden. Setzt man hinter die Anweisung ein B, dann wird die Linie als Diagonale eines Rechtecks angesehen, und das Rechteck ausgebildet. Kommt hinter das B noch ein F, wird das so gebildete Rechteck in der angegebenen Farbe ausgemalt.

```
LINE (X,Y)-(X1-Y1)
LINE (X,Y)-(X1,Y1),,B
LINE (X,Y)-(X1,Y1),,BF
```

CIRCLE: mit diesem Befehl wird es möglich, Kreise, Ellipsen oder nur Segmente dieser Figuren zu zeichnen. Dazu geben wir einerseits einen Punkt ein, der den Mittelpunkt kennzeichnet. Danach folgen Radius, Farbe, Anfangs- und Endpunkte des Kreissektors und das Verzerrungsverhältnis der Ellipsenteile:

```
CIRCLE(X,Y),Radius: zeichnet normalen Kreis
CIRCLE(X,Y),Radius,,3.14,6.28: Halbkreis
CIRCLE(X,Y),Radius,,,,4: Ellipse
```

Der Anfangs- und der Endpunkt des Kreissektors wird im Bogenmaß angegeben. Bevor wir viele Worte über die mathematischen Grundlagen verlieren, zeigen wir die Werte anhand einer Skizze. In Bild 11.2 ist ein Kreis ersichtlich, welcher die Werte bestimmter Punkte angibt. Bogenmaß wird normalerweise immer im Verhältnis zur Zahl PI angegeben, doch hat unser Computer diese Zahl nicht ausdrücklich. Wir müssen uns daher mit 3.1415 behelfen.

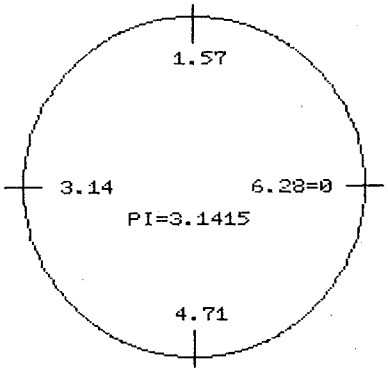


Bild 11.2

Weiters gibt es noch eine Unmenge von Anweisungen, die sehr komfortables Arbeiten möglich machen: DRAW, GET, PUT und viele andere. Wir gehen jedoch nicht näher darauf ein, sondern widmen uns einem weiteren Phänomen in der Computerwelt, dem SPRITE.

Sprites

Das Wesen dieser SPRITES erscheint kompliziert, weil die Erklärung recht abstrakt ist, aber man sieht ziemlich bald, daß diese Einrichtung für "eingefleischte" Videospiel-Programmierer unersetzbar ist.

8*8-SPRITES

FIGUR DES SPRITES	BITMUSTER	ASCII ZAHL	ZEICHEN
1 1	01000001	65	A
1 1	01000010	66	B
1 1	01000100	68	D
1 1	01001000	72	H
1 1	01001000	72	H
1 1	01000100	68	D
1 1	01000010	66	B
1 1	01000001	65	A

SPRITE WIRD MIT ABDHHDBA DEFINIERT

Bild 11.3

16*16-SPRITES

FORM DES SPRITES	BITMUSTER		ASCII-CODES		ZEICHEN	
1 1 1 1	01000001	01000001	65	65	A	A
1 1 1 1	01000010	01000010	66	66	B	B
1 1 1 1	01000100	01000100	68	68	D	D
1 1 1 1	01001000	01001000	72	72	H	H
1 1 1 1	01001000	01001000	72	72	H	H
1 1 1 1	01000100	01000100	68	68	D	D
1 1 1 1	01000010	01000010	66	66	B	B
1 1 1 1	01000001	01000001	65	65	A	A
1 1 1 1	01000010	01000001	66	65	B	A
1 1 1 1	01000100	01000010	68	66	D	B
1 1 1 1	01001000	01000100	72	68	H	D
1 1 1 1	01010000	01001000	80	72	P	H
1 1 1 1	01010000	01001000	80	72	P	H
1 1 1 1	01001000	01000100	72	68	H	D
1 1 1 1	01000100	01000010	68	66	D	B
1 1 1 1	01000010	01000001	66	65	B	A

Bild 11.4

DAS SPRITE WIRD NUN MIT ABDHHDBABDHPPHDBABDHHDBAABDHHDBA DEFINIERT

SPRITES kann man sich als Objekte vorstellen, welche am Bildschirm in eigenen Ebenen frei beweglich sind.

Wir haben nur die Aufgabe, die Figur des SPRITES zu definieren, und es in einer Ebene auf den Bildschirm zu setzen. Duch Änderung von zwei Koordinaten (X und Y) läßt sich die ganze Figur am Schirm bewegen.

Sehen wir uns nun die Eigenheiten dieser SPRITES an. Beim Spectravideo darf man diese Objekte entweder in einer 8*8- oder in einer 16*16-Matrix definieren. Die Figur wird durch einen bestimmten String definiert. Jedes Zeichen dieses Strings kann man ja in den ASCII-Code umwandeln. Dieser Code läßt sich binär darstellen. Diese duale Darstellung zeigt dem Bediener dann die spätere Form des SPRITES, 1 heißt "Punkt leuchtet", 0 "Punkt leuchtet nicht".

Diese Zeichenkette, von der gesprochen wurde, ist beim 8*8-Sprite 8 Zeichen lang, beim 16*16-Sprite 32 Zeichen lang. Wie die einzelnen Zeichen angeordnet sind, zeigen uns die Bilder 11.3 und 11.4.

