
ACORN
COMPUTER
Kaindl
A. Berghofer
Band 1

INZOM

ACORN COMPUTER

Band 1
zur ORF-
Sendereihe
"COMPUTERFAMILIE"

für die Folgen 1–5

23. November 1984
30. November 1984
7. Dezember 1984
14. Dezember 1984
21. Dezember 1984

ACORN COMPUTER

„UNSERE ERSTE BEGEGNUNG"

FOLGE 1

Sie werden es mir nicht glauben! Letztens bemerkte ich, daß meine Frau eifersüchtig ist. Auf wen? Auf meinen COMPUTER. Und warum? Weil ich mich viel zu viel mit IHM beschäftige. Sie sagt, wenn ich mich mit derselben Hingabe mit ihr beschäftigen würde, dann würde sie glücklich sein. Na ja, sie hat schon recht. Nur eines beschäftigt mich: ist ER, mein COMPUTER, glücklich?
Ich werde IHN fragen und tippe auf seiner Tastatur mit spitzen Fingern vorsichtig

```
BIST DU GLÜCKLICH?
```

Voll Erwartung blicke ich auf den Schirm! Und? Nichts, rein gar nichts. Ist das eine Art? Meine Frau meint skeptisch: du mußt die RETURN-Taste drücken, damit schließt du deine Eingabe ab! Gut, ich tu's. Und was sehe ich am Schirm?

```
Mistake
>_
```

So, das hab ich davon. Ich versuch's mit Gefühl und ER meldet bloß – Mistake! Fehler! Irrtum, mein Freund.
Meine Frau versucht's. Sie tippt

```
PRINT "Er wird's schon noch lernen!"
```

und sagt lächelnd zu mir: Schau, wenn ER dir eine g'scheite Antwort geben soll, dann mußt du IHM aber auch eine Chance geben. Und ER kann sich nur mit einer PRINT-Anweisung uns mitteilen. Sagt's und tippt zum Schluß noch auf die RETURN-Taste. Ihr folgt unser COMPUTER, denn er gibt am Schirm aus

```
Er wird's schon noch lernen!
>_
```

Also, der amüsiert sich doch nur über uns zwei! ER äfft uns einfach nach. Oder nicht? Schreibt ER nur das an, was wir IHM anschaffen? Das muß ich rauskriegen. Ich tippe

```
PRINT Dich krieg ich schon noch RETURN
```

und was erscheint am Bildschirm

```
No such variable
>_
```

Das kann doch nicht wahr sein! Meine Frau lächelt nur sanft und zeigt auf den Schirm: Ohne Anführungszeichen (" ") wird er doch nur verwirrt. Und das lehnt er ab! Gut, gut, ich seh' es schon ein. Werd's mir merken. Jetzt muß es gelingen. Eifrig tippe ich ein

```
PRNT "Du bist ein kleiner Schelm!" RETURN
```

Lakonisch gibt ER aus

```
Mistake
>_
```

Jetzt ist der Ofen bei mir aus. Ein verstohlener Blick auf meine Frau – sie lächelt. Das wurmt mich. Ich muß es doch rauskriegen. Aufmerksam kontrolliere ich den Bildschirm – doch sie ist schneller. Sie meint mit ihrer liebevollen Stimme: PRNT ist halt nicht PRINT! Okay, okay, ich hab's schon kapiert.
Aber irgendwann muß ich doch meinen Mann stellen. Im Brustton der Überzeugung vermelde ich: Im Rechnen soll ER Spitze sein! Und tippe

```
PRINT "42 • 12" RETURN
```

Und zu meiner Frau gewandt: so viele Monate lebe ich nun schon auf dieser Welt! Gespannt blicke ich auf den Bildschirm

```
42 • 12
>_
```

Na klar, kein „Mistake". Aber halt, gerechnet hat ER nicht. Ein Blick auf meine Holde – sie lächelt schon wieder. Direkt einfühlsam sagt sie: Deine PRINT-Anweisung ist einwandfrei. Nur, wenn du rechnen willst, dann darfst du keine Anführungszeichen setzen! Jetzt verstehe ich nur mehr „Bahnhof". Einmal sind die " " zuviel und dann wieder zuwenig. Aber ihre Geduld scheint grenzenlos zu sein. Sie tippt

PRINT 42 * 12 RETURN

und ER schreibt folgsam

504
>_

Aha, so geht das. Sofort tippe ich eifrig

PRINT 42 * 12 * 365 * 24 RETURN

So viele Stunden lebe ich nun schon auf dieser schnöden Welt. Was zeigt der Schirm?

4 415 040
>_

Herrlich ich kann meinen COMPUTER programmieren! Das tut gut. Und was sagt jetzt sie dazu? Statt einem anerkennenden Blick meint sie: so schaut's aber schöner aus – und tippt

PRINT "Du lebst schon"; 42*12*365*24 ; "Stunden!" RETURN

He, he, da wird sie sich aber wundern! Da kriegt sie jetzt endlich auch eine „Mistake"-Meldung. Und was tut ER? ER folgt ihr auf's Wort und schreibt am Schirm

Du lebst schon 4 415 040 Stunden!
>_

Also, das versteh' ich nicht. Doch sie sagt sanft: Schau, der Schlüssel zu dem Geheimnis liegt in dem ; Zeichen. Damit trenne ich in der PRINT-Anweisung den Textteil (innerhalb der " ") von dem Rechenteil. Klingt plausibel. Ich frag' mich nur, woher sie das alles weiß?

Mikrocomputer

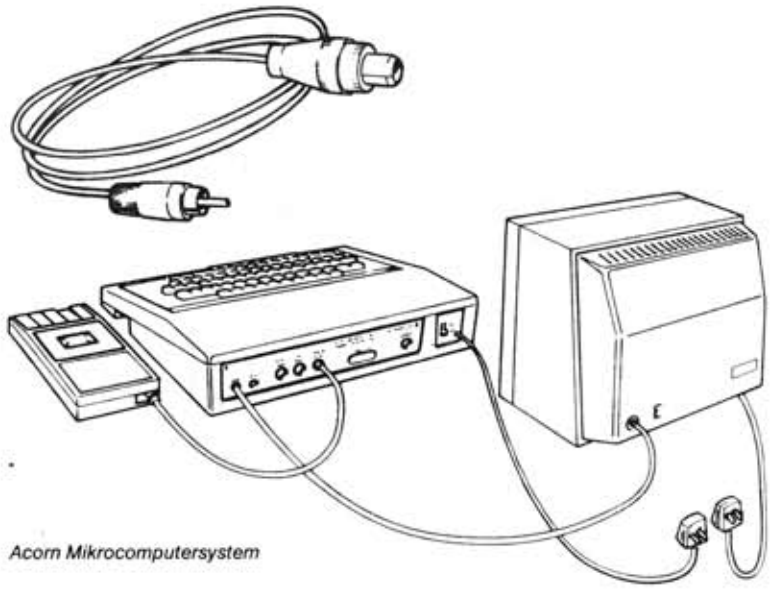
Mikrocomputer sind eine spezielle Bauform von Computern. Der Kern eines Mikrocomputers ist ein Mikroprozessor, enthält die wesentlichen Elemente der Zentraleinheit eines Computers.

Computer können in drei Bereiche gegliedert werden:

- *Die Zentraleinheit*
- *Der Arbeitsspeicher*
- *Die Ein/Ausgabeeinheiten*

Die Zentraleinheit eines Computersystems ist eine komplexe elektronische Baugruppe. Bei Mikrocomputern besteht sie aus einem einzigen hochintegrierten Schaltkreis, dem Mikroprozessor. Die Zentraleinheit ist in der Lage, aus dem angeschlossenen Arbeitsspeicher Befehle auszulesen und diese auszuführen. Jede Zentraleinheit kann nur eine bestimmte Anzahl von unterschiedlichen Befehlen verstehen und ausführen. Diese Befehle, sie werden auch Maschinenbefehle genannt, müssen für den Computer in elektronischer Form gespeichert werden, damit er sie automatisch lesen kann.

Der Arbeitsspeicher enthält die Befehle, die die Zentraleinheit ausführen soll. Die Zentraleinheit ist über elektrische Leitungen mit dem Arbeitsspeicher verbunden und arbeitet mit dem Arbeitsspeicher eng zusammen. Nach dem Einschalten des Gerätes beginnt die Zentraleinheit bei einer bestimmten Adresse im Arbeitsspeicher den ersten Befehl auszulesen und auszuführen. Die Befehle müssen im Arbeitsspeicher in geordneter Reihenfolge stehen, sie werden von der Zentraleinheit in dieser Reihenfolge ausgeführt. Der Arbeitsspeicher kann neben Befehlen auch Zwischenergebnisse aufnehmen. Spezielle Bereiche des Arbeitsspeichers können von der Zentraleinheit beschrieben werden und dienen als Zwischenspeicherbereich für Daten und Variablen.



Acorn Mikrocomputersystem

Bauelemente im Arbeitsspeicher:

Der Arbeitsspeicher eines Mikrocomputers wird durch elektronische Bauelemente realisiert. Je nach Aufgabe eines Speicherbereiches werden unterschiedliche Bauelemente eingesetzt.

Read Only Memory – ROM

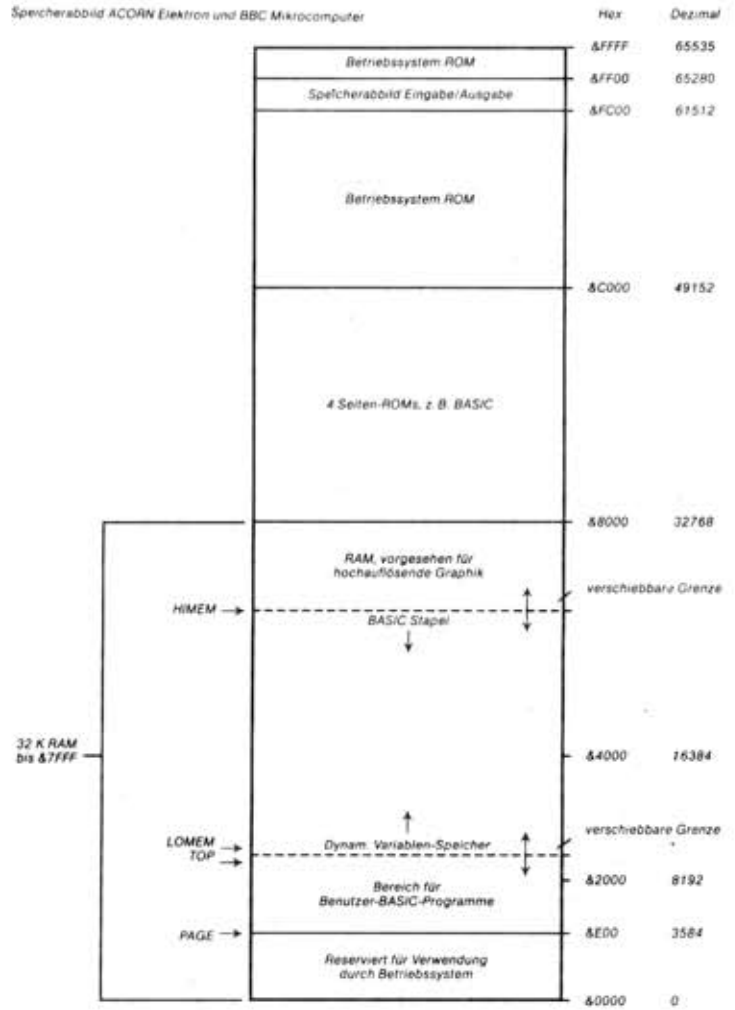
ROM's, das Wort kommt von der englischen Bezeichnung Read Only Memory, sind Speicherbauelemente deren Inhalt bei der Herstellung des Bausteins festgelegt wird. Die in ROM's enthaltenen Befehle und Daten können später nicht mehr verändert werden, sie gehen auch dann nicht verloren, wenn das Bauelement nicht mit Strom versorgt wird.

