Alleinvertrieb für Deutschland

Trommeschläger Computer GmbH
Dieses Programm ist
urheberrechtlich geschützt

PASCAL-80

Phelps Gates

NEW CLASSICS SOFTWARE
239 FOX HILL ROAD-DENVILLE NEW JERSEY 07834-TELEPHONE 201-625-8838

Deutsches Handbuch von Carsten Schmidt

********** PASCAL80 **********

Entgegen der Beschreibung im Handbuch wird PASCAL80 auf einer G-DOS-Diskette mit Minimal-Betriebssystem geliefert.

Sie sollten hiervon unbedingt eine Kopie herstellen und nie mit der Originaldiskette arbeiten!

Anwender anderer DOS sollten die benötigten Files auf eine ihrer Systemdisketten übertragen, von welcher sinnvollerweise zuvor alle nicht benötigten Programme und Systemmodule entfernt wurden (z.B.: BASIC/CMD).

Besitzer von 80-Spur-Laufwerken können die Diskette in ihrem Laufwerk lesen, wenn sie in ihrem DOS die auf der Pascal80-Diskette angegebenen Parameter für Laufwerk 1 einstellen und TI=AL wählen (Doppelschritt bei Spurwechsel).

Im Allgemeinen wird man mit PASCAL/CMD und evtl. CTRLKEY/CMD auf der Arbeitsdiskette auskommen.

Alle übrigen Files sind Utilities bzw. Demo-Programme; alle /SYS-Files gehören zu G-DOS.

==========

Bitte senden Sie uns in jedem Fall das unten anhängende Registrationsformular zurück.

Ohne Registration können wir keinerlei Unterstützung gewähren! Ebenso werden nur Originaldisketten von registrierten Besitzern im Schadensfalle ausgetauscht bzw. instandgesetzt!

PASCAL80 Registration

Seriennummer: K/83- 0065 Kaufdatum: .....

Name des Händlers: .....

Kundenanschrift

Name: .....

Straße: .....

PLZ/Ort: ....

einsenden an: Trommeschläger Computer GmbH
Postfach 2105, 5205 St. Augustin 2

********** PASCAL80 Genie III **********

Bei Anwendung des PASCAL80 auf Genie III sind folgende Besonderheiten zu beachten:

1. Die Files PASCAL/CMD und PASCAL3/CMD sind auf eine Genie III GDOS - Systemdiskette zu übertragen.
   Diese sollte zumindest die /SYS-Files GDOS, INHALT, SYS0 - SYS4, SYS6 - SYS8, SYS16, SYS17, SYS25 und SYS26 enthalten.

   Die /SYS-Files dürfen aber nicht von der PASCAL80-Diskette kopiert werden, da hier Unterschiede zwischen den DOS des Genie I und III bestehen.

   Zum Kopieren sind die Parameter dieses Systems für Laufwerk 1 folgendermaßen einzustellen:

   PD 0 1 TI=AL,TD=A,SP=40,SEK=10,EIB=2,SBIV=17,AEIV=2

2. PASCAL80 ist grundsätzlich durch PASCAL3 zu starten; d.h., auf 'Befehlseingabe:' wird PASCAL3 <NEW-LINE> eingegeben.

   Hierdurch wird PASCAL/CMD in den Speicher geladen und vor dem Start für Genie III modifiziert, sowie der 'ROM' und das DOS dahingehend verändert, daß die CTRL-Taste als solche funktioniert und die Belegung der Pfeil-Tasten wechselt.

3. Hieraus ergeben sich folgende Konsequenzen:

   a) anstelle <SHIFT>-<PFEIL UNTEN> ist <CTRL> zu benutzen

   b) <SHIFT>-<PFEIL HOCH> entspricht <CTRL>-<P>

   c) <SHIFT>-<PFEIL UNTEN> entspricht <CTRL>-<N>

   d) <SHIFT>-<PFEIL LINKS> entspricht <CTRL>-<D>

   e) <SHIFT>-<PFEIL RECHTS> entspricht <CTRL>-<O>

   f) die ursprüngliche Funktion von <SHIFT>-<PFEIL LINKS> (Backspace) ist nur noch über <CTRL>-<X> zu erreichen.

   Alle CTRL-Funktionen sind jedoch unverändert wirksam.

Für den Ablauf mit AUTHOR/SRC und AUTHCODE/CMD compilierter Programme ist PASCAL3/CMD nicht notwendig.

********** INHALT ********** Seite:

Was ist Pascal 80?

Es gibt eine ganze Anzahl von Pascal-Versionen, so zum Beispiel Standard-Pascal, UCSD-Pascal und Tiny-Pascal. Pascal 80 ist eine Implementation des Standard-Pascal für den Video-Genie 1/2 und besitzt einige Erweiterungen, wie auch einige Einschränkungen. Der wichtigste Unterschied zum Standard-Pascal ist das Fehlen der "Variant Records". Die effiziente und kompakte Programmierung von Pascal 80 ermöglicht, daß sich Monitor, Editor und Compiler gleichzeitig im Speicher befinden können, trotzdem aber noch genug Speicherplatz verbleibt, um bis zu 23K lange Programme schreiben zu können. Während das Programm läuft, sind zusätzlich 9K für Variablen und Arbeitsspeicher verfügbar. Dies ermöglicht, daß Programme geschrieben, compiliert, editiert und erneut compiliert werden können, ohne daß jedoch Zeit für Diskettenzugriff benötigt wird. Dies gibt Pascal 80 eine Benutzerfreundlichkeit, wie sie nur von interpretierten Sprachen, wie Basic oder APL bekannt war und das System zeichnet sich zudem als compilierte Sprache durch wesentlich größere Geschwindigkeiten aus. Pascal-80 ist ideal für kleinere Programme, wurde jedoch nicht als Entwicklungssystem konzipiert, welches Overlaystrukturen oder die Verwaltung großer Disk-Files benötigt.

Über dieses Handbuch

Dieses Manual ist in erster Linie eine Beschreibung , w i e man den Pascal-80 Text-Editor, Monitor und Compiler handhabt. Dies geschieht an einer Anzahl von Beispiel-Programmen, welche die Anwendung der einzelnen Strukturen von Pascal-80 erklären. Es wird jedoch ausdrücklich darauf hingewiesen, daß dieses Handbuch kein Lehrbuch für Pascal darstellt. Anwendern, die noch keine Erfahrung mit Pascal haben, empfehlen wir einschlägige Fachliteratur.

WICHTIGER HINWEIS: An einigen Stellen in diesem Handbuch wird der Benutzer angewiesen, eine CONTROL-Taste zu drücken. Da die neuen Modelle der Video-Genie-Computer über eine solche Taste nicht verfügen, wird sie simuliert, indem der Benutzer gleichzeitig <SHIFT> und <PFEIL ABWÄRTS> drückt.
Um somit CONTROL Q auszuführen, werden gleichzeitig die Tasten <SHIFT>, <PFEIL ABWÄRTS> und <Q> gedrückt.

Bei einigen DOS erzeugt (SHIFT) (PFEIL ABWÄRTS) einen Zeilenvorschub. Sollte dies stören, so kann durch Aufruf von "CTRLKEY" vor "PASCAL" Abhilfe geschaffen werden.

Wie startet man Pascal-80?

Wenn Sie Pascal-80 auf dem Genie 1 einsetzen, brauchen Sie lediglich die mitgelieferte Diskette in Laufwerk 0 einzulegen und mit RESET zu booten. Das Pascal-80-System läuft herstellermäßig unter DOSPLUS. Um Pascal-80 auf Disketten mit anderem DOS (z.B NEWDOS oder LDOS) zu kopieren, müssen Sie jedes File einzeln kopieren. Benutzen Sie den Befehl DIR um die Files auszulisten. Dann kopieren Sie mit COPY (Filename):0 :1 , gesetzt den Fall, daß sie zwei Laufwerke besitzen. Haben Sie nur ein Laufwerk, geben Sie COPY0(Filename) ein. Darauf folgen Sie bitte den auf dem Bildschirm erteilten Anweisungen.

Diese Version von Pascal-80 wurde getestet und arbeitet mit: TRSDOS, DOS Plus, NewDOS 80 (auch Version Zwei), NewDOS, LDOS und DoubleDOS. Weil Pascal-80 über eigene Tastaturroutinen verfügt, kann es jedoch vorkommen, daß bestimmte Features, wie z.B. Bildschirm-Ausdruckroutinen unter Pascal-80 mit bestimmten Systemen nicht laufen.

Um eine Kopie Ihrer Pascal-80-Diskette anzufertigen, geben Sie BACKUP ein, beantworten die auf dem Schirm erscheinenden Fragen und folgen den Anweisungen.

WICHTIGE HINWEISE:

1) Die Index-Feldvariablen in Records müssen globale Variablen sein, lokale Variablen sind nicht gestattet.
2) Die in einem Record definierten Variablennamen dürfen nicht mehrmals benutzt werden.
3) An bestimmten Stellen ist das Einsetzen von Leerzeichen (Spaces) nicht unbedenklich, im Gegensatz zum Standard-Pascal. Beispiel: Es darf innerhalb von zuweisenden Operatoren oder Keywords kein Leerzeichen stehen.

Speicherbelegung

START: 5200H, ENDE: AC00H, TRA: 7000H
Die Transferadresse TRA konfiguriert den Computer und springt zur normalen Startadresse 7000H. Der Programmtext beginnt bei AC00H, der Stack (Liegt im Speicher ganz oben) benötigt 256 bytes. Die Symboltabelle arbeitet unterhalb des Stack. Das Programm ist folgendermaßen organisiert:

5300H	PCODE-INTERPRETER
	EDITOR
	COMPILER
	PROGRAMMTEXT
	SYMBOLTABELLE
	STACK
(optional)	INITIALISIERUNGSROUTINEN

Wird das Programm mit der X-Option ausgeführt, beginnt der Pcode bei 8000H.

Cursor-Wahl

Pascal-80 ist mit einem blinkenden, unterstreichenden Cursor versehen. Um einen blinkenden Block zu erhalten, verändern Sie die Speicherposition A0D9 (File-Sektor 50, Bytes 19 und 1A) von 00 00 zu 18 18. Um einen nichtblinkenden Cursorblock zu erhalten, verändern Sie die Speicherposition A0C1 (File-Sektor 4E, Bytes 02 und 03) von (C3) D0 A0 zu (C3) 00 70.

Kleinschrift

Kleinschriftroutinen sind im Pascal-80 durch das DOSPLUS-System nicht implementiert. Haben Sie Pascal-80 jedoch unter einem Dos mit Kleinschrifttreiber laufen, werden kleine Buchstaben in Variablennamen und Keywords werden wie Großschrift behandelt. Pascal-80 unterscheidet nicht zwischen begin, BEGIN, Begin und bEgIN.

Programmieren in Pascal-80

Nachdem Sie eine Sicherheitskopie der Original-Diskette angefertigt haben und das Original am einem sicheren Ort verwahrt haben, können Sie Ihr erstes Programm schreiben. Booten Sie die Kopie der Pascal-80-Diskette. Nun sind Sie im EDITOR-Teil von Pascal-80 und auf dem Schirm erscheint folgendes Bild:

P A S C A L - 8 0

E - EDITOR
Q - QUIT (TO DOS)
K - KILL (CLEAR EDITOR)
C - COMPILE PROGRAM IN EDITOR
R - RUN PROGRAM IN EDITOR (COMPILE IF NECASSARY)
S - SAVE PROGRAM IN EDITOR
L - LOAD PROGRAM IN EDITOR
A - APPEND TEXT TO EDITOR
W - WRITE OBJECT CODE TO DISK (COMPILE IF NECASSARY)
X - EXECUTE PROGRAM FROM DISK

Drücken Sie <E>, um in den Texteditor zu gelangen. Jedes Pascal-80-Programm m u ß folgende Teile besitzen:

1) das Keyword PROGRAM
   (Gewöhnlich gefolgt von einem NAME)
2) ein ";" (Semikolon)
3) das Keyword BEGIN
4) gefolgt manchmal vom Keyword END
5) und einem "."

Geben Sie jetzt dieses Programm ein:

```
Program Print (OUTPUT);
Begin
    Writeln ('Hallo, IHR NAME')
End.
```

Schließen Sie die Eingabe jeder Programmzeile mit <NEW LINE> ab; dies teilt dem Computer das Ende der Zeile mit. Setzen Sie für "IHR NAME" Ihren eigenen Namen ein und vergessen Sie nach Eingabe der letzten Zeile die Eingabe von <NEW LINE> nicht, anderenfalls wird der Editor das END. - Statement ignorieren.

Haben Sie einen Eingabefehler gemacht, bewegen Sie den Cursor mit den Pfeiltasten an die Stelle, wo Sie der Fehler ist und geben Sie die Zeile ab dieser Stelle neu ein.

