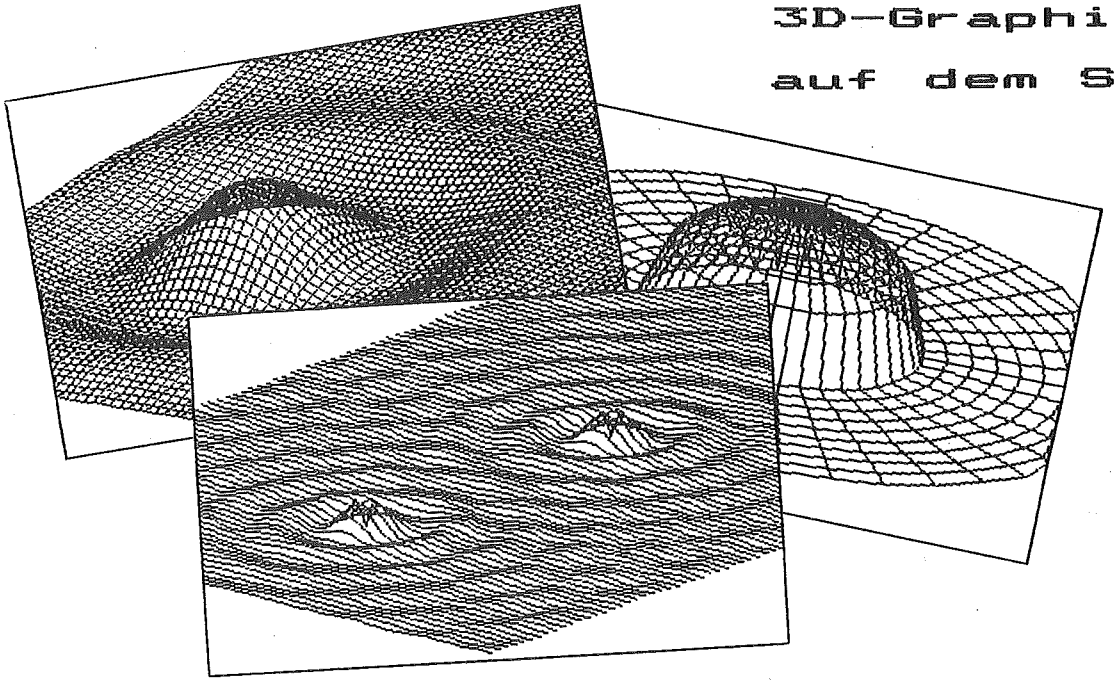
SVI JOURNAL

Die Zeitschrift des Spectra-Video-Club Austria



3D-Graphik auf dem SVI

FEBRUAR 1985

Datenstrukturen und ihre Programmierung

INHALT

Heft 2/85

S 15.-

Lieber SVI-Journal-Leser!

Es gibt wieder einige Neuigkeiten in unserem Club! Zum Ersten: Wir haben schon die "Hundert-Mitgliedergrenze" überwunden. Ab nun hat unser Club über 100 Mitglieder. Dies ist natürlich erfreulich, doch viele Leute werfen manchmal auch kleine Probleme auf. Aus diesem Grund haben wir uns zu folgenden Neuerungen "durchgerungen":

In einer der nächsten Ausgaben des SVI-Journals wollen wir eine Liste aller Clubmitglieder beilegen. Selbstverständlich werden die Listen nur den Heften beigelegt, die an Clubmitglieder gehen. "Außenstehende" bekommen von uns weiterhin keine Daten über Mitglieder. Es hat sich nämlich gezeigt, daß viele Personen von im SVI-Journal veröffentlichten Programmen begeistert sind. Doch wie bekommen sie nun das Programm? Sie müssen sich mühsam durchfragen, wer denn der Herr X sei, wo er zu erreichen sei und wie man zu dem und dem Programm kommen könne. Deshalb wollen wir diese Listen verschicken!

Weiters sind wir der Meinung, daß zwei Clubabende pro Monat etwas wenig sind. Um mehr Mitgliedern die Möglichkeit zu geben, diesen Abenden beizuwohnen, werden wir die Anzahl der ("ordentlichen") Clubabende erhöhen. Ab nun soll jede Woche einer stattfinden. Dies hat den Vorteil, daß mehr Leute Gelegenheit haben, am Computer zu arbeiten. Es hat sich nämlich gezeigt, daß sich einige "enttäuschte" Clubabend-Besucher vom Treffen zurückgezogen haben. Sie haben ihr Heil außerhalb der Clubzeiten gesucht und während der Geschäftszeit der Firma Wehsner die Computer "belagert". Wir wollen nun auch diese "diskriminierten" Clubmitglieder zufriedenstellen und erhöhen die Anzahl der Clubabende. Selbstverständlich dürfen Mitglieder noch immer ihre diversen Ausdrucke auch während der Geschäftszeit der Firma Wehsner herstellen, aber stundenlang vorm Computer sitzen, muß denn das sein?

Kommen wir wieder zu etwas Anderem. Wir haben selbstverständlich schon nationale und internationale Fäden geknüpft und die SVI-Bewegungen im In- und Ausland verfolgt. Nun veröffentlichen wir auch die Adressen von anderen Clubs. Vielleicht bilden sich auch hier rege Informationsaustäusche.

Weiters haben wir einige neue Programme auf Lager. Eines nennt sich SWEEP, ist auf jeder neuen SVI-CP/M-Systemdiskette zu finden und wird im Inneren dieses Heftes erklärt.

Spectravideo hat durch den speziellen Aufbau der RAM-Speicher dafür gesorgt, daß eine neue Art der Speichererweiterung ermöglicht wird, die sogenannte "Software-Erweiterung". Richtig gelesen, man braucht keine teure Hardware mehr, sondern sättigt anhand von Programmen seinen Mehrbedarf an Speicher.

Scherz beiseite, natürlich ist hier ein Kniff. Bis jetzt wurden ja alle SVI-Besitzer damit vertröstet, daß sie zwar 80KByte Speicher hätten, daß fürs BASIC jedoch nur knapp 29K zur Verfügung ständen. Einige findige Leute haben nun Programme erstellt, die die zweiten 32K einsatzfähig machen. Durch Benutzung eines Befehls (entweder CMD oder XSWITCH oder andere) wird der Inhalt der beiden Banken vertauscht. Eines dieser Programme kam schon im vorigen Heft - Kapitel Befehlserweiterung - heraus. Die genaue

```
*****************************************
*                                       *
*  Die nächsten Clubabende:             *
*                                       *
*   Mi, dem 27. Februar 1985, ab 19 Uhr *
*   Di, dem  5. März    1985, ab 19 Uhr *
*   Sa, dem 16. März    1985, ab 17 Uhr *
*   Mi, dem 20. März    1985, ab 19 Uhr *
*   Di, dem 26. März    1985, ab 19 Uhr *
*                                       *
*  wie immer im Computer-Studio,        *
*  1040 Wien, Paniglgasse 18-20.        *
*  Nichtmitglieder sind willkommen.     *
*  Ende jeweils ca. 22 Uhr!             *
*                                       *
*  Aktivitäten an den Clubabenden:      *
*  Arbeiten an Spectravideo-Systemen,   *
*  Informationsaustausch zwischen Club- *
*  mitgliedern, Gelegenheit zum Aus-    *
*  drucken von Programmlistings.        *
*                                       *
*  Telefonische Auskünfte über den Club *
*  und seine Aktivitäten erhalten Sie   *
*  unter der Telefonnummer 65 88 93.    *
*                                       *
*****************************************
```

Funktionsbeschreibung dieser Art von Programm folgt in einem der nächsten Hefte. Also, ab nun haben SVI-Computer nicht mehr 29199 Bytes free, sondern 58398 (ungefähr 2*300 Bytes müssen noch für das Programm selber abgezogen werden).

Im übrigen haben wir uns auch beim SVI-Journal wieder unsere Gedanken gemacht. Die Informationsflut holt auch uns ein. Deshalb haben wir unsere Zeitschrift um vier Seiten erweitert. Wir betrachten es als Kundenservice, daß der Preis nicht gleichzeitig ebenfalls gestiegen ist.

Wie vielleicht schon bemerkt, haben wir auch eine neue Serie angefangen. Die wichtigsten Datenstrukuren und ihre Programmierung werden in der gleichnamigen Serie erklärt.

Ihr SVI-Journal-Chefredakteur Gerhard Fally!

```
*****************************************
*                                       *
*     Clubadressen im In- und Ausland!  *
*                                       *
*    Österreich:                        *
*      Computer Club SV-328             *
*      Weizengasse 2                    *
*      A-4063 Hörsching                 *
*                                       *
*    Niederlande:                       *
*      SpectraVideo Computer Users Club *
*      Postbus 202                      *
*      NL-2300 NA LEIDEN                *
*                                       *
*    Finnland:                          *
*      SVI-Club Finnland                *
*      p.A. N.Kelzenberg / Tenavatie 5 D*
*      SF-00760 Helsinki                *
*                                       *
*    Kanada:                            *
*      SV-ACCESS                        *
*      P.O.Box 5524, Station A,         *
*      Toronto, Ontario,                *
*      CANADA M5W 1N7                   *
*                                       *
*****************************************
```

In dieser Folge warten wieder interessante Befehle auf uns. Neben dem Befehl MID$ (nicht zu verwechseln mit der gleichnamigen Funktion!) lernen wir auch REM, SWAP, TRON/TROFF und andere Anweisungen kennen.

Wir haben schon einmal mit einem MID$ gearbeitet. Es gibt nämlich auch eine Funktion gleichen Namens. Während die Funktion allerdings den Mittelteil eines Strings in eine andere Variable kopiert, tauscht das Kommando MID$ den Mittelteil durch einen anderen aus. Dadurch ergeben sich auch leichte Änderungen in der Syntax des Befehles:

MID$(String, Beginn, Anzahl)=Ausdruck

So sieht die vollständige Form der Anweisung aus. "Anzahl" darf auch weggelassen werden.

Die Funktion ist ganz einfach. Im angegebenen String wird ab dem Wert "Beginn" die Zeichenkette "Ausdruck" eingesetzt. Dabei wird der Mittelteil des ursprünglichen Strings überschrieben (also nicht eingefügt, sondern ersetzt).

Natürlich gibt es auch hier wieder Sonderfälle: Wenn die Zahl der Zeichen von "Ausdruck" größer ist, als im String untergebracht werden kann, so wird der Rest abgeschnitten. Deshalb wird der ursprüngliche String nie länger, als er vorher war. Man kann jedoch mit "Anzahl" die Zahl der ausgetauschten Zeichen ebenfalls beschränken. In diesem Fall werden dann nicht mehr als "Anzahl" Zeichen übernommen.

```
10 INPUT A$
20 INPUT B$
30 MID$(A$,5)=B$
40 PRINT A$:RUN
```

Uns ist ja schon seit geraumer Zeit bekannt, daß es im BASIC GOTO und GOSUB gibt. Aber es gibt nicht nur diese einfachen Sprünge, sondern auch einen weit komplexeren Sprungbefehl. Er nennt sich "ON...GOTO (GOSUB)...". Bei diesem Befehl geben wir eine natürliche Zahl an (also 1,2,3,4,5,..), der Computer wählt dann analog zu der Zahl eine der hinter GOTO/GOSUB gelisteten Sprungadressen. Ist die Zahl eins, wird die erste Adresse angesprungen, ist sie zwei, die zweite und so weiter.

Die Zahl kann natürlich wieder auch ein mathematischer Ausdruck sein, eventuell SQR(4). Der Computer schneidet vom Rechenergebnis alle Dezimalstellen ab. So kommt er auf eine natürliche Zahl, auch wenn SQR(8) hinter ON gestellt wird.

Wie reagiert der Computer nun bei einer Fehlbedienung der Anweisung? Sind weniger Adressen vorhanden, als für eine Zahl gebraucht würde, so nimmt sich der Computer den nächstfolgenden Befehl vor. Das Gleiche passiert bei einer Null als Rechenergebnis. Weniger glimpflich verläuft die Sache, wenn der Wert negativ oder größer als 255 ist. Hier steigt der Computer mit "Illegal function call" aus dem Programm aus. Einen lustigen Effekt erzielt man, wenn man zwischen zwei Beistrichen keine Zahl einfügt. Der Computer zählt dann dieses "Nichts" zwischen den Kommas als Sprungadresse, wenn er eine andere Zahl anspringt. Kommt die Wahl jedoch auf dem Leerraum zu stehen, wirft er einem einen "Syntax error" an den Kopf. Beispiele folgen natürlich auch wieder:

Der ganz normale Fall einer "ON...GOTO..."-Anweisung sieht so aus:

```
10 INPUT "ZAHL";A
20 IF A>255 OR A<0THENPRINT"FALSCHE EINGABE
":GOTO 10
30 ON A GOTO 40,50,60,70,80,90
35 IF A=0 THENPRINT A"=0"ELSEPRINT A"IST GR
OESSER ODER GLEICH 7"
36 GOTO 10
40 PRINT A"IST 1":GOTO10
50 PRINT A"IST 2":GOTO10
60 PRINT A"IST 3":GOTO10
70 PRINT A"IST 4":GOTO10
80 PRINT A"IST 5":GOOT10
90 PRINT A"IST 6":GOTO10
```

Dieses Programm hat an sich wieder wenig Sinn, es zeigt nur schön die Selektion der einzelnen Werte durch Anspringen verschiedener Zeilen.

Nun kann es einem ohne Weiteres passieren, daß man, wie oben erwähnt, Formeln einsetzen muß. Leicht gemacht, wir ändern obiges Programm wie folgt:

```
15 A=INT(SQR(A))
```

Nun kann man sich fragen, daß wir dieses INT(SQR(A)) ja auch hinter ON schreiben könnten. Dies geht aber nur solange reibungslos, solange der Term keine negativen Werte annimmt. Bei INT(SQR(A)) ist dies gegeben. Wir hätten also im ON...GOTO... die Formel ohne Bedenken einsetzen können. Wir haben trotzdem die Form einer Zuweisung gewählt, da sonst die Anzeige PRINT A"IST X" völlig falsch gewesen wäre.

Wenn wir nun starten, sehen wir die gewünschten Effekte.

```
ZAHL? 4
2 IST 2
ZAHL? 8
2 IST 2
ZAHL? 36
6 IST 6
ZAHL? 100000000000
FALSCHE EINGABE
ZAHL? 100
10 IST GROESSER ODER GLEICH 7
ZAHL?
```

Denken wir noch einen Schritt weiter. Es kann bei mehr oder weniger umfangreichen Formeln auch vorkommen, daß man zum Beispiel weiß, daß die Zahlen 1,2,3,5,6 und 9 vorkommen können. 4,7 und 8 sind aber gänzlich durch das Programm ausgeschlossen. In so einem Fall wäre es Speichervergeudung, wenn man bei den Werten 4,7 und 8 Phantasieadressen einsetzen würde. In so einem Fall läßt man diesen Platz leer. Wir simulieren dies in unserem Programm durch Wegnahme von der Zahl Drei. Dies machen wir folgendermaßen:

```
25 IF A=3 THEN GOTO 10
30 ON A GOTO 40,50,,70,80,90
```

Der Betrieb des Programms ist solange un-

verändert zu dem vorigen Versuch, bis drei eingegeben wird. Hier wiederholt der Computer einfach die Eingabe.

In den oben genannten Versuchen läßt sich übrigens ebenso GOSUB statt GOTO einsetzen.

Um einen fließenden Übergang zum nächsten Befehl zu haben, werden wir uns überlegen, wie wir den Leuten, die unsere Programme später ansehen werden, das Lesen dieser erleichtern können. Nichts einfacher als das! Mit REM dürfen beliebige Informationen im Programm untergebracht werden. Der Computer überspringt diese Zeile einfach. Die Informationen gehen ihn ja nichts an. Wir fügen in unserem Programm auch einige REM-Zeilen ein. Hinter REM darf alles stehen, der Interpreter wird den Befehl nie beachten.

```
 1 REM *************************
 2 REM *                       *
 3 REM *    ON...GOTO-DEMO     *
 4 REM *   XAVER BIRNSTINGEL   *
 5 REM *     VERSION 1.2       *
 6 REM *      FEBR. 1985       *
 7 REM *                       *
 8 REM *************************
 9 REM *** EINGABE UND UEBERPRUEFUNG ***
27 REM *** EIGENTLICHER SPRUNG ***
37 REM *** EINSPRUNGTABELLE ***
```

Mit diesen paar Informationen haben wir folgendes erreicht:

Der Bediener weiß, wer wann welches Programm gemacht hat, beziehungsweise, welche Version es schon ist.

Der Bediener bekommt die einzelnen Abschnitte des Programms gegliedert und erklärt. Er weiß genau, wann welcher Programmteil beginnt und aufhört.

Die drei Sterne sind lediglich eine Hilfe, die Dokumentation im Buchstabenwirrwar eines Programms besser auszumachen. Es ist die Aufgabe des Programmierers, nach eigenem Ermessen solche REM-Zeilen zu setzen. Will jemand ein "gepacktes" (möglichst geringer Speicherverbrauch) Programm schreiben, dann wird er auf REM-Zeilen verzichten. Bei Programmen, die zur Einschulung von Anfängern dienen, wird man REM-Zeilen in Mengen verwenden. Notwendig sind sie jedoch nicht unbedingt.

Wir steigen aber schon auf den nächsten Befehl um. SWAP ist äußerst interessant. Die Arbeitsweise ist schnell umrissen. SWAP vertauscht die Inhalte zweier angegebener Variablen. Zugelassen sind alle Variablentypen, doch müssen beide verwendete Variablen vom gleichen Typ sein. SWAP A$, B! ist zum Beispiel verboten.