Random Access Memory – RAM

Speicherbereiche, die wechselnde Daten oder Programme aufnehmen sollen, werden aus sogenannten Schreib/Lesespeichern aufgebaut. Schreib/Lesespeicher werden nach der englischen Bezeichnung Random Access Memory auch RAM, genannt. Diese Speicherbausteine können von der Zentraleinheit eines Computers beschrieben werden, sie behalten ihren Inhalt solange sie mit Strom versorgt werden. Die Information in einem RAM geht auch durch das Auslesen durch die Zentraleinheit nicht verloren, sie wird erst gelöscht, wenn sie durch die Zentraleinheit überschrieben wird.

Der größte Teil des Arbeitsspeichers eines vom Anwender programmierbaren Mikrocomputers besteht aus Schreib/Lesespeicher. In diesen Schreib/Lesespeicher kann der Anwender selbst Programme einschreiben, die dann von der Zentraleinheit ausgeführt werden. Sowohl zur Eingabe von Programmen in diese Schreib/Lesespeicher, wie auch zur Ausführung durch die Zentraleinheit, benötigt der Anwender allerdings Programme die im Mikrocomputer schon nach dem Drücken des Einschaltknopfes vorhanden sein müssen. Diese Programme, die zum Betrieb des Computers unbedingt notwendig sind, werden Betriebsprogramme genannt, sie sind in ROM's enthalten.

Sowohl der Acorn Mikrocomputer, wie auch der Acorn Elektron, verwenden 64 Kilobyte Arbeitsspeicher. Die obere Hälfte dieses Speichers ist für ROM's reserviert. In der Grundausstattung sind das Betriebssystem ROM, sowie ein BASIC Interpreter ROM enthalten. In freie Stecksockel können ROM's für spezifische Programme wie z. B. Tabellenkalkulation oder Textverarbeitung eingesetzt werden. Die untere Hälfte des Arbeitsspeichers enthält Schreib/Lesespeicher. In diesen Speicher werden sowohl BASIC Programme, wie auch Zwischenergebnisse aus BASIC in Variablen abgelegt. Der obere Teil des Schreib/Lesespeichers ist für die hochauflösende Graphik reserviert. Je nach der Auflösung des gewählten Graphik Modes wird dafür ein bestimmter Speicherblock verwendet.



Ein/Ausgabegeräte

Programmierbare Mikrocomputer sollen von Menschen bedient werden können. Dementsprechend sind die Ein/Ausgabegeräte, die an Mikrocomputer angeschlossen sind, für die Kommunikation mit Menschen ausgelegt. Das wichtigste Eingabemedium für den Mikrocomputer stellt die Tastatur dar. Wer bereits mit einer Schreibmaschinentastatur vertraut ist, wird mit der Tastatur des Acorn B Mikrocomputers oder des Acorn Elektron, mühelos zurecht kommen. Zusätzlich zu einer Schreibmaschinentastatur sind im Acorn B Mikrocomputer eine Reihe spezieller Tasten angeordnet, deren Funktion einfach beherrscht werden kann.

Die Tastatur des BBC Microcomputers

Nach dem Einschalten des Computers kann der Bediener über die Tastatur Befehle an den Computer geben. Das Betriebssystem, das ist das Programm, das sofort nach dem Einschalten im Computer abläuft, liest jedes auf der Tastatur gedrückte Zeichen ein. Eingaben müssen durch die Return-Taste abgeschlossen werden. Wenn ein Wort eingegeben und abgeschlossen wird, überprüft das Betriebssystem ob es sich dabei um einen direkt ausführbaren Befehl handelt. Wenn man sich bei der Eingabe genau der Schreibweise eines der Befehle bedient hat, die der Mikrocomputer in der Grundbetriebsart auszuführen in der Lage ist, wird der Befehl durch das Betriebssystem ausgeführt. Wurde eine Zeichenkette eingegeben, die vom Betriebssystem nicht als Befehl interpretiert werden kann, meldet der Computer über sein Ausgabemedium, dem Bildschirm, daß ein Fehler passiert ist. Er gibt die Meldung:

MISTAKE

aus.

Beim BBC Mikrocomputer sowie beim Elektron ist das eigentliche Computersystem mit Mikroprozessor, Arbeitsspeicher und Ein/Ausgabeschnittstellen mit der Tastatur in einem Gehäuse untergebracht. Externspeicher und Bildschirm müssen daran angeschlossen werden.

Das wichtigste Ausgabemedim im Mikrocomputer ist der Bildschirm. Mit den Computern kann entweder ein normaler Standard-Fernseher verwendet, oder ein Hochqualitätsbildschirm, auch Monitor genannt, angeschlossen werden. Wenn ein normales Schwarz/Weiß oder Farbfernsehgerät verwendet wird, muß dieses auf den Kanal im UHF Sendebereich eingestellt werden, auf dem der Acorn Mikrocomputer sein Bild „sendet". Dies ist der UHF Kanal 36.

Auf dem Fernsehgerät oder Monitor stellt der Computer im Grundbetriebszustand normale Schriftzeichen dar. Über die vielfältigen Möglichkeiten Graphiken und bewegte Bilder am Bildschirm zu erzeugen, wollen wir in der Folge 11 der Unterlagen zur Fernseh-Computerfamilie berichten.

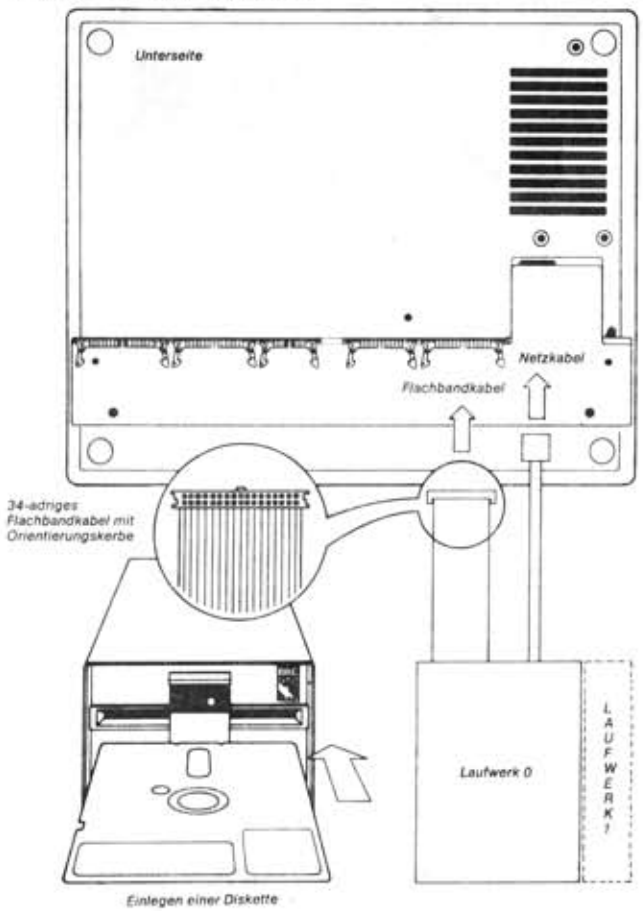
Externspeicher

Da vom Bediener in den Computer eingegebene Programme im Schreib/Lesespeicher des Mikrocomputers nur solange erhalten bleiben als die Versorgungsspannung einge-

schaltet ist, muß zur Archivierung dieser Programme oder zur Archivierung von Daten die mit Programmen verarbeitet werden sollen, ein sogenanntes Externspeichermedium verwendet werden. Darunter versteht man Geräte die es erlauben, Programme und Daten aufzuzeichnen und diese unabhängig von einer Versorgungsspannung zu archivieren.

Mit dem Acorn B Mikrocomputer wie auch mit dem Elektron wird entweder ein Kassettenrecorder für Standard-Tonbandkassetten verwendet, oder aber eine sogenannte Diskettenstation angeschlossen. Bei beiden Aufzeichnungsverfahren werden die Informationen auf magnetisierbarem Datenträger aufgezeichnet. Bei einem Kassettenrecorder werden die Informationen hintereinander durch Töne verschlüsselt auf eine normale Audiokassettte aufgezeichnet, bei einer Floppy Disk erfolgt die Aufzeichnung in magnetischer Form auf eine flache Magnetscheibe.

BBC Mikrocomputer mit Floppy Disk Laufwerk



Einlegen einer Diskette

Fertige Programme können entweder von Kassette oder einer Floppy Disk eingelesen werden. Einige Standardprogramme werden gemeinsam mit ihren Programmbeschreibungen auch in Festwertspeichern, ROM's, geliefert. Diese Bauelemente müssen dann im Computer in bestimmte Sockel eingesetzt werden.

„UNSER ERSTES PROGRAMM"

FOLGE 2

Hallo COMPUTER – Freundinnen und Freunde! Ihr müßt wissen, heute hat meine Frau ganz dringende Besorgungen zu erledigen. So sitze ich ganz allein vor IHM, meinem COMPUTER. Er blinkt mich mit seinem Cursor, einem _, am Bildschirm fröhlich an. Als wenn ER mir signalisieren wollte – Trau' dich doch! Und ob ich's versuche. Flugs tippe ich ein

```
PRINT 45 +12 +365 +24 +3600 RETURN
```

Im selben Zug schreibt ER an

```
1.702944 E 10
```

Sehen sie, so geht's im Leben. Ich hab doch fast dasselbe wie voriges Mal eingetippt und statt einer schönen runden Zahl schreibt ER solches Zeug. Wie? Sei meinen, das ist schon in Ordnung? Aha, in der sogenannten „Reellen Zahlendarstellung". Das E 10 signalisiert, daß die Zahl 1.702944 noch mit 10 Nullen aufzubessern ist. Warten sie, ich rechne das schnell aus – es ergibt 17 029 440 000, also 17 Milliarden, 29 Millionen und 440 Tausend Sekunden. So lange lebt nun meine Frau schon. Toll, was? Aber, wenn ich's noch für Tante FRIEDA berechnen will, dann muß ich nochmals die ganze PRINT-Anweisung eintippen. Ehrlich, das wird mir auf die Dauer zuviel. Sie sagen, das geht einfacher? Mit einer Nummer vor der PRINT-Anweisung, unter der der COMPUTER sie dann für mich aufhebt? Aha, eine Nummer. Irgendeine? Na gut, ich probier's.

```
1Ø PRINT "Meine Frau lebt bereits "; 45 +12 +365 +24 +36ØØ; "Sekunden" RETURN
```

Nun, da haben wir's, ER zeigt keine Reaktion. Man sollte doch nicht jedem Glauben schenken. Wie? Ich kann's wieder mit dem LIST-Befehl sichtbar machen? Also gut, ein letztes Mal probier ich's noch und tippe

```
LIST RETURN
```

und tatsächlich, ER „listet", d.h. schreibt am Schirm.

```
1Ø PRINT "Meine Frau lebt bereits"; 45 +12 +365 +36ØØ; "Sekunden"
```

Gut, aber wenn ich für die Tante FRIEDA diese Berechnung durchführen will, dann muß ich ja trotzdem alles neu eintippen. Nein? Das geht mit der INPUT-Anweisung? Ja, aber wie! Was, zuerst muß ich im Speicher alles löschen? Mit NEW, sagen sie. Okay, ich tippe

```
NEW RETURN
```

Nichts sagt mein Computer drauf. Schnell tippe ich noch

```
LIST RETURN
```

Der Bildschirm bleibt leer. Meine schöne Programmzeile 10 ist weg. Also, auf sie höre ich noch einmal! Was wollte ich denn nur – ah ja, löschen. Ist auch geschehen! Sie haben recht. Zuerst brauchen wir eine INPUT-Anweisung, mit der wir auch Text ausgeben können. Nur, wie das geht? Gut, sie sagen an und ich tippe

```
1Ø INPUT "Wie heissen Sie", Name $ RETURN
```

Also, für mich ist das noch „Bahnhof". Sie sagen, es ist nicht so schlimm? Nun, dann erklären sie einmal!
Also, die INPUT-Anweisung veranlaßt zuerst den COMPUTER, den Text zwischen den Ausführungszeichen am Schirm mit einem anschließenden Fragezeichen auszugeben. Damit signalisiert ER uns, jetzt bist du dran – gib etwas über die Tastatur ein! Mit der RETURN-Taste schließen wir nun unsere Eingabe ab. Das muß ich ausprobieren. Wie? Mit dem RUN-Befehl! Mach' ich und tippe

```
RUN RETURN
```

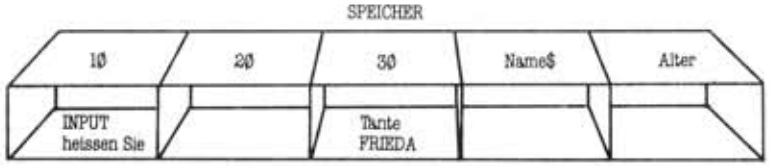
und schon steht am Bildschirm

Wie heissen Sie?