Jetzt können wir das Programm compilieren. Dazu drücken Sie zunächst die Taste <BREAK>. Es erscheint folgendes Menü:
<QUIT TOP NEXT PREVIOUS OPEN DELETE CANCEL LINE ERASE FORMAT>
Drücken Sie CONTROL Q, dies bewirkt, daß Sie zum Monitor-Menü zurückkehren:

P A S C A L - 8 0

E - EDITOR
Q - QUIT (TO DOS)
K - KILL (CLEAR EDITOR)
C - COMPILE PROGRAM IN EDITOR
R - RUN PROGRAM IN EDITQR (COMPILE IF NECASSARY)
S - SAVE PROGRAM IN EDITOR
L - LOAD PROGRAM IN EDITOR
A - APPEND TEXT TO EDITOR
W - WRITE OBJECT CODE TO DISC (COMPILE IF NECASSARY)
X - EXECUTE PROGRAM FROM DISK

Nun haben Sie drei unterschiedliche Möglichkeiten, Ihr Programm zu compilieren. Wenn Sie <C> drücken, wird das Programm lediglich compiliert, drücken Sie <R>, so wird das Programm compiliert und ausgeführt. Wenn Sie <W> drücken, wird das Programm compiliert und Sie erhalten die Meldung:
ENTER FILESPEC <BREAK> TO ABORT
Wenn Sie jetzt einen Filenamen eingeben, wird Ihr Programm auf Diskette abgespeichert.
Geben Sie <R> ein. Es wird eine Folge von Meldungen ausgedruckt, ähnlich wie:

```
00:31:32 COMPILATION STARTING
00:31:32 COMPILATION  COMPLETE
00:31:32 RUNNING
Hallo Carsten
00:31:32 EXECUTION COMPLETE
PRESS <NEW LINE> TO GO ON
```

Haben Sie in Ihrem Programm diverse Fehler, hält der Compiler an der betreffenden Stelle an und auf dem Schirm erhalten Sie eine Mitteilung, in welcher Zeile sich der Fehler befindet. Außerdem wird ein Pfeil (Oder Ä) und ein Errorcode an die mögliche Position des Fehlers gedruckt. Der Fehler kann zum Beispiel in der vorherigen Zeile oder in dem Kommando vor dem Pfeil stecken. Wenn Sie nun <NEW LINE> drücken, um fortzufahren, kommen Sie zurück zum Monitor-Editor. Wenn Sie <E> drücken, kehren Sie zurück zum Editor, mit der Zeile, in welcher der Fehler erschien. Um ein Programm zu korrigieren, brauchen Sie lediglich die Zeile mit dem Fehler, gefolgt von <NEW LINE> neu einzugeben. Nun versuchen Sie nochmals, das Programm zu compilieren. Das Handbuch beinhaltet Tabellen, wo Sie den ausgedruckten Errorcode nachschlagen und Ihr Programm entsprechend abändern können.

Files und WRITELN

(OUTPUT) ist eine Filedefinition. Diese weist den Compiler an, ein File für die Bildschirmausgabe zu reservieren.
Allerdings ist es in Pascal-80 nicht notwendig, OUTPUT als File zu definieren, weil Pascal-80 drei Typen von Files von Grund auf schon implementiert hat: INPUT, OUTPUT, und LP. INPUT definiert die Information, die von der Tastatur eingegeben wird; OUTPUT definiert Information, die auf den Bildschirm auszugeben ist und LP bedeutet Druckerausgabe. Aus diesen Gründen ist es nicht notwendig, diese Files in einem Pascal-80-Programm zu definieren, aber aus Gründen besserer Lesbarkeit und Übersicht gehört es einfach zum guten Programmierstil.

WRITELN, gesprochen "Write line" druckt einen Text auf dem Bildschirm aus. Der Text muß zwischen Klammern und dem Apostrophzeichen eingeschlossen sein, wie im Beispiel gezeigt. Die Funktion WRITE ist WRITELN ähnlich, der einzige Unterschied ist, daß bei WRITELN eine zusätzliche Leerzeile nach der Ausführung des Befehls ausgedruckt wird. WRITE gibt dem Programmierer die Möglichkeit, einen Text neben den nächsten auszudrucken, während WRITELN den nachfolgenden Text in der nächsten Zeile ausdruckt.

Einführung in die Editierung von Programmen

Wir wollen unser Beispielprogramm modifizieren. Kehren Sie zum Editor zurück. (Im Monitor-Menü <E> drücken.).
Um das Programm auf den Bildschirm zu holen, drücken Sie CONTROL T (<SHIFT> & <PFEIL ABWÄRTS>&<T>).
Dieser Befehl positioniert den Cursor an den Programmanfang im Text-Editor. Das Programm sieht momentan folgendermaßen aus:

```
Program Print (OUTPUT);
Begin
    Writeln ('Hallo,NAME')
End.
```

Nun wollen wir das Programm ändern:

```
Program Print (OUTPUT);
Var count:integer;
Begin
    For count := to 50 do
    writeln ('Hallo,NAME')
end.
```

Einfügen

Zuerst bewegen Sie den Cursor mit den Pfeiltasten auf das B in BEGIN. Jetzt drücken Sie CONTROL L. Diese Aktion erzeugt eine Leerzeile im Programmtext. Drücken Sie CONTROL L nochmals, um eine zweite Leerzeile zu erzeugen.
Nun geben Sie ein:
Var count:integer
und drücken Sie <NEW LINE>; so wird die neue Programmzeile in den Programmtext übernommen.
Jetzt bewegen Sie den Cursor auf das W in WRITELN. Drücken Sie wieder CONTROL L, um eine Zeile einzufügen.
Geben Sie nun ein:
For count:=1 to 50 do
und drücken Sie <NEW LINE>. Drücken Sie CONTROL Q, dann <R> zum compilieren. Wenn Sie alles richtig gemacht haben, druckt der Rechner Ihren Text 50 mal aus.
Wenn Sie eine Errormeldung erhalten, versuchen Sie herauszufinden, in welcher Weise sich Ihr eingegebenes Programm von dem Beispielprogramm, wie es im Manual abgedruckt ist, unterscheidet; speziell in der Zeile, wo der Pfeil steht oder der vorherigen Zeile. Nun drücken Sie <NEW LINE>, darauf <E>,um zum Editor zurückzukehren. Jetzt drücken Sie CONTROL Q und <R>, um das Programm nochmals zu compilieren.

Die Löschfunktion

Wir wollen unser Programm ändern von

```
Writeln ('Hallo,NAME')
```

in

```
Write (count,'Hallo NAME')
```

Positionieren Sie den Cursor auf das W in WRITELN und drücken Sie CONTROL E. Dies löscht unsere alte Zeile. Nun drücken Sie CONTROL L, um eine neue Zeile einzufügen. Geben Sie ein:

```
    Write (count,'Hallo NAME')
```

Sicherlich wäre es einfacher gewesen, den Cursor auf das L in WRITELN zu positionieren, die Zeile neu einzugeben und mit <NEW LINE> die neue Zeile zu Übernehmen; wir wollten jedoch die Löschfunktion demonstrieren; daher löschten wir die alte Zeile und fügten eine neue ein.
Drücken Sie nun CONTROL Q, dann <R> zum compilieren und Starten. Was hat die Änderung bewirkt?

Neue Elemente sind VARiablen, der Zuweisungs-Operator und DO LOOPs. Wir haben unserem Programm zwei neue Elemente beigefügt. Das erste ist die Variablenzuweisung:

```
Var count : integer
```

VAR ist ein Pascal-Keyword, welches den nachfolgenden Ausdruck als Variable für das Programm definiert. In unserem Beispielprogramm nannten wir diese Variable COUNT und definierten sie als INTEGER-Variable. In Pascal-80 umfaßt INTEGER einen Zahlenbereich ganzer Zahlen von -32768 und 32768. Beachten Sie bitte den Doppelpunkt (:) zwischen COUNT und INTEGER. Er ist dort absolout notwendig.
Das andere neue Element ist die DO LOOP:

```
For count:=1 to 50 do
```

Die DO-LOOP bewirkt, daß während des Programmlaufes der Variablen COUNT Werte von 1 bis 50 zugeordnet werden; immer in Einerschritten inkrementierend. Das := in der do-Loop ist der zuweisende Operator. Er teilt dem Rechner mit, daß er einer Variablen Werte zuordnen soll. In diesem Fall werden der Variablen COUNT die Werte 1, 2, 3, ... 49, 50 zugeordnet.
Der Zuweisungsoperator kann auch in der Form

```
COUNT :=995
```

benutzt werden, um der Variablen COUNT den Wert 995 zuzuweisen.
Wenn Sie noch nie einen Error erhalten haben, gehen Sie doch mal in den Editor und verstümmeln oder verändern Sie eines der Keywords wie z.B. PROGRAM, BEGIN, WRITELN oder END).
Beispiel: positionieren Sie den Cursor auf das D in END und geben ein Leerzeichen ein. Somit steht in dieser Zeile ein EN .
Nun versuchen Sie das Programm zu compilieren, so daß Sie die Fehlermeldung sehen können. Vergessen Sie nicht, später in den Editor zurückzugehen und den Fehler wieder zu korrigieren, da wir unser Programm noch weiter benutzten.

Abspeichern und Laden von Programmen

Wir wollen unser Beispielprogramm auf Diskette abspeichern, um die Disk-Funktionen des Monitors zu demonstrieren. Es bestehen grundsätzlich zwei Möglichkeiten, Programme abzuspeichern,

a) als Sourcefiles (Text) und
b) als Objectfiles (compiliert)

Kehren Sie zum Monitor-Menü zurück. Haben Sie Ihr Programm gerade laufen lassen, können Sie mit <NEW LINE> zum Monitor zurückkehren. Befinden Sie sich im Editor, müssen Sie CONTROL Q drücken, um in den Monitor zurückzugelangen.

Abspeichern eines Textfiles

Zuerst werden wir den Quellcode (Sourcecode) unseres Programmes abspeichern, dessen Text sich im Editor befindet. Ausgehend vom Monitor-Menü, drücken Sie <S>. Folgende Meldung erscheint auf dem Schirm:

```
ENTER FILESPEC <BREAK> TO ABORT
?
```

Geben Sie nun einen Filenamen ein. Ein Filename aus höchstens acht Ziffern oder Zeichen, beginnend mit einem Zeichen. An den Filenamen kann der Schrägstrich (/), gefolgt von ein bis drei Zeichen angehängt werden. Sinnvoll ist es, die Kürzel /SRC oder /SOU für abzuspeichernden Sourcecode und /PAS odr /OBJ für compiliertes Pascal zu verwenden. Es bleibt Ihnen jedoch freigestellt, irgendein - oder auch gar kein Kürzel anzuhängen. Wir geben ein:

```
HALLO/SRC
```

und <NEW LINE>. Jetzt wird der Quellcode unseres Pascal-Programmes vom Monitor auf Diskette abgespeichert.

Das Abspeichern von Source-Files

Jetzt wollen wir unser Sourcefile zurückladen. Gehen Sie in den Editor (<E> im Monitor), um sicherzugehen, daß kein Programm darin steht und tippen Sie CONTROL Q, um zum Monitor zurückzukehren. Geben Sie nun <L> ein, und es erscheint:

```
ENTER FILESPEC <BREAK> TO ABORT
?
```

Geben Sie ein:

```
HELLO/SRC
```

; oder wie Sie Ihr Programm genannt haben und drücken Sie <NEW LINE>. Nun lädt das Programm, testen Sie dies, indem Sie nach Beendigung des Ladevorganges in den Editor gehen. Sie sehen nun Ihr Programm, so wie Sie es abgespeichert haben. Auf diese Weise sollte es Ihnen jetzt möglich sein, auch eigene Programme abzuspeichern.

Das Abspeichern von Objectfiles

Das Abspeichern eines compilierten Pascal-80 Programmes erfolgt ähnlich wie das Abspeichern eines Sourceprogrammes. Ausgehend vom Monitor-Menü wählen Sie die Option <W>. Wiederum werden Sie gefragt:

ENTER FILESPEC <BREAK> TO ABORT
?

Vielleicht möchten Sie den selben Filenamen verwenden, nur mit einem anderen Anhängsel. Der Computer bewertet die Files als völlig unterschiedlich.
Geben Sie ein :

HALLO/PAS

oder Ihren eigenen Filenamen und drücken Sie <NEW LINE>.
Jetzt wird der Object-Code Ihres Programmes auf Diskette abgespeichert.

Das Einlesen von Obectfiles

Um ein compiliertes Programm von Diskette zu laden und auszuführen, wählen Sie im Monitor die Option <X>. Wieder fragt Sie das System:

ENTER FILESPEC <BREAK> TO ABORT
?

Geben Sie ein:
HALLO/PAS

oder den Filenamen, den Sie gewählt haben und drücken Sie <NEW LINE>. Nun lädt das Programm und wird ausgeführt.
Obwohl das Laden eines compilierten Programms praktisch genauso wie das Laden eines Sourcefiles funktioniert, gibt es doch zwei grundlegende Unterschiede:
Das Programm wird nach dem Laden automatisch gestartet und nach Beendigung der Ausführung wird zum DOS zurückgekehrt. Außerdem überschreibt das Programm den Compilerteil von Pascal-80, so daß ca. 9K mehr PLatz für Variablen zur Verfügung stehen.

Eigene Ein-/Ausgabetreiber

Pascal-80 beachtet die HIMEM-Eingabe in der Speicherzelle 4049H, so daß Sie Ihre eigenen Ein-/Ausgabetreiber benutzen können.

Einschränkungen von Pascal-80

Folgende Funktionen von Standard-Pascal sind in Pascal-80 nicht implementiert :

"Variant Records"

Das WITH- Statement

Die Anweisungen NEW und DISPOSE Filebuffer; die Prozeduren GET und PUT

Die Prozeduren PACK und UNPACK (Alle Datenstrukturen sind gepackt)

Die Prozedur PAGE (Ersetzbar durch WRITE(LP,CHR(12)))

Filestrukturen (Z.B. ARRAY OF FILE , etc.)

