23 Include-Dateien, Units und Overlays

Gliederung

23.1 Vorbemerkungen

Die Instrumente *Include-Dateien, Units* und *Overlays* dienen zum Erzeugen größerer Programme, was z.B. schon notwendig wird, wenn der Quelltext die 64-KByte-Grenze überschreitet.

Zu Include-Dateien

- Mit dem Compilerbefehl {**$I** *dateiname* } kann eine beliebige Pascal-Quelltextdatei in das aktuelle Programm eingefügt werden. Hinweis: Zwischen der geschweiften Klammer/AUF und dem Dollarzeichen des Compilerbefehls dürfen keine Blanks stehen.
- Die Deklaration der Include-Datei muß *vor* dem Ausführungsteil des aktuellen Programms stehen.
- Die Include-Datei muß einen abgeschlossenen Programmblock enthalten, d.h. vollständige Deklarationen von Konstanten, Typen, Variablen, Prozeduren und Funktionen.
- Es können mehrere Include-Dateien aufgeführt werden.
- Include-Dateien dürfen selbst auch wieder Include-Dateien bis zu einer 15-fachen Schachtelungstiefe enthalten.
- An Prozeduren und Funktionen von Include-Dateien können in üblicher Weise Parameter übergeben werden, d.h. mit Wert oder mit Adresse (»*var*«).
- Bei jeder Compilation des aktuellen Programms muß auch die Include-Datei compiliert werden, was Zeit kostet.

Zu Units

- Units sind compilierte Programm-Module mit der Extension »*.TPU*« (Turbo-Pascal-Unit) bzw. »*.BPU*« (Borland-Pascal-Unit) Der Quelltext der Unit beginnt mit dem reservierten Wort »*unit*« und enthält einen Interface-Teil, einen Implementationsteil und einen optionalen Initialisierungsteil.
- Der **Interfaceteil** beginnt mit dem reservierten Wort »*interface*«. Er enthält die öffentlichen Deklarationen, die somit von anderen Programmen genutzt werden können. Der Interfaceteil stellt somit die Software-Schnittstelle zu anderen Programmen dar; der Anwender der Unit muß die *öffentlichen* Deklarationen kennen, nicht aber deren Implementierung. Der Interface-Teil kann beliebige *öffentliche* Deklarationen enthalten (andere Units, Konstanten, Typen, Variablen, Prozeduren und Funktionen).

- Der **Implementierungsteil** beginnt mit dem reservierten Wort »*implementation*«. Er enthält die eigentliche Programmierung der im Interfaceteil aufgeführten Prozeduren und Funktionen. Der Implentierungsteil kann beliebige *private* Deklarationen enthal-

ten (Units, Konstanten, Typen, Variablen, Prozeduren und Funktionen). Die *privaten* Deklarationen sind dem Benutzer der Unit nicht zugänglich.

- Der **Initialisierungsteil** ist optional. Falls vorhanden, dann ist er mit »*begin*« einzuleiten. Der Initialisierungsteil ist im wesentlichen für die Initialisierung von öffentlichen Variablen vorgesehen und wird deshalb nur selten gebraucht.
- Mit Units kann man Turbo-Pascal in einfacher Weise um häufig benötigte Prozeduren und Funktionen erweitern.
- Turbo-Pascal selbst stellt acht Standard-Units zur Verfügung (*SYSTEM*, *CRT*, *DOS*, *PRINTER*, *OVERLAY*, *GRAPH*, *TURBO3* und *GRAPH3*). Die ersten fünf sind in der Datei *Turbo.TPL* gespeichert, die restlichen drei in der Datei *Graph.TPU*.
- Mit Ausnahme der Standard-Unit *SYSTEM* müssen alle anderen Units und somit auch die eigenen Units bei Gebrauch mit »*uses*« deklariert werden.
- Eine (compilierte) Unit kann bis 64 KByte groß sein.

Zu Overlays

- Mit Units können zwar sehr große Programmpakete entwickelt werden; durch die Beschränkung des DOS-Speichers auf 640 KByte kann es aber doch zu Problemen kommen, da Teile des Speichers vom Betriebssystem belegt sind. Einen Ausweg bietet bei geeigneter Programmstruktur die Overlaytechnik, mit der im Prinzip grenzenlos große Programme entwickelt werden können.
- Unter Overlay-Technik versteht man das Nachladen von verschiedenen Programm-Modulen in immer den gleichen Teil des Arbeitsspeichers. Der Zeitpunkt des Nachladens wird durch das Programm selbst gesteuert. Beim Nachladen wird das nicht mehr benötigte alte Modul im Arbeitsspeicher überschreiben; es steht aber auf dem Datenträger (Platte/Diskette) nach wie vor zur Verfügung und kann bei Bedarf auch wieder geladen werden.