Auch hier gibt es wieder ein paar Beispiele:

```
10 INPUT A
20 INPUT B
30 PRINT "EINGEGEBENE ZAHLEN:"A;B
40 SWAP A,B
50 PRINT "NACH SWAP:"A;B
60 GOTO 10
```

Nach RUN sehen wir folgendes:

```
?45
?65
EINGEGEBENE ZAHLEN: 45  65
NACH SWAP: 65  45
?
```

Das Gleiche kann man nun auch mit Zeichenketten (Strings genannt) tun. Hier möchte ich auf die Ausgabe 1/84 unseres SVI-Journals hinweisen, das damals mit SWAP die sogenannten Bubble-Sort-Programme auf die Beine gestellt hatte. Mit diesen Entwicklungen kann man Strings nach dem Alphabet sortieren.

Nebenbei: Die "Bubble-Sorts" sind ziemlich langsam, deshalb hat ein Clubmitglied nun ein schnelleres BASIC-Programm zum Sortieren entworfen. Selbstverständlich ist das SVI-Journal ganz aktuell dabei und hat dieses Programm auch schon in dieser Ausgabe veröffentlicht.

Als letzten Befehl der "normalen" Anweisungen lernen wir TRON/TROFF (TRace ON/OFF) kennen. Mit dieser Anweisung wird es möglich, sich die Zeilennummer auflisten zu lassen, die der Computer gerade abarbeitet. Man kann dann den Arbeitslauf des Computers verfolgen und bei der Suche nach Fehlern effizienter arbeiten.

Eingeschaltet wird die Anzeige mit TRON, ausgeschaltet mit TROFF.

Wir können dies bei unserem Programm von vorhin ausprobieren. In eckigen Klammern wird die jeweilige Zeilennummer geschrieben, die anderen Bildschirmausgaben werden unverändert ausgeführt.

Kommen wir zu einem neuen Kapitel, den Ein-/Ausgabe-Anweisungen. Hier gibt es als ersten den BEEP-Befehl:

Bei Ausführung dieser Anweisung ertönt ein kurzer Ton, mit einer Frequenz von ungefähr 1000 Hz. Er hört nach einer Viertelsekunde wieder auf. Dieser Ton wird auch durch CHR$(7) oder CTRL-G erzeugt. Besonders abwechslungsreich ist dieser Befehl nicht, ja man kann nicht einmal viel falsch machen. Außer einem "ordinären" "Syntax error" läßt sich keine Fehlermeldung erzeugen.

Interessanter wird es bei BLOAD und BSAVE. Diese beiden Befehle dienen zum Abspeichern oder Lesen von Maschincodeprogrammen oder sonstigen Datenansammlungen auf Kassette oder Diskette. Wird eine Eins oder eine Zwei mit Doppelpunkt vor den Dateinamen gesetzt, werden die Disks angesprochen, andernfalls kommt der Kassettenrekorder zum Handkuß.

Das Format des BSAVE:

BSAVE Dateiname, Anfang, Ende, Start

Der Dateiname dürfte klar sein. "Anfang" gibt die Adresse des ersten abzuspeichernden Bytes an. "Ende" wird für das letzte gebrauchte Byte verwendet. Als Option gibt es noch "Start". Der Computer startet dann bei "BLOAD Dateiname,R" ab der angegebenen Startadresse.

BLOAD lädt abgespeicherte Werte wieder in den Computer. Wieder folgt hinter dem Befehlswort der Dateiname. Danach kann man mit ",R" angeben, ob das MC-Programm gleich gestartet werden soll. Wenn ja, dann wird die Anfangsadresse als Startadresse genommen, außer, es wurde schon beim Abspeichern ein "Start" definiert.

Das deutsche Handbuch verspricht übrigens bei diesem Befehl etwas zu viel. So wird auch erklärt, daß man eine Adresse angeben darf, ab der die Datei in den Speicher transferiert wird. Leider hält sich der Computer nicht an diese schöne Beschreibung. Wenn man eine Startadresse angibt, ist er sichtlich irritiert und behält die Datei auf Band. Sie wird weder auf den "Originalplatz" noch zur eingetippten Adresse überspielt.

In der nächsten Folge setzen wir mit den Lade- und Speicherbefehlen fort.

Assemblerfortsetzungskurs Nr.:008

In einer der ersten Ausgaben wurden die Register des Z-80 behandelt. Dabei sind ebenfalls die Vorzüge gegenüber den anderen 8 Bit-Prozessoren gezeigt worden, die sich nicht nur durch die große Anzahl der Register hervorheben, sondern auch durch die beiden Indexregister IX und IY. Diese beiden recht interessanten Register wurden bisher noch nicht besprochen, was jetzt aber nachgeholt werden soll.

Die beiden Register sind vollwertige 16 Bit-Register, mit denen man einfach auf größere Blöcke von Daten zugreifen kann. Sie sind genauso zu handhaben wie das Registerpaar HL. Bei der indirekten Adressierung ist es jedoch möglich, ein zusätzliches Byte anzugeben, welches zu der angegebenen Adresse dazuaddiert wird. Zur Wiederholung wird noch einmal erklärt, was die indirekte Adressierung ist: Bei der indirekten Adressierung, wird keine feste Adresse angegeben (zum Beispiel: LD A,(10A5H)), sondern die Adresse durch ein Registerpaar ersetzt, in dem sich eine Adresse befindet (zum Beispiel: LD C,(HL). Hier beinhaltet HL die Adresse).

Das zusätzliche Byte wird Offsetbyte genannt. Das Offsetbyte ist im Zweierkomplement anzugeben. Wie nun zum Beispiel das Register D aus (IX) zu laden ist, sieht folgendermaßen aus: LD D,(IX+00). "00" ist hierbei das Offsetbyte, das zu IX addiert wird, bevor D daraus geladen wird. LD B,(IX+5DH) bewirkt, daß B aus IX+5DH geladen wird. Da die Zahlen im Zweierkomplement anzugeben sind, ist natürlich auch folgendes möglich: LD C,(IX-45H). Überall, wo indirekt aus den Indexregistern geladen werden soll, muß man ein Offsetbyte angeben. Die gerade angeführten Beispiele mit dem IX Register sind natürlich ebenso mit dem IY Register durchführbar.

Gehen wir doch gleich einmal zur Praxis über und erstellen ein recht simples und leicht zu begreifendes Programm.

```
100 REM           ORG C800h
110 REM           ;
120 ' LOCATE : Diese Routine positioniert
130 ' den Cursor an der angegebenen Stelle,
140 ' wobei [H] die Spalte und [L] die Zeile
150 ' angibt.
160 ' CUR_POS enthaelt die aktuelle Cursor-
170 ' position.
180 REM LOCATE    EQU  393EH
190 REM CUR_POS   EQU  FA03H
200 REM           ;
210 REM           LD   HL,(CUR_POS)
220 REM           PUSH HL ;Rette alte
230 REM                   ;Cursorpositon.
240 ' IX zeigt auf die Tabelle TAB_CHR,
250 ' die aus 3 Eintraegen besteht. Jede
260 ' Eintragung enthaelt 3 Informationen:
270 ' Die ersten 2 Bytes geben die Position
280 ' an, das dritte Byte gibt das auszu-
290 ' gebende Zeichen an.
300 REM           LD   IX,TAB_CHR
310 REM           LD   A,(LEN)
320 REM           LD   B,A
330 REM           ;
340 REM LOOP      LD   H,(IX+00);lies Spalte
350 REM           LD   L,(IX+01);lies Zeile
360 REM           CALL LOCATE ;positionieren
370 REM           LD   A,(IX+02);lies Zeichen
380 REM           RST  18H ;Ausgabe
390 ' Erhoehe den Zeiger um 3 Bytes =>
400 ' IX zeigt nun auf naechste Eintragung
410 REM           INC  IX
420 REM           INC  IX
430 REM           INC  IX
440 REM           DJNZ LOOP
450 REM           POP  HL ;hole alte
460 REM         ;Cursorpositon vom Stapel.
470 REM           LD   (CUR_POS),HL
480 REM           RET
490 REM           ;
500 ' Die Variable LEN enthaelt die Anzahl
510 ' der Elemente der Tabelle.
520 REM LEN       DB   3
530 ' Hier beginnt die Tabelle fuer die
540 ' Zeichen und deren Positionen.
550 REM TAB_CHR   DB   1,1,"A"
560 REM           DB   20,10,"B"
570 REM           DB   38,22,"C"
580 REM           ;
590 REM           END
```

```
C800  2A 03 FA E5 DD 21 27 C8  *....!'.
C808  3A 26 C8 47 DD 66 00 DD  :&.G.f..
C810  6E 01 CD 3E 39 DD 7E 02  n..>9...
C818  DF DD 23 DD 23 DD 23 10  ..#.#.#.
C820  EB E1 22 03 FA C9 03 01  .."..... 
C828  01 41 14 0A 42 26 16 43  .A..B&.C
```

Das eben angeführte Programm setzt ASCII-Zeichen an bestimmte Stellen des Bildschirmes. Dabei braucht man nur die X- und Y-Koordinaten anzugeben und danach das auszugebende ASCII-Zeichen folgen zu lassen. Bei dem Label "LEN" steht die Anzahl der auszugebenden ASCII-Zeichen und bei "TAB_CHR" beginnt die Tabelle, bei der die Koordinaten und das ASCII-Zeichen stehen. Dabei wird zuerst die X- und dann die Y-Koordinate abgespeichert. Bei jedem Durchlauf der Schleife "LOOP", beinhaltet IX die Adresse, bei der die X-Koordinate steht. Ins Register H wird dann die X-Koordinate geladen und danach L aus (IX+01) "gefüllt". Bei der Adresse IX+01 befindet sich die Y-Koordinate. Nun wird zur LOCATE-Routine verzweigt, bei welcher der Cursor positioniert wird. Nachdem das ASCII-Zeichen aus (IX+02) gelesen worden ist, wird dieses mit dem Befehl "RST 18H" auf den Bildschirm ausgegeben. (Sie erinnern sich? RST 18H macht einen Restart zur Adresse 18. Beim Spectravideo ist dies die Bildschirmausgabe eines Zeichens.) Jetzt wird IX erhöht und auf die nächsten Koordinaten gestellt. Am Ende der Schleife wird B, das die Anzahl der auszugebenden Zeichen beinhaltet, dekrementiert und auf Null überprüft. Sind noch nicht alle Zeichen ausgegeben, so wird die Schleife nocheinmal durchlaufen.

Am Anfang wird die alte Cursorposition mit PUSH HL auf den Stapel gelegt, um diese zu "retten", und am Ende wird die alte Cursorposition vom Stapel geholt und wieder in die Systemvariable eingetragen.

Das Programm liegt bei C800H und man kann es mit dem USR-Befehl aufrufen, wie in der letzten Ausgabe erklärt wurde. In der nächsten Ausgabe werden dann endlich die wichtigen "logischen" Befehle erklärt, ohne die ein guter Programmierer nicht auskommt.

2.3 "Massenspeicher"-Format

Als Massenspeicher verwendet das SVI-CP/M 5 1/4 Zoll-Minidisketten. Abhängig von den eingesetzten Laufwerken werden sie entweder ein- oder beidseitig beschrieben. Es ergeben sich somit nach der Formatierung entweder 40 oder 80 Spuren. Bei letzteren werden die Spuren einfach auf der zweiten Seite weitergezählt. In beiden Fällen ist die Spur 0 in einfacher Dichte mit 18 Sektoren zu je 128 Bytes und alle weiteren Spuren in doppelter Dichte mit 17 Sektoren zu je 256 Bytes formatiert. Anders als bei den Spuren, beginnt die Sektornummerierung mit 1.

2.4 Hardware-Voraussetzungen
2.4.1 Laufwerke

Wie schon in der Einleitung erwähnt, handelt es sich bei CP/M um ein diskettenorientiertes Betriebssystem. Daher ist das Vorhandensein von mindestens einem Laufwerk Bedingung. Mehrere, in erster Linie Kopierprogramme, setzen jedoch ein zweites Laufwerk voraus. Aber auch beim Erstellen von Sicherheitskopien bringt es mehr Komfort.

Für CP/M-Anwender, die nur mit einem Laufwerk arbeiten, gibt es z.B. ein Diskettenkopierprogramm "1Copy", das ein Duplizieren von ganzen Disketten ermöglicht. Allerdings muß man, wegen des begrenzten Speicherplatzes im Computer, dabei mehrmals Disketten wechseln.

Umständlicher wird es beim Kopieren von einzelnen Programmen auf eine andere Diskette. In so einem Fall wird dafür meistens das DDT (Dynamic-Debugging-Tool) verwendet. Mit diesem muß das Programm zuerst in den Speicher des SVI-328 geladen, die Länge bestimmt, und nach einem Warmstart mit dem CP/M-Befehl "SAVE" auf die andere Diskette geschrieben werden. Da CP/M auf unerlaubten Diskettenwechsel prüft, soll ein ^C vor dem Speichern nicht vergessen werden.

Die Größe der auf diese Weise zu bewegenden Programme ist jedoch durch den freien Speicherplatz im Computer beschränkt. Will man also Programme kopieren, die größer als etwa 45 KByte sind, muß man die ganze Diskette kopieren und nicht benötigtes löschen.

2.4.2 80Zeichen-Karte

Ein sich etwas stärker bemerkbar machender Kompromiß ergibt sich beim Arbeiten ohne 80Zeichen-Karte. Gehen doch die meisten kommerziellen CP/M-Programme von einem Sichtgerätformat mit 80 Zeichen zu 24 Zeilen aus. Im Falle einer solchen Darstellung würde sich die Information einer Zeile beim Fernseher auf zwei Zeilen aufteilen und so ein vernünftiges Arbeiten unmöglich machen. Manche Programme wie z.B. Turbo-Pascal lassen sich allerdings auch auf 40 Zeichen installieren.

Von der größeren Übersicht am Bildschirm und der bei vielen Programmen verwendeten 80Zeichen-Darstellung ausgehend, ist jedoch der Einsatz einer 80Zeichen-Karte empfehlenswert.

Standardmäßig wird die Karte mit dem ASCII-Zeichenvorrat geliefert. Unter Verzicht auf die eckigen und geschwungenen Klammern ist sie auch mit deutschen Umlauten lieferbar. Es gibt auch eine Version, bei der mittels eines Tastschalters zwischen beiden Zeichensätzen umgeschaltet werden kann.

2.4.3 Zusammenfassung

Bedingung für das Arbeiten in CP/M ist also mindestens ein Laufwerk. Ein zweites und die 80Zeichen-Karte sind auf jeden Fall eine nützliche Erweiterung.

Das Spectravideo-CP/M beinhaltet auch die Ansteuerung eines Parallel-Druckers, der vor allem bei Textverarbeitung, aber auch bei Entwicklungsarbeiten von größeren Programmen unverzichtbar ist. Bestandteil des BIOS ist weiters die Bedienung der seriellen Schnittstelle, die vor allem dem Datenaustausch mit anderen Computern, aber auch der Ansteuerung eines seriellen Druckers dienen kann.

3. Die Initialisierung des CP/M
3.1 Die Systemspuren

Als Systemspuren werden bei der CP/M-Diskette die Spuren 0, 1 und 2 bezeichnet. Das sind jene Spuren, die vor dem eigentlichen Arbeiten mit CP/M gelesen werden müssen, da sie das Betriebssystem beinhalten. Diese Spuren werden während des Arbeitens mit CP/M nur mehr bei einem Warmstart (^C) gelesen.

3.2 Der Boot-Vorgang

Wie schon einleitend erwähnt versucht der BASIC-Interpreter nach dem Einschalten des SVI-328 ein Boot-Programm von der Diskette in den Speicher ab C100H zu laden. Dieser Versuch findet, natürlich erfolglos, auch beim Fehlen einer Diskettenstation statt. Soll eines der beiden verfügbaren Betriebssysteme zur Anwendung kommen, muß sich vorerst ein solches Boot-Programm von Spur 0, Sektor 1 laden lassen.

Bis hierher ist der Vorgang bei eingelegter CP/M- bzw. Disk-BASIC-Diskette genau gleich. Erst eines der beiden in den Speicher gelesenen Boot-Programme bewirkt das weitere Einlesen des auf den folgenden Sektoren befindlichen Systems.

Beim CP/M geht das Laden des Betriebssystems etappenweise vor sich. Die Aufgabe des Boot-Programms ist nur das Laden des BIOS, des hardwareabhängigen Teils des CP/M, in den Speicher ab E600H. In diesem Teil befindet sich bereits die KALTSTART-Routine, die sowohl für die Initialisierung aller notwendigen Betriebsparameter als auch der Peripherie (80Zeichen-Karte, serielle Schnittstelle) verantwortlich ist.

Das Boot-Programm gibt nach getaner Arbeit dieser BIOS-Routine die Kontrolle über das ganze System. Nachdem sie die notwendigen Initialisierungen abgeschlossen hat, kommt die ebenfalls bereits im BIOS enthaltene WARMSTART-Routine zum Zug. Diese lädt dann ihrerseits in der letzten Etappe den restlichen Teil des Betriebssystems in den Speicher ab D000H.

Im Folgenden finden Sie ein kommentiertes Listing des BIOS-Boot-Programms.