So, jetzt heißt's Farbe bekennen. Was wollte ich denn – ja, ich sollte die Tage im Leben meiner Tante FRIEDA berechnen. Munter wird gleich neben dem Fragezeichen eingetippt

Wie heissen Sie? Tante FRIEDA RETURN

Na ja, nur zeigt ER keinerlei Reaktion. Wie heißt's so schön, der COMPUTER, dein unbekanntes Wesen. Sie sagen, er hat's im Speicher abgelegt. Könnten sie das näher erklären? So wie in einer Büroablage mit einzelnen gekennzeichneten Fächern?



Jetzt verstehe ich langsam – mein Computer ist ein Sammler. Wie ein Staubsauger nimmt ER alles, was ich IHM eintippe, in seinen Speicher auf. Und mit der PRINT-Anweisung krieg ich's wieder zu sehen. Das probier ich aus! Mit

PRINT Name $ RETURN

schreibt ER am Bilschirm

Tante FRIEDA

Okay, kapiert. Na ja, da läßt sich schon einiges damit anfangen. Was würden sie zu dem sagen

2Ø INPUT "Wie alt sind Sie", Alter RETURN

3Ø PRINT Name $; "lebt bereits"; Alter +12 +365; "Tage!" RETURN

So, und noch schnell ein

RUN RETURN

damit ER die Anweisungen in der Reihenfolge ihrer Zeilennummern ausführt. Schon meldet der Bildschirm

Wie heissen Sie? Tante FRIEDA RETURN

Wie alt sind Sie? 72 RETURN

Tante FRIEDA lebt bereits 315 360 Tage!

Das geht doch prima, stimmt's?

Programme

Als Programme bezeichnet man eine geordnete Folge von Anweisungen an einen Computer. Die Anweisungen sind so geordnet, daß das vom Anwender gewünschte Ergebnis dadurch erzielt wird, daß der Computer die Befehle in ihrer logischen Reihenfolge ausführt.

Ebenen der Programmierung

Die Zentraleinheit eines Mikrocomputers, der Mikroprozessor, kennt nur wenige, sehr einfache Befehle. Viele Mikroprozessoren haben in ihrem Befehlsvorrat beispielsweise keine Multiplikation, sondern nur die Addition und Subtraktion realisiert. Dabei bezieht sich die Ausführung eines Befehles im Mikrocomputer immer auf Operanden, also Daten die genau einer bestimmten Stellenzahl, der sogenannten Wortlänge des Mikroprozessors entsprechen. Diese Stellenanzahl beträgt oft 8 binäre Stellen, das bedeutet, daß mit einem Additions- oder Subtraktionsbefehl von der Zentraleinheit auf einmal nur Zahlen zwischen 0 und 255 verknüpft werden können. Kompliziertere Operationen wie Multiplikation und Division, sowie die Operation an Zahlen mit höherer Darstellungsgenauigkeit, müssen daher durch Befehlsfolgen gebildet werden.

Durch die erwähnte Einschränkung in der Darstellungsgenauigkeit bei Verwendung der Grundbefehle der Mikroprozessoren, sowie der Einfachheit dieser Befehle, ist die Programmierung von Computern auf diesem Niveau äußerst schwierig. Jeder Befehl auf dieser Ebene, der sogenannten Maschinensprache, wird durch eine symbolische Buchstabenbezeichnung dargestellt.

Assembler

Assembler Programme bestehen aus Befehlsfolgen, die als Befehle den Grundbefehlsvorrat des Mikrocomputers verwenden. Der Programmierer muß, um solche Programme zu schreiben, für jeden Befehl, den der Mikroprozessor auszuführen in der Lage ist, ein symbolisches Wort niederschreiben. Um ein derartiges Programm von einem Computer tatsächlich ausführen zu lassen, muß es allerdings in binäre Form übersetzt werden. Die Zentraleinheit muß ihre Befehle als binär verschlüsselte Worte aus dem Arbeitsspeicher des Mikroprozessors auslesen. Eine entsprechende Anweisungsfolge ist daher für den Menschen sehr schwer verständlich.

Die Übersetzung aus den Assembler-Bezeichnungen, die der Programmierer niederschreibt, in die für den Prozessor verständlichen Binärcodes, erfolgt natürlich durch ein entsprechendes Computerprogramm im Computer selbst. Dieses Übersetzungsprogramm wird als Assembler bezeichnet, der Vorgang des Übersetzens als Assemblieren.

Da bei der Programmierung in der Assembler-Sprache eines Mikroprozessors jeweils die Befehle verwendet werden müssen, die der Prozessor tatsächlich auszuführen in der Lage ist, sind diese Programme auch typenspezifisch.

Höhere Programmiersprachen

Um die Programmierung für den Anwender zu vereinfachen und zu vereinheitlichen, wurden sogenannte höhere Programmiersprachen definiert. Höhere Programmiersprachen bestehen aus einer Menge von Befehlen und Funktionen, die ohne Rücksichtnahme auf einen bestimmten Computertyp festgelegt wurden. Auch die Zahlendarstellung für arithmetische Operationen entspricht einem Standard und ist nicht an die Wortlänge der Zentraleinheit gebunden, von dem das Programm dann tatsächlich ausgeführt wird.

Unter Verwendung dieser Programmiersprachen kann der Programmierer eines Computers Programmiersprachen lernen und Programme erstellen, ohne genau über das Innenleben seines Computers Bescheid zu wissen. Um diese Programme jedoch von einer bestimmten Zentraleinheit ausführen zu lassen, müssen sie natürlich in die Maschinensprache der Zentraleinheit übersetzt werden. Mikroprozessoren sind ja nicht in der Lage direkt die Anweisungen höherer Programmiersprachen auszuführen. Dieser Übersetzungsvorgang, der aus einem Programm in höherer Programmiersprache eine Folge von Maschinenbefehlen erzeugt, die die Zentraleinheit eines Computers tatsächlich auszuführen in der Lage ist, wird durch Übersetzungsprogramme, die im Betriebssystem eines Computers enthalten sind, durchgeführt.

Compiler

Als Compiler bezeichnet man ein Übersetzungsprogramm, das ein in höherer Programmiersprache geschriebenes Programm quasi in einem Arbeitsgang in ein äquivalentes Programm, bestehend aus Befehlen die eine bestimmte Zentraleinheit auszuführen in der Lage ist, übersetzt. Erst wenn dieses, vom Compiler erstellte Maschinenprogramm in den Arbeitsspeicher des Computers geladen wird und ausgeführt wird, kann die Wirkung des Programmes festgestellt werden. Die Programmerstellung mit einem Compiler umfaßt daher drei Schritte:

ACORN COMPUTER

- *Programmeingabe*
- *Übersetzung*
- *Ausführung*

Interpreter

Vor allem im Zusammenhang mit der Programmiersprache BASIC wird, wie auch im Elektron und im Acorn Mikrocomputer, ein Interpreter verwendet. Dabei ist die Übersetzung einzelner Zeilen des BASIC Programms und deren Ausführung ineinander verschachtelt. Im Gegensatz zum Compiler entsteht kein Programm im Maschinencode, das dem BASIC Programm entspricht. Jede Zeile wird für sich übersetzt, der Maschinencode der dieser Zeile entspricht, kurzfristig zwischengespeichert und sofort ausgeführt. Danach wird die nächste Zeile auf diese Art und Weise bearbeitet.

Der Vorteil eines Interpreters liegt in seiner einfachen Verwendung und in der kurzen Zeit mit der man Ergebnisse erzielen kann. Schon nach dem Schreiben auch nur einer Programmzeile kann dieses Programm in einem Zug übersetzt und ausgeführt werden.

BASIC

Nach dem Einschalten des Acorn Mikrocomputers befindet man sich im BASIC Sprachsystem. In diesem Grundzustand unterscheidet der Computer zwischen zwei Arten von Befehlen:

- *Steuerbefehle, auch Kommandos genannt, die er sofort ausführt.*

Diese Steuerbefehle sind nicht Elemente der Programmiersprache BASIC, sie dienen dazu dem Betriebssystem Anweisungen zu geben, um bestimmte Funktionen aufzurufen.

- *Programmbefehle in Form von numerierten Anweisungen*

Dies sind die eigentlichen Programme, die der Computer im Arbeitsspeicher ablegt um sie auf Abruf der Reihe nach auszuführen. Nach dem Eingeben von Programmen wird die Ausführung durch die Eingabe des Steuerbefehls

```
RUN
```

gestartet.

Die wichtigste Regel bei der Verwendung der Programmiersprache BASIC besteht darin, daß jede Programmzeile mit einer Zeilennummer beginnen muß. Der Computer führt die Befehle in der Reihenfolge der Zeilennummern in der sie stehen durch.

BASIC Anweisungen

INPUT – Dateneingabe während der Programmausführung

Beschreibung:

Die Anweisung INPUT dient zur Eingabe von Zahlen oder Zeichen über die Tastatur während der Programmausführung. Wenn die Anweisung INPUT ausgeführt wird, wird das Programm solange angehalten bis Zeichen über Tastatur eingegeben werden und diese Eingabe durch Drücken der RETURN-Taste abgeschlossen ist. Die eingegebenen Zeichen werden im Arbeitsspeicher des Computers abgelegt und können unter dem Variablennamen der in der Anweisung INPUT verwendet wurde, weiter verwendet werden.

Da das Programm bei Ausführung der Anweisung INPUT angehalten wird, erscheint am Bildschirm ein Anforderungszeichen um den Bediener darauf aufmerksam zu machen, daß eine Eingabe erwartet wird.

Beispiel:

```
10 INPUT XY
```

LET – Definition einer Variablen

Beschreibung:

Die Anweisung LET dient zur Wertzuweisung an eine Variable. Dabei wird das Ergebnis

einer Berechnung oder eines Ausdrucks im Arbeitsspeicher abgelegt und für die weitere Verwendung im Programm unter dem mit der Anweisung LET verwendeten Variablennamen zugreifbar gemacht.

Das Wort LET ist nicht obligatorisch, es kann auch weggelassen werden.