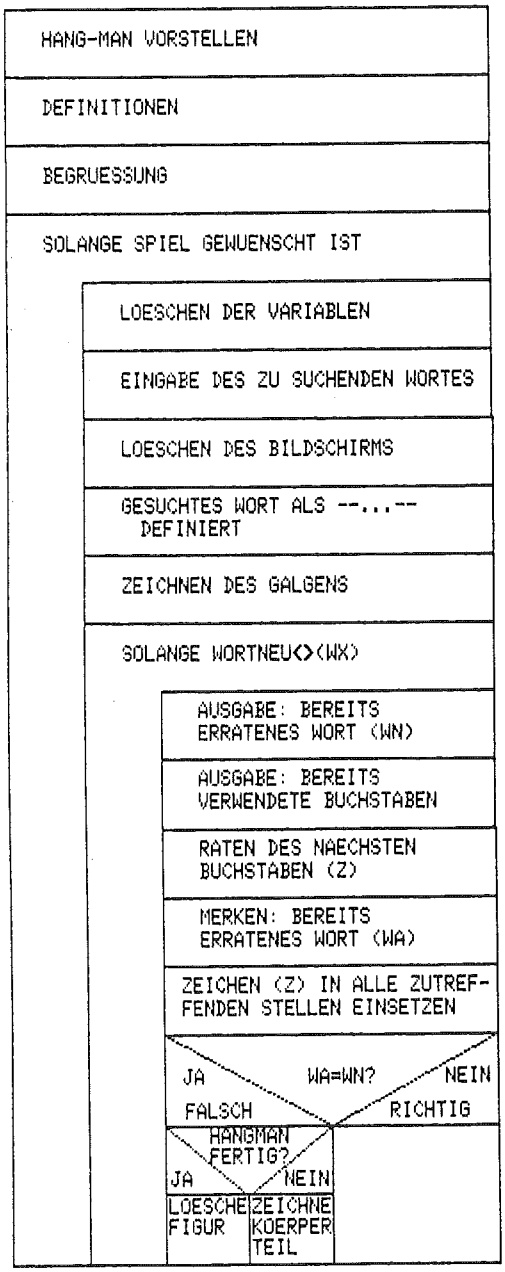


Bild 11.5

Mit "SPRITE$(Nummer)=Ausdruck" läßt sich so ein SPRITE nun definieren. Mit "PUTSPRITE Ebene, (X,Y), Farbe, Nummer" wird das SPRITE auf den Bildschirm gesetzt.

Man darf 32 SPRITES gleichzeitig definieren und 32 gleichzeitig am Bildschirm darstellen. Dabei darf immer nur ein SPRITE auf einer Ebene "verweilen". Dadurch ergibt sich die Anzahl der Ebenen: 32.

Sehen wir uns nun das Programm HANGMAN an.

Programm Hangman

Wieder wird zuerst die Spielbeschreibung gegeben, danach verlieren wir einige Worte über die Programmierung.

Am Anfang wird die Figur, die für den Galgen bestimmt ist, auf dem Bildschirm präsentiert. Danach gibt ein Teilnehmer des Spieles einen Namen ein, während alle anderen nicht auf den Fernseher blicken. Nur die Länge des Namens wird angezeigt.

Die Spieler, die "weggeschaut" haben, müssen nun das Wort erraten. Für jeden falschen Buchstaben wird ein Teil mehr des "Galgenvogels" aufgehängt.

Hängt der ganze Mensch, noch bevor das Wort erraten ist, haben die Spieler verloren.

Zur Programmierung:

In Bild 11.5 erkennen wir das Struktogramm für das Spiel.

Nach dem üblichen Anfang und der Begrüßung beginnt das eigentliche Spiel mit der Eingabe des zu suchenden Wortes.

Nun wird in einer Schleife immer wieder ein Zeichen eingegeben. Vorher zeigt der Computer das bisher erratene Wort an. Nun wird an allen Stellen, an denen der Buchstabe vorkommt, der Buchstabe eingesetzt. Wenn der Buchstabe falsch geraten war, wird ein Teil des Menschen am Galgen gezeichnet. Dies geht solange vor sich, bis das erratene Wort gleich dem vorher eingegebenen ist.

Danach beginnt das Spiel wieder mit der Eingabe des nächsten Wortes.

Zum Schluß zeigen wir ein Bild, in dem das Programm gerade "mitten in der Arbeit steckt" (Bild 11.6).