Fortsetzung folgt

```
                        TITLE   'BOOT-ROUTINE FÜR DAS CP/M-BIOS'

                                .Z80                    ;es folgen Z80-Mnemonics
0000'                           ASEG                    ;Absoluter Maschinencode
                                ORG     0C100H          ;Beginn des Boot-Programms

E600                    COLDBT  EQU     0E600H          ;KALTSTARTADRESSE DES CP/M

C100                    BOOT:
C100    3E 0F                   LD      A,0FH           ;I/O-Port B des Programmierbaren Sound-Chips
C102    D3 88                   OUT     (88H),A         ;==>(Bänke u. Caps-Lock) adressieren
C104    3E DD                   LD      A,11011101B     ;Bank 21 (RAM v.  0-7FFFH) einschalten;
C106    D3 8C                   OUT     (8CH),A         ;Gleichzeitig wird das Basic-ROM weggeschaltet

C108    31 0100                 LD      SP,0100H        ;Stapelzeiger setzen

C10B    06 11                   LD      B,17D           ;Lese über RDMSEC 17 Sektoren mit je 128 Bytes
C10D    CD C127                 CALL    RDMSEC          ;von Spur 0 (beginnend ab E600H) ins RAM;
                                                        ;Einfache Dichte noch vom Ersten Boot gesetzt;
                                                        ;Die notwendigen Parameter (Erster Sektor, dessen
                                                        ;Einleseadresse und die Größe) sind schon im
                                                        ;Programmcode gesetzt; Diese werden jedoch nach
                                                        ;dem Spurwechsel aktualisiert

C110    3E 00                   LD      A,0             ;Für folgende Sektoren doppelte Dichte wählen
C112    D3 38                   OUT     (38H),A         ;Ins DICHTE-Register schreiben

C114    21 0100                 LD      HL,256D         ;Folgende Sektoren haben 256 Bytes; Wichtig
C117    22 C14F                 LD      (BYTSEC+1),HL   ;fürs Aneinanderreihen der Daten im RAM!
                                                        ;wird direkt in den Programmcode geschrieben!

C11A    3E 12                   LD      A,18D           ;Bereite Erhöhen der Spur nach Lesen des
C11C    32 C15B                 LD      (MAXSEC+1),A    ;Sektor 17 vor (soweit wird aber nicht gelesen)
                                                        ;wird direkt in den Programmcode geschrieben!

C11F    06 02                   LD      B,2             ;Lese über RDMSEC 2 Sektoren mit je 256 Bytes
C121    CD C127                 CALL    RDMSEC          ;von Spur 1 (an die zuvor eingelesenen Sektoren
                                                        ;anschließend) ins RAM

C124    C3 E600                 JP      COLDBT          ;Springe zur Kaltstartroutine des CP/M
                                                        ;BIOS-BOOTVORGANG ABGESCHLOSSEN

C127                    RDMSEC:
C127    C5                      PUSH    BC              ;Rette Sektorzähler

C128    3E 02           SETSEC: LD      A,2             ;2 = erster zu lesender Sektor (1 war Boot-Routine)
C12A    D3 32                   OUT     (32H),A         ;wird während des Programmlaufs erhöht (SETSEC+1)!

C12C    3E 80                   LD      A,80H           ;Lesebefehl für Floppy-Disk-Controller
C12E    D3 30                   OUT     (30H),A         ;ins Command-Register

C130    0E 33                   LD      C,33H           ;C zeigt aufs Daten-Register (für Z80-Befehl "INI")

                                REPT 4                  ;Verzögerung nach Kommando an FDC
                                EX      (SP),HL
                                ENDM                    ;in unserem Fall bei 3.58 MHz
C132    E3              +       EX      (SP),HL
C133    E3              +       EX      (SP),HL
C134    E3              +       EX      (SP),HL
C135    E3              +       EX      (SP),HL
                                                        ;4x19 Taktzyklen = ca. 18 usek

C136    21 E600         SETBUF: LD      HL,0E600H       ;E600H = Startadresse fürs Einlesen des BIOS
                                                        ;Wird während des Programmlaufs, nach dem Einlesen
                                                        ;eines Sektors um eine Sektorgröße erhöht (SETBUF+1)

                                                        ;Hier wird der gewählte Sektor tatsächlich gesucht
                                                        ;und in den Speicher gelesen

C139                    WAIT:
C139    DB 34                   IN      A,(34H)         ;Lese das Interrupt- und Datenanforderungssignal
C13B    87                      ADD     A,A             ;verschiebe um 1 nach links
C13C    38 07                   JR      C,READY         ;wenn Bit 7 (INTRQ) "1" war ==> Lesen beendet
C13E    F2 C139                 JP      P,WAIT          ;wenn Bit 6 (DRQ) "0" war (Positiv) ==> warten
C141    ED A2                   INI                     ;wenn DRQ "1" war ==> Byte aus Daten-Register lesen
C143    18 F4                   JR      WAIT            ;lese solange bis INTRQ "1" wird
```

```
C145  DB 30              IN    A,(30H)        ;Lese Status-Register, Prüfe "Record not found"-,
C147  E6 1C              AND   00011100B      ;"Checksum Error"- und "Lost Data" Flagge
C149  20 DD              JR    NZ,SETSEC      ;Bei Fehler (ungleich 0) neuerlicher Leseversuch

C14B  2A C137            LD    HL,(SETBUF+1)  ;Lade Einleseadresse des zuletzt eingelesenen
                                              ;Sektors (erstes Byte)

C14E  11 0080    BYTSEC: LD    DE,128D        ;128 = Bytes pro Sektor bei Spur 0 (Einfache
                                              ;Dichte); wird ab Spur 1 auf 256 gesetzt

C151  19                 ADD   HL,DE          ;Errechne neue Einleseadresse für nächsten Sektor
C152  22 C137            LD    (SETBUF+1),HL  ;und lege sie in Leseroutine ab

C155  21 C129            LD    HL,SETSEC+1    ;Lade vorhergehende Sektornummer
C158  34                 INC   (HL)           ;erhöhe sie
C159  7E                 LD    A,(HL)         ;und lege sie in Leseroutine ab

C15A  FE 13      MAXSEC: CP    19D            ;Beim ersten Durchgang (Spur 0) wird nach Sektor 18
                                              ;(Ende der physikalischen Sektoren) ein Wechsel auf
                                              ;die nächste Spur veranlaßt; Beim zweiten Durchgang
                                              ;würde nach Sektor 17 die Spur erhöht werden;
                                              ;Soweit wird aber nicht gelesen

C15C  38 1B              JR    C,DECSCZ       ;Wenn Sektornummer kleiner als (MAXSEC+1) kein
                                              ;Spurwechsel

C15E  3E 52              LD    A,52H          ;"STEP IN"-Befehl für FDC
C160  D3 30              OUT   (30H),A        ;ins Command-Register

C162  36 01              LD    (HL),1         ;Nach Spurwechsel mit Sektor 1 beginnen

                         REPT 16              ;Verzögerung für die Zeit des Spurwechselns
                          EX   (SP),HL
                         ENDM                 ;in unserem Fall bei 3.58 MHz
C164  E3           +      EX   (SP),HL
C165  E3           +      EX   (SP),HL
C166  E3           +      EX   (SP),HL
C167  E3           +      EX   (SP),HL
C168  E3           +      EX   (SP),HL
C169  E3           +      EX   (SP),HL
C16A  E3           +      EX   (SP),HL
C16B  E3           +      EX   (SP),HL
C16C  E3           +      EX   (SP),HL
C16D  E3           +      EX   (SP),HL
C16E  E3           +      EX   (SP),HL
C16F  E3           +      EX   (SP),HL
C170  E3           +      EX   (SP),HL
C171  E3           +      EX   (SP),HL
C172  E3           +      EX   (SP),HL
C173  E3           +      EX   (SP),HL
                                              ;16x19 Taktzyklen = ca. 85 usek

C174             BUSY:
C174  DB 30              IN    A,(30H)        ;Lese Status-Register des FDC
C176  1F                 RRA                  ;Warte bis "Busy" gleich "0"
C177  38 FB              JR    C,BUSY

C179             DECSCZ:
C179  C1                 POP   BC             ;Hole Sektorzähler vom Stapel
C17A  10 AB              DJNZ  RDMSEC         ;Vermindere Sektorzähler um 1 und springe
                                              ;solange ungleich 0 zur Leseroutine
C17C  C9                 RET                  ;wenn gleich 0 ins Hauptprogramm zurück

                 ;Abkürzungen:  PSG           Programmierbarer Sound-Generator
                 ;              FDC           Floppy-Disk-Controller

                 ;Erklärungen:  REPT <exp>     Wiederhole <exp> mal die Befehle (repeat) die
                 ;                             zwischen REPT und ENDM stehen

                 ;              ENDM          Ende des Makros (steht am Ende jedes Makros)
                 ;                            Beides sind Makro-Anweisungen an den verwendeten
                 ;                            Assembler

                                 END           ;Ende der Quell-Datei
```

No Fatal error(s)

Es gibt seit einiger Zeit ein sehr nützliches Programm auf allen CP/M-Disketten für die SVI-Computer. Es nennt sich SWEEP und hilft beim Arbeiten mit Dateien. Die Operationen auf Diskette werden erleichtert. Da es leider völlig undokumentiert herausgegeben wurde, holen wir dieses Manko nun nach und bieten eine Beschreibung.

Das Programm SWEEP ermöglicht das Löschen beziehungsweise Kopieren von mehreren vom Anwender markierten Dateien (=tagged files) auf einmal. Dateien können sowohl zwischen den Laufwerken als auch zwischen verschiedenen Benutzerebenen (=User) kopiert werden. Beim Aufruf von SWEEP erfolgt zunächst die Anzeige der zur Auswahl stehenden Kommandos (werden weiter unten erklärt), danach die Angabe des aktiven Laufwerkes sowie der Benutzerebene, die Anzahl der Dateien auf der Diskette, der von ihnen benötigte Speicherplatz, die noch vorhandene Speicherkapazität und schließlich die Anzeige der (alphabetisch geordneten) Dateien mit fortlaufender Nummer, Laufwerk/Benutzerebene, Name, Dateityp und Länge. Die letzte angezeigte ist die aktuelle Datei, auf die sich die meisten der unten angeführten Befehle beziehen.

Der Aufruf erfolgt in der Form:

SWEEP ((Laufwerk:)(Dateiname (.Dateityp))
(Benutzerebene:))

Ausdrücke in Klammern sind wahlweise, man beachte jedoch die Schachtelung! Ebenso müssen alle Abstände eingegeben werden.)

Wird das Programm nur mit SWEEP aufgerufen, so werden das voreingestellte Laufwerk und die Benutzerebene verwendet und nach der Beendigung mit X (siehe unten) wieder automatisch angewählt.

Laufwerk: Hier kann nur A: oder B: stehen.

Dateiname: Die "Joker"-Zeichen * und ? werden akzeptiert. Befindet sich die Datei nicht auf der Diskette/Benutzerebene, dann wird die alphabetisch unmittelbar folgende Datei angezeigt.

Dateityp / Benutzerebene: Für den und die gilt dies ebenfalls. Wird eine Ebene zwischen 0 und 9 angesprochen, so ist vor die Zahl ein Leerzeichen oder eine 0 zu setzen. Wird "?" angegeben, müssen ebenfalls zwei Zeichen verwendet werden.

Hier nun die Befehle im einzelnen:

A: Dateien, die nach dem Kopieren mit M durch das Zeichen "#" zwischenmarkiert wurden, erhalten wieder ihre "*"-Markierung.

B: Die Datei vor der zuletzt angezeigten wird aktuelle Datei.

C: Kopiert die aktuelle Datei auf das angegebene Laufwerk bzw. die Benutzerebene (Kopiere Quellaufwerk/Benutzerebene: Dateiname.Dateityp ===⇒ Ziellaufwerk/Benutzerebene).
Bei Angabe von " V" wird die Kopie mit dem Original auf Übereinstimmung verglichen (Kopiere ...mit Überprüfung).
Sind Quelle und Ziel identisch, erfolgt die Meldung "Warnung: Quelle=Ziel. Für richtiges Inhaltsverzeichnis Diskette neu anwählen". Der Eintrag erscheint erst nach erneutem Anwählen des Laufwerks (siehe L). Das letzte Zeichen des Dateityps ist dabei ein "$".

D: Löscht die aktuelle Datei. Wird die Frage "Löschen (Y/N)?" mit Y (ohne Enter!) beantwortet, erscheint die Anzeige "Gelöscht". N bricht den Befehl ab.

E: Löscht alle markierten (T) oder unmarkierten (U) Dateien. Lautet die Antwort auf die Frage "Sollen die Dateien angezeigt werden?" Y, so wird bei jeder Datei gefragt, ob sie gelöscht (Y), nicht gelöscht (N) oder der Befehl abgebrochen (=A) werden soll.

L: Wählt neues Laufwerk/Benutzerebene an (siehe Anmerkung zu Laufwerk bzw. Dateiname). Enthält der Zielbereich keine Datei, wird eine entsprechende Fehlermeldung ausgegeben. Es sind dann nur die Befehle L und X möglich.

M: Kopiert alle markierten Dateien auf eine andere Diskette/Benutzerebene. Ist die derzeitige Benutzerebene ein "*" d.h. alle Dateien werden angezeigt, und wird wieder auf dieselbe Diskette kopiert, so erscheint eine Fehlermeldung wie bei C. Außer in den Fällen, wo die Datei auf dieselbe Benutzerebene kopiert würde, ist die Meldung gegenstandslos und kann ignoriert werden (siehe auch C).

R: Benennt die aktuelle Datei entweder sofort oder bei Eingabe von "*" oder "?" die Datei "Alter Name" in "Neuer Name" um. Der Dateityp muß ebenfalls eingegeben werden. Existiert eine Datei "Neuer Name" bereits auf der Benutzerebene, so wird eine Fehlermeldung ausgegeben. Alle anderen Fehler führen zum Abbruch des Befehls.

S: Freier Speicherplatz auf der Diskette in Kilobyte.

T: Markiert die aktuelle Datei für nachfolgende Lösch- oder Kopieroperationen durch ein "*". Die Summe der markierten Dateien in Kilobyte wird angezeigt. ACHTUNG: Wird vor einer der beiden Operationen das Laufwerk/Benutzerebene gewechselt, so werden die bereits gesetzten Markierungen wieder gelöscht.

U: Löscht Markierungen wieder. Die Summe der markierten Dateien in Kilobyte wird angezeigt.

X: Rücksprung zum Betriebssystem

?: Anzeige des Menues

ENTER oder Leertaste: Nächste Datei

In dieser Folge will ich auf die Programmstruktur eingehen, und erklären, wie man Fehler mit dem Turbo-Editor beseitigt.

Im Gegensatz zu BASIC ist PASCAL eine strukturierte Sprache, das heißt, es sind sehr viele Sprung- und Verzweigungsbefehle vorhanden, die ein übersichtliches Programmieren erlauben. Außerdem ist PASCAL eine "Block-strukturierte" Sprache, sie ist also in Blöcken aufgebaut. Ein Block ist ein Programmteil, in dem es lokale Variable (Variable, die außerhalb des Blockes nicht definiert sind) gibt, und der nach außen hin abgeschlossen ist. Ein Block wirkt ähnlich wie eine "Black Box". Es gibt fest definierte Variable, die übergeben werden, und bestimmte Variable werden zurückgeliefert. Der Programmierer muß sich aber nicht kümmern, welche Variablen im Block intern verwendet werden. Man kann zum Beispiel oft gebrauchte Unterprogramme getrennt abspeichern und bei Bedarf einladen, man muß sich aber dann nicht mehr den Aufbau ansehen. Es kommt nämlich zu keiner Fehlfunktion, wenn man die gleichen Variablen im Hauptprogramm verwendet.

Jetzt aber wieder ein bißchen Praxis. Ich will ein Programm schreiben, das die Blockstruktur von PASCAL deutlich zeigt. Da es in PASCAL keine Funktion x^y gibt, will ich ein Programm entwerfen, das nach Eingabe einer Zahl die Potenzen bis x^10 berechnet.

```
PROGRAM potenzen;

VAR basis,ergebnis:REAL;
    exponent:INTEGER;

PROCEDURE potenz (bas:REAL;
                  exp:INTEGER);
  VAR zaehler:INTEGER;
  BEGIN
    ergebnis:=1;
    FOR zaehler:=1 TO exp DO
      ergebnis:=ergebnis*bas;
    bas:=10;
  END;

BEGIN
  CLRSCR;
  WRITE('Basis=? ');
  READLN(basis);
  FOR exponent:=1 TO 10 DO
    BEGIN
      potenz(basis,exponent);
      WRITE(basis:8:4,' hoch ',exponent);
      WRITELN(' = ',ergebnis:8:4);
    END;
  (*WRITELN (zaehler);*);
END.
```

Ganz zu Beginn werden die Variablen "basis", "ergebnis" und "exponent" definiert. Dann verwenden wir einen neuen Befehl: "PROCEDURE". Mit diesem Befehl wird ein Unterprogramm eingeleitet. Die Syntax lautet:

```
PROCEDURE name (variablenliste);
```

"name" bezeichnet den Namen der Prozedur, mit dem sie aufgerufen werden kann. "variablenliste" enthält alle Variablen, die übergeben werden sollen. Bei unserem Beispiel sind dies die Variablen "basis" und "exponent". Diese Variablen erhalten im Inneren der Prozedur die Namen "bas" und "exp", die aber nur innerhalb der Prozedur gültig sind.