Beispiel:

```
100 LET TAG $ ="DIENSTAG"
210 LET X =5 * Y
```

PRINT – Bildschirmausgabe

Beschreibung:

Die Anweisung PRINT dient zur Ausgabe von numerischen und alphanumerischen Zeichen auf dem Bildschirm.

Nach dem BASIC Schlüsselwort PRINT können in einer durch Beistriche getrennten Liste Variablennamen oder Zeichenketten oder Steueranweisungen aufgelistet sein. Die dem Schlüsselwort PRINT folgenden Elemente bezeichnet man daher auch als Druckliste. Bei der Ausführung der Anweisung PRINT werden die Inhalte der Variablen am Bildschirm ausgegeben. Alle Zeichen die zwischen Anführungszeichen stehen werden genauso ausgegeben wie angeschrieben. Werden Elemente in der Druckliste durch Kommas getrennt, bewirkt dies das nach einem Element soviele Leerstellen nachgesetzt werden, daß das folgende Element in die nächste „Zehnerspalte" am Bildschirm gesetzt wird. Ein Strichpunkt nach einem Element bewirkt, daß das folgende Element in der selben Zeile direkt angeschlossen wird.

Beispiel:

```
10  PRINT "HALLO", 2 * 3
100 PRINT A; "SCHILLINGE"
```

„UNSER HAUSHALTSBUCH"

FOLGE 3

Gestern habe ich meine Bank besucht. Ich sag' Euch, ein Drama. Der Kontostand sieht ja ganz passabel aus, nur mit dem Vorzeichen stimmt etwas nicht. Ich hätte nie gedacht, daß dieses lächerliche Zeichen (-) mich so bedrücken könnte. Mein geplanter schöner Urlaub – ade! Auch mein altes Auto muß noch durchhalten, ob es will oder nicht?
Okay, jetzt hilft kein Bedauern. Nun muß etwas geschehen. Ich werde mir ab nun jede meiner Ausgaben fein säuberlich notieren. Dann kann ich einfach feststellen, wo der wunde Punkt in meiner Ein- und Ausgabenrechnung liegt. Es wird halt mit der Zeit ganz schön viel zum Rechnen sein. Mit meinem COMPUTER ist das kein Problem. Ich werde ein Programm schreiben, das dann die Gegenüberstellung der Ausgaben zu den Einnahmen vornimmt. Nur eines beschäftigt mich – wie mach' ich's? Ein lieber Freund sagte mir vor kurzem, er verwende nur fertige Programme – sogenannte COMPUTER-Werkzeuge, auch TOOLS genannt. Diese seien leicht anzuwenden und für seine Bdürfnisse völlig ausreichend. Das ist mir die ganze Zeit nicht aus dem Kopf gegangen. Nach kurzem Stöbern in meinen COMPUTER-Unterlagen fiel mir ein Programm-Modul in die Hände, das mich nach kurzem Einlesen begeistert hat. Es heißt VIEWSHEET und wird mit eintippen von

- `SHEET RETURN`

```
CONTENTS =
0      .....A.....B.....C.....D.....E
.....1
.....2
.....3
.....4
.....5
.....6
.....7
.....8
.....9
.....10
```

gestartet. Der Schirm löscht sich und es erscheint ein Raster

Sie müssen sich ein riesiges Arbeitsblatt vorstellen, von dem wir den linken oberen Ausschnitt sehen. Auf diesem Blatt – elektronisch natürlich – können wir jetzt beliebige Texte notieren und jegliche Berechnungen durchführen. Jede Eintragung wird automatisch in jeder Beziehung, in der die Eintragung vorkommt, berücksichtigt. Gigantisch, was man damit alles so schaffen kann. Na, wie wär's, wir könnten doch unser Haushaltsbuch damit realisieren. Wer wagt, gewinnt. Ich tippe erst einmal eine Überschrift. Wie? Nun, dort, wo der weiße Balken grad steht, wird das eingetippte Wort mit der RETURN-Taste eingetragen. Mit der –> Taste wird der Balken nach rechts verschoben und das nächste Wort eingetippt. Das sieht dann so aus

```
CONTENTS =

0       .....A.....B.....C.....D.....E
.....1  ********************************
.....2  *    HAUSHALTS-BUCH    *  86/6
.....3  ********************************
.....4  ▬▬▬
.....5
.....6
.....7
```

Sie werden's schnell heraußen haben, die Tasten mit den Pfeilen aufwärts, abwärts, links und rechts steuern, den Balken in die gewünschte Richtung bzw. ins gewünschte Feld. So, das wichtigste ist geschehen, der Titel ist gefunden. Das andere ist eine Kleinigkeit. Zumindest wollen wir's hoffen. Also, als nächstes kommen 4 Spalten. In die erste Spalte tragen wir einfach das Datum jeder Ausgabe ein. Logisch! Und in die zweite Spalte kommt die Bezeichnung der Ausgaben hinein, wie z. B. Fleisch oder Miete. In die 3. Spalte tippen wir den Kaufpreis. Ja, und in der 4. Spalte legen wir das Konto fest, unter dem die Ausgaben vermerkt oder besser „verbucht" werden. Ich würde für die Betriebsausgaben, wie Miete, Telefon, eine 1, fürs Essen eine 2, für die Bekleidung eine 3, fürs Auto eine 4, usw. vergeben. So, und jetzt tippen sie schön! Das sieht dann so aus

```
CONTENTS =

0       .....A.....B.....C.....D.....E.....F
.....1  ********************************
.....2  *    HAUHALTS-BUCH     *
.....3  ********************************
.....4  <---Nur hier eingeben--->
.....5  -----  -----  -----  -----  -----  -----
.....6  Datum  Einkauf    Preis  Konto  Betrieb Nahrung
.....7                                      1      2
.....8  -----  -----  -----  -----  -----  -----
.....9  84-06-02  Fleisch   224.00      2
....10  84-06-03  Mantel   1600.00      3
....11  84-06-04  Vase      333.00      5
....12  84-06-05  Meinl      34.00      2
....13  84-06-06  Kino      123.00      6
....14
```

Sieht doch schon ganz gut aus, nicht? Sie müssen jetzt nicht ihren Ehrgeiz demonstrieren und die Ausgaben der letzten fünf Monate hier eintippen. Dazu haben wir schon Zeit. Jetzt wollen wir die Rechenleistung unseres „Elektronischen Arbeitsblattes" demonstrieren. Wir fahren mit dem „Balken" unter die letzte Position der Spalte und nun soll uns der COMPUTER die Summe der Ausgaben ermitteln. Wir tippen einfach C9C13 mit einem abschließenden RETURN ein und schneller als sie atmen können steht die Summe der Einzelpreise am Schirm. Toll, was? Ich habe bloß die Koordinaten des ersten und des letzten Feldes in der Spalte für die Preise angegeben. So einfach geht das. Wenn jetzt die Ausgaben in die einzelnen Konto-Spalten eingetragen werden sollen, so soll das der COMPUTER natürlich selbst machen. Flugs steuern wir unseren Balken in die Spalte „E9", die wir für die Betriebskosten vorgesehen haben, und zwar in die erste Zeile unserer Ausgaben. Nun müssen wir IHN befragen: wenn (IF) die links stehende Kontobezeichnung mit der Nummer unserer Kontospalte übereinstimmt (also die Ausgabe unter die Betriebskosten fällt) dann (THEN) soll der Betrag in das Feld eingetragen werden. Im anderen Fall (ELSE) soll nichts eingetragen werden. Dazu tippen wir an dieser Stelle des „Elektronischen Arbeitsblattes" die Formel


```
IF (D9 =E7, C9, 0)
```

So geht's hurtig dahin und in kurzer Zeit sieht's dann so aus

```
**********************************************************************
*                  HAUSHALTSBUCH                                     *
**********************************************************************
**********************************************************************
*                  <----- Nur hier eintragen ----->                  *
**********************************************************************
```

Datum	Einkauf	Preis	Konto	Betrieb 1	Nahrung 2	Kleider 3	KFZ 4	Wohnung 5	Freizeit 6
84-06-02	Fleisch	224,00	2		224,00				
84-06-03	Mantel/m	1600,00	3			1600,00			
84-06-04	Vase	333,00	5					333,00	
84-06-05	Meindl	34,00	2		34,00				
84-06-06	Kino	123,00	6						123,00
84-06-07	Museum	50,00	6						50,00
84-06-08	Benzin	356,00	4				356,00		
84-06-09	Miete	3000,00	1	3000,00					
84-06-10	Hausvers	350,00	1	350,00					
84-06-11	Meindl	234,00	2		234,00				
84-06-12	Tasche	126,00	1	126					
84-06-13	Strom	1000,00	1	1000,00					
84-06-14	Telefon	876,00	1	876,00					
84-06-15	Wein	899,00	2		899,00				
84-06-16	Bücher	233,00	6						233,00
84-06-17	Bücher	465,00							
		9893,00		5352,00	1391,00	1600,00	356,00	333,00	406,00

Das läßt sich nun für Monate erstellen. Es können genauso Jahresübersichten

```
****************************************************************
*                    HAUSHALTSBUCH                             *
****************************************************************
```

EINNAHMEN >>>>>>>>>>>>> AUSGABEN >>

Monat	Gehalt	Gesamt	Betrieb	Nahrung	Kleider	KFZ	Wohnung	Freizeit	Beruf
1	12000,00	15500,00	6000,00	3000,00	1000,00	1500,00	500,00	1500,00	2000,00
2	12000,00	12900,00	6000,00	3100,00	200,00	1500,00	0,00	100,00	2000,00
3	18000,00	15300,00	6000,00	3200,00	2400,00	1500,00	0,00	200,00	2000,00
4	12000,00	27200,00	6000,00	3300,00	400,00	1500,00	10000,00	4000,00	2000,00
5	12000,00	13700,00	6000,00	3400,00	600,00	1500,00	0,00	200,00	2000,00
6	18000,00	33520,00	6000,00	4000,00	5000,00	1500,00	20,00	15000,00	2000,00
7	12000,00	13800,00	6000,00	3900,00	300,00	1500,00	0,00	100,00	2000,00
8	12000,00	14600,00	6000,00	3800,00	1200,00	1500,00	0,00	100,00	2000,00
9	18000,00	13979,00	6000,00	3700,00	234,00	1500,00	200,00	345,00	2000,00
10	12000,00	18120,00	6000,00	3600,00	20,00	1500,00	0,00	5000,00	2000,00
11	18000,00	15175,00	6000,00	3500,00	155,00	1500,00	2000,00	20,00	2000,00
12	12000,00	14299,00	6000,00	3400,00	1299,00	1500,00	0,00	100,00	2000,00
1983	168000,00	208093,00	72000,00	41900,00	12808,00	18000,00	12720,00	26665,00	24000,00
Anteil %		100,00	34,60	20,14	6,15	8,65	6,11	12,81	11,53
Vorschau:									
Zunahme %	4,50	17,49	20,00	20,00	10,00	20,00	10,00	10,00	20,00
	175560,00	244492,30	86400,00	50280,00	14088,80	21600,00	13992,00	29331,50	28800,00

sowie einfache Bilanzen erstellt werden. Das sieht doch schon richtig professionell aus, nicht?

Tabellenkalkulation

Tabellenkalkulationsprogramme sind Softwarewerkzeuge, auch Tools genannt. Es handelt sich dabei um Standardprogramme, die für häufig wiederkehrende Arbeiten mit dem Computer ausgelegt sind. Der Anwender solcher Programmwerkzeuge hat die Möglichkeit das Softwarewerkzeug an seine eigenen Bedürfnisse anzupassen. Dabei muß er Befehle, die ähnlich die einer Programmiersprache sind, eingeben.