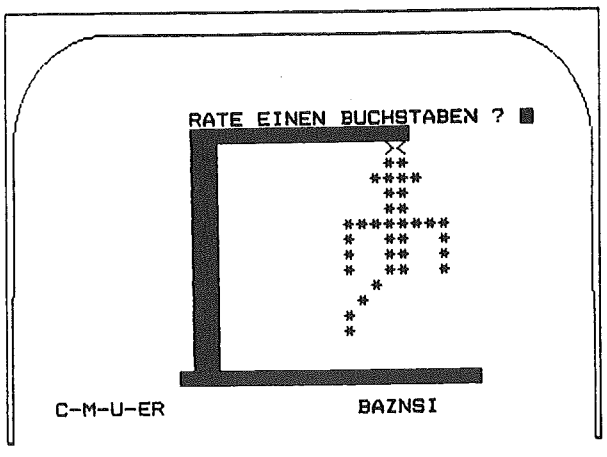


Bild 11.6

```
10 REM
20 REM BUCHSTABENSPIEL HANGMAN
30 REM ======================
40 REM
50 REM
60 CLS
70  PRINT"           333333333333
80  PRINT"           %         ><
90  PRINT"           %          **
100 PRINT"           %         ****
110 PRINT"           %          **
120 PRINT"           %          **
130 PRINT"           %         ****
140 PRINT"           %       * ** *
150 PRINT"           %       * ** *
160 PRINT"           %       * ** *
170 PRINT"           %       * ** *
180 PRINT"           %          **
190 PRINT"           %         *  *
200 PRINT"           %         *  *
210 PRINT"           %         *  *
220 PRINT"           %        **  **
230 PRINT"           %
240 PRINT"           %
250 PRINT" ......................... "
255 REM LEFT GRAPH + O = .
256 REM LEFT GRAPH + T = 3
257 REM LEFT GRAPH + M = %
260 F$="-":REM FUELLZEICHEN FUER DAS ER
STE ZU ERRATENDE WORT
270 LOCATE25,8
280 PRINT "HANGMAN"
290 LOCATE1,21:PRINT"BELIEBIGE TASTE DR
UECKEN ";
310 A$=INPUT$(1):REM BELIEBIGE TASTE
320 REM
1000 REM
1010 REM PROGRAMMBEGINN
1020 REM ==============
1030 REM
1040 WN$=""
1050 G$=""
1060 WA$=""
1070 V=0
1080 CLS
1090 REM
1100 REM
1110 INPUT "NEUES WORT ";WXXX$
1120 CLS
1130 L=LEN(WXXX$)
1140 REM
1150 REM
1160 FOR S=1TO L:WN$=WN$+F$:NEXT S
1170 REM
1180 REM
1190 X$="/"
1200 X1=10:X2=25:Y=1:GOSUB5090
1210 X1=9:X2=30:Y=19:GOSUB5090
1220 X=10:Y1=1:Y2=18:GOSUB5120
1230 X=11:Y1=1:Y2=18:GOSUB5120
1240 X$=">":X=24:Y=2:GOSUB5060:X$="<":X
=25:Y=2:GOSUB 5060
2000 REM
2010 REM NAECHSTER VERSUCH
2020 REM ==================
2030 REM
2040 LOCATE1,21:PRINT WN$
2050 REM
2060 REM
2070 LOCATE21,21:PRINT G$
2080 REM
2090 REM
2100 LOCATE10,0 :INPUT "RATE EINEN BUCH
STABEN";Z$
2110 Z$=LEFT$(Z$,1)
2120 REM
2130 REM
2140 REM
2150 WA$=WN$
3000 REM
3010 REM PRUEFSCHLEIFE
3020 REM =============
3030 REM
3040 FORS=1TOLEN(WXXX$)
3050  IF MID$(WXXX$,S,1)<>Z$ THEN 3080
3060  REM
3070  WN$=LEFT$(WN$,S-1)+Z$+RIGHT$(WN$,
L-S)
3080 NEXT S
3090 REM
3100 IF WA$=WN$ THEN 3160
3110 REM
3120 REM
3130  IF WN$<>WXXX$ THEN 2040
3140  LOCATE10,21:PRINT"GEWONNEN !!!":G
OTO4290
3150 REM
3160 REM
3170 V=V+1:
3180 G$=G$+Z$
3190 X$="*"
3200 ON V GOSUB 4050,4080,4100,4120,414
0,4160,4190,4220,4240,4260
3210 IF V<9 THEN 2040
3220 REM
3230 REM
3240 X$=" ":FOR W=1 TO 2000:NEXT W
4000 REM
4010 REM ZEICHNEN DES HANGMAN
4020 REM ====================
4030 REM
4040 REM KOPF
4050 Y1=3:Y2=5:X=24:GOSUB 5120:X=25:GOS
UB5120
4060 X1=23:X2=26:Y=4:GOSUB5090:IF X$<>"
 " THEN RETURN
4070 REM HALS
4080 X1=24:X2=25:Y=6:GOSUB5090:IFX$<>"
"THEN RETURN
4090 REM SCHULTRN
4100 X1=21:X2=28:Y=7:GOSUB5090:IF X$<>"
 " THEN RETURN
4110 REM ARME
4120 Y1=8:Y2=10:X=21:GOSUB5120:X=28:GOS
UB 5120:IFX$ <>" " THEN RETURN
4130 REM KOERPER
4140 X=24:Y1=8:Y2=11:GOSUB5120:X=25:GOS
UB 5120:IFX$<>" "THEN RETURN
4150 REM LINKES BEIN
4160 X=23:Y=12:GOSUB5060:X=22:Y=13:GOSU
B 5060
4170 X=21:Y1=14:Y2=15:GOSUB 5120:IFX$<>
" "THEN RETURN
4180 REM RECHTES BEIN
4190 X=26:Y=12:GOSUB 5060:X=27:Y=13:GOS
UB 5060
4200 X=28:Y1=14:Y2=15:GOSUB5120:IFX$<>"
 "THEN RETURN
4210 REM LINKER FUSZ
4220 X1=19:X2=20:Y=15:GOSUB5090:IF X$<>
" "THEN RETURN
4230 REM RECHTER FUSZ
4240 X1=29:X2=30:Y=15:GOSUB5090:IF X$<>
" "THEN RETURN
4250 REM ENDE
4260 LOCATE1,21:PRINT "DU HAENGST, DAS
WORT WAR "WXXX$"
4270 REM
4280 REM
4290 LOCATE10,22:PRINT"NOCH EIN SPIEL (
J/N)";:SP$=INPUT$(1)
4300 IF SP$="j" OR SP$="J" THEN 1040 EL
SE END
5000 REM
5010 REM UNTERPROGRAMME
5020 REM ==============
5030 REM
5040 REM
5050 REM
5060 LOCATEX,Y:PRINT X$;: RETURN
5070 REM
5080 REM
5090 LOCATEX1,Y:FOR I=0 TO X2-X1:PRINTX
$;:NEXT I:RETURN
5100 REM
5110 REM
5120 FORI=0TO Y2-Y1:LOCATEX,Y1+I:PRINTX
$;:NEXT I:RETURN
```

Alles in Butter

Die Zeiten, in denen Schriftsteller mit Tinte und Feder arbeiten mußten, sind vorbei. Heutzutage schreibt man mit elektrischen Schreibmaschinen. Trotzdem ist diese Art des Schreibens schon wieder veraltet. Wenn man nälich auf einer normalen Schreibmaschine einen Text schreibt, und man fabriziert einen Fehler, dann muß man manchmal das ganze Blatt noch einmal schreiben.

Mit Computern geht dies viel einfacher und besser. Wir tippen unseren Text anhand eines Textverarbeitungssystems in den Speicher oder auf Diskette, ändern was wir wollen und so viel wir wollen, und drucken zum Schluß mit einem Drucker den ganzen überarbeiteten Text in einem aus.

Das Korrigieren von Texten wird somit ein Kinderspiel. Ebenso das Kopieren, man läßt den Text ganz einfach zweimal ausdrucken.

WordStar

Wir haben auf unserem Gerät auch die Möglichkeit, so ein Textverarbeitungsprogramm zu verwenden. Es nennt sich WordStar. Wie alle bisher genannten Tools läuft WordStar unter CP/M. Sehen wir uns nun dieses Programm etwas näher an:

Das Prinzip von WordStar ist einfach. Man kann Texte schreiben, korrigieren, drucken und löschen. Das Programm ist so ausgereift, daß es auch eine Fehlbedienung nicht aus der Fassung bringen kann. Es läßt dem Bediener in kritischen Passagen immer die Möglichkeit offen, das eben Gesagte zu wiederrufen.

Wir haben zwei Arten von Texten. Mit "D" werden "Dokumentar"-Files geöffnet. Nun kann man mit allen Raffinessen wie automatischem Zeilenumbruch, Formatieren eines Absatzes, variabler Zeilenbreite und so weiter arbeiten.