Die Variablen können beliebig verändert werden, ohne daß das eine Auswirkung auf das Hauptprogramm hat (Zeile: bas:=10;). Genauso verhält es sich mit der Variable "zaehler". Sie wird innerhalb der Prozedur definiert und muß daher nicht schon im Hauptprogramm definiert sein. Die Zeile "(*WRITELN (zaehler);*)" hat keine Wirkung, da "(*" und "*)" einen Kommentar einschließen, der nicht beachtet wird. Man kann auch geschlungene Klammern verwenden. Wenn man aber die Klammern weggibt, wird die Zeile als Befehl interpretiert, und beim Compilieren gibt TURBO einen Fehler aus, da "zaehler" nicht definiert ist. Bei dieser Gelegenheit kann man einen großen Pluspunkt des TURBO-Editors bemerken. Wenn TURBO einen Fehler gefunden hat, wird der Cursor nach einem Betätigen der Taste "ESC" gleich an die Stelle des Fehlers gestellt. So kann man einfach und rasch Fehler beseitigen.

Ein anderer Befehl kommt in diesem Programm auch vor: Die FOR-NEXT-Schleife des BASIC. In PASCAL sieht sie allerdings etwas anders aus. Eingeleitet wird sie fast so wie in BASIC:

```
FOR variable := anfang TO ende DO
```

Die Variable "variable" ist die Schleifenvariable, die hochgezählt wird. In PASCAL ist es aber nur möglich, die Variable um 1 zu erhöhen. Eine Option "STEP" wie in BASIC ist nicht möglich. Wenn man hinunter zählen will, muß man statt "TO" "DOWNTO" verwenden.

```
FOR variable := anfang DOWNTO ende DO
```

Nach dem DO am Ende der Zeile folgt jetzt ein "BEGIN". Mit "BEGIN" wird der Schleifenkörper eingeleitet, der dann mit "END;" wieder abgeschlossen wird. Die Schleife sieht dann so aus:

```
FOR variable := anfang TO ende DO
  BEGIN
    .
    .     Anweisungen
    .
  END;
```

Ein "NEXT" wie in BASIC gibt es nicht. Das "END;" bezeichnet das Ende der Schleife. Es gibt auch noch eine Kurzform der FOR-Schleife, wenn nur eine Anweisung wiederholt werden soll.

```
FOR variable := anfang TO ende DO
  befehl;
```

Beide Formen sind in dem Potenzen-Programm verwendet worden.

Es gibt aber noch einen wichtigen Unterschied zum BASIC. In BASIC wird jede Schleife mindestens einmal durchlaufen, in PASCAL nicht unbedingt. Ein Beispiel zur Verdeutlichung:

```
10 FOR A= 10 to 1
20 PRINT A
30 NEXT A
```

Diese Schleife wird in BASIC einmal durchlaufen, es erfolgt also die Ausgabe "10". In PASCAL sieht das so aus:

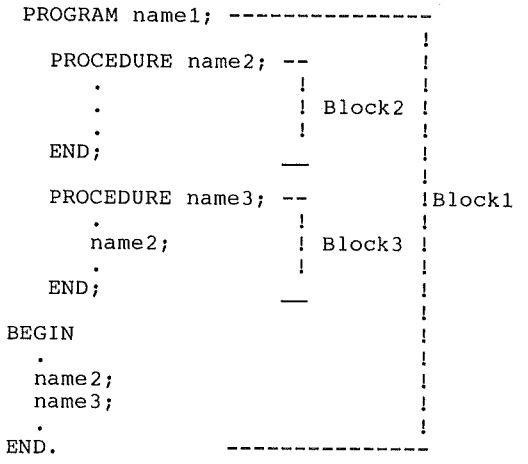
```
FOR a := 10 TO 1 DO
 WRITELN(a);
```

Diese Schleife wird nicht durchlaufen, es erfolgt also keine Ausgabe. In PASCAL darf man auch, im Gegensatz zu BASIC, die Schleifenvariable innerhalb der Schleife nicht verändern.

Nun aber wieder zu unserem Programm. Sehen wir uns den Aufbau genauer an. Der erste Befehl im Hauptprogramm ist "CLRSCR". Das ist eine eingebaute Prozedur, die ein Löschen des Bildschirms bewirkt. Dann wird nach einer Eingabe gebeten, die in der Variablen "basis" abgelegt wird. Mit Hife einer FOR-Schleife werden dann die Potenzen berechnet. Dabei wird ein Unterprogramm, die Prozedur "potenz", aufgerufen. Man muß sich nicht um den inneren Aufbau der Prozedur kümmern, man verwendet sie wie eine Befehlserweiterung. Man muß nur die Prozedur vor ihrem ersten Aufruf definieren. Besonders ist noch die Variable "ergebnis". Sie ist im Hauptprogramm definiert worden, aber nicht im Unterprogramm, und doch wird sie dort verwendet. Die Erklärung ist einfach:

Da die Variable nur im Hauptprogramm definiert wurde, gilt sie auch im Unterprogramm. Anders würde es sich verhalten, wenn man im Unterprogramm auch eine Variable "ergebnis" definiert hätte. Dann wären das zwei unabhängige Variable.

Jetzt will ich noch etwas genauer auf die Blockstruktur eingehen. Im allgemeinen sieht ein PASCAL-Programm so aus:



Man sieht, daß Blöcke beliebig ineinander geschachtelt werden dürfen. Man kann auch Unterprogramme von Unterprogrammen aus aufrufen. Wie sieht es aber mit der Gültigkeit von Variablen aus? Variable, die in Block1 definiert worden sind, also im Hauptprogramm, gelten grundsätzlich auch in den Unterprogrammen und können auch dort verändert werden. Das gilt aber nur, wenn nicht Variablen mit demselben Namen in den Unterprogrammen selbst definiert worden sind. Dann gelten die im Hauptprogramm definierten Variablen nicht in den Unterprogrammen und umgekehrt. Variable, die in einem Unterprogramm definiert worden sind, gelten nur dort, und sonst nirgendwo.

Im Zusammenhang mit der Blockstruktur will ich auch den GOTO-Befehl erklären, obwohl man ihn eigentlich in PASCAL nicht verwenden sollte, da durch viele GOTO-Befehle ein Programm unleserlich wird.

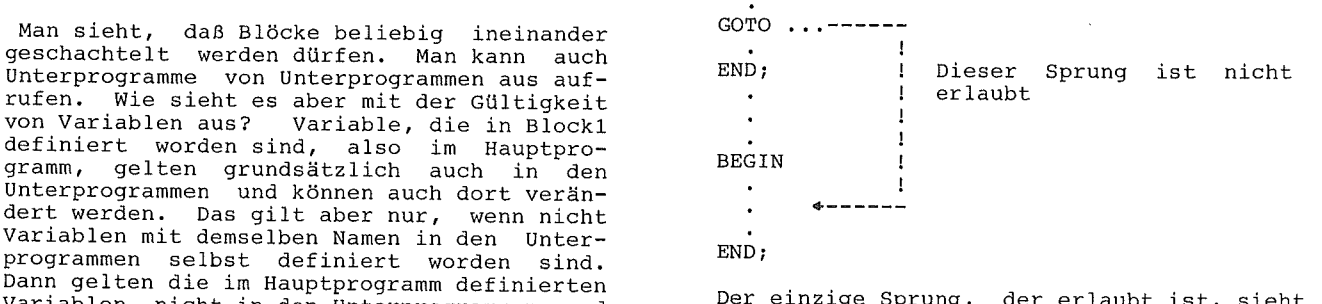
Anders als in BASIC gibt es in PASCAL keine Zeilennummern, und die GOTO-Befehle haben sogenannte "Labels" als Ziel. Ein Label muß, bevor es verwendet wird, definiert werden. Das geschieht durch den Befehl "LABEL name;". "LABEL sprung;" besagt, daß irgendwo im Block, wo dieses Statement steht, das Label "sprung" vorkommt. Wenn also in einem Unterprogramm Labels benötigt werden, müssen sie auf jeden Fall in diesem Unterprogramm definiert werden. Labels aus dem Hauptprogramm oder aus anderen Unterprogrammen haben keine Wirkung. Ein Beispiel:

```
PROGRAM test1;
  LABEL ziel;

  PROCEDURE unterprogramm1;
    LABEL ziel;
      BEGIN
        WRITE('Unterprogramm Nr.');
        GOTO ziel1;
        WRITELN('2');
  ziel: WRITELN('1');
      END;

BEGIN
     WRITE('Das ist ');
     GOTO ziel1;
     WRITE('nicht ');
ziel: WRITELN('das Hauptprogramm');
     WRITELN('');
     unterprogramm1;
END.
```

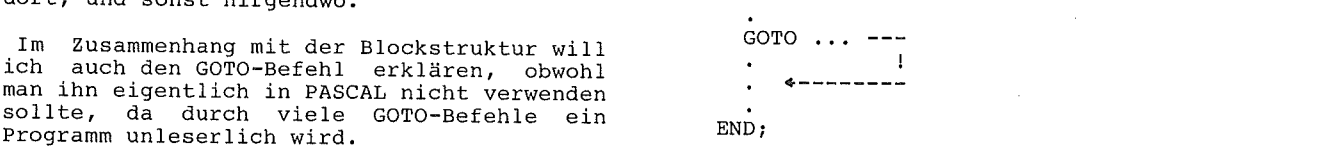
Obwohl im Haupt- und im Unterprogramm ein Label mit demselben Namen erzeugt wurde, läuft das Programm dennoch. Labels muß man immer dann definieren, wenn auch Variable definiert werden können. Dabei ist es in TURBO-PASCAL, nicht wie in anderen PASCAL-Versionen, egal, ob zuerst Variable und dann Labels, oder umgekehrt definiert werden. Der GOTO-Befehle sieht also so aus:

```
BEGIN
      LABEL marke;
         .
         .
      GOTO marke;
         .
         .Dieser Programmteil wird nicht
         .durchlaufen
         .
marke: anweisungen;
         .
         .
END;
```

Eine Einschränkung gibt es aber noch: Man kann nicht aus Blöcken herausspringen.



Der einzige Sprung, der erlaubt ist, sieht so aus:



Mehrere Labels werden so definiert:

```
LABEL marke1,marke2,.....marke7;
```

In der nächsten Folge werde ich die anderen Verzweigungsbefehle von PASCAL erklären, und natürlich gibts auch wieder etwas Praxis.

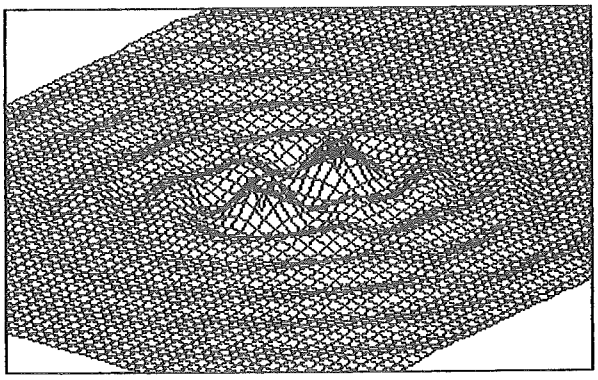
RAFAEL RAZIM

3D-Graphik auf dem SVI

Wer hat nicht schon beeindruckende Abbildungen von Paraboloiden, Ellipsoiden oder anderen geometrischen Figuren gesehen? Das BASIC-Programm "3D-Graphik" ermöglicht die Darstellung beliebiger Flächen, die lediglich mit mathematischen Formeln definiert sein müssen.

Mit "3D-Graphik" wird es möglich, gekrümmte Flächen zu zeichnen. Dazu braucht man nur die dazugehörige Formel einzugeben. Es ermöglicht andererseits beliebige Funktionen zweier unabhängiger Veränderlicher graphisch darzustellen. So kann man seiner Phantasie freien Lauf lassen, interessante Bilder erstellen, das räumliche Denken trainieren und lernen, mit Funktionen umzugehen.

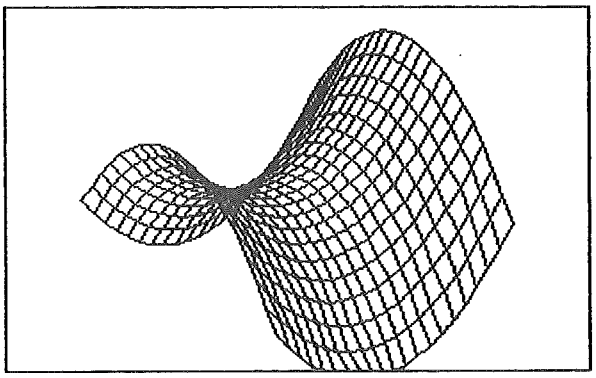
Eine Fläche ist eine Punktmenge, die von zwei Parametern abhängt, lernt man in der Geometrie. Aber wie soll man eine Fläche im dreidimensionalen Raum auf einem Bildschirm, der ja selbst nur ein zweidimensionales Gebilde ist, zeichnen? Üblicherweise stellt man die Schnittkurven dieser Fläche mit Ebenen, denke man nur an die Höhenschichtenlinien auf Landkarten, dar, oder man zeichnet Kurven, die auf der Ebene liegen und dadurch zustande kommen, daß man einen der oben erwähnten Parameter konstant hält und den anderen variiert.



Auf jeden Fall benötigt man zur Darstellung von Funktionen ein Koordinatensystem. Eines der gebräuchlichsten ist das kartesische System, welches aus drei Achsen besteht, die aufeinander senkrecht stehen und rechtswendig zugeordnet sind. Daumen, Zeigefinger und Mittelfinger der rechten Hand bilden ein Rechtssystem. Davon macht man zum Beispiel als Merkregel in der Elektrotechnik, die ich studiere, Gebrauch. Man verfällt dann aber bei einem Test oft auf den Fehler, die linke Hand heranzuziehen, und schon ist alles falsch. Aber Spaß beiseite. Mein Programm benützt so ein System. Der Anwender muß nun die realen Koordinaten des Ursprungs, das ist der Schnittpunkt der drei Koordinatenachsen angeben. Die Eingabe erfolgt in den gewohnten Bildschirmkoordinaten. Weiters ist die Angabe der Winkel der Achsen zur Horizontale notwendig.

Allerdings gibt es verbotene Winkelkombinationen, der Computer frägt dann aber nochmals. Die Winkel selbst werden übrigens in Altgrad eingegeben (360 Altgrad ergeben eine volle Umdrehung.). Die Achsen selbst werden auch gezeichnet. Falls das nicht erwünscht ist, kann man die Zeilen 1170-1190 löschen.

Doch nun zum Wichtigsten, zur Eingabe der Funktion. Die Zeilen 200-270 sind dafür reserviert. Die Eingabe selbst kann nicht während des Programmlaufs erfolgen, da die Funktion einprogrammiert werden muß. Am besten ist es, das Programm nach dem Laden kurz laufen zu lassen, dann zu stoppen und die Funktionstaste 2 zu drücken. Einige Erklärungen und die reservierten Programmzeilen werden gelistet.



Wie muß nun die Funktion aussehen? Antwort: Das Programm verarbeitet drei Arten der Funktionsdarstellung:

1. explizite Darstellung
2. Darstellung in Zylinderkoordinaten
3. Parameterdarstellung

Fangen wir mit der einfachsten, der expliziten Darstellung an. Sie hat folgende Gestalt: z=f(x,y). Der Variablen z wird je nach x- und y-Wert und je nach Funktionsvorschrift f ein bestimmter Wert zugewiesen. Man kann sich das auch so vorstellen: Jedem Punkt in der x-y-Ebene wird eine Höhe z zugewiesen. Der Punkt auf der Fläche hat dann die Koordinaten: (x,y,z). Das Programm zeichnet die Linien x=konst. und y=konst., also die Schnittkurve der Fläche mit den Ebenen x=R,2R,3R,... und y=R,2R,3R,...R ist der Raster, der auch eingegeben werden muß. Je größer der Raster gewählt wird, desto eckiger werden die Kurven, andererseits ist die Graphik dadurch schneller fertig.

Auch ist die Eingabe eines Maßstabs notwendig, da ja das Programm das gewünschte Verhältnis von Einheitsstrecke zu Bildschirmpunkt kennen soll. Ich habe die z-Achse, die ja immer senkrecht ist, als Bezugsstrecke gewählt. Gefragt wird die Länge vom Ursprung zum oberen Bildschirmrand. Ich testete auch andere Arten der Maßstabseingabe, machte aber mit dieser die besten Erfahrungen, da man sich darunter etwas vorstellen kann.

Ich habe auch die Möglichkeit offen gelassen, Zylinderkoordinaten zu verwenden. Hier ist z eine Funktion des Winkels und des Radius r: z=f(r,u). Wiederum werden die Linien r=konst. und u=konst gezeichnet.

Allerdings haben beide Darstellungsarten den Nachteil, daß eben jedem Punkt in der x-y-Ebene nur eine Höhe zugeordnet werden kann. Das verhindert die Darstellung von Kugeln, Ellipsoiden und vieler anderer interessanter geometrischer Figuren. Daher verwende ich auch die allgemeinste Form der mathematischen Beschreibung einer Fläche, die Parameterdarstellung. Wie oben bereits erwähnt ist die Fläche als Punktmenge, die von zwei Parametern abhängt, definiert. Diese Parameter habe ich in meinem Programm u und v genannt. Die Parameterdarstellung sieht so aus:

```
x=f(u,v)
y=g(u,v)
z=h(u,v)
```

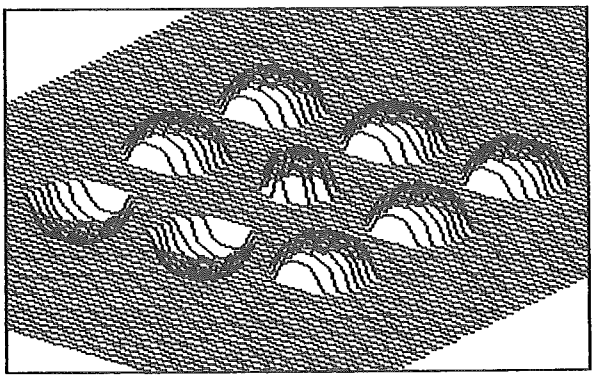
Man sieht statt einer Funktion drei, allerdings macht sich dieser Mehraufwand bezahlt.