23.2 Demo Include-Dateien

```
program Pas23021;  { Demo Kap 23.1: Include-Dateien }

uses
  CRT;

{$I C:\Student\Pas23022.PAS  >>>>> die Include-Datei mit
    Zugriffspfad, der ggf. anzupassen ist. }

var
  i:    Integer;
  Name: string;

begin
  ClrScr;
  Name := ' Anton';

  Write('Hier Hauptdatei ...................: ');
```

```
  for i := 1 to 5 do
    Write('*');
  WriteLn(Name);

  WriteLn('Hier Konstante aus der Include-Datei: ', NN);

  IncludeDemo(Name);  { Prozedur aus Include-Datei }

  Write('Hier Hauptdatei ...................: ');
  for i := 1 to 5 do
    Write('-');
  WriteLn(Name);

  { Die Bildschirmausgabe: }
  { |Hier Hauptdatei ...................: ***** Anton   }
  { |Hier Konstante aus der Include-Datei: Huber Toni    }
  { |Hier Prozedur  aus der Include-Datei: !!!!! Anton   }
  { |Hier Hauptdatei ...................: ----- Huber   }

  repeat
  until ReadKey <> '';
end.
```

```
 { Include-Datei »Pas23022.PAS«.  Kein "program" bei Include-Datei }
const
  NN = 'Huber Toni';

procedure IncludeDemo(var s: string);
var            { Parameterübergabe hier mit Adresse !!! }
  i: Integer;
begin
  Write('Hier Prozedur  aus der Include-Datei: ');
  for i := 1 to 5 do
    Write('!');
  WriteLn(s);
  s := ' Huber';
end;
```

23.3 Demo Units

```
program Pas23031;  { Demo Kap. 23.3:  Units }

uses
  CRT, PAS23032;   { ... mit der eigenen Unit »PAS23032.TPU« }

var
  x, y: Real;

begin
  ClrScr;

  extWriteXY(10, 10, 'Eine Prozedur aus der Unit »PAS23032.TPU«');
         { Die Bezeichner der externen Routinen lasse ich zur
           besseren visuellen Erkennung mit "ext" beginnen.
           Ansichtssache! }

  WriteLn; WriteLn;

  x := 100.0;    { Man teste auch mit:  x := -100.0; }
  y := extLog10(x);
```

```
  Write('Der dekadische Logarithmus von ', x:6:2, ': ', y:6:2);

  repeat
  until ReadKey <> '';
end.
```

```
unit Pas23032;     { Quelltext: »Pas23032.PAS«. Mit "Compile/Compile" }
                   { oder "Alt-F9" auf "Destination Disk" compilieren }

interface          { -----  1. Unit-Teil: Interface ----------------- }
                   { ... Öffentliche Deklarationen .................. }
                   { Hier: Eine Prozedur und eine Funktion   ...      }

  procedure extWriteXY(Spalte, Zeile: Byte; Meldung: string);

  function  extLog10(x: Real): Real;


implementation     { -----  2. Unit-Teil: Implementation ------------ }

  uses            { Private Deklaration einer Standard-Unit wegen }
    CRT;          { späterem »GotoXY« in der Implementierung      }

  procedure Abbruch(z: Real);   { Private Deklaration }
  begin                         { einer Prozedur      }
    WriteLn;
    WriteLn('Das Argument des Logarithmus ist mit ', z);
    WriteLn('kleiner oder gleich Null. Abbruch nach Return ...');
    ReadLn;
    Halt(1);  { >>>>>>>>>>>> }
  end;

  procedure extWriteXY;  { Implementierung der Prozedur »extWriteXY« }
  begin
    GotoXY(Spalte, Zeile);
    Write(Meldung);
  end;

  function extLog10;     { Implementierung der Funktion »extLog10«  }
  begin                  { Hier: Dekadischer Logarithmus }
    if x > 0
       then extLog10 := Ln(x)/Ln(10)
       else Abbruch(x);  { Aufruf einer privaten Prozedur }
  end;

{ begin }          { -----  3. Unit-Teil: Initialisierung ----------- }
                   {        Optional. Hier nicht vorhanden.           }
end.               { -----  Ende der Unit -------------------------- }
```

23.4 Demo Overlays

Fehlt noch ... (kommt auch nicht mehr! kha)