Mit "N" werden die "Nicht-Dokumentar"-Files bearbeitet. Sie haben etwas weniger Komfort, man hat jedoch keine störenden Steuerzeichen im Text. Diese Art wählt man zum Editieren von Programmfiles (z.B. für Compiler, Assembler usw.), wobei man ja die Steuerzeichen nicht brauchen kann.

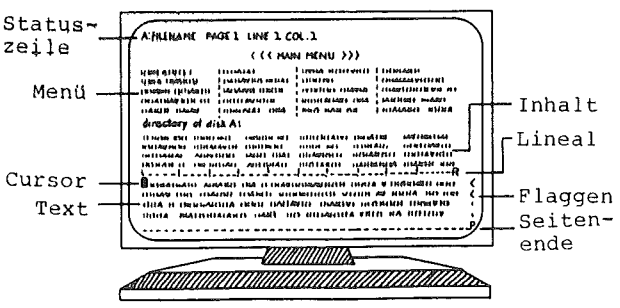


Bild 12.1

Wir werden mit D-Texten arbeiten. Wenn wir nun in so einen Text eingestiegen sind, dann können wir anfangen zu schreiben. Wir haben beim Spectravideo zwar keine direkten Umlaute, doch sind diese unter den Klammern rechts vom "P" "versteckt". Diese Tasten entsprechen den Normzeichen für den deutschen Zeichensatz. Die 80 Zeichen-Karte von SVI wird auf Wunsch auch mit dem deutschen Zeichensatz geliefert. Man hat also sowohl am Bildschirm als auch am Drucker die Umlaute zur Verfügung.

Ansonsten verhält sich unsere Tastatur wie eine Schreibmaschine. Wir können nun die Zeilenbreite und die Zeilenabstände mit "CTRL O" und danach entweder "L" (linken Rand einstellen), "R" (rechten Rand einstellen) und "S" (Zeilenabstand wählen) korrigieren. Am oberen Rand des Bildschirms sehen wir ein Lineal, welches uns immer die exakten Grenzen angibt, bis wohin wir schreiben dürfen. Wird diese Grenze überschritten, zieht WordStar das Wort in die nächste Zeile und füllt den leergewordenen Raum mit der bisher geschriebenen Zeile. Dadurch wird der Zwischenraum zwischen den einzelnen Wörtern zwar größer, die Zeilenränder sind jedoch nicht "ausgefranst".

Weiters gibt uns eine Statuszeile (Bild 12.1) noch Auskunft über die Position des Cursors, den Text, der gerade bearbeitet wird und den Zeilenabstand. Ebenso wissen wir, in welcher Seite wir uns befinden. Die Seitenlängen können wir definieren. Zuletzt gibt WordStar auch noch an, ob man sich im Einfügemodus befindet oder nicht. Dies wird durch ein Schild "INSERT ON" angezeigt. Im Einfügemodus wird der Text, der rechts vom Cursor steht, beim Schreiben vorgeschoben. So wird ein Einfügen von Buchstaben, Wörtern und sogar ganzen Absätzen ermöglicht. Manche Schriftsteller arbeiten nur im Einfügemodus, um bei Tippfehlern schnell kleine Korrekturen anbringen zu können.

Will man während des Schreibens einen Absatz machen, drückt man ENTER.

Soweit das normale Schreiben. WordStar kann aber noch viel mehr.

Bevor wir nun die einzelnen Befehlsgruppen durchbesprechen, noch ein wenig zur Organisation dieses Tools. WordStar ist ein bedienergeführtes Programm daß heißt, dem Bediener wird immer vor Augen geführt, was er im Moment machen darf und was nicht. Hier stehen jedoch drei Hilfstufen zur Verfügung.

In der ersten sieht der Bediener ständig die Kopfleiste und die wichtigsten Kommandos, die verwendet werden können. Drückt er eine Anweisung, die noch weitere Befehle nach sich zieht, zeigt ihm der Computer nun die nächsten Schritte.

Für etwas Fortgeschrittenere bietet WordStar im Schreibmodus nur die Kopfleiste an, erst bei Drücken einer Anweisung sieht man die Möglichkeiten, die man zum Weitermachen hat.

Für ganz professionelle gibt es auch die Variante ohne Hilfe. Keine Kopfleiste steht zur Verfügung.

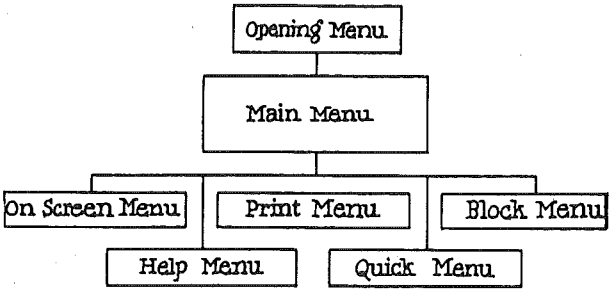


Bild 12.2

Je weniger Hilfe man in Anspruch nimmt, desto größer wird natürlich der Auschnitt des Textes am Schirm. Deshalb wird dies zur Auswahl gestellt.

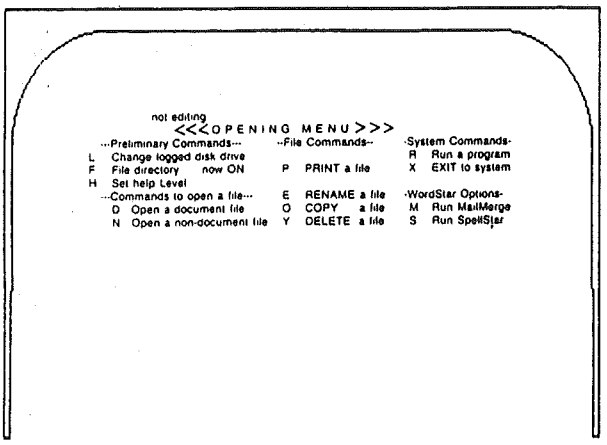


Bild 12.3

Die Gliederung des Programms zeigt die Darstellung in Bild 12.2. Die Menüs sind in den Abbildungen dieser Folge dargestellt. Sie werden vom Hauptmenü her durch Eingabe von CTRL und einen der Buchstaben K, J, Q, P oder O erreicht.