Nochmals, warum Parameterdarstellung? Denken wir nur an den Kreis: er wird implizit durch die Gleichung $x^2+y^2=r^2$ beschrieben. Eine eindeutige explizite Darstellung existiert nicht. Man kann lediglich die Äste eines Kreises angeben: y=+sqr(r²-x²) und y=-sqr(r²-x²). Oder man gibt den Kreis mittels Parameterdarstellung an:

```
x=r*cos(u)
y=r*sin(u)
```

Setzen Sie in die obige implizite Gleichung ein. Sie sehen, diese Darstellung genügt der impliziten Kreisgleichung. Genauso ist es mit der Darstellung von Flächen. Die Kugel ist explizit nicht darstellbar, die implizite Gleichung lautet: $x^2+y^2+z^2=r^2$, r=Kugelradius. Die Parameterdarstellung ermöglicht es aber, die Kugel doch richtig ins Bild zu bringen:

```
x=r*cos(u)*sin(v)
y=r*sin(u)*sin(v)
z=r*cos(v)
```



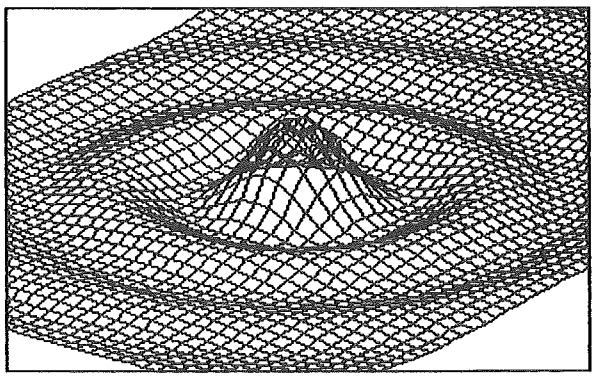
Ich habe mich bemüht, das Programm übersichtlich zu gestalten, damit man es leicht ändern kann. Es besteht im wesentlichen aus vier Teilen: dem Teil am Anfang, der die darzustellende Funktion enthält, dem Eingabeteil, der vom Benützer die notwendigen Informationen erfrägt, dem Rechenteil, der die Lage der Achsenkreuze, die Verzerrungsfaktoren i,j,k und einiges mehr berechnet, der übrigens auch den kniffligsten Teil des ganzen Programms darstellt, und dem Graphikteil, der die Funktion auf dem Bildschirm ausplottet.

Falls Sie Ihre Kreation auf Diskette oder Kassette speichern möchten, drücken Sie, bevor sie in den Kommandomodus zurückkehren, Funktionstaste 1. Und nun viel Spaß beim Experimentieren.

Norbert Rohringer

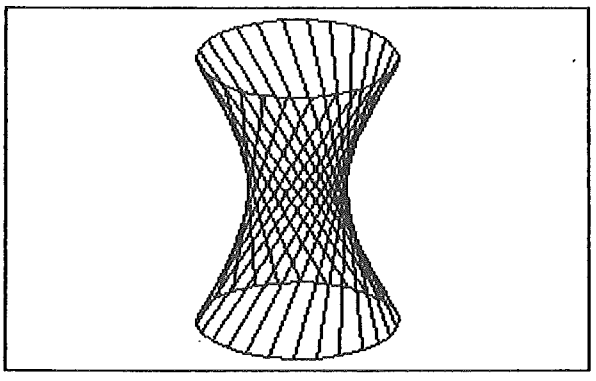
```
10 '*********************************
20 '*                               *
30 '*          3D-GRAPHIK           *
40 '*                               *
50 '*     von Norbert ROHRINGER     *
60 '*                               *
70 '*********************************
80 '
85 GOTO300
90 'Folgende Funktionsarten werden
100 'verarbeitet:
110 '    1. explizite Darstellung:
120 '       z=f(x,y); z.B.:z=3*x+sin(y)
130 '    2. Zylinderkoordinaten:
140 '       z=f(r,u); z.B.:z=1/r+sin(u)
150 '    3. Parameterdarstellung:
160 '       x=f(u,v); z.B.:x=cos(u)*v
170 '       y=g(u,v); z.B.:y=sin(u)*v
180 '       z=h(u,v); z.B.:z=tan(u)*v
190 'geben sie die Funktion ein:
200 :
210 :
220 :
230 :
240 :
250 :
260 :
270 :
280 RETURN
290 '
300 'Eingabeteil
310 '
315 SCREEN0,0
316 ONSTOPGOSUB3190:STOP ON
320 DEFSNG A-D,G-Z
330 DEFINTE,F
340 ONERRORGOTO3210
350 CLS
360 PRINTSTRING$(1,180);STRING$(37,174);STRI
NG$(1,168)
370 PRINTSTRING$(1,197);SPC(9)"3 D - G R A P
 H I K";SPC(9)STRING$(1,200)
380 PRINTSTRING$(1,169);STRING$(37,171);STRI
NG$(1,170)
390 PRINTSPC(2)"Basic - 2.Nov. 1984"
400 PRINT
410 ONKEYGOSUB3130:KEYON
420 KEY2,"list90-270"+CHR$(13)
430 '
440 'Eingabe des Achsenkreuzes
450 '
460 PRINTSPC(2)"Eingabe des Koordinatensyste
ms"
470 PRINTSPC(4)"1. Koordinaten des Ursprungs
:"
480 PRINTSPC(8)"horizontal (0<a<255): ";:INP
UTEU
490 PRINTSPC(8)"vertikal (0<b<196):";:INPUTF
U
500 PRINTSPC(4)"2. Winkel der Achsen zum Hor
izont:"
510 PRINTSPC(8)"x-Achse: ";:INPUTA
520 PRINTSPC(8)"y-Achse: ";:INPUTB
530 IFABS(A+B-90)<1E-04THENPRINTSPC(2)"Falsc
he Eingabe der Achsenwinkel:":PRINTSPC(6)"Se
hstrahl parallel z-Achse":GOTO510
540 PRINTSPC(4)"4. Masstab: z<=: ";:INPUTP8
550 PRINTSPC(4)"5. Background: ";:INPUTC1
560 PRINTSPC(2)"Eingabe der Funktionscharakt
eristik"
570 PRINTSPC(4)"1. Kartesische Koordinaten"
580 PRINTSPC(4)"2. Zylinderkoordinaten"
590 PRINTSPC(4)"3. Paramterdarstellung"
600 PRINTSPC(8)"Eingabe: ";:L$=""
610 L$=INKEY$:IFL$=""THEN610
620 PRINTL$:F3=VAL(L$)
630 IF(F3=1)THENPRINTSPC(6)"Darstellungsbere
ich"ELSEGOTO670
640 PRINTSPC(8)"voll/halb (1/2); ";:N$=""
650 N$=INKEY$:IFN$=""THEN650
660 F4=VAL(N$):PRINTN$:PRINTSPC(8)"Raster: "
;:INPUTG
670 IF(F3=2)THENPRINTSPC(6)"Parameterbereich
"ELSEGOTO690
680 PRINTSPC(8)"r laeuft bis: ";:INPUTRM:PRI
NTSPC(8)"Raster: ";:INPUTGR:PRINTSPC(8)"Rast
er u: ";:INPUTGU
690 IF(F3=3)THENPRINTSPC(6)"Parameterbereich
"ELSE720
```

```
700 PRINTSPC(8)"u laeuft von: ";:INPUTUN:PRI
NTSPC(16)"bis: ";:INPUTUM:PRINTSPC(16)"Raste
r: ";:INPUTGU
710 PRINTSPC(8)"v laeuft von: ";:INPUTVN:PRI
NTSPC(16)"bis: ";:INPUTVM:PRINTSPC(16)"Raste
r: ";:INPUTGV
720 PRINT" Fall sie die Graphik speichern"
721 PRINT" moechten druecken sie, wenn sie
722 PRINT" fertig ist, eine Funktionstaste"
725 '
730 'Rechenteil
740 '
750 PI=3.14159
760 A=A*PI/180
770 B=B*PI/180
780 TA=TAN(A)
790 TB=TAN(B)
800 SA=SIN(A):CA=COS(A)
810 SB=SIN(B):CB=COS(B)
820 I=(1/CA)*(1/SQR(1+TA/TB))
830 J=(1/CB)*(1/SQR(1+TB/TA))
840 K=(2-I*I-J*J):IFK<0THENPRINTSPC(4)"Falsc
he Winkelkombination":GOTO10
850 K=SQR(K)
860 PS=FU/(K*P8)
870 '
880 'Achsenberechnung
890 '
900 E2=FU+EU*TA
910 IF(E2>191)THEN(E2=191):E1=EU-(191-FU)/TA
:F1=1:GOTO930
920 E1=0
930 E4=(255-EU)*TB+FU
940 IF(E4>191)THEN(E4=191):E3=EU+(191-FU)/TB
:F2=1:GOTO960
950 E3=255
960 '
970 P1=PS*J*CB:P2=PS*I*CA
980 P4=PS*J*SB:P5=PS*I*SA
990 P3=PS*K
1000 '
1010 'Maximalwerte der Koordinaten
1020 '
1030 IFF1=0THENXM=EU/P2ELSEXM=(191-FU)/P5
1040 IFF2=0THENYM=(255-EU)/P1ELSEYM=(191-FU)
/P4
1050 IFF1=0THENXN=-FU/P5ELSEXN=(EU-255)/P2
1060 YN=-EU/P1
1070 Y1=FU+P4*YN
1080 IFY1<0THENYN=-FU/P4
1085 '
1090 'Plotten der Funktion
1100 '
1110 COLOR5,C1,C1
1120 SCREEN1
1130 LINE(0,0)-(255,0)
1140 LINE(0,192)-(255,192)
1150 LINE(0,0)-(0,192)
1160 LINE(255,0)-(255,192)
1170 LINE(EU,FU)-(E1,E2),3
1180 LINE(EU,FU)-(E3,E4),3
1190 LINE(EU,FU)-(EU,0),3
1200 '
1210 '
1230 ONF3GOSUB1270,2370,2720
1240 '
1250 PLAY"m5000cegec"
1260 GOTO1260
1270 '
1280 'Ausgabe: explizit
1290 '
1300 '1.Quadr. y-Richt.
1310 '
1320 FORX=0TOXMSTEPG
1330 Y=0:GOSUB200
1340 E1=EU-X*P2:E2=FU-P3*Z+X*P5
1350 FORY=GTOYMSTEPG
1360 GOSUB200
1370 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
1380 LINE(E1,E2)-(E,F)
1390 E1=E:E2=F
1400 IFF>191THENNEXTXELSENEXTY
1410 NEXTX
1420 '
1430 '1.Quadr. x-Richt.
1440 '
1450 FORY=0TOYMSTEPG
1460 X=0:GOSUB200
1470 E1=EU+Y*P1:E2=FU-P3*Z+Y*P4
1480 FORX=GTOXMSTEPG
1490 GOSUB200
1500 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
1510 LINE(E1,E2)-(E,F)
1520 E1=E:E2=F
1530 IFF>191THENNEXTYELSENEXTX
1540 NEXTY
1550 '
1560 '2.Quad. y-Richt.
1570 '
1580 FORX=-GTOXNSTEP-G
1590 Y=0:GOSUB200
1600 E1=EU-X*P2:E2=FU-P3*Z+X*P5
1610 FORY=GTOYMSTEPG
1620 GOSUB200
1630 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
1640 LINE(E1,E2)-(E,F)
1650 E1=E:E2=F
1660 IFF>191THENNEXT
```



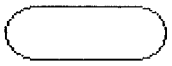
```
1670 IFE>255THENNEXTXELSENEXTY
1680 NEXTX
1690 '
1700 '2.Quadr. x-Richt.
1710 '
1720 FORY=0TOYMSTEPG
1730 X=0:GOSUB200
1740 E1=EU+Y*P1:E2=FU-P3*Z+Y*P4
1750 FORX=-GTOXNSTEP-G
1760 GOSUB200
1770 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
1780 LINE(E1,E2)-(E,F)
1790 E1=E:E2=F
1800 IFE>255ORF<0THENNEXTYELSENEXTX
1810 NEXTY
1820 IFF4=2THENRETURN
1830 '
1840 '3.Quad. y-Rict.
1850 '
1860 FORX=0TOXMSTEPG
1870 Y=0:GOSUB200
1880 E1=EU-X*P2:E2=FU-P3*Z+X*P5
1890 FORY=-GTOYNSTEP-G
1900 GOSUB200
1910 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
1920 LINE(E1,E2)-(E,F)
1930 E1=E:E2=F
1940 IFF>191ORE<0THENNEXTXELSENEXTY
1950 NEXTX
1960 '
1970 '3.Quad. x-Richt.
1980 '
1990 FORY=0TOYNSTEP-G
2000 X=0:GOSUB200
2010 E1=EU+Y*P1:E2=FU-P3*Z+Y*P4
2020 FORX=GTOXMSTEPG
2030 GOSUB200
2040 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
2050 LINE(E1,E2)-(E,F)
2060 E1=E:E2=F
2070 IFF>191ORE<0THENNEXTYELSENEXTX
2080 NEXTY
2090 '
2100 '4.Quad. y-Richt.
2110 '
2120 FORX=-GTOXNSTEP-G
```

```
2130 Y=0:GOSUB200
2140 E1=EU-X*P2:E2=FU-P3*Z+X*P5
2150 FORY=-GTOYNSTEP-G
2160 GOSUB200
2170 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
2180 LINE(E1,E2)-(E,F)
2190 E1=E:E2=F
2200 IFF<0THENNEXTX
2210 IFE<0THENNEXTXELSENEXTY
2220 NEXTX
2230 '
2240 '4.Quad. x-Richt.
2250 '
2260 FORY=0TOYNSTEP-G
2270 X=0:GOSUB200
2280 E1=EU+Y*P1:E2=FU-P3*Z+Y*P4
2290 FORX=-GTOXNSTEP-G
2300 GOSUB200
2310 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
2320 LINE(E1,E2)-(E,F)
2330 E1=E:E2=F
2340 IFE>255ORF<0THENNEXTYELSENEXTX
2350 NEXTY
2360 RETURN
2370 '
2380 'Zylinderkoordinaten
2390 'z=f(u,r)
2400 '
2410 'Linien u=const.
2420 '
2430 FORU=0TO6.28318STEPGU
2440 R=0:GOSUB200
2450 CU=COS(U):SU=SIN(U)
2460 E1=EU:E2=FU-P3*Z
2470 FORR=GRTORMSTEPGR
2480 GOSUB200
2490 X=R*CU:Y=R*SU
2500 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
2510 LINE(E1,E2)-(E,F)
2520 E1=E:E2=F
2530 IF(E>255)OR(E<0)THENNEXTU
2540 IF(F<0)OR(F>191)THENNEXTU
2550 NEXTR
2560 NEXTU
2570 '
2580 'Linien r=const.
2590 '
2600 FORR=GRTORMSTEPGR
2610 U=0:GOSUB200
2620 E1=EU-P2*R:E2=FU-P3*Z+P5*R
```

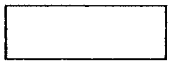


```
2630 FORU=GUTO6.4STEPGU
2640 GOSUB200
2650 X=R*COS(U):Y=R*SIN(U)
2660 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
2670 LINE(E1,E2)-(E,F)
2680 E1=E:E2=F
2690 NEXTU
2700 NEXTR
2710 RETURN
2720 '
2730 'Ausgabe: parametrisiert
2740 '
2750 'Linien u=const.
2760 '
2770 FORU=UNTOUMSTEPGU
2780 V=VN:GOSUB200
2790 E1=EU+Y*P1-X*P2:E2=FU-P3*Z+Y*P4+X*P5
2800 FORV=VN+GVTOVMSTEPGV
2810 GOSUB200
2820 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
2830 LINE(E1,E2)-(E,F)
2840 E1=E:E2=F
2850 NEXTV
2860 NEXTU
2870 '
2880 'Linien v=const.
2890 '
2900 FORV=VNTOVMSTEPGV
2910 U=UN:GOSUB200
2920 E1=EU+Y*P1-X*P2:E2=FU-P3*Z+Y*P4+X*P5
2930 FORU=UN+GUTOUMSTEPGU
2940 GOSUB200
2950 E=EU+Y*P1-X*P2:F=FU-P3*Z+Y*P4+X*P5
2960 LINE(E1,E2)-(E,F)
2970 E1=E:E2=F
2980 NEXTU
2990 NEXTV
3000 RETURN
3110 '
3120 RETURN
3130 '
3140 'Speichern des Bildes
3150 '
3160 SAVE"1:scr I",S
3170 RETURN
3180 '
3190 COLOR 15,4,5:SCREEN0,1:END
3200 '
3210 RESUMENEXT
```

Symbole in Flussdiagrammen



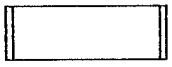
Anfang, Ende



Verarbeitung

Ein-/Ausgabe

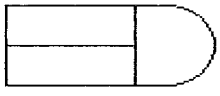
Entscheidung



Unterprogramm

Verbindung

WHILE,REPEAT-Schleife



FOR...NEXT-Schleife

Es ist auch im BASIC mitunter nützlich, über die Datenstrukturen bescheid zu wissen. Aus diesem Grund werden wir uns nun mit den grundlegenden Strukturen beschäftigen und ihre Programmierung erklären.