Die Anpassung an eine Aufgabe erfordert jedoch in der Regel keine Programmierkenntnisse und ist vom Anwender leicht durchzuführen.

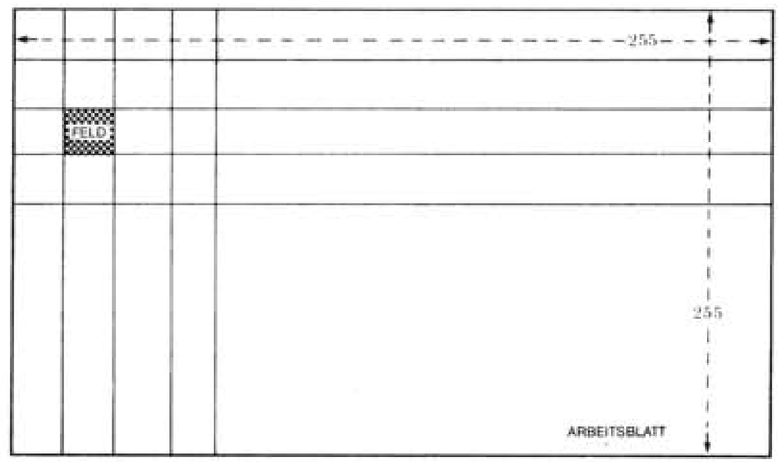
Mit Hilfe eines Tabellenkalkulationsprogrammes kann der Computerbenützer Manipulationen an Zahlen vornehmen, die in einem Arbeitsblatt in Zeilen und Spaltenform angebracht sind. Derartige Berechnungen kommen bei Kalkulationen gleichermaßen wie in Rechnungsformularen vor.

Für den Acorn Mikrocomputer steht das Kalkulationsprogramm „VIEWSHEET" zur Verfügung. Dieses Programm ist in einem elektronischen Baustein, einem ROM, fest einprogrammiert. Der Baustein muß in einen freien Stecksockel des Mikrocomputers eingesetzt werden. Je nachdem, in welchen Stecksockel er eingesetzt wird, wird entweder BASIC oder das eingesetzte VIEWSHEET sofort nach Einschalten des Acorn Mikrocomputers ausgeführt. Personen, die häufig Kalkulationen durchzuführen haben, können dadurch sofort nach dem Einschalten das Tabellenkalkulationsprogramm aktivieren.

Arbeitsblatt

Mit Hilfe von VIEWSHEET stellt der Bildschirm einen Ausschnitt aus einem großen Arbeitsblatt dar. Dieses Arbeitsblatt enthält 255 Zeilen und 255 Spalten. Daraus ergeben sich, festgelegt durch Zeilen und Spaltennummern, Felder. Durch spezielle Funktionen kann die Spaltenbreite im Arbeitsblatt verändert werden. Damit wird auch die Größe der Felder verändert.

VIEWSHEET verwaltet diese Felder und erlaubt die Eingabe von Werten, Zahlen oder Zeichen in diese Felder.



Wenn mit dem Arbeitsblatt gearbeitet wird, kann jedes Feld eine der folgenden drei Informationen aufnehmen:

- Eine Zahl
- Eine Zeichenkette, auch „LABEL" genannt
- Eine Formel

Durch die Möglichkeit für einen Feldinhalt anstelle eines Zahlenwertes eine Formel anzu-

geben aus der sich der Feldinhalt berechnet, können Feldinhalte von anderen Feldinhalten abhängig gemacht werden. Eine Formel im Feld D1 kann beispielsweise angeben, daß dieses Feld D1 die Summe der Zahlen in den Feldern A1 und B1 aufnehmen möge. Wenn diese Definition einmal festgelegt ist, kann in D1 kein Zahlenwert mehr direkt eingesetzt werden, der an der Stelle D1 am Arbeitsblatt angezeigte Wert errechnet sich eben aus den beiden angegebenen Feldern. Wird einer der beiden Ausgangszahlenwerte verändert, verändert sich auch der Zahlenwert im Ergebnisfeld.

Durch weitreichende Verwendung dieser Verknüpfungsmöglichkeiten lassen sich in den verfügbaren 255 Zeilen und Spalten komplizierte Berechnungsvorgänge darstellen.

Die Arbeit mit VIEWSHEET zerfällt in zwei Teile:

- Den Entwurf des Arbeitsblattes und
- das Arbeiten mit dem Arbeitsblatt

Entwurf

Durch den Arbeitsblattentwurf wird die Grundlage für die spätere Berechnung der einzelnen Feldinhalte gelegt. Viele Felder werden in der Regel als Eingangszahlenfelder festgelegt, in sie werden beim späteren Arbeiten mit dem Tabellenkalkulationsprogramm die Eingangsgrößen als Zahlenwerte eingegeben.

Um diese Eingaben zu erleichtern können zwischen oder neben diesen Feldern, die Zahlenwerte aufnehmen sollen, auch Textfelder stehen, die lediglich die Funktion von Überschrifts- und Anleitungszeilen haben.

Die sogenannten „Formelfelder" werden beim Entwurf des Arbeitsblattes mit den gewünschten Formeln beschrieben. Diese Formeln können natürlich auch länger sein als die Spaltenbreite eines Feldes angibt. Die Spalte selbst muß später nur den Zahlenwert aufnehmen, der sich aus der Formel errechnet. Die Formel selbst jedoch kann sehr komplex sein.

Wenn ein Arbeitsblatt vorbereitet ist und das Grundschema für die gewünschten Berechnungen durch Festlegung der Formeln in den einzelnen Feldern durchgeführt wurde, kann diese Arbeitsblatteinteilung auch auf Externspeichermedium abgelegt werden. Ein so eingeteiltes Formular kann zu späteren Zeitpunkten immmer wieder aufgerufen und mit neuen Zahlenwerten beschrieben werden.

Kalkulation

Arbeiten mit dem Arbeitsblatt bedeutet die Eingabe und Modifikation von Zahlen in einem bereits festgelegten Schema. Durch die Eingabe von Ausgangsgrößen werden die Zahlenwerte in den durch Formeln definierten Feldern automatisch berechnet. Rechenergebnisse, wie sie Feldern zugeordnet werden, können durch spezielle, beim Entwurf des Arbeitsblattes eingegebene Funktionen auch auf Dateien des Externspeichermediums aufgezeichnet werden. Feldinhalte oder Gruppen von Feldern können durch spezielle Funktionen auch ausgedruckt werden.

Sowohl für den Entwurf des Arbeitsblattes, wie auch für das tatsächliche Arbeiten mit dem Arbeitsblatt, stehen in VIEWSHEET mächtige Funktionen zur Verfügung, die es dem Anwender erleichtern, die richtigen Funktionen aus der möglichen Funktionsvielfalt einzugeben.

VIEWSHEET belegt die Funktionstasten des Mikrocomputers. Mit dem Programm selbst wird eine Schablone mitgeliefert, die die Belegung der Funktionstasten zeigt.

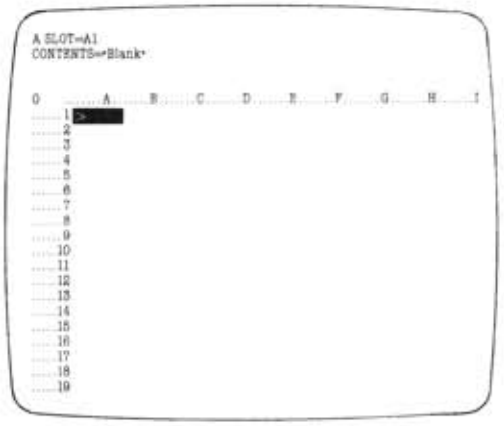
CTRL ►	AUTO ENTRY	DELETE COLUMN	DELETE ROW							
SHIFT ►	RECALCULATE MODE	INSERT COLUMN	INSERT ROW	COLUMN HEADING	ROW HEADING	PROTECT COLUMN	PROTECT ROW	RECALCULATE	JUSTIFY LABEL	DELETE SLOT
ViewSheet	REPLICATE	EDIT WINDOW	SET WINDOW	DELETE END OF LINE	BEGINNING OF LINE	END OF LINE	EDIT SLOT FORMAT	GO TO SLOT	INSERT CHARACTER	DELETE CHARACTER

Nach dem Einschalten des Computers befindet sich VIEWSHEET im sogenannten „COMMAND MODE"

In diesem Betriebszustand können Verwaltungsfunktionen aufgerufen werden, sowie VIEWSHEET auch wieder durch Aufruf des BASIC Systems verlassen werden.

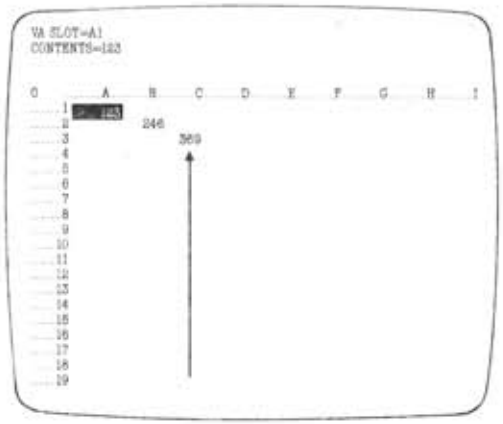
Die wichtigste Betriebsart von VIEWSHEET ist der „ARBEITSBLATTMODE". Durch die Taste „ESCAPE" wird zwischen dem „COMMAND" und „SHEETMODE" hin- und hergeschaltet. Nach Aufruf des Arbeitsblattes wird am Bildschirm die leere Spalten- und Zeileneinteilung dargestellt. Die Zeilen sind numeriert, die Spalten tragen Buchstaben und Buchstabenkombinationen zwischen A und IU.

Die erste Zeile am Bildschirm zeigt die Feldbezeichnung in der sich der Feldcursor gerade befindet. In dieses Feld können Eingaben direkt geschrieben werden. In unserem Beispiel befindet sich der Eingabecursor auf dem Feld A1. Der Cursor kann durch die Cursortasten über den ganzen Bildschirm bewegt werden.



In der zweiten Zeile wird neben dem Wort CONTENTS der Inhalt des durch den Feldcursor aufgesuchten Feldes angezeigt. In unserem Beispiel ist die Zeile unbeschrieben, der Inhalt wird als „BLANK" dargestellt. Wenn Felder aufgesucht werden, deren Inhalt durch eine Formel bestimmt ist, zeigt das Feld im Arbeitsblatt den errechneten Wert, die Zeile zwei jedoch die diesen Feldinhalt bestimmende Formel.

Der Bildschirm kann 19 Zeilen, sowie 9 Spalten der Standardeinteilung des Arbeitsblattes darstellen. Dies ist jedoch nur ein Ausschnitt aus dem gesamten Arbeitsblatt. Wenn durch fortgesetzte Eingabe oder Cursorbewegung eine Zeile bzw. Spalte aufgesucht werden soll die außerhalb des dargestellten Fensters liegt, bewegt sich das Fenster automatisch über das Arbeitsblatt. Durch Drücken der „SHIFT-TASTE", gemeisam mit einer Cursortaste, kann das ganze Fenster über das Arbeitsblatt bewegt werden.