Die Befehlsgruppen

Es gibt folgende Befehlsgruppen:

CTRL-K (wie Kommando): In dieser Gruppe sind alle Speicherkommandos untergebracht. Ebenso lassen sich Suchbefehle und Blocktransferbefehle finden. Zuletzt gibt es in dieser Sparte noch die Datei-Manipulationen. Man kann diese verschiedenen Anweisungen unter das Motto Block-Kommandos stellen,

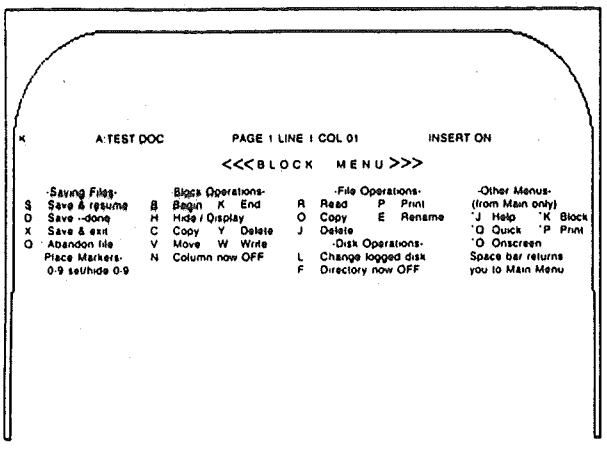


Bild 12.4

daher das "K". Eine Zusammenfassung der Befehle finden Sie im "BLOCK MENU" in Bild 12.4.

CTRL-O (wie Onscreen=auf den Bildschirm): Hier finden wir alle Befehle, die etwas mit der Bildschirmausgabe zu tun haben. So wird zum Beispiel Zeilenbreite und Zeilenabstand eingestellt. Diverse Steuerzeichen können ein- oder ausgeschalten werden, und vieles andere mehr. Das "ONSCREEN MENU" finden Sie in Bild 12.5.

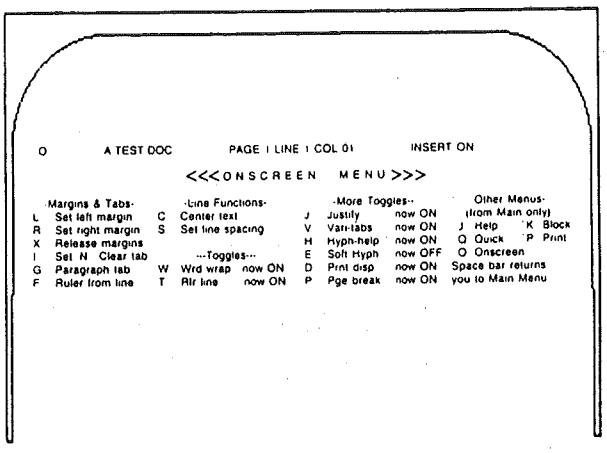


Bild 12.5

CTRL-Q (wie Quick=Schnell): Hier finden sich alle Fähigkeiten, Textteile zu suchen, umzuändern oder schnell an den Anfang von Zeilen oder Texten zu gelangen. So bilden die Tasten "E","D","S" und "X" ohne CTRL-Q einen Cursortastenersatz (aber zusammen mit CTRL). Mit CTRL-Q wandert der Cursor zwar in die gleiche Richtung wie vorher, nur gleich bis an den Rand des Bildschirms. CTRL-Q S läßt den Cursor an den linken Rand des Bildschirms bewegen. Das zugehörige Menü ist in Bild 12.6 dargestellt.

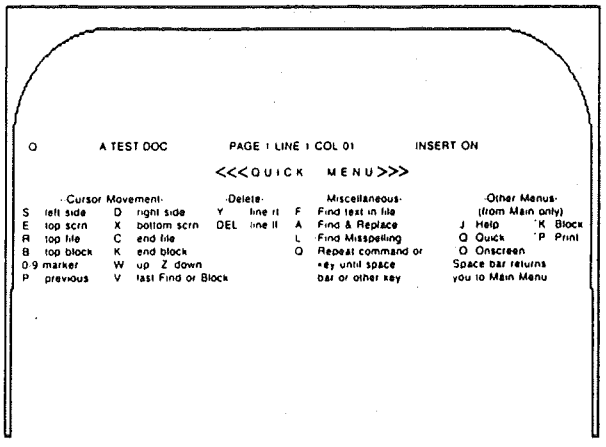


Bild 12.6

CTRL-J: Mit dieser Befehlsgruppe werden alle Funktionen zur Hilfe des Benutzers überwacht. Hier kann man genau bestimmen wie viel man angezeigt haben will.

Zuletzt gibt es noch CTRL-P: Hier wird die Ausgabe an den Drucker überwacht. Man kann zum Beispiel mit CTRL-B einen Text für Fettdruck bestimmen. Am Bildschirm sieht man nur das Steuerzeichen, erst am Drucker erkennt man den Fettdruck. In dieser Weise gibt es noch einige solcher Spezialeffekte (Bild 12.7).

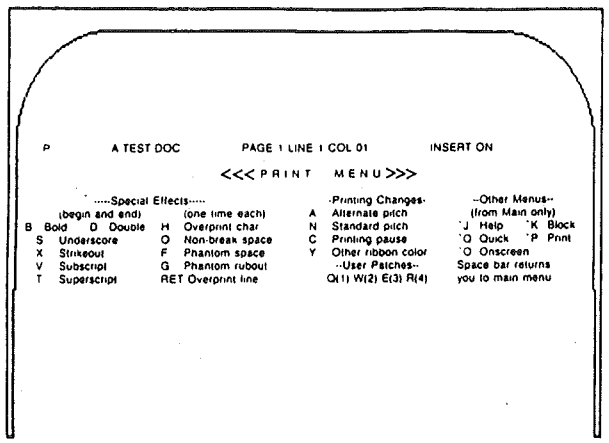


Bild 12.7

Das Hauptmenü

Wenden wir uns nun dem Hauptmenü zu (Bild 12.8) und sehen wir uns jene Befehle an, welche normalerweise am häufigsten verwendet werden, die Editierkommandos:

Bei jedem Buchstaben muß die CTRL-Taste gedrückt werden. Da haben wir zuerst die Cursorbewegungen: S (Zeichen nach links), A (Wort nach links), E (Zeile hinauf), X (Zeile hinunter), D (Zeichen nach rechts) und F (Wort nach rechts).

Danach folgen die Bildschirmscrollbefehle: Z (Zeile hinauf), W (Zeile hinunter), C (Bildschirmseite hinauf) und R (Bildschirmseite hinunter).