Um die Übersichtlichkeit und Leistungsfähigkeit eines Computer-Programms zu erhöhen und zugleich dessen Komplexität in übersehbaren Grenzen zu halten, ist es von höchster Wichtigkeit, jeweils die vorteilhafteste "Datenstruktur" zu verwenden. Doch dies allein ist nicht genug. Es müssen zusätzlich zur Behandlung(Abarbeitung) der gewählten Datenstrukturen die geeignetsten Programmierverfahren(Techniken) angewendet werden.

Beispiele von Datenstrukturen sind:

* Lineare Listen (arrays)
* Mehrdimensionale Listen (matrices)
* Stapeln (stacks, LIFO=Last In First Out)
* Schlangen (queues, FIFO = First In First Out)
* Bäume (trees)
* Graphen (graphs)

u.a.

Als Verfahren zum sinnvollen Arbeiten mit diesen Strukturen nennen wir:

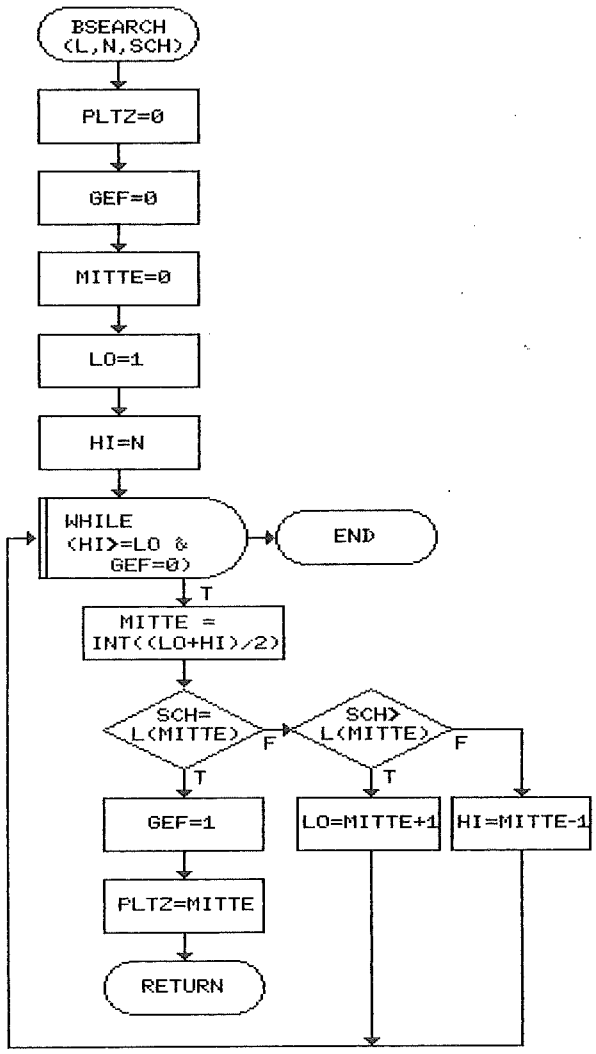
* Sortieren (sort)
* Suchen (search)
* Verschmelzung (merge)
* Vereinigung (union)
* Durchschnitt (intersection)

u.a.

Die am meisten verwendete Datenstruktur ist die lineare Liste. Eine Liste ist eine Ansammlung von gleichartigen Elementen. Elemente können wiederum, wie dies der Fall in höheren Programmiersprachen wie PL/I, PASCAL, COBOL und anderen ist, auch Strukturen selbst sein. Als Operationen an linearen Listen können alle oben genannten Verfahren angewendet werden. Deshalb werden die meisten Programmbeispiele in dieser Artikelserie solche Listen verwenden.

Sortieren ist einer der häufigsten Programmierprozesse. Man versteht darunter die Herstellung einer gewissen Ordnung zwischen den Elementen einer Struktur. Das Kriterium dieser Ordnung nennt man Schlüssel (key). Ein Sortierverfahren beruht also darauf, eine Liste nach einem Schlüssel zu ordnen. Es gibt zwei Arten von Sortierverfahren, interne und externe. Interne Verfahren laufen ausschließlich im Hauptspeicher ab. Dabei kann es zu Speicherplatzproblemen kommen. Externe Verfahren sind geeignet, um große Mengen von Daten zu sortieren, und machen von peripheren Speichern wie Disketten oder Kassetten Gebrauch.

Wir werden ausschließlich interne Verfahren besprechen. Dabei sind zwei Faktoren von großer Bedeutung. Zum einen ist die Anzahl der gleichartigen Einzelschritte (Befehle) wichtig. Deren Anzahl ist nämlich direkt proportional zur Zeit, die notwendig ist, um eine Struktur nach einem Kriterium zu sortieren. Zum zweiten muß der Speicherbedarf berücksichtigt werden. Je mehr Speicherplatz benötigt wird, desto schwerer können gewisse Verfahren angewendet werden. Oft muß man Kompromisse schließen.



Neben dem Sortieren soll das Suchen (search) als zweithäufigste Operation eingestuft werden. Eine Liste oder eine Datei wird nach einem bestimmten Suchkriterium abgesucht, nämlich nach dem Schlüssel (key). Wenn eine Liste oder eine Datei nicht sortiert ist, dann bleibt als einzig mögliche Lösung der sequentielle Suchweg. Dies ist aber eine zeitraubende Angelegenheit. Schnellere Verfahren verkürzen die Bearbeitungsvorgänge erheblich und stellen geeignete Werkzeuge für die Lösung vieler Probleme dar. Auch hier gibt es interne und externe Verfahren.

Die Programme werden im SVI-BASIC präsentiert. Doch gerade weil in letzter Zeit Compiler für höhere Programmiersprachen wie PASCAL oder COBOL auf den Markt gekommen sind, bringen wir zusätzlich die Flußdiagramme, um den Besitzern solcher Compiler die Möglichkeit zu geben, diese Verfahren in ihrer Lieblingssprache zu implementieren. Dabei werden wir nicht ausschließlich die von ANSI (American National Standards Institute) vorgeschlagenen Zeichen verwenden, sondern zusätzlich mehrsagende Zeichen wie die von Forsythe, Keenan, Organick und Stenberg gebrauchen, die im Buch "Computer Science" angeführt sind. Eine Zusammenstellung finden Sie auf Seite 15.

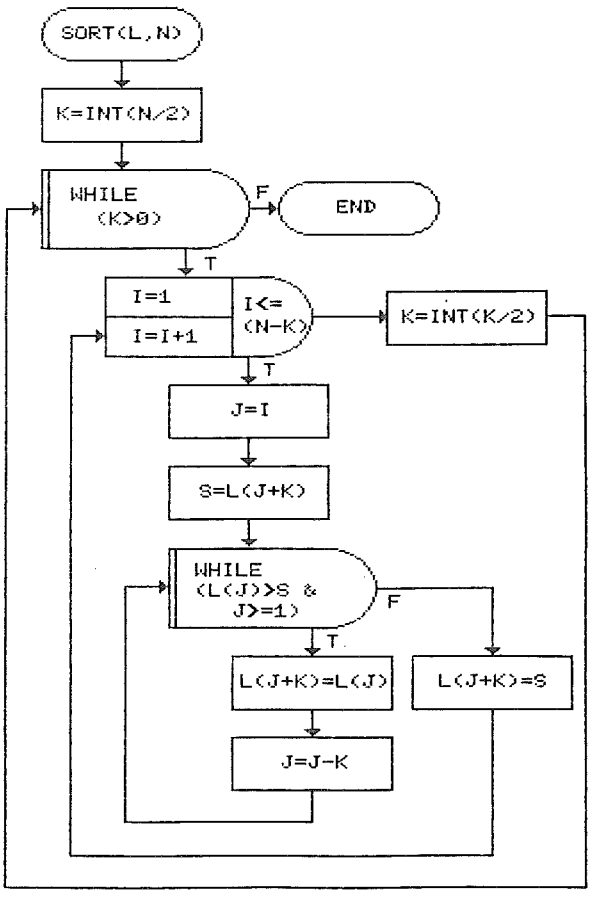
Zum Binarysearch-Algorithmus:

In einer sortierten Liste wird nach dem Wert des Schlüssels (key) gesucht. Vorerst wird die Liste mit LO und HI markiert. LO entspricht dem Index 1(=erstes Glied der Liste) und HI dem Index N(=letztes Element).

Nun wird die Liste halbiert, und zwar nach der Formel MITTE=INT((LO+HI)/2). Danach wird der Schlüssel mit dem in der MITTE der Liste gelegenen Element verglichen. Wenn sie übereinstimmen, dann haben wir ja den gewünschten Platz gefunden. Wenn der Schlüssel aber kleiner als der Wert von MITTE ist, dann setzen wir neue Suchgrenzen - LO bleibt gleich, HI=MITTE-1, neue MITTE=((LO+HI)/2) - und wiederholen wie oben. Wenn der Schlüssel größer als der Wert des Elements in der Position MITTE ist, dann ändern sich die neuen Suchgrenzen wie folgt: LO=MITTE+1, HI bleibt gleich und MITTE=((LO+HI)/2). Der Faltungsvorgang wird solange wiederholt, bis wir entweder gefunden haben, wonach wir suchten, oder bis HI kleiner als LO wird, wobei das bedeutet, daß der gesuchte Schlüssel nicht in unserer Liste ist.

Zum Shellsort-Algorithmus:

Beim sogenannten "Bubblesort" werden nebeneinanderliegende Elemente verglichen. Wenn das Element mit dem Index "N" größer ist als das mit dem Index "N+1", dann werden sie ausgetauscht. Die Schritte beim Bubblesort sind dann mit $Z=(N^2-N)$ anzugeben. Shellsort vergleicht nicht von Anfang an nebeneinanderliegende Elemente, sondern Elemente, die um K=INT(N/2) auseinander liegen. Wenn es notwendig ist, dann tauscht dieses Programm dann auch die beiden verglichenen Daten aus. Dieser Vorgang wiederholt sich, nachdem K halbiert wurde, solange K größer Null ist. Die Anzahl der gleichartigen Schritte und damit die Zeit ist $Z<2*N*^2\log(N)$. Um einen direkten Vergleich zu erhalten, nehmen wir an, daß eine Liste mit hundert Elementen zu sortieren ist. Bubblesort braucht 9900 Zeiteinheiten, Shellsort kommt mit weniger als 1300 Zeiteinheiten aus.




```
1000  '''''''''''''''''''''''''
1010  '    binary search      '
1020  '     written by        '
1030  '      C.Gaganas        '
1040  '                       '
1050  '''''''''''''''''''''''''
1060 CLS
1070 DEFINT A-Z
1080 MX=52 ' Maximalanzahl der Elemente
1090 DIM A$(MX)
1100 ' Test Data
1110 DATA A,B,C,D,E,F,G,H,I,J,K,L,M
1120 DATA N,O,P,Q,R,S,T,U,V,W,X,Y,Z
1130 DATA a,b,c,d,e,f,g,h,i,j,k,l,m
1140 DATA n,o,p,q,r,s,t,u,v,w,x,y,z
1150 FOR I=1 TO MX
1160 READ A$(I)
1170 NEXT
1180 INPUT "Schluessel";SCH$
1190 PLTZ=0
1200 GEF=0
1210 MITTE=0
1220 LO=1
1230 HI=MX
1240 IF HI < LO OR GEF=1 THEN GOTO 1280
1250 MITTE=(LO+HI)\2
1260 IF SCH$=A$(MITTE) THEN GEF=1:PLTZ=MITTE
:GOTO 1280 ELSE IF SCH$ > A$(MITTE) THEN LO=
MITTE+1 ELSE HI=MITTE-1
1270 GOTO 1240
1280 IF GEF=0 THEN PRINT "Nicht gefunden" EL
SE PRINT "Gefunden in Position:";PLTZ;"VALUE
:";A$(PLTZ)
1290 END
```

```
1000  '''''''''''''''''''''''''''''''
1010  ' Shellsort einer Liste L(n)  '
1020  ' written by C.Gaganas        '
1025  '                            '
1030  '''''''''''''''''''''''''''''''
1035 PRINT "Index  Array "
1040 DEFINT A-Z
1050 CLS
1060 PRINT "=========================="
1070 PRINT "= SHELL SORT ALGORITHMUS ="
1080 PRINT "=  written by C.Gaganas  ="
1090 PRINT "=========================="
1100 PRINT:PRINT
1110 INPUT "Wie viele Elemente";N
1120 DIM L(N),U(N)
1130 GOSUB 1280 ' Test Data
1140 K=INT(N/2)
1150 IF K<=0 THEN GOTO 1250
1160 FOR I=1 TO N-K
1170 J=I
1180 S=L(J+K)
1190 IF J <1 THEN GOTO 1210
1200 IF L(J)>S AND J>=1 THEN L(J+K)=L(J):J=J
-K:GOTO 1190
1210 L(J+K)=S
1220 NEXT I
1230 K=INT(K/2)
1240 GOTO 1150
1250 GOSUB 1345 ' Sortiertes Feld ausgeben
1260 END
1270 ' Daten fuer das Test-Array
1280 DATA 125,398,34,19845,345,776,129,2,55,
67,1,457,349,2,4,1,1000,683,11,99
1285 DATA 12,1893,3129,567,9910,2345,916,55,
32,0,576,943,24,13,1453,836,11199
1287 DATA 12398,3419,8453,4577,6,129,24,5367
,1457,3492,4110,300,683
1288 DATA 8398,9999,3584,7777,9,921,24,6677,
4157,9292,1010,301,863
1290 RESTORE 1280
1300 FOR I=1 TO N
1310 READ L(I):U(I)=L(I)
1320 NEXT I
1330 RETURN
1340 ' Sortiertes Feld ausgeben
1345 CLS
1347 PRINT "  Index  Unsortiert  Sortiert"
1348 PRINT "  ---------------------------"
1350 FOR I=1 TO N
1360 PRINT USING "#######    ";I;U(I);L(I)
1370 NEXT I
1380 RETURN
```

```
*****************************************
*                                       *
*                QMERGE                 *
*                                       *
*****************************************
```

Der Befehl MERGE gestattet es, BASIC-Programme oder Teile derer zu dem aktuellen Programm hinzuzuladen, ohne daß der Programmspeicher vorher gelöscht wird. Leider setzt der Befehl MERGE voraus, daß das Programm in einem bestimmten Format abgespeichert ist. Die Befehlswörter werden nicht in ein oder zwei Bytes lange Token umgewandelt und auch Zahlen und Konstante werden so abgespeichert, wie sie in einem Listing erscheinen. Dadurch ist der Interpreter beim Laden gezwungen, jede Zeile einzulesen, sie umzuwandeln (Befehlswort in Token) und sie in den Programmspeicher einzufügen. Deshalb ist MERGE langsam. Dafür werden die Zeilen auch dort eingefügt, wo sie hingehören.

Meine Methode, diesen Prozeß des Hinzuladens schneller zu machen, liegt darin, den BASIC-Anfang direkt hinter die letzte Zeile des aktuellen Programmes zu setzen, dann das andere Programm mit LOAD oder CLOAD zu laden und dann den BASIC-Anfang wieder hinunterzusetzen. Dafür muß man aber in Kauf nehmen, daß die Zeilen eventuell nicht richtig eingefügt werden. Daraus resultiert, daß man Programmteile nur in aufsteigenden Zeilennummern hinzuladen sollte. Ansonsten könnte es Ihnen nämlich passieren, daß auf die Zeilen 1500 bis 2410 auf einmal die Zeilen 142 bis 855 folgen. Mit RENUM könnten Sie keine Abhilfe schaffen, weil auch die Verteilung im Programmspeicher falsch wäre.

Um nun ein weiteres Programm hinzuzuladen, müssen Sie, nachdem Sie das Maschinenprogramm eingegeben haben, mit 'PRINT USR1(0)' den BASIC-Anfang verschieben. Danach laden Sie das Programm (mit CLOAD oder LOAD), und hernach setzen Sie mit 'PRINT USR2(0)' den BASIC-Anfang wieder hinunter. Um also ein Programm hinzuzuladen sind folgende Schritte nötig: 1) PRINT USR1(0)
2) Laden des Programmes(mit LOAD oder CLOAD)
3) PRINT USR2(0).

Zum Maschinenprogramm: Den Assembler so umzumodeln, daß er ein formatiertes Listing einschließlich Opcodes ausdruckt, ist schwierig, und deshalb bleibts vorläufig, wie es ist. Deshalb lasse ich auch das Hex-Dump abdrucken, damit Sie nicht in seitenlangen Tabellen nachgucken müssen. Die Routine USR1 müssen Sie mit 'DEFUSR1=&HFFC0' und USR2 mit 'DEFUSR2=&HFFC2' definieren.

Der Aufruf von USR1 mittels PRINT zeigt nichts an, das Programm springt nämlich zum BASIC-Befehl NEW. Bei USR2 kommt das übergebene Argument wieder zurück. Sollten Sie die Reihenfolge von USR1-USR2 nicht einhalten, kommt es zur Fehlermeldung 'Illegal function call'. Diese Fehlererkennung wurde deshalb so gewählt, weil bei zweimaligen USR1-Aufruf der momentane BASIC-Anfang abgespeichert wird. Allerdings könnte dann USR2 nicht mehr erkennen, wo der ursprüngliche BASIC-Anfang gelegen ist und würde deshalb einen falschen Wert in die Systemvariable des BASIC-Anfangs eintragen.