Andere Einschränkungen:

Felder sind auf 256 Einträge beschränkt; sind sie numerisch, mussen sie sich im Zahlenbereich von 0 bis 255 befinden.

Der Identifier einer Prozedur oder Funktion ist nicht erlaubt als Parameter einer anderen Prozedur oder Funktion.

Kein Ausdruck, der als Werteparameter zulässig ist, darf 510 Bytes übersteigen, sofern es sich nicht um einen VAR-Parameter handelt.

Integervariablen, die Feldgrenzen in Records beschreiben, müssen globale Variablen sein.

Eckige Klammern werden auf dem Genie 1 folgendermaßen definiert: Klammer auf: (.
Klammer zu : .)

ODER: CONTROL S und CONTROL K produzieren die Zeichen Ä und Ü, die vom Compiler als eckige Klammern interpretiert werden.

Pascal-80 benutzt (* und *) für Kommentare.

Leerzeichen sind an gewissen Stellen delikat, wo sie im Standard-Pascal unwichtig sind; so darf der Zuweisungsoperator := nicht durch ein Leerzeichen getrennt werden (: =).

Integervariablen sind beschränkt auf den Bereich Ganzer Zahlen von -32768 bis 32767.

Erweiterungen des Standard-Pascal:

Felder von Zeichen können mit Pascal-80 mit einem einzigen Ausdruck gedruckt werden. Beispiel:

Wenn STRING definiert ist als ARRAY(.1..10) OF CHAR
ist der Ausdruck WRITE(STRING) gleichbedeutend mit
FOR I :=1 TO 10 DO WRITE(STRING(.I.))

READ UND WRITE können in Files, die keine Textfiles sind, anstelle von GET und PUT verwendet werden.

Bei der Verwendung von Stringkonstanten in Zuweisungen und bei Vergleichen von Zeichenfeldern kann die rechte Konstante kürzer als die Liste auf der linken Seite sein und wird, wenn nötig, automatisch mit Leerzeichen aufgefüllt.

Ist:
VAR STRING := ARRAY(.1..10)OF CHAR
dann ist STRING :='NAME' gültig und STRING >'NA' ergibt logisch wahr (TRUE)

(Das Argument auf der rechten Seite muß wenigstens zwei Zeichen besitzen.)

PROC und FUNC sind gültige Abkürzungen für PROCEDURE und FUNCTION. Pascal-80 besitzt einige fest vorgewählte Konstanten: MIMINT (-32768), MAXINT (32767), PI, FALSE, und TRUE. REAL-Variablen haben 14-Stellige Genauigkeit; REAL6-Variablen bieten optional dazu die Möglichkeit an, bei Verwendung umfangreicher Variablenfelder Speicherplatz zu sparen (4 Bytes statt 8). REAL6-Variablen, die nicht zu einem Array oder einem Record gehören, dürfen in einer Funktion oder Prozedur nicht als Werteparameter benutzt werden.
Die Standard-Files INPUT und OUTPUT brauchen in einem PROGRAM-Statement nicht extra erklärt zu werden; auch der Name eines Programmes ist optional. Das vordefinierte File LP dient zur Druckerausgabe.

Zusätzliche Prozeduren:

Folgende Prozeduren sind Erweiterungen des Standard-Pascal. Sie werden später in den betreffenden Abschnitten des Handbuches genauer erläutert.
CLS löscht den Bildschirm, CLOSE schließt Files. SEEK(Ausdruck, Filename) positioniert ein benanntes File zu einem Record innerhalb eines Files. INKEY ermittelt den Wert einer gedrückten Taste. CALL(Adresse, Wert) lädt den Akku mit einem Wert, springt zur angegebenen Adresse und kehrt mit einem Wert im Akku zurück. MEM ermittelt den noch vorhandenen freien Speicherplatz. PEEK(Adresse) ermittelt den Wert einer Speicherzelle und POKE(Addresse, Wert) transferiert einen Wert in eine Speicherzelle. FP(Ausdruck) ermittelt eine Zahl als Bruch des Exponenten dieser Zahl und EX(Ausdruck) übergibt den Exponenten einer REAL-Zahl an eine Variable. Mit der INCLUDE-Funktion kann man eine Source-Prozedur von Diskette in ein Programm im Editor hineincompilieren. Zusätzlich zu diesen Prozeduren besitzt Pascal-80 Routinen für Cursorpositionierung GOTOXY(Horizontalpos., Vertikalpos.) und für Graphik: PSET(Horizontalpos., Vertikalpos.) ; PRESET(Horizontalpos., Vertikalpos.) und POINT(Horizontalpos., Vertikalpos.), sowie Zufallszahlengenerierung. Die CASE-Anweisung ist zweifach erweitert: Eine ELSE-Bedingung darf benutzt werden und wird ausgeführt, wenn kein anderes CASE erfüllt wird. Ist kein CASE erfüllt und ein ELSE nicht vorhanden, wird ohne eine Fehlermeldung zur nächsten Operation weitergesprungen. REAL- und INTEGER-Ausdrücke werden in folgendem Format ausgegeben: WRITE(Ausdruck, Länge, Stellen). Einen Länge von -1 bedeutet exopnentielle Darstellung und läßt die Stellenanweisung (Falls vorhanden) unberücksichtigt. Die Länge 0 ergibt ein Leerformat, genauso wenn das Format nicht definiert wurde. Das Leerformat druckt die Zahl mit einem Leerzeichen vor der Zahl und sovielen Stellen nach dem Komma, wie nötig.

Die EDITOR- Funktionen von Pascal-80

Das Pascal-80- System verfügt über einen sehr komfortablen Editor. Der eingegebene Text wird in einem Textspeicher innerhalb des Computerspeichers abgespeichert und kann vom Monitor compiliert und/oder auf Diskette abgespeichert werden. Da der Bildschirm nicht den gesamten Textspeicher auf einmal darzustellen vermag, zeigt der Editor nur 15 Zeilen Programmtext auf einmal. Mit den Pfeiltasten wird der Cursor bewegt. Ist die obere Position des Bildschirms erreicht, wird der Bildschimausschnitt bis an den Textanfang gescrollt. Das gleiche gilt für die Cursorbewegung nach Unten.
Während des Programmiervorgangs künnen Zeichen mit <SHIFT> <LINKSPFEIL> gelöscht werden. Genauso können Fehler einfach überschrieben werden. Hierbei haben die Pfeiltasten, wie die anderen Tasten auch, Autorepeat. Haben Sie alle Fehler korrigiert, wird mit <NEW LINE> die Programmzeile, wo der Cursor steht, in den Textspeicher übernommen. Hierbei ist es egal, ob der Cursor am Ende der Zeile steht oder nicht.
Drücken Sie in einer geänderten Programmzeile, wo noch kein <NEW LINE> gedrückt wurde CONTROL C oder bewegen Sie den Cursor mit dem Hoch- oder Abwärtspfeil, so erscheint die Zeile so, wie sie im Textspeicher steht.
(Anmerkung: bei der neuen Pascal-80-Version ist es nicht mehr erforderlich, zur Übernahme eines Programmtextes <NEW LINE> zu drücken; auch wenn Sie die aktuelle Zeile mit <HOCHPFEIL> oder <PFEIL ABWÄRTS> bzw. den Subkommandos NEXT, QUIT, PREVIOUS oder TOP verlassen haben, wird der Text übernommen.)

Die EDITOR- Kommandos

Im Editor-Mode verfügen Sie über 12 nützliche Kommandos. Während des Arbeitens können Sie sich die Editorbefehle am unteren Bildschirmrand auflisten lassen. Drücken Sie einfach <BREAK>. Um dieses "Menü" zu löschen, drücken Sie <CLEAR>.
Nach <BREAK> erscheint:
QUIT TOP NEXT PREV OPEN DEL CANC LINE ERA FORM BLOCK WRITE
Diese Kommandos werden aufgerufen, indem man den ersten Buchstaben des Befehlswortes drückt, während die Tasten <SHIFT> und <PFEIL ABWÄRTS> gedrückt gehalten werden.
Bei der Benutzung der Editierkommandos ist es gleichgültig, ob sich das Menü auf dem Bildschirm befindet oder nicht.

CONTROL T positioniert den Cursor an den Anfang des Arbeitsspeichers.

CONTROL N gibt die nächsten 15 Zeilen des Programmtextes aus.

CONTROL P gibt die vorangehenden 15 Zeilen des Textes aus.

CONTROL Q kehrt zum Monitor zurück.

CONTROL L setzt eine Leerzeile ein, wo der Cursor sitzt.

CONTROL E löscht die Zeile, wo der Cursor sich befindet.

CONTROL F aktiviert die Tabulatorfunktion, welche Ihnen hilft, Ihr Pascal-Programm zu strukturieren. Der Editor speichert die aktuelle Cursorposition ab und positioniert den Cursor bei Druck auf <NEW LINE> unter die abgespeicherte Stelle. Dies gestattet, Programmzeilen der gleichen Verschachtelungsebene einzugeben, ohne jedoch den Tabulator ständig neu setzen zu müssen.

CONTROL C macht Korrekturen rückgängig, sofern noch kein <NEW LINE> gedrückt wurde oder die Zeile noch nicht mit den Pfeiltasten etc. verlassen wurde.

CONTROL O ermöglicht, an der aktuellen Position des Cursors Text einzufügen.

CONTROL D löscht ein Zeichen an der aktuellen Cursorposition.

CONTROL W gibt den Programmtext, beginnend bei der Zeile, wo der Cursor steht auf den Drucker aus. Findet der Editor beim Drucken das Markierungszeichen "§", so wird der Druckvorgang abgebrochen, anderenfalls wird der gesamte Programmtext bis zum Ende ausgedruckt.

CONTROL B ist ein Transferbefehl, der in Verbindung mit dem APPEND- Kommando des Monitors benutzt werden kann. APPEND hängt ein File an das Ende des aktuellen Files an. CONTOL B transferiert den gesamten Text von der Blockmarkierung bis ans Ende des Files zur angegebenen Position. Dazu positionieren Sie die Markierung, das "§"-Zeichen an den Anfang der Zeile, bei der der Transfer beginnen soll. Dann bewegen Sie den Cursor zu einer Leerzeile, wo der neue Text eingesetzt werden soll und drücken Sie CONTROL B. Der Textblock wird transferiert und gleichzeitig wird das Markierungszeichen gelöscht.
Um kleine Stücke des Programmtextes zu transferieren müssen Sie dieses Kommando zweimal durchführen.

CONTROL S und CONTROL K erzeugen die Zeichen Ä und Ü, die vom Compiler als eckige Klammern interpretiert werden.

CONTROL U erzeugt das Unterstreichungszeichen (Darf in Variablennamen benutzt werden.)

Der Pascal-80 MONITOR

Das erste, was Sie nach dem Starten von Pascal-80 zu sehen bekommen, ist das Menü des Monitors:

P A S C A L - 8 0

E - EDITOR
Q - QUIT (TO DOS)
K - KILL (CLEAR EDITOR)
C - COMPILE PROGRAM IN EDITOR
R - RUN PROGRAM IN EDITOR
S - SAVE PROGRAM IN EDITOR
L - LOAD PROGRAM IN EDITOR
A - APPEND TEXT TO EDITOR
W - WRITE OBJECT CODE TO DISK
X - EXECUTE PROGRAM FROM DISK

Das Menü des Monitors erlaubt eine einfache Handhabung der einzelnen Operationen.
Das Kommando <K> fragt, ob Sie ein im Editor stehendes Programm löschen wollen:
ERASE TEXT (Y/N)
Um den Programmtext zu löschen, geben Sie <Y> ein.
Wenn Sie die Kommandos <S>, <L>, <A>, <W> oder <X> eingeben, erscheint auf dem Schirm folgende Frage:
ENTER FILESPEC <BREAK> TO ABORT
?
Geben Sie nun den Namen des zu bearbeitenden Files an.
Die Eingabe hat folgendes Format:
FILENAMEN/anh.CODEWORT:nr
FILENAME ist ein Name von Einem bis Acht Buchstaben und Zahlen der es Ihnen ermöglicht, Ihr File zu kennzeichnen und identifizieren. Das erste Zeichen des Filenamen muß ein Buchstabe sein, die restlichen Zeichen können Ziffern oder Buchstaben sein. Innerhalb des Filenamen dürfen keine Leerzeichen oder Symbole wie z.B. Fragezeichen stehen.
/anh ist ein Anhängsel, welches Sie zusätzlich einsetzen können, um Ihre Programme zu kennzeichnen. Es besteht aus einem Schrägstrich (/) , gefolgt von ein bis drei Buchstaben oder Zahlen. Das erste Zeichen nach dem Schrägstrich muß ein Buchstabe sein. Das Anhängsel ist ein optionaler Zusatz und kann auch weggelassen werden.
mit einem Codewort gesicherten Programm erhalten. Wie bei Filenamen muß das erste Zeichen des CODEWORTS ein Buchstabe sein, der Rest kann aus Buchstaben oder Ziffern bestehen. Listet man so gescützte Files mit DIR, so werden sie durch den Filenamen und ein "P" hinter dem Namen angezeigt. Auch CODEWORT ist ein optionaler Zusatz.
nr gibt an, welches Laufwerk angesprochen weden soll. :nr ist entweder Null oder Eins.