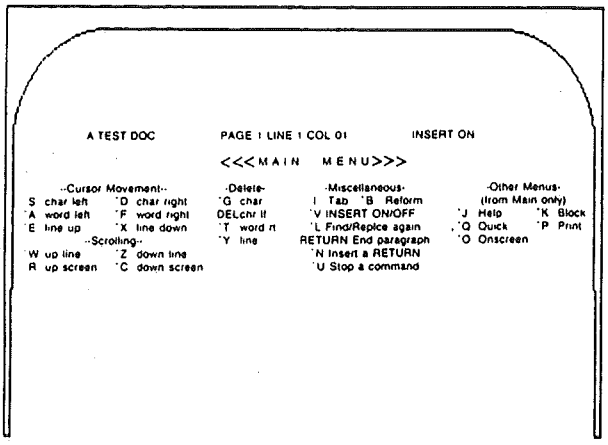


Bild 12.8

Weiters finden sich die Löschbefehle: G (lösche Zeichen unter Cursor), DEL (lösche Zeichen links vom Cursor), T (lösche Wort vom Cursor an) und Y (lösche Zeile, in welcher der Cursor steht).

Zuletzt erkennen wir noch einige andere Kommandos: Mit B formatiert man einen Absatz. Dabei versucht der Computer, so wenig Freiraum wie möglich zwischen den Wörtern zu belassen. Aus diesem Grund kann man beim Formatieren auch Wörter abteilen. Um dem Benützer die Arbeit zu erleichtern, gibt der Computer Abteilungsvorschläge, die aber nicht immer stimmen.

Mit V schaltet man den Einfügemodus ein oder aus. I erzeugt einen Tabulatorsprung. U wird zum Löschen des derzeit laufenden Kommandos gebraucht. N erzeugt eine Leerzeile. L sorgt zuletzt für Such- und Änderungsbefehle.

Bevor wir uns nun den praktischen Gebrauch von WordStar ansehen, wollen wir uns nochmals dem NO FILE-Menü (Bild 12.3) widmen, an dem wir schon am Anfang vorbeigekommen sind, als das Programmwerkzeug initialisiert wurde.

Hier kann man die einzelnen Texte zur Bearbeitung auswählen. D und N kennen wir schon, doch es gibt noch einige andere auch.

Während wir bis jetzt immer CTRL drücken mußten, brauchen wir dies in diesem Menü nicht zu tun, warum?

Im Textmodus schreiben wir, daher kann jeder Buchstabe, der als Steuerzeichen fungiert, auch als normaler Text vorkommen. Wenn wir kein CTRL vorgesehen hätten, wüßte der Computer nie, ob mit dem gedrückten Buchstaben ein normales Zeichen oder eine Anweisung gemeint ist. Da aber die CTRL-Taste im Text nicht verwendet wird, eignet sie sich glänzend dazu, für die Befehle verwendet zu werden.

Im "OPENING MENU" hingegen schreiben wir keine Texte, jedes Zeichen ist eine Anweisung. Das CTRL ist nun überflüssig!

Kommen wir jedoch wieder zu dem NO FILE-Menü zurück. Wir haben zuerst einmal die drei folgenden Befehle zur Verfügung: L ist Laufwerk wechseln, F schaltet das Inhaltsverzeichnis ein oder aus und H setzt die verschiedenen Hilfsstufen.

Weiters gibt es noch Y (Datei löschen), O (Datei kopieren), E (Datei umbenennen) und P (Datei ausdrucken).

Wenn man ein Programm laufen lassen möchte, dann tut man dies mit R. Zwei spezielle Programme, "Mailmerge" und "SpellStar" werden durch die eigenen Anweisungen M und S gestartet.

Um aus dem WordStar wieder in das CP/M auszusteigen muß man X drücken.

Nun haben wir aber wieder genug theoretisiert. Nun stürzen wir uns auf den Praxisbetrieb. Wir möchten einen Brief schreiben, haben als Vorlage schon ein nettes Gedicht von Eugen Roth, nur das geeignete Briefpapier fehlt uns. Macht nichts, wir schreiben nicht auf Brief-, sondern auf Computerpapier, per WordStar.

Wir schreiben einen Brief

Wenn wir den Computer mit mit unserer CP/M-Diskette initialisiert haben, tippen wir WS und ENTER. Danach erscheint das Titelbild unseres Programmes. Wenn wir ENTER oder die Space-Taste betätigen, verschwindet es, und das NO FILE-Menü erscheint. Da unser Brief ein ganz normaler Text ist, werden wir ihn mit sämtlichen Steuerzeichen schreiben, daher drücken wir D. Nun geben wir den Namen des Files ein, "BRIEF". Schreiben wollen wir den Brief in einer Zeilenbreite von 40 Zeichen. Daher tippen wir CTRL-O L 2 und CTRL-O R 41. Der linke Rand auf 2, der rechte Rand auf 41, das ergibt genau 40 Zeichen Breite.

Nun lassen wir noch die Seitennummerierung durch ".op" "unter den Tisch fallen". Wir wollen ja keine Seitennummern im Brief. Ebenso stellen wir am Anfang ein ".mt0" und

ein ".mb4". Damit korrigieren wir später beim Drucken den Freiraum, der auf dem Blatt über der ersten und unter der letzten Zeile frei bleibt.

Nachdem wir nun unser Werk getan haben, beginnen wir, unseren Brief zu schreiben.

Dabei sehen wir, daß die einzelnen Wörter, welche über den Rand hinausstehen, in die nächste Zeile geschoben werden. Der Text sieht insgesamt so aus:

"Ein Mensch schreibt mitternächtig tief an die Geliebte einen Brief, der schwül und voller Nachtgefühl. Sie aber kriegt ihn morgenkühl, liest gähnend ihn und wirft ihn weg. Man sieht, der Brief verfehlt den Zweck. Der Mensch, der nichts mehr von ihr hört, ist seinerseits mit Recht empört und schreibt am hellen Tag gekränkt und saugrob, was er von ihr denkt. Die Liebste kriegt den Brief am Abend, soeben sich entschlossen habend, den Menschen dennoch zu erhören - der Brief muß diesen Vorsatz stören. Nun schreibt die Grobheit abzubitten, der Mensch noch einen zarten dritten und vierten und fünften, sechsten, siebten der herzlos schweigenden Geliebten. Doch bleibt vergeblich alle Schrift, wenn man zuerst daneben trifft."