Eingabe von Informationen

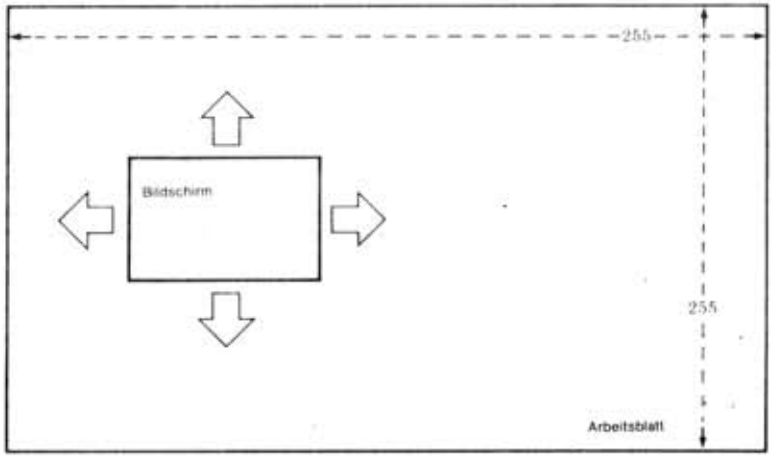
Zahlenwerte

Zahlenwerte können in jedes durch den Feldcursor aufgesuchte Feld einfach durch Eintippen der entsprechenden Zahl und Drücken der Return-Taste eingegeben werden.

Formeln

Auch Formeln können einfach ohne besondere Vorkehrungen in ein Feld eingegeben werden. VIEWSHEET erkennt nach Drücken der Return-Taste, daß es bei der eingegebenen Sequenz durch das Vorhandensein mathematischer Operatoren, wie z. B. dem Plus- oder Malzeichen um keine einfache Zahl, sondern um eine Formel handelt und speichert diese entsprechend ab. Das Feld zeigt den errechneten Wert. Zur Berechnung von Feldinhalten stehen neben den einfachen arithmetischen Grundfunktionen auch komplexe mathematische Funktionen, die Maxima, Minima und Mittelwerte berechnen, Winkelfunktionen und logarithmische Funktionen darstellen können, zur Verfügung. Durch einen entsprechenden Ausdruck kann auch eine Vergleichsentscheidung eingesetzt werden, die einen Feldinhalt von der Erfüllung einer angegebenen Bedingung abhängig macht. Diese Funktion arbeitet ähnlich der BASIC Anweisung IF/THEN.

Alle Zeichenketten, die von VIEWSHEET weder als Zahlen noch als Formeln erkannt werden können, werden als Texte betrachtet. Diese Texte dienen in der Regel als Trennungszeichen und Überschriften.



Unser Beispiel zeigt ein einfaches Arbeitsblatt, in dem die Felder A1 und B2 mit Zahlenwerten beschrieben wurden. Das Feld C3 hingegen enthält eine Formel nach der sein Inhalt aus A1 + B2, also der Summe der beiden eingetragenen Werte berechnet wird.

„WIR LERNEN ITALIENISCH"

FOLGE 4

So! Heute wird einmal so richtig programmiert. Und zwar in BASIC! Sie sind neugierig? Fein! Wir haben in unseren ersten beiden Folgen doch schon die ersten Anweisungen dieser Sprache kennen und lieben gelernt. Es waren die Anweisungen PRINT und INPUT – jene Art und Weise, in der wir miteinander sprechen können. Als Gesprächspartner ist natürlich ER gemeint, unser COMPUTER. Naja, und da wieder einmal ein Urlaub in Italien fällig ist, so habe ich mir gedacht – wenn schon Vokabel trainieren, dann mit IHM. Warum nicht? ER kann mich doch die einzelnen Wörter der Reihe nach abprüfen. Wenn ich's richtig errate, dann werde ich mit einem freundlichen „O.K." belohnt. Im Fehlerfall bessert ER mein unrichtig eingetipptes kulant gegen das richtige aus. Mit seiner elendslangen Geduld, ohne dabei müde zu werden. Ich kann mir das schon gut vorstellen. ER fragt mich nach der Übersetzung des Wortes „warten" und ich antworte lässig mit „aspettare". Nur mit der Zeit kenne ich dann schon die Reihenfolge auswendig. Okay, dann soll ER uns die Vokabel stets neu durchmischen, bevor ER uns prüft. Selbstverständlich wollen wir am Beginn wählen können, ob ER uns Deutsch-Italienisch oder Italienisch-Deutsch prüft. Und eine Mitteilung, wieviele der Vokabel wir bereits richtig erraten haben, das wäre doch auch nicht das Schlechteste.
Na, dann gehen wir die Sache an. Zuerst brauchen wir eine Aufmunterung von IHM. ER soll uns was Nettes am Schirm schreiben, wie z. B.

```
Happy Computing!
Wie wär's mit einem Training in
DEUTSCH - ITALIENISCH - VOKABELN
------------------------
```

Anschließend lassen wir uns ein wenig in die Bedienung des Programms einweisen. Wir wollen das „BEDIENER-Führung" bezeichnen. Ja, natürlich müssen die Vokabeln gemischt werden. Und das sieht dann so aus:

```
Dieses Programm prüft deine
Kenntnisse Italienischer Vokabeln!

Beim Start erhältst du die Frage,
ob du Deutsch oder Italienisch
geprüft werden möchtest.

Nach jedem Wort, ob falsch oder
richtig beantwortet, wird dir
die richtige Antwort angezeigt.

Zu Beginn drücke irgendeine Taste!
```

Fein, versuchen's wir. Wir tippen leicht auf die Leertaste (SPACE-Taste) und der Schirm wechselt seine Ausgabe

```
Es sind derzeit 40 Vokabeln gespeichert!

Nun habe ich die Vokabel gemischt!

Deutsche oder Italienische Wörter?
( 'D' oder 'I' )
```

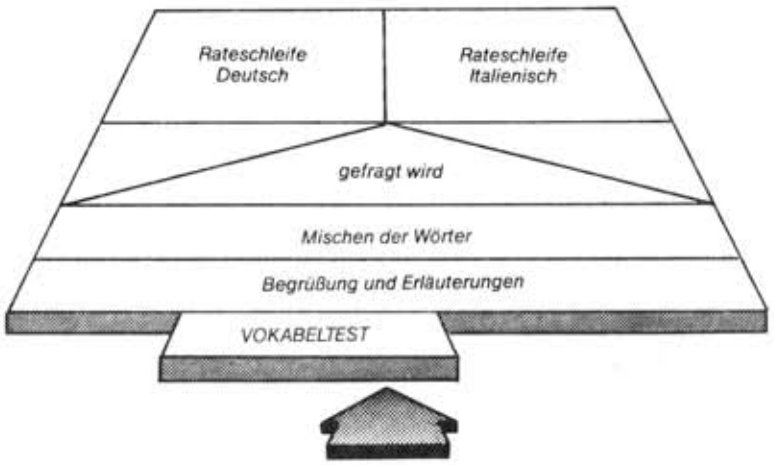
Jetzt wollen wir uns zuerst für die Italienische Gangart entscheiden und tippen ein I auf der Tastatur. Schon wieder wechselt der Schirm

```
DOMENICA

              -->?

40 Rest
```

Na, wissen sie das richtige Wort? Macht nichts. Sie werden's bald können – ER wird sie schon hochtrainieren. Ich tippe für sie SONNTAG ein und sein O.K. freut mich. Ungeduldig, wie so ein COMPUTER nun mal ist, zeigt ER am Schirm schon das nächste Vokabel. Dabei ist ER so nett, uns in der unteren Ecke noch die Anzahl der noch abzuprüfenden Wörter anzuzeigen. So geht's dahin und in Nullkomma-Nix sprechen sie wie ein waschechter Italiener. Nicht schlecht, was? Sie werden nun fragen, wie unser COMPUTER das alles so spielend macht. Tja, das ist eben die Aufgabe unseres Programms. Wie wir uns das vorstellen sollen? Ganz einfach. Wir skizzieren uns den Ablauf in einem Diagramm – nennen wir's einfach STRUKTOGRAMM. Damit können wir das nachempfinden, was unser COMPUTER alles so anstellt.



So, Sie wollen es noch genauer wissen? Nehmen wir die Lupe und sehen wir im „LISTING" nach.

Stark, womit sich der ARME da abplagen muß. Mich interessiert da die Anweisung in der Zeile 2090

```
2090 INPUT GERATEN$
```

Sie kennen diese Anweisung? Damit gibt ER doch sein Fragezeichen aus und wartet auf ihre Eingabe, stimmts? Mit der abschließenden RETURN-Taste legt er das Wort in dem Speicherfach mit der Bezeichnung GERATEN$ ab. Mit der darauffolgenden Programmzeile 2130

```
2130 IF GERATEN$ =DEUT$ (WORT) THEN 2140 ELSE 2170
```

startet unser COMPUTER einen Vergleich zwischen dem von Ihnen eingegebenen Wort und der richtigen Übersetzung, gespeichert in dem Speicherfach mit der Aufschrift DEUT$. Im Falle, daß Sie richtig geraten haben, dann – THEN – macht ER mit der Anweisung in der Programmzeile 2140 weiter und gibt am Bildschirm ein O.K. aus.

```
2140 PRINT " O.K. "
```

Im Fehlerfall – ELSE – setzt ER mit den Anweisungen ab der Zeile 2170 fort. Dabei gibt ER die richtige Übersetzung aus und hofft, daß Sie sich's endlich merken!!!

Variablen

Variablen dienen zum Speichern von Zahlen und Zeichen im Computer. Ein einfaches Beispiel für die Verwendung von Variablen ist die Speicherung von Zwischenergebnissen. Bei der Berechnung von Löhnen in einem Lohnverrechnungsprogramm interessiert z. B.

auch die Gesamtsumme aller Löhne. Dazu muß das Programm zuerst die einzelnen Löhne berechnen und zuletzt die Summe dieser Löhne bilden. Um die Summe bilden zu können, müssen allerdings die Einzellöhne zwischengespeichert werden. Um einzelne, eben variierende Größen im Gedächtnis zu behalten, werden im Computer sogenannte „Variablen" eingesetzt.

Variablen können mit nahezu beliebigen Werten belegt werden, die in diesen Variablen so lange gespeichert bleiben, bis ein neuer Inhalt eingeschrieben wird.

```
10 LET Z=5
20 PRINT Z
30 LET Z=9
40 PRINT Z
```

Die Befehlsfolge in unserem Musterbeispiel zeigt die Verwendung der Variablen Z. Schon durch Verwendung der Variablenbezeichnung wird der zugehörige Speicherplatz für eine Variable reserviert. In der Zeile 10 wird dieser Variablen der Wert 5 zugeordnet, der in der Zeile 20 auch ausgedruckt wird. In der Zeile 30 hingegen ordnen wir Z einen neuen Wert, nämlich 9 zu, dieser Wert wird durch die Zeile 40 ausgedruckt.

Der Computer kann viele verschiedene Variablen gleichzeitig speichern. Diese Variablen können nahezu beliebige Namen haben, also nicht einfach nur Z, wie in unserem Beispiel. Denkbar wäre auch eine Zuordnung.

```
10 MEINALTER = 30
```

Dabei ist zu beachten, daß der Variablenname MEINALTER ohne Leerzeichen zwischen den Wörtern MEIN und ALTER geschrieben wurde. Namen von Variablen unterliegen den folgenden vier Einschränkungen:

- Innerhalb eines Variablennamens darf kein Leerzeichen stehen.
- Das erste Zeichen eines Variablennamens muß immer ein Buchstabe sein, der Rest kann beliebig viele Zahlen enthalten.
- Interpunktionszeichen, das sind Punkt, Komma, Ausrufe- und Fragezeichen, dürfen in einem Variablennamen nicht verwendet werden. Das Unterstreichungssymbol hingegen ist erlaubt.
- Namen von Variablen dürfen nicht mit Schlüsselwörtern aus der BASIC Programmiersprache beginnen.

Beispiele dafür sind die Schlüsselwörter PRINT, oder LET aber auch das Schlüsselwort TO, das häufig versehentlich verwendet wird.