Monitorfunktionen

Mit <E> kommen Sie in den Editiermodus. Falls der Editor Text enthält, erscheint dieser auf dem Bildschirm. Falls während des Compiliervorgangs ein Fehler im Programm festgestellt wird, stoppt der Rechner. Wenn Sie nun in den Editor zurückgehen, befindet sich der Cursor in der Zeile, wo der Fehler auftrat. In manchen Fällen, so z.B. wenn in der vorangehenden Zeile ein Semikolon fehlte, befindet sich der Fehler in der vorangehenden Zeile.
Mit <Q> gelangen Sie zum DOS zurück. Dies gestattet Ihnen, normale DOS-Kommandos auszuführen wie z.B DIR (Listet die auf der Diskette befindlichen Files auf) oder FREE (Ermittelt den noch zur Verfügung stehenden freien Diskettenspeicherplatz) oder PRINT (File auf Drucker ausgeben).
Mit <Q> können Sie ins DOS gehen, ohne daß Ihr Programm zerstört wird. Wollen Sie wieder in das Pascal-System zurück, geben Sie PASCAL ein. das Programm wird geladen und der Speicher ist gelöscht. Wollen Sie jedoch, daß Ihr Programm im Editor stehen bleibt, geben Sie ein: PASCAL und halten <NEW LINE> gedrückt, bis das Programm lädt. Mit <E> können Sie nun Ihr unzerstörtes Programm redigieren.
<K> löscht den Inhalt des Editors. Um unbeabsichtigtes Löschen zu verhindern, fragt der Rechner nach Eingabe von <K>:
ERASE TEXT? (Y/N)
Drücken Sie jetzt <Y), wird das Programm im Editor gelöscht. Ein Druck auf eine andere Taste bewirkt die Rückkehr zum Monitor, ohne daß das Programm gelöscht wird.
Mit <R> führen Sie ein im Editor stehendes Source-Programm aus. Wurde das Programm noch nicht compiliert oder sind Sie nach dem letzten Start des Programms zum Editor zurückgekehrt, wird das Programm automatisch compiliert, bevor es ausgeführt wird.
Mit <S> wird ein Programm, das im Editor steht auf Diskette abgespeichert. Sie werden gefragt, ob Sie an den Filenamen ein Anhängsel wie z.B /SRC oder /SOU anhängen wollen. (Dies ist manchmal recht nützlich, damit Sie später Sourcefiles von anderen Filetypen unterscheiden können.)
Wollen Sie das File doch nicht abspeichern, können Sie mit <BREAK> anstelle der Eingabe eines Filenamen zum Hauptmenü zurückkehren.
Mit <L> wird ein Sourcefile von Diskette in den Editor geladen. Sie werden vom System nach dem Namen des zu ladenden Files gefragt. Wissen Sie den Filenamen nicht mehr, können Sie mit <Q> ins DOS zurückkehren und mit DIR die Files auslisten lassen. Wenn Sie jetzt PASCAL eingeben und <NEW LINE> drücken, um zum Monitor zurückzukehren können Sie Ihr Programm wie beschrieben laden. Wollen Sie kein File einlesen, können Sie mit <BREAK> zum Monitor zurückkehren.

Mit <A> können Sie ein File von Diskette laden, ohne daß Ihr Programm im Editor zerstört wird. Das neue File wird einfach an das File im Editor angehängt. Sie müssen beachten, daß der Compiler die Übersetzung als abgeschlossen betrachtet, wenn er zum ersten END- Statement im Programm kommt. Deshalb ist es wichtig, daß Sie Ihr Programm nach der Ausführung der Option <A> auf eventuell überflüssige ENDs durchsehen.
Nach Eingabe von <A> fragt der Rechner, wie das anzuhängende File heißt. Wenn Sie statt eines Filenamen <BREAK> eingeben, kehren Sie zum Monitor zurück. Versuchen Sie ein File zu laden, welches kein Pascal-Sourcefile ist, erhalten Sie die Fehlermeldung BAD FORMAT.
Wollen Sie ein File von Diskette in Ihr aktuelles Programm einfügen, können Sie dieses mit der INCLUDE- Funktion oder dem Transferbefehl <B> des Editors erreichen.
Mit <W> können Sie compilierte Objectcode-Files auf Diskette abspeichern. Wurde Ihr aktuelles Programm im Editor noch nicht compiliert, wird es compiliert und Sie werden nach einem Filenamen gefragt. Mit <BREAK> können Sie zum Monitor zurückkehren. Um das Programm zu spezifizieren, können Sie ein Anhängsel, wie z.B. /OBJ oder /PAS benutzen.
Das Ausführen eines compilerten Programmes gibt Ihnen die Möglichkeit, 9K zusätzlichen Speicherplatz, der sonst von Editor und Compiler benötigt würde, für Variablen und Speicherplatz Ihres Programmes zu verwenden. Außerdem benötigt ein compiliertes Programm auf Diskette weniger Speicherplatz als sein Sourcecode und Sie sparen die Zeit, die Sie sonst benötigen würden, um ein Sourceprogramm compileren zu lassen.
Mit <X> können Sie ein compiliertes Pascal-Programm von Diskette ausführen lassen. Sie werden nach einem Filenamen gefragt, können aber mit <BREAK> zum Monitormenü zurückkehren. Das geladene Programm überschreibt, je nach dem, wie lang es ist, den Editor und Compiler von Pascal-80. Dies spart Speicherplatz, erfordert aber, daß Sie nach der Ausführung Ihres Programms PASCAL neu starten. Daher kehrt das ausgeführte Programm nicht zum Monitor, sondern zum DOS zurück.
Wenn Sie versuchen, ein File zu laden, welchs kein Pascal-Objectfile ist, erhalten Sie die Fehlermeldung BAD FORMAT.

Compileroptionen

Es gibt sechs nützliche Compilerbefehle, die es gestatten, Compilerausgaben auf den Drucker zu geben, Compilerausgaben zu unterdrücken (Außer Fehlermeldungen) , den freien Speicherplatz für Symboltabellen oder den Stack auszugeben, jedes Byte compilierten Objectcodes auf Drucker auszugeben oder alle Variablen vor der Übersetzumg auf Null zu setzen und/oder alle Diskettenoperationen zu verifizieren.
Die Funktionen heißen HARDCOPY, NOLIST, MEMORY, CODE, ZERO und VERIFY. Sie können durch ihre Anfangsbuchstaben abgekürzt werden, da der Compiler nur den ersten Buchstaben überprüft. Die Befehle oder Abkürzungen der Befehle müssen in Großschrift erfolgen. Um die Compileroptionen anzuwenden, setzen Sie die von Ihnen ausgewählten Compilerbefehle ganz an den Anfang Ihres Pascal-Programms. Der Compiler bewertet alles, was vor dem Keyword PROGRAM steht als Anweisung für den Compiler.
Die Instruktionen durch alle gültigen Begrenzungszeichen, wie z.B. Leerzeichen oder "/" getrennt werden. Somit ist die Abkürzung H/C/Z identisch mit
HARDCOPY CODE ZERO.
Die ersten vier Optionen betreffen Druckerausgaben:
HARDCOPY oder H gibt alle Ausgaben auf Drucker aus.
NOLIST oder N unterdrückt Compilerlistings, außer den Fehlermeldungen des Compilers.
MEMORY oder M fügt zu jeder vom Compiler ausgegebenen Zeile zwei Hexadezimalzahlen hinzu, welche den Speicherbedarf des Compilerstacks und der Symboltabelle anzeigen:
0000 00F4 5C4B PROGRAM; BEGIN END.
Die erste Zahl teilt Ihnen mit, wieviele Bytes vor dieser Zeile schon compiliert worden sind. Die zweite Zahl gibt an, wieviel Bytes Speicherplatz für den Compilerstack zur Verfügung stehen und die dritte Zahl beschreibt den für die Symboltabelle zur Verfügung stehenden Speicherplatz.
CODE oder C weist den Compiler an, jedes Byte compilierten Codes als Hexadezimacodes im Compilerlisting auszudrucken. Dieses Listing ist jedoch nicht vollständig: In bestimmten Fällen, z.B. bei Vorwärtssprüngen, erzeugt der Compiler Platzhalter und setzt die richtigen Werte später ein. Der P-Code ist selbstrelozierend und vom Compiler erzeugte relative Sprungadressen werden vor der Ausführung jedes Blocks durch absolute RAMadressen ersetzt.
Die zwei übrigen Compileroptionen betreffen die Programmausführung:
VERIFY oder V verifiziert alle Diskettenschreiboperationen. Pascal-80 setzt sich über alle Verifizierungsinstuktionen. welche unter dem DOS VERIFY gegeben werden, hinweg.

ZERO oder Z bewirkt, daß alle Variablen in einem compilierten Programm auf Null gesetzt werden, bevor das Programm ausgeführt wird. Benutzen Sie diese Option nicht oder weisen Sie den Variablen in Ihrem Programm keinen Anfangswert zu, erhalten die Variablen irgendwelche Werte, die noch vom letzten Programm im Speicher stehen. Dies kann zu falschen Ergebnissen führen.
Das Pascalwort FORWARD ist ein Compileroption. FORWARD wird benutzt, um eine Funktion oder Prozedur nach Ihrem ersten Erscheinen zu deklarieren, ohne daß ein Error ausgegeben wird. Vor und hinter FORWARD muß ein Semikolon stehen.
Beispiele:

```
PROCEDURE IRGENDWAS;FORWARD;
PROCEDURE KALENDER( MONAT, DATUM , JAHR: real); FORWARD
FUNCTION GERADE (var NUMBER: integer):boolean; FORWARD
```

INCLUDE (*$ FILENAMEN*)
Die Include-Funktion gestattet entweder ein ganzes Programm oder einzelne Prozeduren in ein Programm im Editor hineinzucompilieren. Beispielsweise wenn Sie ein Programm geschrieben haben, das so lang ist, daß nicht genug Platz zum compilieren ist, können Sie dieses Programm als Source-Code von Diskette in Ihr Programm compilieren.
Wenn Ihr Programm beispielsweise PROGRAMM/SRC heißt, löschen Sie zunächst mit dem Monitorkommando <K> den Text im Editor und geben Sie folgendes in den Editor ein:

```
(*$ PROGRAMM/SRC*)
```

Dies ist das Einzige, was im Editor stehen muß; natürlich können Sie noch Compileroptionen ergänzen. Wenn Sie nun in den Monitor gehen und die Optionen <C>, <W>, oder <R> ausführen, wird das Programm von Diskette anstatt aus dem Programmspeicher compiliert.
Wollen Sie dagegen Prozeduren in Ihr Programm einbinden, speichern Sie zunächst die Prozedur als Sourcefile ab. Das Demoprogramm COINTOSS/SRC gibt Ihnen Beispiele an.
Wollen Sie den Sourcecode in Ihr Programm einbinden, anstatt die Include-Funktion zu benutzen wenn Sie ein Object-Code-File anlegen, benutzen Sie die <A>-Funktion des Monitors, um die gewünschte Prozedur an das Ende Ihres Programms anzuhängen. Dann gehen Sie in den Editor und platzieren die Prozedur mit dem Transferbefehl des Editors an die gewünschte Position.

Konstante in Pascal-80

Pascal-80 besitzt 5 eingebaute Konstanten: True, False, MinInt (kleinste erlaubte Integerzahl), MaxInt (größte erlaubte Integerzahl) und Pi. Außerdem können Sie Ihre eigenen Konstanten definieren. Das folgende Beispielprogramm und der Ausdruck des Probelaufs zeigt die eingebauten Konstanten von Pascal-80 und wie man eigene Konstanten definiert.

```
PROGRAM Constants (Output);
Const Two = 2.0;
Var Diameter,Circumference : real;
Begin
Writeln(Two);
Writeln(True);
Writeln(False);
Writeln(MinInt);
Writeln(MaxInt);
Writeln(Pi);
Diameter :=Two;
Circumference :=Pi * Diameter;
Writeln( Circumference);
End.
```

Probelauf:

```
2
TRUE
FALSE
-32768
32767
3.1415296535898
6.2831853071796
```

Variablentypen

Pascal-80 läßt sechs verschiedene Variablentypen zu: Boolean, Integer, Char, Real, Real6 und Text. Außerden können Sie mit dem Type-Statement eigene Variablentypen definieren. Globale Variablen müssen am Programmanfang deklariert werden und können im ganzen Programm benutzt werden. Lokale Variablen können in einer Prozedur oder Funktion deklariert werden und arbeiten nur in dieser. Trotzdem können Sie dieselben Variablennamen innerhalb verschiedener Prozeduren ohne Probleme benutzen. Variablennamen können beliebige Längen besitzen und alle Buchstaben sind signifikant, so werden die Variablen VariableNummerFünfzehn und VariableNummerSechzehn als unterschiedliche Variablen erkannt und behandelt.
Boolean-Variablen sind entweder True (wahr) oder False (falsch)
Integer-Variablen müssen ganze Zahlen zwischen -32768 (MinInt) und 32767 (MaxInt) sein.
Char- oder Charactervariablen sind Zeichen, Zahlen, Symbole, Leerzeichen usw. Auch mit der CHR- Funktion können Zeichen erzeugt werden.
Real-Variablen besitzen 14-Stellige Genauigkeit und alle Berechnungen, auch logarithmische, trigonometrische und arithmetische werden mit 14-Stelliger Genauigkeit vorgenommen.
Real6-Variablen können z.B. in großen Variablenfeldern eingesetzt werden, um Speicherplatz zu sparen. (vier Bytes Speicherbedarf statt acht bei Real-Variablen)
Die Verwendung von Real6-Variablen bringt jedoch keine schnellere Programmausführung mit sich.
Wichtiger Hinweis: Real6-Variablen, die zu einem Array (Variablenfeld) oder einem Record gehören, dürfen in einer Funktion oder Prozedur nicht als Werteparameter verwendet werden.
Text-Variablen sind gepackte Files, die aus einzelnen Zeichen bestehen.
Beispiele für Variablendeklarationen:

```
program Variables (Input, Output);

      Const Length = 16;
      Type Number = Array (. 0..Length .) of integer;
      SetC = Set of Char;

  Var N:Number;
      C:Char;
 I.J.K :Integer;
      B:Boolean;
      S:SetC;
      R:Real;
     R6:Real6;
   Flag:Boolean;
      A:Array (. 0..3 .) of Number;
      T:Array (. 1..1000 .) of char; ...
```

Funktionen und Operatoren

Es mag leicht sein, Funktionen und Prozeduren zu verwechseln, speziell die, welche in der Sprache implementiert sind. Während beides wichtige Unterprogramme sind, ist der wichtige Unterschied zu einer Funktion der, daß eine Funktion einen bestimmten Wert ermittelt. Pascal besitzt sowohl arithmetische und logische Operatoren, inklusive trigonometrische und logarithmische Funktionen, wie auch Konvertierungs, Filehandhabungs und Ordnungsfunktionen etc. Ein Operator ist einer Funktion ähnlich; auch er ermittelt einen einzelnen Wert, aber er ist gewöhnlich binär und benötigt einen Operanden auf jeder Seite, wie bei 2 + 2 .