Nun kann man CTRL-B eintippen, um diesen Text zu formatieren. Bei "verfehlt" wird dann abgeteilt.

"Ein Mensch schreibt mitternächtig tief an die Geliebte einen Brief, der schwül und voller Nachtgefühl. Sie aber kriegt ihn morgenkühl, liest gähnend ihn und wirft ihn weg. Man sieht, der Brief verfehlt den Zweck. Der Mensch, der nichts mehr von ihr hört, ist seinerseits mit Recht empört und schreibt am hellen Tag gekränkt und saugrob, was er von ihr denkt. Die Liebste kriegt den Brief am Abend, soeben sich entschlossen habend, den Menschen dennoch zu erhören - der Brief muß diesen Vorsatz stören. Nun schreibt die Grobheit abzubitten, der Mensch noch einen zarten dritten und vierten und fünften, sechsten, siebten der herzlos schweigenden Geliebten. Doch bleibt vergeblich alle Schrift, wenn man zuerst daneben trifft."

Doch in Falle eines Gedichtes ist der Zeilenumbruch und das Formatieren des Absatzes nicht schön. Wir werden daher den rechten Rand weit hinausschieben, und die einzelnen Verszeilen mit ENTER abschließen.

"Ein Mensch schreibt mitternächtig tief
an die Geliebte einen Brief,
der schwül und voller Nachtgefühl.
Sie aber kriegt ihn morgenkühl,
liest gähnend ihn und wirft ihn weg.
Man sieht, der Brief verfehlt den Zweck.
Der Mensch, der nichts mehr von ihr hört,
ist seinerseits mit Recht empört
und schreibt am hellen Tag gekränkt
und saugrob, was er von ihr denkt.
Die Liebste kriegt den Brief am Abend,
soeben sich entschlossen habend,
den Menschen dennoch zu erhören -
der Brief muß diesen Vorsatz stören.
Nun schreibt die Grobheit abzubitten,
der Mensch noch einen zarten dritten
und vierten und fünften, sechsten, siebten
der herzlos schweigenden Geliebten.
Doch bleibt vergeblich alle Schrift,
wenn man zuerst daneben trifft."

Der Rand ist jetzt natürlich ausgefranst, da wir den Zeilenausgleich unterbunden haben. Dafür steht jede Verszeile nun in einer eigenen Bildschirmzeile.

Nachdem unser Werk fertig ist, speichern wir es mit CTRL-K D ab. Nun erscheint das Hauptmenü. Wir geben P ein, und können unseren Text auf den Drucker leiten.

Vor dem Drucken müssen wir allerdings noch eine Reihe von Fragen beantworten.

Zuerst geben wir den Namen des Textes ein, bei uns "BRIEF". Danach drücken wir dreimal ENTER, so beantworten wir die ersten drei Fragen. Unter diese drei Fragen fallen auch die Begrenzungen, falls man nur einen Teil des Textes ausdrucken möchte.

Danach geben wir ein N ein, daß heißt, wir wollen keinen Seitenvorschub. Nun kommt wieder ein ENTER, da wir die Steuerzeichen im Text ausgeführt sehen wollen und nicht wie am Bildschirm abgebildet. Zuletzt müssen wir noch entscheiden, ob wir eine Pause zum Papierwechsel haben wollen oder nicht.

Wenn alles eingestellt ist, wird dem Computer mit einem ENTER der Befehl gegeben, mit dem Drucken anzufangen.

Während der Drucker übrigens seine Arbeit verrichtet, können wir uns anderen Texten widmen. Der WordStar läßt zu, daß man Files bearbeitet, ja sogar das Laufwerk wechselt, während im Hintergrund ein anderer File gedruckt wird.

Nun haben wir ein Beispiel im Praxisbetrieb des WordStar kennengelernt. Der interessierte Leser, der mit diesem Programm arbeitet, wird in seiner Arbeit noch viele nützliche Funktionen von WordStar kennenlernen, die wir hier aus Platzmangel nicht erläutern können.

Es sind nämlich schon ganze Bücher über dieses sehr gute Textverarbeitungsprogramm geschrieben worden.

Nachdem wir nun in 13 Folgen einen kleinen Einblick in die Computerwelt erhalten haben, und die Familie Dischek bei ihrem Weg zur Computerfaszination begleitet wurde, stellt sich nun die Frage, welche Zukunft bietet sich für den Einsatz der Home- und Personal-Computer?

Mikrocomputer

Neben den Großcomputern werden sehr viele Mikrocomputer in immer weiteren Bereichen arbeiten. Dabei darf man sich aber diese Computer nicht als Jobkiller und "brutaldenkende" Roboter vorstellen, sondern als mehr oder weniger versteckte Hilfe für den Menschen.

Die meisten kleinen Prozessorsysteme werden ohne Wissen der Benutzer arbeiten, ähnlich den Elektromotoren in unseren heutigen Haushalten. Viele Menschen wissen gar nicht, daß bessere Kassettenrekorder bis zu drei Elektromotoren besitzen. Trotzdem verrichten diese Geräte ihren Dienst. Viele werden auch nicht bemerken, wie viele kleine Steuerungssysteme dafür sorgen, daß die diversen Waschmaschinen, Mixer und Herde ihren Dienst gefahrlos und korrekt verrichten. Der große Bruder mit dem noch größeren Bildschirm ist in dieser Sparte sehr weit von uns entfernt!

Heimcomputer

Daneben werden unsere Wohnungen bestimmt mit Heimcomputern ausgestattet werden, ähnlich dem Fernseherboom, der bei uns seit einigen Jahren wütet. Wir werden per Teletext-Systemen einkaufen "gehen" können und uns beliebige Informationen beschaffen. Während der Sinn der kleinen Prozessorsysteme zweifelsohne einleuchtend ist, wird es bei dieser Heim-Computerisierung schon problematischer, diese Entwicklung zu befürworten.

Tatsache ist jedoch, daß wir vor dem Computer keine Angst haben zu brauchen. Der "blutrünstige" Roboter, der uns Menschen haushoch überlegen ist, wird noch lange auf sich warten lassen. Wir Menschen werden jedoch lernen müssen, den Computer sinnvoll und mit Maß zu verwenden.