Es wird Sie vielleicht wundern, daß die zwei Routinen ganz oben im Speicher untergebracht sind. Dies wurde deshalb so gewählt, um sie nicht irgendwo mitten im RAM liegen zu haben.

```
FFC0  3A FB FF A7 C2 9E 0F CD  :.......
FFC8  F3 FF 2A 4A F5 22 F9 FF  ..*J."..
FFD0  2A EE F7 2B 2B 2B 22 4A  *..+++"J
FFD8  F5 ED 7B DD F7 3B 3B C3  ..{..;;.
FFE0  57 65 3A FB FF A7 CA 9E  We:.....
FFE8  0F CD F4 FF 2A F9 FF 22  ....*.."
FFF0  4A F5 C9 3E AF 32 FB FF  J..>.2..
FFF8  C9 00 00 00 00 00 4D 53  ......MS
```

```
10 REM          org ffc0h
11 REM          ;
12 ' BAS_BEG und PRG_END enstprechen
13 ' den Angaben der letzten Clubzeit-
14 ' schrift (Seite 13).
15 REM bas_beg  equ f54ah
16 REM prg_end  equ f7eeh
17 ' NEW : Einsprungsadresse des NEW-
18 ' Befehls.
19 REM new      equ 6557h
20 ' STK_PTR enthaelt den Stapelzeiger,
21 ' er wird vor jeder Befehlsausfuehrung
22 ' dort abgespeichert.
23 REM stk_ptr  equ f7ddh
24 REM illegal  equ 0f9eh
25 ' FLAG_OFF : Einsprung um FLAG zu
26 ' loeschen.
27 REM flag_off equ flag_on+1
28 REM          ;
29 ' Bei FFC0H erfolgt der Einsprung, um
30 ' den BASIC-Anfang direkt hinter die
31 ' letzte Zeile zu verlegen. Die ganze
32 ' Spielerei mit FLAG soll nur bewirken,
33 ' dass pro USR1 nur ein USR2 aufgerufen
34 ' wird.
35 REM USR1       ld a,(flag)
36 REM           and a
37 REM           jp nz,illegal
38 REM           call flag_on
39 ' Die Variable OLD_BAS dient zum Retten
40 ' des aktuellen BASIC-Anfangs.
41 REM           ld hl,(bas_beg)
42 REM           ld (old_bas),hl
43 REM           ld hl,(prg_end)
44 REM           dec hl
45 REM           dec hl
46 REM           dec hl
47 ' Der neue BASIC-Anfang wird eingetragen.
48 REM           ld (bas_beg),hl
49 ' Jetzt wird der Stapel wie fuer eine
50 ' Interpreterroutine hergerichtet und
51 ' mit 'JP NEW' der Programmspeicher
52 ' geloescht, ohne Ihr Programm wirklich
53 ' zu loeschen. Klar ?
54 REM           ld sp,(stk_ptr)
55 REM           dec sp
56 REM           dec sp
57 REM           jp new
58 REM           ;
59 ' Wieder die Spielerei mit FLAG, dann
60 ' wird der alte BASIC-Anfang neu ein-
61 ' getragen.
62 REM USR2      ld a,(flag)
63 REM           and a
64 REM           jp z,illegal
65 REM           call flag_off
66 REM           ld hl,(old_bas)
67 REM           ld (bas_beg),hl
68 REM           ret
69 REM           ;
70 ' FLAG_ON und FLAG_OFF sind eine kleine
71 ' Demonstration dafuer, welche Tricks
72 ' die MICROSOFT-Programmierer verwenden.
73 ' Der Aufruf von FLAG_ON duerfte ja klar
74 ' sein, aber FLAG_OFF ?
75 ' Nun, bei den EQU-Anweisungen wurde ja
76 ' FLAG_OFF als FLAG_ON+1 definiert, und
77 ' der Einsprung bei FLAG_OFF erfolgt
78 ' bei einem 'XOR A' (=AFH)
79 ' In Falle von FLAG_ON hat der Akku den
80 ' Inhalt AFH und im Falle FLAG_OFF 00H.
81 ' Danach wird FLAG mit dem Akku geladen
82 ' und zurueckgesprungen. Im System
83 ' passiert gleiches bei TRON/TROFF,
84 ' AUTO, Cursor ON/OFF etc.
85 REM flag_on   ld a,afh
86 REM           ld (flag),a
87 REM           ret
88 REM           ;
89 REM old_bas   defw 00
90 REM flag      defb 00
91 REM           ;
92 REM           end
```

Tips und Tricks für den SVI-728

Wir wollen keinen SVI-Computer vernachlässigen, deshalb werden wir von nun an auch für den SVI-728 Wissenswertes bringen. In der ersten Folge dieser neuen "Serie" behandeln wir das Schreiben von Texten am Graphik-Screen. Was nämlich bei der "3er-Reihe" mit LOCATE funktioniert, ist beim SVI-728 etwas schwieriger.

Die User von SVI-328-Computersystemen haben es ja leicht, wenn sie am Graphikschirm Text ausgeben wollen. Einfach LOCATE angeben und mit PRINT den Text schreiben, fertig ist die Angelegenheit. Beim SVI-728 kann man das LOCATE jedoch nicht für die Graphik verwenden. Hier muß man andere Wege einschlagen.

Wir lösen das Problem ganz einfach, indem wir eine Datei eröffnen und diese auf den Bildschirm schicken. Im untenstehenden Programm sieht man in Zeile 110 die OPEN-Anweisung. "GRP:" spricht die Graphik an. Als nächstes müssen wir den Punkt positionieren, ab dem geschrieben werden soll. Normalerweise kann man dies am einfachsten mit PSET machen. Theoretisch lassen sich aber auch LINE, CIRCLE und alle anderen Graphikbefehle dazu verwenden. Man kann auch, wenn man den Text an irgendeine Graphik unmittelbar anschließen will, überhaupt keine Positionierung vornehmen, der Computer nimmt dann den letzten gezeichneten Punkt als Anfangsort.

Nachdem wir die Stelle markiert haben, können wir Datensätze auf den Bildschirm schicken. Dies geht, wie allgemeinbekannt, mit "PRINT #X,...". Man kann auch die Flächen zuerst säubern, bevor man etwas schreibt. Der Computer tut dies nicht von alleine. Mit einem "LINE...,Backgroundf.,BF" ist dies leicht bewerkstelligt.

Das untenstehende Programm zeigt die wichtigsten Funktionen. Das CLOSE in 200 wird erst dann ausgeführt, wenn man nichts mehr auf den Bildschirm schicken will. Selbstverständlich kann man die Datei auch während des ganzen Programms geöffnet lassen. Nur muß man dann darauf achten, daß keine zweite Datei mit gleicher Nummer geöffnet wird.

Es liegt nun am interessierten Leser, das Programm für seine eigenen Bedürfnisse entsprechend abzuändern.

```
100 SCREEN 2
110 OPEN "GRP:" FOR OUTPUT AS #1
120 PSET (90,90)
130 PRINT #1,"Guten Tag!"
140 PSET (90,100)
150 PRINT #1,"Guten Morgen!"
160 FOR T=1 TO 1000:NEXT
170 LINE (90,100)-(215,108),4,BF
180 PSET (90,100)
190 PRINT #1,"Schoenes Wetter!"
200 CLOSE #1
210 GOTO 210
```

****

Die SCREEN-Aweisung beim SVI-728:

Beim SVI-328 kennen wir drei verschiedene Arten der Bildschirmdarstellung. Beim MSX-Computer SVI-728 gibt es deren vier, und zwar:

SCREEN 0: Textmodus, 40 Zeichen pro Zeile, kann mit dem WIDTH-Befehl bis auf 1 Zeichen/Zeile reduziert werden.
SCREEN 1: Textmodus, 32 Zeichen pro Zeile, kann mit dem WIDTH-Befehl bis auf 1 Zeichen/Zeile reduziert werden.
SCREEN 2: hochauflösende Graphik (SCREEN 1 beim SVI-328), 256*192 Bildpunkte
SCREEN 3: niedrigauflösende Graphik (entspricht SCREEN 2 beim SVI-328), 64*48 Bildpunkte.

Tastatur-Klick ein- und ausschalten:

Beim SVI-728 wird der Tastatur-Klick durch Setzen bzw. Löschen des 3. Parameters in der SCREEN-Anweisung ein- bzw. ausgeschaltet (zum Vergleich: beim SVI-328 "CLICK ON" bzw. "CLICK OFF").

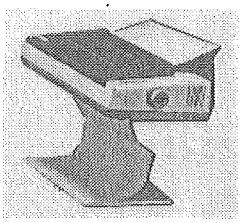
Anzeige der Funktionstastenbelegung:

Die Anzeige der Belegung der Funktionstasten kann mit "KEY OFF" ausgeschaltet werden. Mit "KEY ON" wird die Anzeige wieder eingeschaltet. Beim SVI-328 ist dafür der 2. Parameter der SCREEN-Anweisung im Textmodus zuständig.

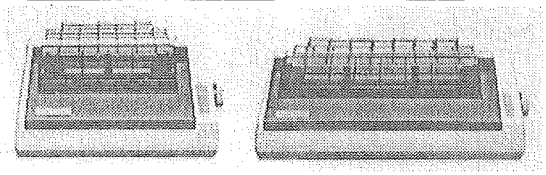
W i r h a b e n i h n !

den N e u e n von s t a r

S G 1 0 / 1 5



* 120 Zeichen pro Sekunde
* Near letter Quality
* IBM-PC kompatibel
* 7 Graphik Ausdruckarten
* Papiereinzug auch von unten

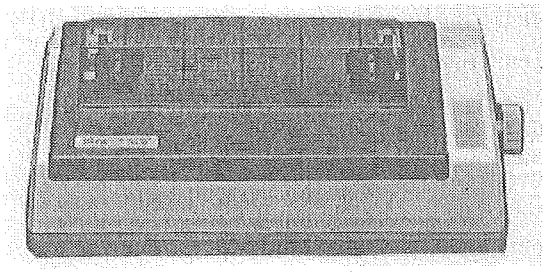


SG-10 SG-15

Mit der neuen s t a r Druckerserie SG 10/15 setzt s t a r neue Maßstäbe.Neben den bekannten s t a r Attributen wie Selbsttest, frei programmierbaren Zeichen,permanentes Unterstreichen und Einzelnadelansteuerung bieten wir jetzt auch Hex-Dump,durchgestrichene Null und Zeilenspeicher Dump.

16 k-Byte Puffer

lassen beim SG 15 kaum noch Wünsche offen. Und s t a r Drucker sind schnell wie ein Südwester,aber kaum lauter als ein laues Sommerlüfterl.



Sie kennen Ihn noch nicht ?
Dann müßen sie Ihn kennenlernen den
S G 1 0 von s t a r
bei Ihrem s t a r Fachhändler schon um

erstarliche 9.576.- incl.MwSt.

star

```
*****************************************
*                                       *
*              Programme                *
*                                       *
*****************************************
```

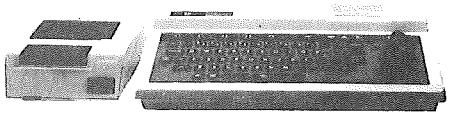
Sie erinnern sich vielleicht noch an das Suchrätsel in Heft 1/84. Es hat unser Clubmitglied Herrn Stocker zur Entwicklung eines "Suchrätsel-Generators" inspiriert. Herr Gaganas ermöglicht mit seinem Programm die Eingabe eines unsichtbaren Codes.

```
10 REM ********************************
20 REM **                            **
30 REM **       WORTRAETSEL          **
40 REM **  (c) by Andreas STOCKER    **
50 REM **     WIEN, 1984 10 16       **
60 REM **                            **
70 REM ********************************
80 RESTORE
90 PRINT :PRINT
100 CLEAR 3000
110 CLS
120 DEF FNA(Z)=INT(RND(1)*Z+1)
130 TW=35
140 INPUT "MOECHTEST DU DIE LOESUNG";X$
150 CLS :INPUT "WIE BREIT SOLL DAS RAETSEL S
EIN";W:MD=W
160 IF W+2<=18 THEN 180
170 PRINT "DAS PASST NICHT ":GOTO 150
180 IF W<1 THEN 150
190 CLS :INPUT "WELCHE HOEHE SOLL DAS RAETSE
L HABEN";L:IF L>W THEN MD=L
200 IF L<1 THEN 190
210 IF L>19GOTO 190
220 CLS :PRINT "WAS IST DIE MAXIMALE ANZAHL
VON WORTEN":INPUT "IM RAETSEL";M
230 IF M>=2 THEN 250
240 PRINT "TUT MIR LEID ES MUESSEN MINDESTEN
S ZWEIWORTE SEIN":GOTO 220
250 PRINT
260 DIM A$(L,W),W$(M)
270 DIM W(M,3),DXY(8,2),DD(28)
280 CLS: INPUT "GIB DIE UEBERSCHRIFT DES RAE
TSELS AN ";XY$
290 CLS: PRINT "GIB NACH JEDEM FRAGEZEICHEN
EIN WORT   EIN"
300 PRINT "TIPPE [-] EIN UM DAS VORHERGEHEND
E WORTNOCH MAL EINZUGEBEN.
 TIPPE [.] WENN DIR KEINE NEUEN WOERTER MEHR
 EINFALLEN."
310 FOR I=1TO M
320 INPUT T$:IF T$="-" THEN I=I-1:PRINT "WIE
DERHOLE ";W$(I);"...":GOTO 320
330 IF T$="." THEN M=I-1:GOTO 530
340 IF LEN(T$)=0 THEN PRINT "EINGABE-FEHLER;
 WIEDERHOLE:":GOTO 320
350 J=1
360 TE$=MID$ (T$,J,1):IF TE$="a" AND TE$<="z
" THEN 430
370 IF TE$<"A" OR TE$>"Z" THEN 390
380 T$=LEFT$ (T$,J-1)+CHR$ (ASC (MID$ (T$,J,
1))+32)+RIGHT$ (T$,LEN (T$)-J):GOTO 430
390 IF TE$=T$ THEN T$="":GOTO 340
400 IF J=LEN (T$) THEN T$=LEFT$ (T$,J-1):GOT
O 440
410 IF J=1 THEN T$=RIGHT$ (T$,LEN (T$)-1):J=
J-1:GOTO 430
420 T$=LEFT$ (T$,J-1)+RIGHT$ (T$,LEN (T$)-J)
:J=J-1
430 J=J+1:IF J<=LEN (T$) THEN 360
440 PRINT "-";T$;"-"
450 IF LEN (T$)<=MD THEN 480
460 PRINT "TUT MIT LEID, DAS IST ZU LANG."
470 PRINT "VERSUCHE EIN ANDERES: ":GOTO 320
480 FOR IZ=1 TO I-1:IF W$(IZ)<>T$ THEN NEXT
IZ:GOTO500
490 PRINT "DU HAST DIES SCHON EINMAL EINGEGE
BEN.  VERSUCHE EIN ANDERES;":GOTO 320
500 W$(I)=T$
510 NEXT I
520 PRINT "FERTIG...";M;"WORTE."
530 PRINT "LASS' MICH JETZT UEBERLEGEN.....
"
540 FOR I=1TO M-1
550 FOR J=I+1 TO M
560 IF LEN (W$(I)) <LEN (W$(J)) THEN HZ$=W$(
I):W$(I)=W$(J):W$(J)=HZ$
570 NEXT J:NEXT I
580 FOR I=1TO 8:READ DXY(I,1),DXY(I,2):NEXT
I
590 FOR I=1TO 28:READ DD(I):NEXTI
600 DATA 0,1,1,1,1,0,1,-1,0,-1,-1,-1,-1,0,-1
,1
610 DATA 2,4,6,8,2,4,6,8,2,4,6,8,2,4,6,8,2,4
,6,8,2,4,6,8,1,3,5,7
620 FOR I=1TO M
630 LN=LEN (W$(I))
640 NT=0
650 SD=DD(FNA(28))
660 SX=FNA(W):X1=SX+(LN-1)*DXY(SD,1):IF X1 <
1 OR X1>W THEN 650
670 SY=FNA(L):X1=SY+(LN-1)*DXY(SD,2):IF X1 <
1 OR X1>L THEN 650
680 NT=NT+1:IF NT<>W*L*2 THEN 730
690 PRINT "KONNTE ";W$(I):PRINT "NICHT IN DA
S RAETSEL EINBAUEN."
700 INPUT "SOLL ICH NOCH MAL VON VORNE ANFAN
GEN";A$
710 IF LEFT$(A$,1)="J" THEN 620
720 W$(I)="":GOTO 810
730 J=SY:K=SX
740 FOR P=1TO LN
750 IF LEN(A$(J,K)) AND A$(J,K)<>MID$ (W$(I)
,P,1) THEN 650
760 J=J+DXY(SD,2):K=K+DXY(SD,1):NEXT P
770 J=SY:K=SX
780 FOR P=1TO LN:A$(J,K)=MID$ (W$(I),P,1)
790 J=J+DXY(SD,2):K=K+DXY(SD,1):NEXT P
800 W(I,1)=SX:W(I,2)=SY:W(I,3)=SD
810 NEXTI
820 FOR I=1TO L
830 FOR J=1TO W
840 IF A$(I,J)="" THEN A$(I,J)=CHR$(FNA(26)+
96)
850 NEXT J:NEXT I
860 FOR I=1TO M-1:FOR J=I+1TO M
870 IF W$(I)<=W$(J) THEN 900
880 HZ$=W$(I):W$(I)=W$(J):W$(J)=HZ$
890 FOR K=1TO 3:HZ=W(I,K):W(I,K)=W(J,K):W(J,
K)=HZ:NEXT K
900 NEXT J:NEXT I
910 GOSUB 920:GOTO 1020
920 INPUT A$:PRINT
930 CLS
940 T=(TW-2*W)/2:PRINT
950 PRINT
960 PRINTTAB((TW-LEN(XY$))/2);XY$
970 PRINT :PRINT
980 FOR J=1TO L:PRINTTAB(T);
990 FOR K=1TO W:IF A$(J,K)="."THEN PRINT ".
";:GOTO 1010
1000 PRINT CHR$ (ASC (A$(J,K))-32);" ";
1010 NEXT K:PRINT :NEXT J
1020 IF LEFT$ (X$,1)="J" THEN 1040
1030 GOTO 1140
1040 REM
1050 FOR I=1TO L:FOR J=1TO W:A$(I,J)=".":NEX
T J:NEXT I
1060 FOR I=1TO M
1070 LN=LEN (W$(I)):J=W(I,2):K=W(I,1)
1080 FOR P=1TO LN
1090 A$(J,K)=MID$(W$(I),P,1)
1100 J=J+DXY(W(I,3),2):K=K+DXY(W(I,3),1):NEX
T P
1110 NEXT I
1120 XY$="HIER IST DIE LOESUNG"
1130 GOSUB 920
1140 END
```

```
1000  ''''''''''''''''''''''
1010  '      password      '
1020  '    written by      '
1030  '    C. Gaganas      '
1040  ''''''''''''''''''''''
1050 CLS
1060 A$=""
1070 I$=INKEY$
1080 IF I$="" THEN GOTO 1070
1090 IF I$=CHR$(13) THEN GOTO 1120
1100 IF I$=CHR$(8) AND A$<>"" THEN PRINT I$;
:A$=LEFT$(A$,LEN(A$)-1):GOTO 1070
1110 IF VAL(I$)<32 OR VAL(I$)>127 THEN A$=A$
+I$:PRINT ".";:GOTO 1070
1120 PRINT:PRINT A$
1130 GOTO 1060
```

Die Super-Computer.