Die folgenden Beispiele von Variablennamen würde ihr ACORN Computer ohne weiteres akzeptieren.

```
LET ALTER=38
LET dieses_Jahr=1983
LET LaengeDESBalkens=18
LET KM_Stand=13280
LET Wert5=16.1
LET Gewicht4=0.00135
LET Huhn2En3=61.6
```

Nicht jedoch die nachstehend angeführten unzulässigen Namen.

```
LET Fussball Ergebnis=3   [Leerzeichen im Namen]
LET Wer?=6                [Fragezeichen]
LET 4.Wert=16.3           [beginnt mit einer Zahl]
LET TORE=23               [beginnt mit TO]
LET PRINT=1234.56         [PRINT ist ein Schlüsselwort aus BASIC]
```

Zeichenkettenvariablen

Obwohl wir „22" üblicherweise eine Zahl nennen, stellt „22" auch ein Zeichen dar. Es besteht darin kein Unterschied zu AA, AB oder XY.

Beim Arbeiten mit Computern ist es sehr wichtig zwischen Zahlen und Zeichen zu unterscheiden. Rechnen kann man nur mit Zahlen, nicht aber mit Zeichen. Ein einziges Symbol kann manchmal als Zahl, manchmal als Zeichen zu verstehen sein. In einer Datumsangabe kommen auch Zahlen vor, sie stellen aber eine Zeichenkette dar, mit der normalerweise nicht gerechnet wird. Ein anderes Beispiel können wir mit der erwähnten Zahl 22 bilden. Wir können die Zahl 22 durch 2 teilen und erhalten 11, solange wir 22 als Zahl betrachten. Wenn wir aber beispielsweise von einem „Gleis 22" sprechen, dann ist es nicht sinnvoll das Zeichen 22 durch 2 zu teilen. Ein Resultat „Gleis 11" gibt es daher nicht.

Der Computer unterscheidet zwischen diesen beiden Arten von Variablen. Variablen die Zahlen enthalten, werden numerische Variablen genannt. Variablen die Zeichenketten aufnehmen können, nennt man „String Variable" oder „Zeichenketten Variable". Sie dienen zum Speichern von Zeichenketten, wie Wörtern. Damit man diese Variablen leicht erkennt, müssen ihre Namen immer mit einem $-Zeichen enden. Nachstehend einige Beispiele für String Variablen.

```
X$ = "HALLO"
TAG$ = "SONNTAG 3 JANUAR"
NAME$ = "ALEX"
```

Ein Beispiel zeigt die Verwendung einer Zeichenkettenvariablen.

```
10 LET X$="HALLO"
20 PRINT X$
```

GOTO – Verzweigung auf eine Programmzeile

Funktion:

Die Verzweigungsanweisung GOTO veranlaßt das Programm einen unbedingten Sprung auf eine angegebene Zeilennummer außerhalb der normalen Reihenfolge im Programm durchzuführen. Die Verarbeitungsreihenfolge ändert sich also.

GOTO ist einfach im Gebrauch, sollte jedoch sehr vorsichtig verwendet werden. Durch zu viele Sprünge im Programm wird dieses sofort sehr unübersichtlich.

Die Acorn Version der Programmiersprache BASIC erlaubt die Anweisung GOTO auch mit Variablen. Das bedeutet die Zeilenummer in der GOTO-Anweisung kann eine Variable sein, die wir auch Zielpunktvariable nennen können.

Ist die Anweisung auf die verzweigt werden soll nicht einfach durch Angabe einer Zeilennummer oder einer entsprechenden Variablen spezifiziert, sondern aufgrund eines mathematischen Ausdruckes zu errechnen, dann muß dieser Ausdruck in Klammern gesetzt werden.

GOTO kann auch als Kommando zum Starten eines Programmes verwendet werden, ohne daß dabei die Variableninhalte zerstört würden.

Beispiele:

```
GOTO 330
100 GOTO XYZ
200 GOTO (START X 6-1)
```

IF THEN – logische Abfrage

Durch die Anweisung IF THEN wird eine Testbedingung gesetzt, von deren Ergebnis die darauffolgende Operation des Computers abhängt.

Eine solche Anweisung ermöglicht es dem Computer bei der Ausführung eines Programms zwischen verschiedenen Möglichkeiten zu wählen, wobei die Entscheidung vom Ergebnis einer Abfrage abhängig ist.

Ein Beispiel:

```
IF H > 3 THEN PRINT "JUHU"
```


In unserem Beispiel hängt die Ausführung der Anweisung PRINT „JUHU" vom Ergebnis des Ausdruckes H>3 ab. Je nachdem ob zum Zeitpunkt der Programmausführung der Inhalt der Variablen H größer als 3 ist, wird entweder die Anweisung PRINT „JUHU" ausgeführt und der Text JUHU auf dem Bildschirm geschrieben, oder aber das Programm setzt mit der nächsten Anweisung fort.

Die BASIC Schlüsselwörter IF und THEN sind Teile der

IF... THEN... ELSE Struktur.

Das Wort ELSE ist jedoch ein nicht obligatorisches Element und kann in Sonderfällen auch entfallen.

In der IF... THEN... ELSE Anweisungsstruktur kann zusätzlich zu der Anweisung die nach THEN steht, sie wird ausgeführt wenn die Bedingung erfüllt ist, eine Anweisung angegeben werden, die dann ausgeführt wird, wenn die Bedingung nicht erfüllt ist. Diese alternative Anweisung wird durch das Schlüsselwort ELSE eingeleitet.

Beispiel:

```
780 IF X =6 THEN PRINT "GUT" ELSE PRINT "SCHLECHT"
```

„DER TOTO-MILLIONÄR"

FOLGE 5

Also, meine COMPUTER-Ecke kann sich sehen lassen. Wobei meine Wunschliste noch vieles aufzeigt. Aber woher nehmen? Da hilft nur ein TOTO-Zwölfer! Ein Blick in die Tipliste läßt meine Hoffnungen auf den absoluten Nullpunkt sinken. Ich bin nun mal kein Fußball-FRAEK, wie soll ich da über Sieg oder Niederlage entscheiden? Chancenlos! Oder doch nicht? Mein ratloser Blick trifft meinen COMPUTER. Irgendwie zwinkert ER mir aufmunternd zu. So, als würde ER zu mir sagen – versuch's mit mir! Warum eigentlich nicht? Wenn ich IHM die letzten Spielergebnisse der involvierten Fußballmanschaften angeben würde, so könnte ER doch den optimalen TOTO-Tip ermitteln. Also, ich würde da einmal das Spielergebnis des letzten Treffens der jeweiligen Paarungen nehmen und dann vielleicht die letzten 5 Ergebnisse der einzelnen Teilnehmer. Das müßte doch reichen, nicht? Gut, aber wie sag' ich's IHM, meinem treuen Gefährten? Nun, ganz einfach. Ich werde im Speicher für jedes Ergebnis bei einem Sieg eine 1, ein X bei einem Unentschieden und eine 2 bei einer Niederlage vorsehen. Doch halt, es muß doch ein Unterschied sein, ob eine Mannschaft einen Kantersieg landet oder es grad mit einem 1:0 schafft. Also werde ich die Tordifferenz ebenfalls in die Speichertabelle aufnehmen. Klingt recht einfach, nur, wie soll ich das programmieren? Ein Blick in die Unterlagen meines COMPUTERS weist mich an, diese DATEN in sogenannten DATA-Zeilen zu verpacken! Gleich ein Versuch: wollen wir die Mannschaft 'M1–1' (1. Mannschaft für den 1. Tip) und ihren Gegner 'M1–2' nennen. Nun wird mit der Anweisung

```
DATA 2, 2
```

festgelegt, daß die letzte Partie von der Mannschaft M1–2 gewonnen wurde und das mit einer Tordifferenz von 2 Toren. Ist doch plausibel, nicht? So, die letzten 5 Spielergebnisse der 1. Mannschaft kommen in eine weitere DATA-Zeile

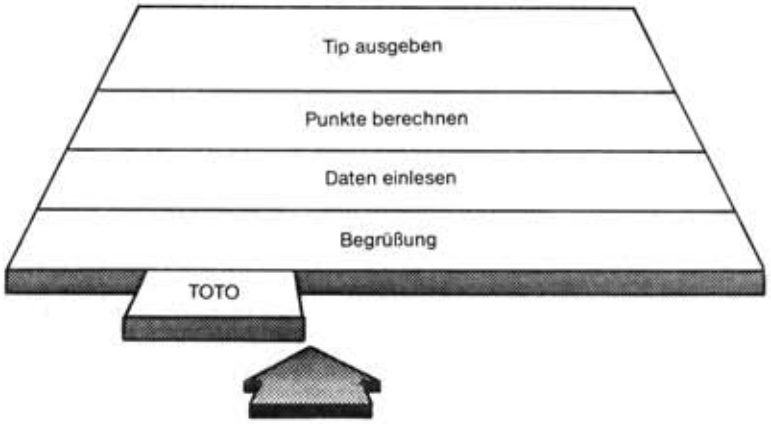
```
DATA M1-1, X,0 1,3 2,1, X,0 1,4
```

Im Klartext heißt das, daß das erste Spiel unentschieden (Tordifferenz 0), das zweite mit 3 Toren Unterschied gewonnen wurde, usw. Na ja, das machen wir jetzt alle 12 Partien unseres Tototips. Da heißt's zwar fleißig tippen – aber was für ein Lohn – ich sehe schon meinen Traumurlaub in der Karibik vor meinen Augen – herrlich!
Doch zurück! Noch ist einiges zu tun. Denn, was soll unser COMPUTER nun mit diesen Daten anfangen? Ich würde vorschlagen, daß ER aus jedem Spiel eine Punkteanzahl ermittelt. 10 Punkte für einen Sieg, 0 für ein Unentschieden und –10 Punkte für eine Niederlage.

ACORN COMPUTER

Für eine Tordifferenz von mehr als 2 Toren gibt's für den Sieger eine 20%-Prämie. Schließlich wollen wir der Mannschaft, die auf eigenem Spielfeld antritt, die Punktezahl um 10% erhöhen (Heimvorteil). Kompliziert? Aber nein, nur ein bißchen Fußball-Arithmetik. Wir wollen aber nicht bürokratisch sein. Ist die Gesamtpunkte-Differenz einer Paarung kleiner als eine „Unentschieden-Schranke" so wird X getippt. Sonst erhält die Mannschaft mit der größeren Punkteanzahl den Sieg zugesprochen.

Jetzt wollen wir unserem COMPUTER einmal über die Schulter schauen. Ich hab mir sagen lassen, ER soll bei seiner Arbeit ungeheuer schnell sein. Stellen sie sich vor, der schuftet im Tausendstel-Sekunden-Rhythmus. Dabei geht IHM nicht einmal die Luft aus. Der macht das glatt Tag und Nacht. Nun ja, jedem sein Plaisierchen. Ich habe seinen Job in einzelne Abschnitte aufgeteilt, die ER jetzt Zug um Zug abarbeiten muß.