Nur wenn der Großteil der Bevölkerung mit den Computern umgehen kann, werden wir einsehen, daß wir nicht von derartigen Systemen beherrscht werden, sondern mit ihnen kommunizieren müssen. Dann werden Computer genauso wie die Erfindung des Rades ein Segen für die Menschheit.

Die einzige Gefahr, die in der Anwendung von Computern liegt, ist die erschreckend große Datenmenge, die jedermann verarbeiten und abrufen kann. Dadurch wird es möglich, vieles zu überwachen. Überwachung kann Segen, kann aber auch Fluch bedeuten!

Für uns sind jedoch die einzelnen Sparten der Heimcomputer wichtig. Deshalb sehen wir uns jetzt die Möglichkeiten näher an, welche uns diese Sparte in Kürze bieten wird.

Für Erziehung und Schulwesen

Zweifelsohne bieten Computer die Fähigkeit, Kindern mit Engelsgeduld verschiedenste Wissensgebiete beizubringen. Als Ergänzung zum heutigen Schulwesen können Computer interesierten Kindern sowohl zu Hause als auch in der Schule neue Dimensionen eröffnen. Man denke nur an die schier unbegrenzten Möglichkeiten, im Physik-Unterricht Experimente zu simulieren, die in natura nicht möglich sind. Oder haben sie schon einmal von einer Exkursion ins nächste schwarze Loch gehört?

Der Computer wird aber das jetzige Schulsystem nicht ersetzen, sondern nur ergänzen können.

Für Freizeit und Hobby

Mit der Fähigkeit, frei programmierbar zu sein, bieten Heimcomputer schon jetzt sehr viel Anreiz, selber zu programmieren oder Spiele ablaufen zu lassen. Man kann den Computer damit auch als Spielgerät oder als ernsthaftes Hobby betreiben.

Für den Haushalt

Jetzt noch Utopie ist die Möglichkeit, per Teletext-System, einzukaufen. Die Einführung wird jedoch nicht allzu lange auf sich warten lassen.

Es gibt aber schon jetzt genug Möglichkeiten, im Haushalt Computer einzusetzen. Wir wollen jetzt gar nicht von den vielen Steuerungen in den Geräten sprechen (Waschmaschinen, Staubsauger und so weiter).

Wie wir schon kennengelernt haben, können Hausfrauen ihre Kassen, Rezeptbücher und vieles andere durch Tools inventarisieren und abrufbereit halten.

Für den Beruf

Auch in professionellen Bereichen brauchen Computer keinen Vergleich zu scheuen. Die Möglichkeit, mit Datenbanken und Textverarbeitungssystemen zu arbeiten, versetzt viele kleine Firmen in die Lage, konkurrenzfähig zu bleiben. Mit modernen Kalkulationsprogrammen können Planungen durchgeführt werden, welche die nötigen Investitionen und den Zeitaufwand von Entwicklungen feststellen. Fehlplanungen werden so verringert. Nicht zuletzt wird auch in der Montage größere Präzision erreicht.

Spectravideo-Computer

Sehen wir uns als letzten Punkt auch noch die Firma unserer Computer an. Welche Möglichkeiten wird Spectravideo in Zukunft bieten?

Obwohl es den SVI-328 erst seit dem Sommer 1983 gibt, wurden bis jetzt schon viele Computer verkauft. Dies mag einerseits auf die gute Hardware des Grundgerätes zurückgeführt werden, bestimmt tragen auch die Erweiterungsmöglichkeiten des Spectravideo-Computers dazu bei.

Mit den Diskettenlaufwerken wird das weltweit verbreitete Betriebssystem CP/M mitgeliefert. Mit diesem ist es möglich, eine Menge professioneller Software auf dem Spectravideo-Computer einzusetzen. Einige Programmwerkzeuge haben wir ja in den vorhergegangenen Folgen besprochen. Darüber hinaus ist die Verwendung zahlreicher Programmiersprachen (Compiler) möglich, wie Cobol, Fortran, Turbo Pascal. Zum Zeitvertreib lohnt es sich jedoch auch, eines der zahlreich verfügbaren Spiele laufen zu lassen.

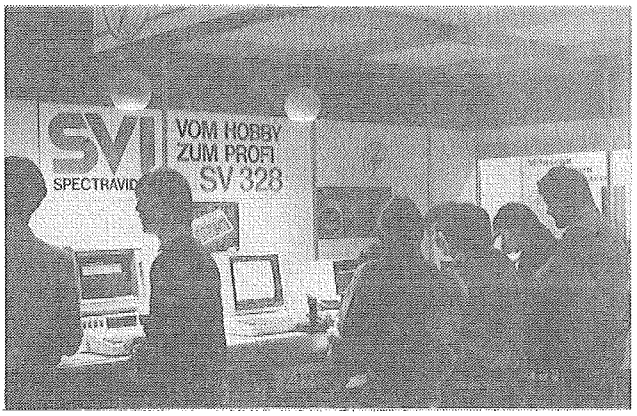
Für echte Spielernaturen gibt es den Spectravideo-Computer SVI-728, der voll MSX-kompatibel ist und so auf einen ungeheuren Video-Spiel-Markt zugreifen kann. Darüber hinaus kann man diesen Computer ebenfalls auf CP/M erweitern.

Ernstere Anwender werden sich jedoch an den mit zwei 320 KByte-Floppy-Disks erweiterbaren SVI-328 halten.

Natürlich wird Spectravideo nicht bei den jetzt vorhandenen Modellen verweilen. Unsere Computerfirma ist am besten Wege, die Marktlücke zwischen Heim- und Personal-Computern zu füllen.

Hoffentlich sind Sie genauso wie die Familie Dischek ein Computerenthusiast geworden. Doch dies ist noch kein Grund, ihre Familie auch zur Computerfamilie 2000 umzumodeln. Verwenden Sie ihren Computer mit Maß, und die Freude bleibt erhalten!

Die in der "Computerfamilie" vorgestellten BASIC-Programme erhalten Sie bei allen SVI-Fachhändlern.



Die Broschüre "Die Computerfamilie" wurde zur Gänze auf einem SVI-328 geschrieben, unter Verwendung von WordStar und Graphikprogrammen.

Idee und Entwurf Dipl.-Ing. Kurt P. Judmann, Text von Gerhard Fally, Redaktion und Gestaltung Heinz Schmid.

Hersteller: HTU-Wirtschaftsbetriebe GmbH. Ges.m.b.H. 1040 Wien.