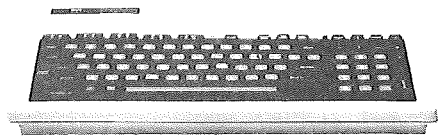
SVI



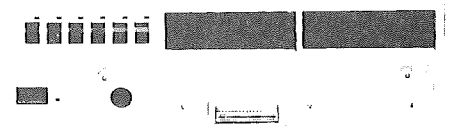
SVI-318 32 K RAM, erweiterbar bis **144 K RAM.** Erweitertes MICROSOFT-BASIC, integrierte Cursorsteuerung **öS 4.990,—**

SVI-904 Datenrecorder, 1800 Baud, Zählwerk, Laufwerksteuerung durch SVI-318 oder 328 inkl. 2 Spielkassetten **öS 990,—**

SVI-318-Set bestehend aus SVI-318 Basisgerät (32 K RAM, MICROSOFT-BASIC), SVI-904 Datenrecorder und Softwarepaket mit 5 Kassetten **öS 5.890,—**



SVI-328 32 K ROM, 80 K RAM, Erweitertes MICROSOFT-BASIC, Schreibmaschinentastatur, 10 Funktionstasten, 10er-Block **öS 6.990,—**



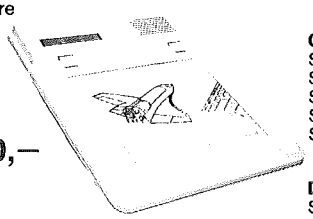
Super-Expander SVI-605, ein eingebautes Diskettenlaufwerk (160 K), Centroniks-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 **öS 12.990,—**

Super-Expander SVI-605 A, zwei eingebaute Diskettenlaufwerke (je 160 K), Centronics-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 **öS 18.990,—**

Super-Expander SVI-605 B, mit Supersoftware-Paket, zwei eingebaute Diskettenlaufwerke (je 320 K), Centronics-Interface, 4 freie Steckplätze, Betriebssystem CP/M 2.2 WordStar, Mailmerge, CalcStar ReportStar, DataStar **öS 25.990,—**

SVI-328 Pro Profisystem bestehend aus: Computer SVI-328, Super-Expander SVI-605 A (inkl. WordStar, Mailmerge, CalcStar, DataStar, ReportStar) Betriebssystem CP/M 2.2. 80-Zeichenkarte SVI-806, **öS 32.290,—**

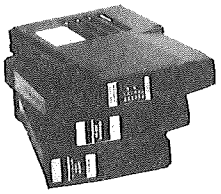
Grafik-Tablett SVI-105, 186 x 158 mm Zeichenfläche, Kassette mit Anwender-Software inkl. **öS 2.090,—**



Erweiterungskarten für SVI-605, A, B

SVI-803 16 K-Speicher-Erweiterung (für SVI-318) **öS 890,—**

SVI-805 RS 232, serielle Schnittstelle **öS 1.790,—**



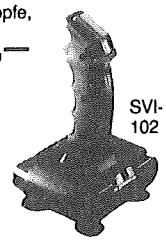
SVI-806 80-Zeichenkarte **öS 2.390,—**

SVI 807 64-K-RAM Speichererweiterung **öS 3.290,—**

Joystick SVI-101, zwei Feuerknöpfe, vier Saugfüße, ergonomischer Handgriff **öS 390,—**

Joystick SVI-102, automatisches Dauerfeuer, zwei Feuerknöpfe, vier Saugfüße **öS 490,—**

(Joystick SVI-101 und SVI-102 auch für Atari und Commodore geeignet)



SVI-102

Die Software

		öS
Kassettensoftware		
SVI-K 110	Einführung in das SVI-Basic inkl. 40seitigem Handbuch	390,—
SVI-K 115	SVI-Dateiverwaltung	290,—
SVI-K 122	SVI-Text	390,—
SVI-K 129	SVI-Termin	290,—
SVI-K 146	Disassembler	590,—
SVI-K 147	Maschinen-Code-Monitor	590,—
SVI-K 148	SVI-Spritegenerator	290,—
SVI-K 149	SVI-Zeichengenerator	290,—
SVI-K 179	Old-Mac-Farmer	390,—
SVI-K 180	Tetra Horror	390,—
SVI-K 181	Tele Bunny	390,—
SVI-K 182	Turboat	390,—
SVI-K 183	SASA	390,—
SVI-K 184	NINJA	390,—
SVI-K 185	Kung-Fu-Master	390,—
SVI-K 188	Armoured Assault	290,—
SVI-K 189	Spectron	290,—
Cartridgesoftware		
SVI-C 220	Sector Alpha	790,—
SVI-C 232	Frantic-Freddy	790,—
SVI-C 236	Music-Mentor	990,—
SVI-C 237	Super-Cross-Force	790,—
SVI-C 291	Flipper-Slipper	790,—
Diskettensoftware		
SVI-D 310	Einf. in das SVI-Basic	590,—
SVI-D 315	SVI-Dateiverwaltung	390,—
SVI-D 322	SVI-Text	590,—
SVI-D 334	SVI-Lager	390,—
SVI-D 348	SVI-Toolkit I (SVI-Spritegenerator u. SVI-Zeichengenerator)	590,—
SVI-D 349	SVI-Toolkit II (Disassembler und Maschinen-Code-Monitor)	1.190,—
SVI-D 359	LISP 80	1.690,—
SVI-D 360	C-Compiler	1.690,—
SVI-D 361	Turbo-PASCAL (Version 2.0)	2.390,—
SVI-D 318	Nevada-FORTRAN (Compiler)	1.390,—
SVI-D 382	Nevada-COBOL (Compiler)	1.390,—
SVI-D 383	Nevada-PILOT (Interpreter)	1.390,—
SVI-D 384	Nevada-EDIT (Editor)	1.390,—
SVI-D 388	Old-Mac-Farmer	390,—
SVI-D 389	Tetra Horror	390,—
SVI-D 390	Tele Bunny	390,—
SVI-D 391	Turboat	390,—
SVI-D 392	SASA	390,—
SVI-D 393	NINJA	390,—
SVI-D 394	Kung-Fu-Master	390,—

Druckeranschlußkabel SVI-205, 1,5 m, für parallele Schnittstelle **öS 590,—**

Diskettenlaufwerk SVI-905, 160 K, zur Erweiterung des Super-Expanders SVI-605 **öS 6.490,—**

Centronics-Interface SVI-802 mit Kabel 205 zum Anschluß an Mini-Expander SVI-602 **öS 3.080,—**

Preise incl. MWSt.

SVI-FACHBERATUNG UND VERKAUF BEI:

Großhandel, Facheinzelhandel **BASTL-COMPUTER-SYSTEME** **2700 Wr. Neustadt,** Hauptplatz 5 Tel. (0 26 22) 22 7 20 - 59 80 Telex 16 522	**DAHMS-PRAKTIKER-ELEKTRONIK** Pilgramgasse 11 **1050 Wien** Tel. (02 22) 54 34 21	**ESV ELEKTROTECHNISCHER SERVICE MBH.** Bayerhamerstraße 19—21 **5020 Salzburg** Tel. (06 62) 74 7 51	**SCHILLER MICRO-COM-BOUTIQUE** Fasangasse 21 **1030 Wien** Tel. (02 22) 78 35 99, 78 56 61	**TARGET ELECTRONIC** Bergstraße 6 **6900 Bregenz** Tel. (0 55 74) 23 7 18
BYTE COMPUTER Favoritenstraße 20 **1040 Wien** Tel. (02 22) 65 73 42 Telex 132 827	**EDV-STUDIO PORSCH** Kinderspittalgasse 13 **1090 Wien** Tel. (02 22) 42 63 44	**FEDCON ELEKTRONIK** Ing. Franz Krenn Kanalplatz 86 **9400 Wolfsberg** Tel. (0 43 52) 42 73	**TARGET ELECTRONIC** Reichsstraße 123a **6800 Feldkirch** Tel. (0 55 22) 21 5 29 Telex 52 300	**WEHSNER GMBH. COMPUTER-STUDIO** Paniglgasse 18—20 **1040 Wien** Tel. (02 22) 65 88 93, 65 78 08

GENERALVERTRETUNG FÜR ÖSTERREICH: MONACOR ELECTRONIC - VERTRIEBS-GESMBH. · 6800 FELDKIRCH · ☎ (0 55 22) 21 9 89 · TELEX 52 300 tarmo

****
* *
* SVI-Programmecke *
* *
****

KUNG FU MASTER

Im Programm "KUNG FU MASTER" muß man als Karate-Kämpfer gegen übermütige Roboter kämpfen. Im ersten Bild befindet man sich in einer Säulenhalle, die man durchqueren muß. Wo man sich gerade befindet, darüber gibt eine Skizze der Halle am rechten oberen Eck Auskunft. Die Roboter, die während des Spieles "antanzen", sind entweder hell oder dunkel gefärbt. Die dunklen werden mit der Hand erledigt (RIGHT-GRPH-Taste). Die hellen Roboter müssen mit dem Fuß "in die Knie gezwungen" werden (Space-Taste). Nach einer gewissen Anzahl von zerstörten Robotern und nach dem Durchqueren der Halle kommt man in den nächsten Modus.

Hier befinden sich Schränke. In diesen Schränken befinden sich drei Teile, die zusammen einen Schlüssel ergeben. Nun muß man die einzelnen Schränke zertrümmern, die Teile suchen und diese in die Vorlage am Bildschirm einsetzen. Dabei werden wieder die hellen Kästen mit dem Fuß und die dunklen per Hand zerstört. Daß auch hier wieder Roboter die Handlung zu stören versuchen, ist klar. Auch sie müssen erledigt werden.

Hat man in diesem Raum die Aufgabe erfüllt, kommt man wieder in eine Säulenhalle. Nach der Säulenhalle darf man wieder Schränke zertrümmern. Dies geht in diesem Rythmus weiter.

Es gibt allerdings verschiedene Schwierigkeitsgrade. So erscheinen ab der dritten Säulenhalle Messer am Bildschirm, die den Spieler töten können. Schafft man auch diese Hürde, werden die Roboter resistenter. Man muß einen Roboter sowohl mit der Hand als auch mit dem Fuß treffen. Schafft man es in einer vorgegebenen Zeit nicht, eine "Blechbüchse" in die Knie zu zwingen, wird sie zu einer rollenden Kugel, die den Karate-Kämpfer auf jeden Fall tötet.

Man wird am Anfang ziemliche Schwierigkeiten haben, die Roboter zu treffen. Deshalb hier einige Tips:

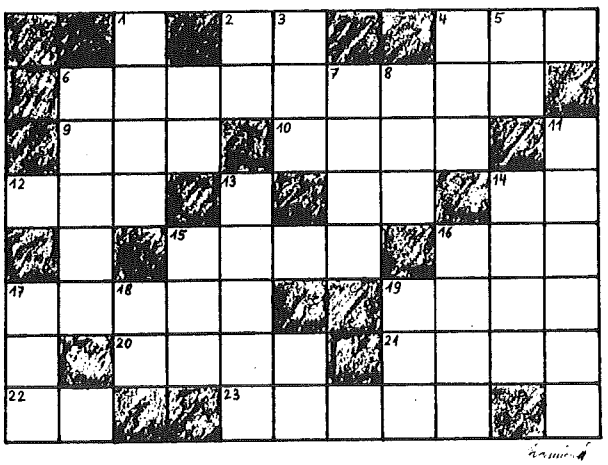
Nach Robotern, die durch die Hand getötet werden sollen, muß man schon dann ausschlagen, wenn man noch etwas weiter von ihnen entfernt ist. Der Kämpfer beugt dabei nämlich seinen Oberkörper nach vor. Die Reichweite steigt dadurch. Wenn man mit dem Fuß schlägt, ist eine wesentlich geringere Distanz notwendig.

Das Gleiche gilt für die Kästen. Mit der Hand schlägt man schon früher aus, als mit dem Fuß. Die Schränke müssen nämlich immer fast genau in der Mitte getroffen werden.

Ebenfalls die Kästen geht der letzte Tip an. Wenn ein Schlüsselteil entdeckt wurde, dürfen solange keine Schränke mehr zerschlagen werden, bis der Teil sicher an der vorgegebenen Stelle plaziert ist. Andernfalls könnte einer der anderen noch nicht entdeckten Teile verlorengehen.

Das Spiel wurde von Spectravideo erzeugt, und ist im Fachhandel erhältlich. Es gibt sowohl eine Disketten- als auch eine Kassettenversion. Beide kosten 390 Schilling.

****
* *
* Rätselecke *
* *
****



WAAGRECHT:

2 Over, Abk.
4 Niedrig, tief, engl.
6 Zeichen, engl.
9 Anweisung für Variable
10 Befehl für Laden
12 Befehl für Zeichenkettenlänge
14 Light Emission, Abk.
15 Spiel, engl.
16 Einheit im Binärsystem
17 Ausgabebefehl
19 Befehl für Fortsetzung des Programmes
20 Verknüpfungsart
21 Befehl für Beschreiben eines Speicherplatzes
22 nicht verwendet, engl. Abk.
23 Eingabebefehl

SENKRECHT:

1 Verzweigungsbefehl
2 Verknüpfungsart, engl.
3 Befehl für Ermittlung des numerischen Wertes
4 Light Egemitting Diode, Abk.
5 Oder, engl.
6 Löschbefehl
7 Verschlüsselung
8 BASIC-Funktion für Tangens
11 Brief, engl.
13 Datenansammlung, Kartei
14 Adjektiv von left
15 Ground, Abk.
16 Kleines Schiff
17 Anschlußstift
18 Input/Output, Abk.
19 Zentraleinheit, engl. Abk.

****

Kassetteninterface für den SVI-318/328!

Erinnern Sie sich noch an die SVI-Journal-Ausgabe vom November 1984? Es wurde damals über ein Kassetteninterface berichtet, mit dem man handelsübliche Kassettenrekorder an die SVI-Computer anschließen kann. Dieses Interface gibt es von nun an auch schon fertig gebaut.

Ungefähre Kosten: S 300,-

Vertrieben wird es von Rafael Razim!

Was bietet Ihnen der Spectra Video Club Austria?

- regelmäßige Clubabende mit Gelegenheit zum Informationsaustausch
- Möglichkeit zum kostenlosen Arbeiten an SVI-Computern während der Clubtreffen und zum Ausdrucken von Programmlistings
- außerordentliche Clubabende mit Vorträgen über Themen rund um Hard- und Software der Spectravideo-Computer
- kostenloser Bezug der monatlich erscheinenden Clubzeitschrift SVI-JOURNAL
- verbilligte Angebote von Spectravideo-Produkten

Mitgliedsbeitrag: Jahresbeitrag S 500,-
für Schüler, Studenten, Lehrlinge S 250,-

Nähere Informationen beim
Spectra Video Club Austria
c/o Computer-Studio
1040 Wien, Paniglgasse 18-20
Telefon (0222) 65 88 93

IMPRESSUM:

Chefredakteur: Gerhard Fally

Ständige freie Mitarbeiter: Rudolf Bolek, Philipp Ott, Rafael Razim, Heinz Schmid, Stephan Traxler, Georg Wolfbauer

Medieninhaber (Verleger): Spectra Video Club Austria, p.A. Computer-Studio, A-1040 Wien, Paniglgasse 18-20, Tel (0222) 65 88 93

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

Herausgeber: Spectra Video Club Austria, p.A. Computer-Studio, A-1040 Wien, Paniglgasse 18-20, Tel. 65 88 93

Erscheinungsweise: monatlich, jeweils zur Monatsmitte, Einzelheft S 15,-

Abonnementpreise:
jährlich S 150,-
halbjährlich S 80,-

Erscheinungsort Wien
Verlagspostamt 1040 Wien