Arithmetische Funktionen

Arithmetische Operatoren (+ - / * DIV MOD)

Die arthmetischen Operatoren beinhalten + für Addition, - für Subtraktion, / für reelle Division , DIV für Integerdivision und MOD um den Rest einer Integerdivision zu ermitteln.

```
program Arithmetic (Output);
var Two, Three : Real;
  Four, Five : Integer;
begin
   Two := 2.0 ; Three :=3.0
   Four :=4; Five :=5;
   write(Two+Three, Four+Five);
   write(Two-Three, Four-Five);
   write(Two*Three, Four*Five);
   write(Two/Three);
   write(Four Div Five, Four Mod Five)
   end.
```

ABS, SQR und SQRT

Die ABS-Funktion ermittelt den Absolutwert, SQR das Quadrat und SQRT ermittelt die Quadratwurzel einer Zahl. Diese Funktionen arbeiten mit Real-, Real6- und Integervariablen.

```
program Numbers (Output)
var    R: Real;
      R6: Real6;
       I: Integer;

begin
  R:= -1.414 ; R6:= 81;
  For I := -3 To 3 Do Writeln(ABS(I));
  writeln(Sqr(R));
  writeln(Sqrt(R6));
```

Vergleichende Operatoren (=, <>, <, <=, >=, >)

Vergleichende Operatoren sind gleich (=), ungleich (<>), kleiner (<), kleiner gleich (<=), größer gleich (>=) oder größer (>). Diese Operatoren dürfen in Vergleichen aller Variablentypen verwendet werden.

```
program Compare (Output);
var One,Two : Real;
    String : Array (. 1..6 .) of Char;
begin
One := 1.0 ; Two := 2.0; String := 'String';
write(One = One, Two = Two);
write(One<>One, One<>Two);
write(One<Two,Two<One);
write(One<=Two,Two<=One);
write(One>=Two,Two>=One);
write(One>Two,Two>One);
write(String='String');
if String > 'St' then
   write(' Das Ganze ist größer als das Teil')
end.
```

Der zuweisende Operator (:=)

In Pascal ist das Gleichzeichen (=) ein zuweisender Operator, obwohl es auch benutzt wird, um Konstanten Werte zuzuweisen oder Typen zu identifizieren. Um einer Variablen Werte zuzuweisen, benutzt Pascal den zuweisenden Operator; ein Doppelpunkt gefolgt von einem Gleichzeichen (:=).
Gemäß dem ISO-Pascal-Standard ist das Einfügen von Leerzeichen vor oder hinter dem zuweisenden Operator irrelevant. Nicht gestattet dagegen ist es, zwischen Doppelpunkt und Gleichzeichen ein Leerzeichen zu setzen (: =).
Es gilt:
Variablenname := Wert

Logische Operatoren (AND OR NOT)

In Pascal-80 ist die Verwendung der Booleschen Standardoperatoren zulässig. Die Verwendung von Klammern entspricht nicht nur der Syntax, sondern macht es auch das Programm übersichtlicher.

```
program Logic (Output);
begin
  writeln((True AND False));
  writeln((True or False));
  writeln( NOT(True));
  writeln(Not(True AND False))
end.
```

Logarithmusfunktionen

EXP und LN

Die Logarithmusfunktionen sind EXP(x), um den natürlichen Logarithmus e zur Basis x zu ermitteln, und LN(x), um den natürlichen Logarithmus einer reellen- oder Integerzahl x zu ermitteln. Obwohl Pascal keinen Exponentialoperator besitzt, kann man diese Funktion mit den Logarithmusfunktionen simulieren.
Beispiel:

```
program Powers (Input,Output);
var Power : integer;
    Number, Result : real;
begin
    readln(Number);
    readln(Power);
    Result := exp(Power * ln(Number));
    writeln(Result);
end.
```

Trigonometrische Funktionen

SIN und COS

Pascal-80 berechnet die Winkelfunktionen nicht in Grad, sondern in Radiant. Dieses Programm demonstriert sowohl die Anwendung von SIN und COS, wie auch die Konvertierung von Grad in Radiant:

```
Program TrigFunctions(Input, Output);
var Degrees,Minutes,Seconds : integer;
   Radians : real;
Begin
  writeln(' Geben SIe einen Winkel ein');
  write ('Grad');
  readln(Degrees);
  write('Minuten');
  write('Sekunden');
  readln(Seconds);
  Radians := Pi * (Degrees + Minutes/60
    + Seconds/3600)/180;
  writeln(' Der Sinus ist,'sin(Radians):10:5);
  writeln(' Der Kosinus ist,'cos(Radians):10:5);
end.
```

ARCTAN

Diese Funktion ermittelt den Arcustangens einer Zahl:

```
program ArcTanDemo (Input,Output);
var Tangent,Degrees : real;
begin
   write(' Was ist der Tangens Ihres Winkels');
   readl(Tangent);
   Degrees := ArcTan(Tangent) * 57.29578;
   write(' Der Winkel beträgt',Degrees:6:1,'Grad.');
end.
```

ArcSin, ArcCos und Tan

Diese Funktionen sind in Pascal nicht standardmäßig implementiert; können aber mit bekannten Funktionen folgendermaßen simuliert werden:

```
function Tan(x:real):real;
begin
  Tan := Sin(x)/Cos(x)
end.
```

```
function ArcSin(x:real):real;
var temp : real;
begin
  temp := sqrt(1-x*x);
  if temp = 0 then
     begin
        if x<0 then ArcSin := -Pi/2
        else ArcSin := ArcTan(x/temp)
end;
```

```
function ArcCos(x:real):real;
begin
  if x=0 then ArcCos := Pi/2
  else if x>0 then ArcCos := ArcTan(sqrt(1-x*x)/x)
      else ArcCos := ArcTan(sqrt(1-x*x)/x)+Pi
end;
```

Speicherverwaltungsfunktionen

PEEK(Adresse)

Mit Peek ermittelt man den Wert einer Adresse. Die Adresse muß eine Integerzahl sein.

```
program Example2(Output);
var Wert,Adresse : integer;
begin
write (' Geben Sie eine Adresse ein');
readln(Adresse);
Wert := Peek(Adresse);
writeln('Inhalt:');
write (Wert);
end.
```

CALL(Adresse,Wert)

Call überträgt einen Wert von Null bis 255 in das A-Register und springt zur angegebenen Adresse. Nach Ausführung der Routine erhält man den aktuellen Inhalt des A-Registers im Integer-Format. Wichtig ist, daß man bei der Benutzung von CALL eine Variable zuweisen muß um den Inhalt von A zu erhalten: ganz egal, ob Sie diesen Wert für Ihre Berechnungen brauchen oder nicht.

```
programm Example3;
var Adress,Byte,Name : integer;
begin;
  Adress:= 73; Byte:= 0
  write (' Irgendeine Taste drücken');
  Name := Call(Adress,Byte);
  writeln(Chr(Name))
end.
```

MEM

Mit MEM ermittelt man den zur Verfügung stehenden freien Speicherplatz(Integer):

```
program Example4(Output);
var Memory : integer;
begin
  Memory := MEM;
  write(Memory)
end.
```

Funktionen zur Filehandhabung

INKEY

Mit dieser Funktion kann man den ASCII-Wert einer gedrückten Taste ermitteln. (Variablentyp Char)
wird keine Taste gedrückt, wird der Wert chr(0) ermittelt.

```
program Example1 (Input,Output);
var Letter := Char;
begin
  repeat Letter := inkey until ord(Letter)>0;
  writeln(Letter)
end.
```

EOF und EOLN

Die Funktionen EOF (Ende des Files) und EOLN (Ende der Zeile) werden werden in Verbindung mit den Prozeduren READ und READLN in einem anderen Kapitel des Handbuchs behandelt.

Ordnungsfunktionen

ORD, PRED und SUCC

Integer-, Boolesche- und Char-Variablen haben eine Eigenschaft gemeinsam, die Real-Variablen nicht besitzen: Sie besitzen einen genau abgegrenzten Bereich möglicher Zahlenwerte und können sowohl einen exakt definierten Vorgänger wie auch einen Nachfolger aus diesem möglichen Wertebereich besitzen.
Mit ORD ermittelt man die Position in einem Datenfeld, mit PRED den Vorgänger einer Variablen eines Datenfeldes und mit SUCC den Nachfolger.
PRED(MinInt) und SUCC(MaxInt) fallen aus dem erlaubten Integerbereich heraus und halten das Programm mit einer Fehlermeldung an.

```
program Ordinals(Output);
var I: Integer;
    C:Char;
begin
  I:= -32000; C:= 'Y';
  writeln(pred(I),succ(I),ord(I));
  writeln(pred(C),' ',succ(C),' ',ord(C));
  writeln(pred(True),succ(True),ord(True));
  writeln(pred(False),succ(False),ord(False))
end.
```

CHR

CHR druckt ein zum ASCII-Zeichenbereich gehörendes Zeichen aus.

```
program Chr;
var ASCII : integer ;
      BigChr : Char
begin
  BigChr := Chr(23);
  cls;
  write(BigChr);
  for ASCII := 32 to 191 do
    write(chr(ASCII)),' ')
end.
```

Konvertierungsfunktionen

ODD, ROUND und TRUNC

ODD ist eine Funktion der Booleschen Algebra und ermittelt den Wert TRUE (wahr), wenn das Argument eine ungerade Integerzahl ist. ROUND rundet eine Real-Zahl auf die nächste ganzzahlige Zahl auf. TRUNC schneidet von einer Real-Zahl die Nachkommastellen ab.

```
program Numbers (Output);
var    N: Real
      I: Integer
begin
  R:=5.6;
  writeln(Round(N),Trunc(N));
  for I:=1 to 10 Do
  begin
    if odd(I) then writeln(I,'ist ungerade.)
  end
end.
```

Vergleich von Variablenfeldern

Elementvergleich, Verknüpfung, Unterschied und Gemeinsamkeit

IN, +, - und *

Mit IN überprüft man, ob ein bestimmtes Element zu einem Variablenfeld gehört, mit + werden zwei Variablenfelder miteinander verknüpft, mit - ermittelt man die Elemente eines Feldes, die nicht zu einem anderen Feld gehören und mit * ermittelt man die Feldelmente, die zu beiden Feldern (Sets) gleichzeitig gehören.

```
program InDemo;(Output)
type S=set of 1..100 ;
var A,B,Union,Intersection,Difference :S;
   I: Integer
begin
  A:=(. 3..5 .);
  B:=(. 5..10 .);
  Union := A+B;
  Intersection := A*B;
  Difference := A-B;
  write('Gemeinsame Elemente='); for I:= 1 to 100 do
    if I in Union then write(I);
  writeln;
  write('Gemeinsame ELemente=');for I:=1 to 100 do
    if I in Intersection then write(I);
  writeln;
  write(' Unterschiedliche Elemente ='); for I:=1 to 100 do
    if I in Difference then write(I);
end.
```

Feldvergleichende Operatoren

Die gewöhnlichen Vergleichsoperatoren, ausgenommen < und > können benutzt werden, um zwei Felder zu vergleichen.