Im Detail sieht dann seine Arbeit so aus

Daten Einlesen

PAARUNGEN: Anzahl der zu erratenden Tips
SPIELZAHL: Anzahl der vorhandenen Resultate
PAARUNG = 1
  Lesen des letzten Resultats
  Lesen der letzten Tordifferenz
  MANNSCHAFT = 1
    Lesen des Mannschaftsnamens
    SPIEL = 1
      Lesen des Tips und der Tordifferenz
    solange SPIEL<SPIELANZAHL SPIEL = SPIEL + 1
  solange MANNSCHAFT<2 MANNSCHAFT = MANNSCHAFT + 1
solange PAARUNG<PAARUNGEN PAARUNG = PAARUNG + 1

PUNKTE BERECHNEN

PAARUNG = 1
  GEGNER = 1
    PUNKTEBERECHNUNG
    Anfangswert der Punktezahlen festlegen
  solange GEGNER<2 GEGNER = GEGNER + 1
  MANNSCHAFT = 1
    SPIEL = 1
      PUNKTEBERECHNUNG
      Punktezahlen erhöhen
        MANNSCHAFT = 1
        ja | nein
        Heimvorteil
    solange SPIEL<SPIELZAHL SPIEL = SPIEL + 1
  solange MANNSCHAFT<2 MANNSCHAFT = MANNSCHAFT + 1
solange PAARUNG< PAARUNGEN = PAARUNG + 1

TIP AUSGEBEN

Titel schreiben und unterstreichen
PAARUNG = 1
  Ausgabe Mannschaftsnamen
  Ausgabe Punktezahlen
    /Punktedifferenz/< Schwelle für Unentschieden
    ja | nein
    Punkte 1>Punkte 2
    ja | nein
    Ausgabe „X" | Ausgabe „1" | Ausgabe „2"
solange PAARUNG<PAARUNGEN PAARUNG = PAARUNG + 1

Gewaltig, was? Jetzt wird's aber Zeit, daß wir unser Glück machen. Wir starten unser Programm und am Schirm erscheint

```
TOTO

ICH ERSTELLE FÜR DICH DEN TOTO-TIP
FÜR DIE NÄCHSTE SPIELRUNDE

DATEN BEREITS IN DATA-ZEILEN EINGEGEBEN

BITTE WARTEN, ICH ARBEITE...
```

Höflich ist ER auch – ich bin schon sehr aufgeregt – da, plötzlich löscht sich der Schirm und es erscheint der Millionentip!

MEIN TOTO-TIP LAUTET:

TEAM 1	: TEAM 2	PUNKTE	TIP
M1-1	:M1-2	72:-60	1
M2-1	:M2-2	2:-71	1
M3-1	:M3-2	0: 0	X
M4-1	:M4-2	45:-38	1
M5-1	:M5-2	-12: 22	2
M6-1	:M6-2	60: 50	X
M7-1	:M7-2	60: 70	X
M8-1	:M8-2	0: 0	X
M9-1	:M9-2	72:-60	1
M10-1	:M10-2	12:-34	1
M11-1	:M11-2	44: 02	1
M12-1	:M12-2	72: 66	2

Doch was hör' ich da im Radio? Die Ergebnisse der Meisterschaft: 1. Spiel – Tip 2! Oh je, aus der Traum.

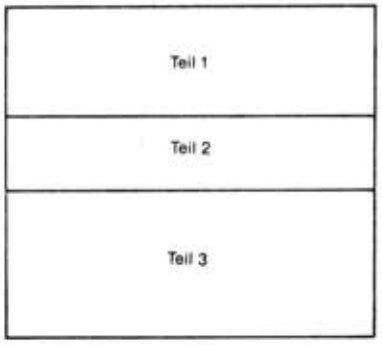
Programmdarstellung

Sowohl bei der Planung eines Programmes vor der Programmierung, wie auch zur Dokumentation eines fertigen Programmes, werden vorzugsweise graphische Methoden verwendet. Mit Hilfe von graphischen Symbolen kann ein Programmablauf sehr übersichtlich dargestellt werden. Um eine überregionale Lesbarkeit solcher Darstellungen zu gewährleisten, werden genormte Symbole und Anordnungen zur Programmdarstellung verwendet. Die am häufigsten verwendeten Darstellungsmethoden sind das Flußdiagramm und das Struktogramm. In beiden Darstellungsarten werden Aktionen, die vom Computer ausgeführt werden, in Symbole gestellt, die hintereinander gereiht werden. Dies entspricht dem Prinzip der sequentiellen Durchführung dieser Aktionen durch den Computer selbst.

Beim Flußdiagramm werden die Symbole nebeneinander bzw. untereinander niedergeschrieben und in der Reihenfolge ihrer Durchführung durch Striche verbunden. Durch die Tatsache, daß auch weit auseinanderliegende Symbole durch Striche miteinander verbunden werden können und so eine Ausführung des zweiten Symbols sofort nach Ausführung des ersten Symboles dargestellt werden kann, werden Flußdiagramme leicht unübersichtlich.

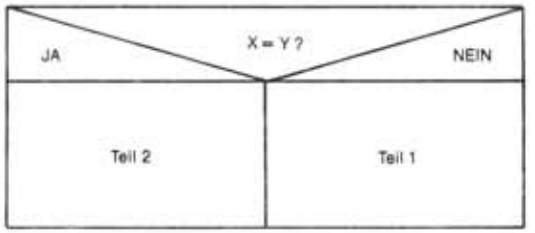
Bei Struktogrammen werden die graphischen Symbole direkt hintereinander und ineinander gesetzt.

Die einzelnen graphischen Symbole können Aktionen beliebiger Mächtigkeit enthalten, es kann sich dabei entweder um einen einzelnen Befehl handeln, der deswegen in ein eigenes Symbol gesetzt wird weil er sehr wichtig ist, oder aber auch einen ganzen Programmteil. Je nachdem, ob das Flußdiagramm oder Struktogramm eine Übersichtsdarstellung oder eine Feinauflösung zeigt, ist die Beschreibung der Aktionen in den einzelnen Symbolen detailliert oder nur übersichtlich.

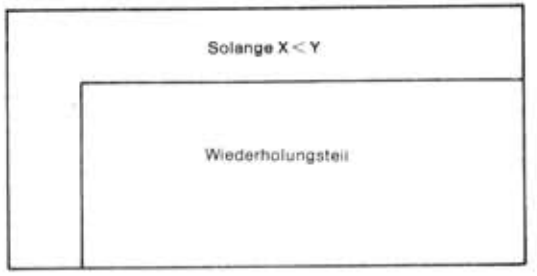


Für besonders wichtige Aktionen sind bestimmte graphische Symbole reserviert.

Ein einfaches Kästchen bzw. hintereinandergestellte Rechtecke bilden eine Sequenz. Die in den Rechtecken enthaltenen Aktionen werden jeweils einmal von oben nach unten ausgeführt.

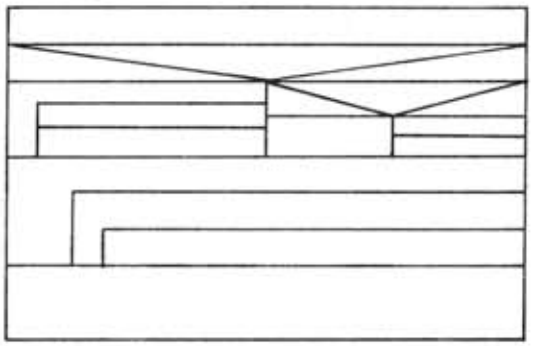


Die Auswahlstruktur erlaubt die graphische Darstellung eines Entscheidungsvorganges. Je nachdem, ob die in die Auswahlentscheidung gestellte Bedingung erfüllt ist, wird entweder der linke Teil der innenliegenden Struktur, oder der rechte Teil durchlaufen.



Die Wiederholungsstruktur erlaubt die graphische Darstellung von Schleifen. In den äußeren Teil der Struktur wird die Wiederholungsbedingung geschrieben, in den inneren Teil, den Wiederholungsteil die Aktionen die, solange die außenliegenden Bedingungen erfüllt sind, wiederholt durchlaufen werden.

Die Aneinanderreihung von Strukturen bilden ein Struktogramm.



Bereiche – Arrays

Als Bereiche, auch Arrays oder Felder genannt, bezeichnet man eine Gruppe von Variablen. Jede dieser Variablen ist im Prinzip von den anderen Variablen der Gruppe unabhängig. Durch die Gesetzmäßigkeit in der Namensbildung können allerdings verschiedene Operationen entweder auf den gesamten Bereich, also auf die Gruppe von Variablen bezogen werden, oder eine Variable aus diesem Bereich gezielt ausgewählt werden.

Namen von Bereichsvariablen werden dadurch gebildet, daß einem normalen Variablennamen in Klammer ein sogenannter Index nachgestellt wird.

Ein Beispiel für eine Bereichsvariable ist:

```
N$(5)
```

Die dargestellte Bereichsvariable ist eine, in unserem Fall die fünfte Variable aus dem Bereich N$. Das bedeutet, daß es auch noch N$(1), N$(2), ... N$(4) geben muß.

Bereiche sind dadurch sehr vorteilhaft zu verwenden, daß ihr Index selbst wiederum durch eine Variable gebildet werden kann. Durch den Befehl

```
30 PRINT N$(I)
```

kann jedes beliebige Element des Bereiches N$ angesprochen werden, je nachdem welchen Inhalt die Variable I zum Zeitpunkt der Ausführung der Befehlszeile 30 hat.

Durch einen eigenen Befehl muß dem Computer mitgeteilt werden, wie groß ein Bereich sein wird, bevor man einzelne Elemente aus diesem Bereich verwendet. Dies geschieht mit der DIM-Anweisung. Nach der BASIC-Anweisung DIM wird der Name des Bereiches und in Klammer nachgestellt der größte Index, also die Anzahl der verwendeten Bereichselemente, angegeben.

Ein Beispiel dafür ist:

```
10 DIM N$(50)
```

```
 10 DIM N$(5)
 20 N$(1)="MITTELMAIR"
 30 N$(2)"SCHULZE"
 40 N$(3)"MUELLER"
 50 N$(4)"SCHMIDT"
 60 N$(5)"BERGER"
 70 PRINT "WELCHEN EINTRAG WOLLEN SIE"
 80 INPUT X
 90 PRINT N$(X)
100 GOTO 70
```

In unserem Beispiel zeigen wir die Verwendung eines Feldes N$. In der Programmzeile 80 kann eingegeben werden, welche der fünf Bereichsvariablen in der Zeile 90 ausgedruckt werden soll.

Schleifen – FOR... NEXT

Mit der Schleifenstruktur wird der Computer dazu veranlaßt, einen Teil eines Programmes, er wird als Schleifeninhalt bezeichnet, beliebig oft durchlaufen.

Eine Schleifenstruktur besteht aus drei Teilen, die anhand eines Beispiels gezeigt werden sollen.

```
10 FOR X =8 TO 20 STEP 2.5 - Schleifenkopf
200 PRINT X, X +X          - Schleifenkörper
30 NEXT X                  - Schleifenende
>RUN          8       16
            10.5      21
              13      26
            15.5      31
              18      36
```

Der Schleifenkopf dient zur Definition der Bedingung die angibt wie oft der Schleifenkörper durchlaufen wird.

Im Schleifenkopf wird eine sogenannte Laufvariable, sie kann einen beliebigen Variablen-

namen tragen, verwendet. Bei Eintritt in die Schleife wird die Laufvariable an den angegebenen Anfangswert, in unserem Beispiel 8 gesetzt und danach der Schleifenkörper durchlaufen.

Der Schleifenkörper kann aus beliebig vielen Zeilen bestehen.

Das Ende des Schleifenkörpers wird durch die Anweisung NEXT festgelegt. Wenn die Programmausführung bei der Anweisung NEXT angelangt ist, geht das Programm zum Schleifenkopf zurück. Dort wird die Laufvariable um die angegebene Schrittweite, in unserem Beispiel 2,5 erhöht und der Schleifenkörper noch einmal durchlaufen. Dies wird solange, eben in Form einer Schleife durchgeführt, bis die Laufvariable den angegebenen Endwert, in unserem Beispiel 20, überschritten hat.

ACORN COMPUTER