= bedeutet , daß zwei Felder identisch sind
<> beschreibt ungleiche Felder
>= : ein Feld enthält ein anderes Feld
<= : ein Feld ist in einem anderen Feld enthalten.

```
program SetCompare;
type JunkFood= set of (Franks,Burgers,Fries,Sodas,Pizza);
Var BurgerPrince : JunkFood;
   Pizza Prince : Junk Food;
   MacFrank : JunkFood;
begin  BurgerPrince := (. Burgers..Sodas .);
    Pizza Prince := (. Sodas..Pizza .);
    writeln;
    writeln(' Pizza Prince und Burger Prince');
    writeln('Gleichheit-',(PizzaPrince=BurgerPrince));
    writeln('Ungleichheit-',(PizzaPrince<>BurgerPrince));
    writeln ;   writeln('  Mac Frank und Burger Prince');
    writeln('Schließt ein-',(MacFrank>=BurgerPrince'));
    writeln('Beeinhaltet-',(MacFrank<=BurgerPrince));
end.
```

Pascal-80-Prozeduren

WRITE und WRITELN

WRITE und WRITELN übergeben Daten in ein File, einschließlich auf Diskette oder die implementierten Filetypen Output(Bildschirmausgabe) und LP(Druckerausgabe). Output ist eine Option, die nicht extra deklariert werden muß- Write('HALLO') druckt das Wort HALLO auf den Bildschirm aus. WRITELN schließt den Ausdruck einer Zeile etc. mit einem 'End of Line-Character' ab (0D Hex oder 13 Dez.) Dies dient bei Diskettenfiles als EOLN-Begrenzung oder weist bei Druckerausgebe den Drucker an, den Druckkopf auf den Anfang der nächsten Zeile zu positionieren. Wenn Sie die Prozedur WRITE benutzen, wird jeder nachfolgende ausgedruckte Text o.ä. ohne Leerzeichen an den vorangehenden Text angehängt. Um ein benanntes File zu schreiben, benutzen Sie den Pascal-Filenamen als erstes Statement im Write-Befehl: Write(LP,'TEST') gibt das Wort TEST auf den angeschlossenen Drucker aus. Die Benutzumg von Write und Writeln in Text-und Recordorientierten Files wird in einem anderen Kapitel beschrieben (--> FILES).
Real-und Integerausdrücke werden werden in folgendem Format ausgedruckt: WRITE(Ausdruck,Feldweite,Stellen).
Eine Feldweite von -1 erzeugt eine Darstellung in wissenschaftlicher Notation und der Stellenparameter wird ignoriert. Eine Feldweite von Null erzeugt das Leerformat, genauso, wenn keine Parameter spezifiziert werden. Das Leerformat druckt die aktuelle Zahl mit einem Leerzeichen vor der Zahl und sovielen Nachkommastellen wie nötig (Bis zu 14 signifikanten Stellen). Feldweite und Stellenparameter werden als Modulo 256 behandelt.
Beispiele formatierter WRITE-Statements:

```
program Example (Output);
var Number : real;
begin
  Number:= 12345.98765
  writeln(Number);
  writeln(Number:-1);
  writeln(Number:-5:0);
  writeln(Number:10:0);
  writeln(Number:10:2);
end.
```

Die Datenausgabe mit WRITE oder WRITELN kann mit der gedrückt gehaltenen <CLEAR>-Taste unterbrochen werden bzw. durch Drücken der <LEERTASTE> während des Compiliervorgangs.

CLS

CLS löscht den Bildschirm:

```
program Example6(Output);
begin
 cls
end;
```

READ und READLN

Ursprünglich war Pascal als ein System konzipiert, welches seine Dateneingaben von Lochkarten, nicht von einem Terminal erhält. Die Funktionen READ, READLN, EOLN, und EOF wurden so modifiziert, daß sie bei der Dateneingabe von der Tastatur anders arbeiten wie bei Dateneingabe von Diskette.
Wird kein Filenamen angegeben, werden Daten von der Tastatur eingelesen: READ(Variable) ist funktionsgleich mit READ(INPUT,Variable). Bei Tastatureingabe ist der einzige Unterschied von READ zu READLN der, daß READLN nach der Eingabe den Cursor auf den Anfang der nächsten Zeile positioniert, während bei READ kein Zeilenvorschub stattfindet.(Analog zu WRITE) Benutzen Sie die READ-Anweisung, können Sie mehrere Daten (Durch ein Leerzeichen getrennt) in einer Zeile eingeben. Bei Diskettenfiles springt READLN zur nächsten EOLN-Marke, und alle dazwischenliegenden Daten werden eingelesen. Das folgende Program liest Zeichen von der Tastatur ein:

```
program Read1 (Input,Output);
var C:Char;
begin
 repeat
  read(C);
  write(C)
 until eoln
end.
```

In diesem Programm dürfen Sie

```
while not eoln do read(C)
```

nicht schreiben, da die Anfangsbedingung beim READ von Pascal-80 ein EOLN (End of Line) - Zeichen ist. (Siehe auch Beispielprogramm "COINTOSS/SRC" für Beispielprozeduren f. READ)

```
program Read2 (Input,Output);
var  C: array (. 1..16 .) of Char;
     I:Integer;
procedure Input;
begin
  I:=I+1;
  read(C(. I .));
end;
begin
   I:=0;
   C:=' ';
   repeat Input until EOLN;
   writeln(C)
end.
```

In Pascal-80 ist EOF nur dann TRUE, wenn das nächst zu lesende Zeichen eine spezielle EOF-Marke ist; diese wird durch das Drücken der <CLEAR)-Taste produziert. EOF erscheint als Graphikblock (8F Hex oder 143 Dez.) nach der Dateneingabe am Zeilenende.
Mögliche Fehlermeldungen bei Verwendung von READ-Statements:
READ PAST EOLN - Sie haben versucht, ein Zeichen nach dem speziellen EOLN-Zeichen einzulesen.
REDO - Sie haben versucht, ein unerlaubtes Zeichen in eine numerische Variable einzulesen oder eine Zahl außerhalb des Zahlenbereichs in eine Integervariable. Geben Sie die korrekten Daten einfach nochmals ein.
In einer Eingabe kann der Programmlauf durch Drücken der <BREAK)-Taste unterbrochen werden. Sie erhalten die Meldung BREAK AT 0000 .

CLOSE

CLOSE ohne angegebene Parameter schließt alle offenen Files. CLOSE(Filenamen) schließt ein mit ' Filenamen' bezeichnetes File. Alle Files werden automatisch geschlossen, wenn das Programm die Programmausführung unterbricht oder während des Programmlaufs ein Fehler auftritt.

```
program Example8 (Fila : 'File/DAT:0');
var Fila : text;
     Message: Array (. 1..26 .) of Char;
begin
  Message :='Beispiel für das Schließen eines Files';
  write(Fila,Message);
  close(Fila)
end.
```

SEEK(Ausdruck,Filenamen)

SEEK weist einem bezeichneten File ein Record zu, dessen Nummer durch den Ausdruck gegeben ist. Records sind nummeriert und beginnen bei Null. Wenn nötig, wird das File vor der Ausführung von SEEK neu eröffnet.

```
Seek(Record),FILEB);
```

POKE(Adresse,Wert)

Mit POKE transferiert man einen Wert von Null bis 255 in eine Speicheradresse. Geben Sie dezimale Adressen an und subtrahieren Sie 65536 von Adressen, die größer als 32767 sind. Somit ist 8000 Hex = -32768 und 9000 Hex = -31746 usw.

```
program Example7
begin
 poke(15365,65)
end;
```

CASE und ELSE

Die Case-Anweisung ist auf zwei Arten erweitert worden. Ein ELSE darf eingeschlossen werden und wird ausgeführt, wenn kein anderes CASE erfüllt wird. Wird kein CASE erfüllt und ist kein ELSE vorhanden, geht das Programm ohne Fehleranzeige zur nächsten Anweisung.

```
program DemonstrateCase (Input,Output);
var ch:char;
   Stop: boolean;
procedure GetChar;
begin
   write (' Wie würden Sie eine Ja/Nein-Frage beantworten?');
   read(ch);
   writeln;
   case ch of
   'J','j' : Writeln('Sie haben mit JA geantwortet');
   'A','a' : Begin
               Stop:=True;
         Writeln('Sie wollen aufhören.')
         end;
    'n','n' : Writeln (' Sie haben mit NEIN geantwortet.')
     else writeln('Ich verstehe Sie nicht!')
   end
 end;
 Begin
   Stop := False;  repeat GetChar until Stop
end.
```

EX(Ausdruck) und FP(Ausdruck)

EX weist den Exponenten einer reellen Zahl einer Variable vom Typ Integer zu, während FP die Zahl als Bruch ihres Exponenten ermittelt.

```
program Example5(Output);
var Number := real;
begin
 Number:=98.123;
 write(EX(Number));
 write(FP(Number))
end.
```

Files

Pascal-80 bietet zwei unterschiedliche Filetypen: Textfiles und Recordorientierte (File Of...) Files. Wenn Sie in Ihrem Programm Disk-Files für Eingabe oder Ausgabe verwenden, ist es notwendig, diese in der Program-Anweisung zu deklarieren. Pascal-80 läßt auch folgendes Format zu:

```
program Example (FileA :'DATAFILE/DAT:1');
```

Wenn Sie dieses Format wählen, wird jeder Zugriff auf FileA in Ihrem Programm automatisch DATAFILE/DAT auf Laufwerk 1 bezogen. Wählen Sie das allgemeine Pascalformat, deklarieren Sie so:

```
program Easier (FileA,FileB);
```

Dies gibt Ihnen FILEA und FILEB als aktuelle DOS-Filenamen. In Pascal müssen Sie Ihre Filenamen als Variablen deklarieren: der Identifier TEXT steht für Textfiles und der Identifier FILE OF... für Recordorientierte Files:

```
program TextFile(FILA);
var Fila : Text
```

oder:

```
program RecordFile(FileB);
var  FileB : file of real
```

Um einen Ausdruck in ein Textfile zu schreiben, benutzen Sie das Format Write(Filenamen,Ausdruck). Es ist in Pascal-80 nicht extra notwendig, das File zu eröffnen; WRITE eröffnet das File automatisch und legt es auf Diskette an. Eine Write-Anweisung hängt den Text stets an das Ende eines Files an und rückt somit an das Ende des Files vor, bevor dort neuer Text abgelegt wird. Sie können die Parameter Feldgröße und Ziffernanzahl angeben, um Ihre Files so auf Diskette abzulegen, wie Sie sie auf dem Bildschirm erscheinen. (Siehe WRITE-Anweisung). Erlaubt ist es auch, mehrere Ausdrücke in dasselbe Write-Statement einzubauen:

```
Write(Filenamen,Ausdruck1,Ausdruck2);
```

Writeln(Filenamen,Ausdruck) arbeitet wie Write, nur daß an das Ende des Ausdrucks eine 'End of Line'-Markierung angefügt wird (EOLN, ASCII 13 , Carriage Return).

Read(Filenamen,Variable) liest von einem File einen Zahlenwert in eine numerische Variable oder ein einzelnes Zeichen in eine Zeichenvariable ein. Existiert das File, ist jedoch nicht eröffnet, wird es von Pascal-80 automatisch eröffnet und startet den Lesevorgang am Fileanfang. Später erfolgende Read-Anweisungen machen weiter, wo vorangehende Read-Anweisungen aufhörten. Es gibt zwei mögliche Fehlermeldungen beim Lesen von Files:
FILE NOT FOUND bedeutet, daß ein File mit dem von Ihnen angegebenen Filenamen nicht auf Diskette existiert;
MISMATCH ist eine Fehlermeldung, die Sie erhalten, wenn Sie unterschiedliche Variablentypen vermischen und z.B. versuchen, Zeichen in eine numerische Variable oder eine reelle Zahl in eine Integervariable einzulesen.
Readln(Filenamen,Ausdruck) liest ein Zeichen oder einen Zahlenwert aus einem File und geht zum Zeilenende.
Reset(Filename) schließt ein File und eröffnet es neu am Anfang. Das nächste Read-Statement liest das nächste Zeichen oder die nächste Zahl ein, aber eine Write-Anweisung geht zum Ende des Files. Wenn ein File noch nicht auf Diskette existiert, wird es mit RESET angelegt.
Close(Filenamen) schließt ein bestimmtes File; Close ohne Angabe eines Filenamen schließt alle noch offenen Files. Files werden automatisch geschlossen wenn ein Fehler auftritt oder das Programm den Programmlauf unterbricht.
ReWrite(Filenamen) löscht ein File, reserviert den Platz und legt ein neues Leerfile mit dem gleichen Filenamen an.
EOF(Filenamen) ist TRUE, wenn das File an einer EOF-Markierung positioniert ist (8F Hex). Ist das File geschlossen, eröffnet EOF das File und kehrt mit einem FALSE zuruck, außer wenn das File leer ist.
EOLN(Filename) ist TRUE, wenn das File an einer 'End of Line' (EOLN)-Markierung positioniert ist (0D Hex). Ist das File geschlossen, wird es eröffnet.
In Pascal-80 können Sie Read und Write mit Nicht-Textfiles benutzen. Die Syntax ist:

```
Write(Filenamen,Variable,Variable...);
 Read(Filenamen,Variable,Variable...);
```

Beide Filenamen und Variablen müssen zum gleichen Typ gehören. Wir haben folgende Deklarationen:

```
type BigOne = array (. 1..50 .) of real;
var FileA : file of BigOne;
   VarName : BigOne;
```

Damit können wir die Variablen auf das File schreiben mit

```
write(FileA,VarName);
```

Das Write-Statement eröffnet ein File oder legt es an, wenn nötig; nachfolgende Write-Anweisungen setzen am hängen Daten an das Fileende an.
Read(Filenamen,Variable) liest eine Variable von einem File ein. Wenn nötig, wird das File geöffnet; jedoch nicht angelegt wenn es noch nicht existiert hatte. Der Filepointer wird nach jeder Read-Anweisung vorgerückt.

Seek(Ausdruck,Filenamen) weist das File einem Record zu, dessen Nummer durch den Ausruck definiert ist. Das erste Record in jedem File ist mit Null beziffert und das zweite Record ist Nummer Eins. Falls nötig, wird das File vor der Ausführung von Seek eröffnet.
Reset(Filenamen) kann mit Record-Orientierten Filenamen benutzt werden und ist gleichbedeutend mit Seek(0,Filenamen) .
ReWrite(Filenamen) und Close(Filenamen) arbeiten bei Record-orientierten Files genauso wie bei Textfiles.
EOLN,EOF, Writeln und Readln sind in Record-orientierten Files undefiniert. Der Versuch, Daten über das Ende eines Files hinaus einzulesen, führt zu undefinierten Ergebnissen.
Beachten Sie den Unterschied zwischen Record-orientierten Files und Textfiles. In einem Textfile werden Daten stets an das Fileende angehängt; in einem Nicht-Textfile erfolgt das Lesen und Schreiben von Daten an derselben Stelle und der Filepointer muß mit dem Seek- oder Reset-Statement gesetzt werden. Dies gestattet ein Aktualisieren Ihrer Daten durch einfaches Überschreiben.
Es ist möglich, ein File zu schließen und als einen unterschiedlichen Filetyp wieder zu eröffnen; z.B. können Sie ein Textfile als ein file of char einlesen. Ihre Pascal-80-Diskette enthält Demonstrationsprogramme, welche File ein-und Ausgabe verdeutlichen. Die Programme heißen MAILIST/SRC und CREATE/SRC. Laden Sie die Programme ruhig mal in den Editor und sehen Sie, wie sie arbeiten.

Utilities zum Packen und Entpacken von Files

Alle Pascal-80-Sourcefiles werden in komprimierter Form abgespeichert. Immer wenn beispielsweise in einem Programmlisting mehrere Leerzeichen auf dem Schirm erscheinen, wird die Anzahl der Leerzeichen bei der Speicherung auf Disk durch ein Byte codiert, welches die Anzahl der Leerzeichen angibt. (Dieser Code ist 80 Hex plus der Anzahl der Leerzeichen) Dieses Verfahren spart zwar Speicherplatz, macht es aber schwierig, Pascal-80 mit einem anderen Editor zu benutzen (Z.B. SCRIPSIT), oder Pascal-80-Programme auf andere Computer oder Pascal-Compiler zu transferieren.
Um diese Probleme zu umgehen, finden Sie auf Ihrer Pascal-80-Diskette zwei Utilities in Maschinensprache:
ASCII/CMD konvertiert ein Pascal-80-File in ein ASCII-File und
TEXT/CMD konvertiert ein ASCII-File in ein Pascal-80-File.
Die Anwendung ist einfach:
Zuerst speichern SIe Ihr Sourcefile auf Diskette ab. Gehen Sie ins DOS und geben Sie ein:
ASCII Filenamen1 TO Filenamen2 (bzw.:)
TEXT Filenamen1 TO Filenamen2
Auf diese Weise können Sie Ihr Programm PROG1/SRC in Laufwerk 0 in ein ASCII-File in Laufwerk 1 verwandeln:
ASCII PROG1/SRC:0 to PROG1/ASC:1
ASCII-Files wurden auf SCRIPSIT getestet. Es gibt eine Einschränkung: ASCII und TEXT sehen keine End-of-File-Markierung vor; braucht Ihr Editor eine solche Markierung am Fileende, müssen Sie diese anhängen.

Graphik

Es gibt für Video Genie 1/2 prinzipiell zwei unterschiedliche Möglichkeiten in Pascal-80 Graphik zu erzeugen: Mit POKE können Sie ein Zeichen direkt in eine Bildschirmspeicheradresse schreiben (15360-16386) und mit CHR können Sie die Standard-ASCII-Zeichen des Video-Genie 1/2 auf den Bildschirm drucken lassen. Beispiel für einfache Graphikerzeugung:

```
Program GraphDemo;
Begin;
   CLS;
   Write(' ',CHR(141));
   Poke(15900,191);
End.
```

Blockgraphik und Zufallszahlenerzeugung
GotoXY, pSET, pRESET, POINT, RND, RNDR, RANDOM

Diese Funktionen sind äquivalent zu den Basic-Funktionen PRINT@, SET, RESET, POINT, RND und RANDOM. RNDR erzeugt eine reelle Zufallszahl zwischen Null und Eins. Die Befehle werden in folgendem Format angewendet:

```
GOTOXY( X-Koordinate, Y-Koordinate)
pSET( X-Koordinate, Y-Koordinate)
pRESET(X-Koordinate, Y-Koordinate)
POINT( X-Koordinate, Y-Koordinate)
Integervariablenname := RND( Positiver ganzzahliger Wert.)
RANDOM
Realvariablenname := RNDR
```

Das Programm COINTOSS/SRC verdeutlicht die Anwendung der obigen Befehle.

Erstellung von CMD-Files

Das AUTHOR-Programmpaket auf der Pascal-80-Diskette ermöglicht Ihnen, aus Pascal-Programmen CMD-Files zu machen. Die so erzeugten Files können direkt vom DOS ausgeführt werden und erscheinen dem Benutzer wie normale Maschinensprachprogramme.
Das AUTHOR-Programmpaket besteht aus zwei Teilen :
AUTHCODE/CMD und einem Pascal-80-Sourcefile AUTHOR/SRC.
Zur Erstellung von CMD-Files gehen SIe folgendermaßen vor:

1) Schreiben Sie ein Programm mit Pascal-80
2) Compilieren Sie Ihr Programm und speichern Sie es mit der <W>-Option des Monitors auf Diskette.
3) Gehen Sie ins DOS, geben Sie AUTHCODE ein und drücken Sie <NEW LINE>.
4) Gehen Sie ins Pascal zurück mit PASCAL <NEW LINE>.
5) Laden Sie mit der Option <L> das Programm AUTHOR/SRC.
6) Drücken Sie <R>, um das Program zu compilieren und zu starten.
7) Folgen Sie den Anweisungen auf dem Schirm.
   Sie werden gefragt:
   a) Ob Sie AUTHCODE/CMD laden wollen;
   b) Ob Sie Ihr File compiliert haben und
   c) Wie Sie Ihr File nennen wollen.
8) Antworten Sie <N> und <Y> und geben Sie einen Filenamen an, der noch nicht auf der Diskette steht. Nun werden Sie gefragt, ob Sie fertig sind, fortzufahren.
   Antworten Sie mit <Y>.
9) Sie erhalten nun weitere Informationen. Lesen Sie diese und drücken Sie <NEW LINE>.
10) Nun befinden Sie sich wieder im Monitor-Menü. Drücken Sie <X> und geben Sie den Namen Ihres Pascal-Object-Files an. Es wird geladen und das Programm schreibt Ihnen Ihr File automatisch als CMD-File.
11) Ist der Rechner fertig, befinden Sie sich im DOS und können nun Ihr Programm testen. Hieß Ihr Programm beispiels weise CHESS/CMD, starten Sie das Programm mit
    CHESS <NEW LINE>.

ACHTUNG: testen Sie Ihr CMD-Programm sorgfältig, um sicherzugehen, daß es korrekt arbeitet! (Neuwertige Diskette verwenden und Programm nach Kaltstart starten.)

Demonstrationsprogramme

Außer den schon erklärten Utilities befinden sich auf der Pascal-80-Diskette weitere Demoprogramme:

HANOI/SRC demonstriert, wie schnell Pascal-80 das bekannte 'Türme von Hanoi' - Problem löst.

PRIME/SRC ist ein Programm, daß nach der Methode des 'Sieb des Eratosthens alle Primzahlen zwischen Eins und 20000 herausfindet.

COINTOSS/SRC simuliert das Werfen einer Münze. Die Münze wird in Serien von 10 Würfen hochgeworfen und die Anzahl der 'Kopf'-Würfe wird als Graphik ausgegeben. Erreicht ein Graph die obere Achse, wird neu skaliert. Wenn Sie das Programm eine Weile laufen lassen, wird der Graph der Gaußschen Normalverteilung (Glockenkurve) erkennbar.

Die Programme CREATE/CMD und MAILIST/SRC dienen als Demonstration für Fileverwaltung. Zuerst wird mit CREATE ein Dummyfile erstellt, dann können Sie es mit MAILIST bearbeiten. Sie haben folgende Kommandos zur Verfügung, während das Programm läuft:

T Hole das erste Record
+ Hole das nächste Record
- Hole das vorangehende Record
L Hole das letzte Record
S Suche Namen
N Hänge ein neues Record an
D lösche aktuelles Record
H Drucke das aktuelle Record als Hardcopy aus
P Drucke die gesamte Liste
Q Zurück zum Monitor oder DOS.

Dieses Mailingprogramm ist gedacht zur Demonstration von Filebehandlung und -Verwaltung; und keineswegs für ernsthafte Anwendung konzipiert. Natürlich bleibt Ihnen frei, es für Ihre eigenen Anwendungen zu modifizieren.

Fehlermeldungen des Compilers

Der Compiler unterbricht, wenn er einen Fehler im Programm findet und positioniert an die betreffende Stelle einen Pfeil. (Oder ein Ä, je nach Zeichensatz)
Dies kann z.B. in der Zeile nach dem Fehler stattfinden (Wenn nach einem Ausdruck ein Semikolon fehlt o.ä.). Ein fehlendes END-Statement kann unter Umständen mehrere Zeilen nicht entdeckt werden.

BAD OPTION

In Pascal-80 wird alles vor dem Keyword PROGRAM als Compileroption interpretiert; siehe COMPILEROPTIONEN.

SYNTAX ERROR

Etwas ist falsch, aber der Compiler weiß auch nicht exakt, was. Dieser Fehler kann z.B. auftauchen, wenn Sie ein Semikolon zwischen zwei Statements vergessen haben oder versucht haben, eine Zahl mit einem Dezimalpunkt zu beginnen. (In Pascal ist

UNDECLARED

Ein Identifier (Gewöhnlich eine Variable) oder ein Label wurde vergessen zu deklarieren. In Pascal müssen alle Variablen in einem VAR-Statement v o r der Benutzung deklariert werden. Die in einem Record gebrauchten Feldindexvariablen müssen globale Variablen sein, lokale Variablen erzeugen die Fehlermeldung UNDECLARED.

DUPLICATE

Sie haben einen Namen im gleichen Block doppelt deklariert; kann auch bedeuten, daß Sie den gleichen Namen in einem Record und gleichermaßen in Ihrem Programm verwendet haben. Pascal-80 gestattet auch nicht, einen Filenamen mit dem Namen eines Standard-Identifiers zu belegen.(Es ist zwar erlaubt, diese Standard-Identifier neu zu definieren, nicht aber als Filenamen)

BAD RANGE

Unlogische Bereichs - oder Feldgrenzenangabe (z.B. (. 10..1 .)

REAL OVERFLOW

Eine reelle Konstante ist fällt aus dem erlaubten Zahlenbereich von 1E-63 bis 1E+63 heraus.

BAD TYPE

Unerlaubte Typendeklaration. In Pascal müssen Parameter als Typen vordefiniert werden;
type Ran = 1..10 ; procedure Test(par:Ran) ist in Ordnung, nicht aber:
procedure Test(par:1..10).
Beachten Sie auch, daß Pascal-80 bestimmte Strukturen, wie z.B File of File nicht unterstützt.

OUT OF MEM

Bedeutet meistens, daß der Compiler nicht mehr genug Speicherplatz für seine Symboltabelle zur Verfügung hat, benutzen Sie die MEMORY-Option des Compilers, um herauszufinden, vieviel Speicherplatz noch verbleibt. Der Fehler tritt auch auf, wenn der Compiler nicht mehr genug Platz zum Abspeichern von Labels (Max. 63) oder Disk-Filenamen (Maximum : 12) besitzt. Außerdem darf ein Programm nicht mehr als 252 verschiedene Skalartypen besitzen und ein Skalar darf aus maximal 255 Elementen bestehen. Es kommt bei kleineren, aber tief verschachtelten Programmen gelegentlich vor, daß dem Compiler der Speicherplatz für den Stack ausgeht, bevor der Speicherplatz für die Symboltabelle knapp wird. Auch hier wird OUT OF MEM ausgegeben; reservieren Sie sofort mehr Platz für den Stack (Auf Kosten des Symboltabellenspeicherplatzes) und compilieren SIe neu. Der zusätzliche Stackspeicher steht bei nachfolgenden Compilierungen weiter zur Verfügung, bis Sie das System neu starten.

MISMATCH

Es wurde eine Operation oder Zuweisung mit inkompatiblen Typen versucht (z.B 'X'+2). Beachten Sie, daß in Pascal zwar rellen Variablen Integerwerte zugewiesen werden können, nicht aber dürfen Integer-Variablen relle Werte zugewiesen werden (Benutzen Sie die Funktion TRUNC). Die Zahl -32768 darf aufgrund der Syntax von Pascal-80 keiner Integer-Variablen zugewiesen werden. Ein MISMATCH kann durch inkompatible Fileoperationen ausgegeben werden; z.B. wie bei EOF(Output) oder wenn versucht wurde, auf einen Filenamen Bezug zu nehmen, der noch nicht in einem VAR-Statement deklariert wurde.

UNRESOLVED GOTO

Die Sprungadresse nach einem GOTO existiert nicht. Dieser Fehler wird stets ganz am Ende der Compilierungsphase ausgegeben, weil der Compiler bis zuletzt hofft, daß das Label irgendwo auftaucht.

STRUCTURE TOO BIG

Es wurde versucht, ein Set mit mehr als 256 Elementen (oder mit Integers außerhalb 0 ..255) zu deklarieren. Oder es wurde versucht, einem Block mehr als 65535 Bytes Speicherplatz zuzuweisen (Gewöhnlich in Form von Arrays). Oder eine Struktur ist so tief verschachtelt, daß der Compiler sie nicht mehr handhaben kann (Array of Array of Array..bis zu Tiefe von ca. 30 Verschachtelungen) oder es wurde versucht, ein Array oder Record mit mehr als 510 Bytes als Werteparameter einzusetzen. (Diese Einschränkung gilt nicht für Variablenparameter)

BREAK

Die Compilierung wurde durch Drücken der Taste <BREAK>

Fehlermeldungen während der Programmausführung

Tritt bei der Ausführung eines Programms ein Fehler auf, wird unterbrochen und das System druckt (In Hex.) die Position aus, wo der Fehler auftrat. diese Position entspricht den Zeilenangaben, die der Compiler während des Compiliervorgangs ausgegeben hat und gestattet ein schnelles Lokalisieren des Fehlers.
Einige Fehlermeldungen sind selbsterklärend:
OUT OF MEM, DIV BY 0, DISK ERROR, BEYOND EOF.
Andere Fehlermeldungen sind:

BAD RANGE

Die Dimension eines Arrays oder der Wert einer Feldgrößen beschreibenden Variablen-ist außerhalb des in Ihrem Programm deklarierten Wertebereichs.

REAL OVERFLOW

Das Ergebnis einer Berechnung ist außerhalb des Zahlenbereichs 1E-64 <= N < 1E+63 (Sowohl Under- wie auch Overflow)

INT. OVERFLOW

Das Ergebnis einer Integer-Operation ist außerhalb des Bereichs -32768 <= n <= 32767. Beachten Sie, daß Pascal nicht automatisch ein Egebnis in eine reelle Zahl verwandelt, wenn es zu groß is.

MISMATCH

Ein unzulässiges Zeichen wurde gefunden, während versucht wurde, eine Nummer von Diskette zu lesen. Oder ein nicht-Integer wurde gefunden, als versucht wurde, einem Integer einen Wert zuzuweisen. Passiert das, wenn eine Zahl von der Tastatur eingelesen wurde (File INPUT), wird die Meldung REDO ausgedruckt.

STRUCTURE TOO BIG

Während des Programmlaufes wurde versucht, ein Set mit mehr als 256 Elementen zu erzeugen, z.B.:
(. A..B .) , wenn A=1 und B=300; oder wenn versucht wurde, einer Setvariablen ein größeres Set zuzuordnen, als für diese Variable deklariert wurde. Weil der Platz für Sets als Vielfaches von 16 angelegt wir, kann ein Set, welches als 0..10 deklariert wurde, bis zu 15 Elementen erhalten, ohne daß eine Fehlermeldung ausgegeben wird.

ILLEGAL JUMP

Illegaler Sprung (Mit GOTO) in eine FOR-Schleife oder ein Case Statement bzw. in eine inaktive Prozedur oder Funktion.
Allgemein können Sie von einer tieferen Verschachtelungsebene in eine höhere zurückspringen (Z.B aus einer Schleife), aber es ist nicht erlaubt, in eine tiefere Verschachtelungsebene hineinzuspringen. Wenn Sie aus einer Funktion herausspringen, ohne der Funktion einen Wert zuzuweisen, wird der Wert Null ermittelt. Pascal-80 gestattet Sprünge von einer FOR-LOOP zur anderen, sofern beide der gleichen Verschachtelungsebene angehören; allerdings ist die Kontrollvariable undefiniert.

Neu in Pascal-80 implementierte Funktionen

Die Kommandos SP und DP

Normalerweise bringt die Verwendung von Variablen mit sechsstelliger Genauigkeit gegenüber Variablen mit 14-stelliger Genauigkeit nur Speicherplatzersparnis, jedoch keine Beschleunigung des Rechenablaufs.
Das Kommando SP aber erhöht die Verarbeitungsgeschwindigkeit um etwa 30%, indem es die Genauigkeit auf 'Single Precision' beschränkt. Das Kommando DP schaltet zurück auf doppelte Genauigkeit. Hier ist ein Beispielprogramm, welchs die Verwendung von SP und DP erläutern soll:

```
program DemonstrateSinglePrecision (output);
var I : integer;
    J,K,L:real;
begin
  cls;
  K:=999; L:=99;
  writeln('Anfang einfache Genauigkeit');
  SP;
  For I:=1 to 500 Do J:= K * L;
  writeln('Fertig');
  writeln('Anfang doppelte Genauigkeit');
  DP;
  For I:= 1 To 500 Do J:= K * L;
  Writeln('Fertig.');
end.
```

Das Wort PACKED

Die Hinzufügung dieses Wortes hat an sich keine Bedeutung; es wird lediglich eine Standardmäßige Definition eines Strings als PACKED ARRAY OF CHAR ermöglicht.

Pointer, NEW, NIL, MARK und RELEASE

Neu in Pascal-80 ist die Implementierung von Pointer-Variablen und Pointer-Symbolen. Pointer-Symbole ist das Zeichen '§' und das sog. 'Caret'- Zeichen (Zu erreichen mit <SHIFT>&(RECHTSPFEIL> im Editor). Pointer-Variablen gestatten eine dynamische Neuverteilung des Speicherplatzes. Für viele Leute ist dies in Bezug auf Stringverwaltung von Vorteil; auch das unten stehende Beispielprogramm demonstriert die Verwendung von Pointern in Verbindung mit Stringfunktionen. Um den für Pascal-80 erforderlichen Code zu begrenzen, werden MARK und RELEASE anstelle des Befehls Dispose verwendet. Dispose löscht in Verbindung mit einer Bezugs-Pointervariablen (Verschachtelte Bezugspointer) Speicherplatz. MARK ermittelt eine Position in einem 'Heap' und RELEASE eliminiert alle 'Heap'-Positionen, welche nach dem letzten MARK-Statement benutzt wurden und zerstört alle inzwischen aufgebauten Datenstrukturen.
(Anmerkung: im Listing bedeutet '-' das 'Caret'-Symbol.)

```
program ConcentrateStrings ( input,output);
type String = record
                Entry : Char;
                Next : -string;
              end;
var S,T,U : string;
    I     : -integer;
procedure WriteS( S : String);
begin
  Write( S. Entry);  write ( S.Next);
  if S.Next = nil then writeln else Writes( S.Next-);
end;
procedure ReadS ( var S : String);
var C: Char;
begin
   read (C);
   S.Entry := C;
   if eoln then S.Next := nil else
     begin
     new(S.Next);
     ReadS(S.Next-);
   end;
end;
procedure Conc(S,T : String; var U : String);
begin
  U.Entry := S.Entry;
  if S.Next = nil then
       begin
         new(U.Next);
         U.Next- := T
        end
```

```
    else
       begin
         new(U.Next);
         Conc(S.Next-,T,U.Next-)
       end;
end;
begin
  repeat
    mark(I);
    ReadS(S); Read S(T);
    WriteS(S); WriteS(T);
    Conc(S,T,U); WriteS(U);
    release(I)
 until false
end.
```

Nicht-standardmäßige Pointerfunktionen

Die Anwendungsmöglichkeiten der Pointervariablen wurde so erweitert, daß nunmehr Operationen, die sonst die Verwendung von 'Variant Records' erfordern würden, möglich sind.
Die normale Syntax der 'Variant Records' erlaubt es einem geschickten Programmierer, die sehr streng definierte Struktur von Pascal zu umgehen. Das folgende Programm, welches nicht in Pascal-80 funktioniert, transferiert R in ein Record als reelle Zahl und verwandelt es in ein Array von Zeichen:

```
var A : Record
          Case 1..2 of
         1 : (One : real);
         2 : Two : Array_(. 1..8 .) of char)
       end;
A.One := R;
for I := 1to 8 do write (ord(A.Two(. I .)));
```

Obwohl Pascal-80 keine 'Variant Records' besitzt, läßt sich obiger Effekt mit folgender Methode simulieren:

```
Var One : §real;
    Two : §array (. 1..8 .) of char;
    New(One);
    One§ := R;
    Two := One;
    For I := 1 to 8 do write(Ord(Two§( . I .)));
```

Hier sind noch einfachere Beispiele für die nicht standardmäßige Typenumwandlung:

```
var I : §char;
    J : §array(. 1..10 .) of char;
  I := J;
  writeln(I);
und:
var A : §integer;
    B : §char;
    A := B;
```

Die Pointervariablen wurden zudem so erweitert, daß nun direkter Zugriff auf Adressen möglich ist (Wie in der Sprache 'C'). Hier ist ein Beispielprogramm, welches in Standard-Pascal nicht funktioniert (Aber in Pascal-80) und mit dem man aktuelle Adressen ausdrucken oder berechnen kann:

```
var A : §integer;
writeln(A);
writeln(A+1);
```

(Anmerkung : In den Programmlistings entspricht das Paragraphenzeichen dem "Klammeraffen" (Rechts neben der P-Taste)

Anhang

****

Pascal-80-Problem:

Der Cursor-Block bleibt während der Programmausführung auf dem Bildschirm stehen. Dieses tritt manchmal auf, wenn Sie eine Prozedur von Diskette ausführen. Die Lösung ist einfach: schalten Sie den Cursor mit dem Kommando
WRITE(CHR(15));
einfach aus. (Ist bei Programmen, die mit dem AUTHOR-Paket umgewandelt wurden, nicht nötig.)

****

ANMERKUNG: Auf Ihrer Pascal-80-Diskette existiert ein File CTRLKEY/CMD. Dieses File ist für die Benutzer von Video-Genie 1/2 - Computern ohne Bedeutung.

P-Codes (Partial List)

5A	Add next two numbers to adress of the start of P-Code and store on Stack (Store Address)
06	Store next two numbers on Stack
B1 02	Pop two numbers from stack Pop next address - Store numbers at that address
B1 08	Place top 8 Stack numbers in address under them
96 02	Get two bytes from address on top of stack Put them on stack
2A 3C	Integer Addition
2A 3F	Integer Subtraction
2A 45	Integer Multiplication
2A 4B	Integer Division
2A 4E	Mod
36 72	Integer ABS
36 42	Integer SQR
36 51	PRED
36 54	SUCC
CF	MEM
0C	Put next 4 bytes on stack Pad with 4 zeros to make 8 byte real number
18	Put 8 byte real on stack
33 18	Real addition
33 1B	Real subtraction
33 21	Real multiplication
33 24	Real division
39 00	Real ABS
39 1E	Real SQR
39 60	SQRT
39 5D	ROUND
39 5A	TRUNC
B4 00 FF	Ceck integer - Between 0 and FF?
B4 00 01	Check Boolean - 0 or 1?
D2	PEEK
D5	CALL
D8 00	POKE
D8 01	GOTOXY
D8 02	PSET
D8 03	PRESET
D8 04	RANDOM
D8 05	SP
D8 06	DP
D8 07	RNDR
D8 08	POINT
D8 09	RND
D8 0A	MARK
D8 0B	RELEASE
D8 0C	NEW
D8 0D	IN HEAP?

Pascal- 80 Memory Map

(Extremly subject to change from revision to revision)

5200 to 523F	Buffer (Multiple Use)
5240 to 52FF	P-Code Routines
5300 to 533F	Buffer
5340 to 54FF	P-Code Routines
5500	Start of P-Code Jump Table
5500 to 652E	P-Code and Floating-Point Routines
572B	Primary P-Code Routine (Restart 08H Vector)
6990 to 6B91	Disk I/O Handler for Monitor
6BA0 to 6DF5	Error Routines
6EE0 to 6FFF	Termination routines
7000	Original Tansfer Address
7000 to 70FF	Jump and Control Table
7100 to 78F0	P-Code for Log & Trig Functions
78F0 to 797C	Editor Functions
797D to 7B45	P-Code Patch Area
7B46 to 82EC	Editor Functions
8000	Find Tokens in Text Buffer
8300 to 85B8	Compiler Utility Routines
85BA to 878C	Compiler Lookup Table
8790 to 9EB3	Compiler Variable Declarations Handler
9EB4 to ABF2	Modifications and Patch Area
9EE0	Brackets
9F09	Control Characters
9F49	Character Insert
9F7A	Character Delete
A080	Transfer Address - Set Up for Model I or III
A280	Include
A368	Editor Patches
A400	Allow Declarations in Records
A47E	Lower Case Routines
A9C3	Pointer Variables
AB00	Pointer Symbols
ABF2 to ABFF	Text Flags
AC00	Text Buffer
	P-Code (Begins at 8000 under X option)
	Local Variables
	Stack (Default 256 Bytes, can be larger)
	Heap (moves Stack down if space required)