Grundlagen wissenschaftlichen Arbeitens

An introduction to the Extended Pascal Language

Unter der Leitung von
Univ.-Prof. Dr. Jens Knoop

Institut für Computersprachen
Fakultät für Informatik Technische Universität Wien

Name : CELIK Mehmet
MatNr : 0127103
Kennzahl : 534
E-Mail : mehmetcelik9@hotmail.com

Einleitung

Pascal ist eine Programmiersprache, die 1968 bis 1972 von dem Schweizer Informatiker Niklaus Wirth entwickelt wurde. Sie ist nach Blaise Pascal benannt. Berücksichtigt man Leistungsfähigkeit, Logische Struktur, Einfachheit und die leichte Erlernbarkeit insgesamt. So dürfte Pascal zu den modernsten heute existierende Programmier Sprache zählen .Pascal ist eine Weiterentwicklung von ALGOL. Das wichtigste Konstruktionsprinzip war, die Sprache so einfach wie möglich zu gestalten, damit sie in der Ausbildung genutzt werden konnte. Gleichzeitig sollte strukturierte Programmierung möglich sein. Alle Variablen müssen vor der Benutzung deklariert werden. Der erste Pascal-Compiler selbst war mit ca. 6000 Codezeilen recht klein und erzeugte eine plattformunabhängige, assemblerähnliche Zwischensprache, den P-Code. Zu den ersten Implementierungen des Pascal-Compilers gehörte Pascal 6000, das auf Großrechnern vom Typ Cyber der Firma Control Data Cooperation (CDC) lief. Pascal zeichnete sich durch die strikte und einfach verständliche Syntax aus, sowie dadurch, dass es keine kontextabhängigen Interpretationen des Codes gab. Somit erlaubte Pascal im Vergleich zu Sprachen wie C und Fortran eine gute Lesbarkeit und verglichen mit Fortran auch eine bessere Unterstützung von strukturierter Programmierung. Einer der wesentlichen Nachteile der ursprünglichen Implementierung war, dass eine Modularisierung im Sinne getrennter Compilierung nicht vorgesehen war - ein Manko, das mit der Weiterentwicklung Modula-2 behoben wurde

Pascal erlebte eine Blütezeit, als die Firma Borland eine Version mit dem Namen Turbo Pascal publizierte, in der die ursprünglich nur zu Lehrzwecken geschaffene Sprache so stark erweitert wurde, dass viele weit verbreitete Programme damit entwickelt wurden. U.a. wurde dabei auch die Objektorientierung in Pascal eingeführt.

Standards

Standards, die sich auf Pascal beziehen:

1. Standard Pascal: ANSI(American National Standard Institution)/IEEE770X3.97-1993 oder ISO(International Standard Institution)7185:1990
2. Extended Pascal: ANSI/IEEE770X3.160-1989 oder ISO/IEC 10206:1991

Unterstützt ISO-7185 Pascal Vollständig und ISO-10206 Extended Pascal fast vollständig

Hallo Welt in Pascal

Das Beispielprogramm *Hallo Welt* sieht *in Pascal* folgendermaßen aus:

```
Program Hallo (Output);
Begin
  Writeln ('Hallo Welt');
  Readln;
End.
```

Extended Pascal enthält einige Größere neue Merkmals, aber vor Berücksichtigung dieser ist es nützlich einen überblick mit einigen Verbesserungen auf zurufen.

Einfache Datentypen

Zu den einfachen Datentypen Zählen die Standardtypen *integer, real, boolean, und char , ,aber werden mit Extended Pascal Complex und String neu eingeführt.*

1. integer (Ganze zahlen)
2. Real (Reelle zahlen)
3. Boolean (wahrheitswerte)
4. Char (Zeichen)
5. Complex (komolexe zahlen)
6. String (zeichen ketten)

. # Konstanten, Typen ,Variablen

1) In einer Konstantendefinition können Namen für konstanten (benamte konstanten) festgelegt werden.

```
const
    Name = konstante; …{nicht leer}
```

Die rechts stehende Konstante ist dabei eine Literal konstante oder eine bereits zuvor Definierte Konstante. Benamte Konstanten Können wie Literal konstanten in einem Programm verwendet werden. Sie sind während der Programmausführung Unveränderlich.

2) In einer Typdefinition können Namen für Typen festgelegt werden.

```
type
    Name = Typ; …{ nicht leer }
```

Der rechts stehende Typ ist dabei ein explizit angegebener Typ oder ein zuvor Definierter Typ.

3) In einer Variablenvereinbarung können Namen für variablen festgelegt werden.

```
var
    Namensliste: Typ; …  {  nicht leer  }
```

Die in der Liste aufgezählten Namen bezeichnen Variablen zum rechts stehenden Typ. Eine Variable kann als symbolische Adresse eines entsprechenden Speicherplatzes interpretiert werden.

if- Anweisung

Die if- Anweisung ermöglicht die wahlweise Ausführung von zwei Anweisungen :
and_then ist ein ISO 10206 Extended Pascal Erweiterung

```
if  logischer Ausdruck   then    Anweisung
                  else    Anweisung {kann entfallen }
```

```
program AndBug;
var p: ^Integer;
begin
        New (p);
        ReadLn (p^);
        if (p <> nil) and (p^ < 42) then          { This is NOT safe! }
        WriteLn ('You"re lucky. But the test could have crashed ...')
end.
```

```
program And_ThenDemo;
    var p: ^Integer;
    begin New (p);
            ReadLn (p^);
            if (p <> nil) and_then (p^ < 42)   then    { This is safe. }
            WriteLn (p^, ' is less than 42')
     end.
```

Case- Anweisung

Im Unterschied zur if- Anweisung erlaubt die case- anweisung oder Aus wahl-anweisung die Ausführung einer Anweisung, welche aus beliebig vielen Alternatinen ausgewählt wird:

```
case    Auswahlausdruck     of
            Auswahlkonstantenliste: Anweisung;….{ nicht leer}
    End
```

Bsp:

Case- Anweisung

```
CASE   a   OF
        `0`…`9`:                digit;
        `A`,`E`,`I`,`O`,`U`:    vowel;
        OTHERWISE               other;

END  {case}.
```

Module

Hauptprogramm können Ausgedehnte Pascal Programme Bauteile bekannte als Module mit einschließen. Ein Modul kann Konstanten, Arten, Variablen, Verfahren und Funktionen durch

genannte Schnittstellen exportieren, und diese Schnittstellen können durch andere Module oder durch das Hauptprogramm importiert werden.
Ein Modul hat zwei Teile: eine Überschrift und ein Block.

Die Modul überschrift enthält Aussagen und Definitionen von irgendeinen Stück, die exportiert werden sollen.

Der Block enthält die Definitionen von irgendeinen exportierten Verfahren oder Funktionen.Dort auch kann Initialisierung sein.

```
MODULE   ONE,
EXPORT   i1 = (lower, upper ) ;
CONST    lower = 0;
         Upper = 11;   {must  be  prime }

VAR      dummy:  Boolean ;

END   { of   module- heading };
END   { of   module- block }  .
```

Strukturierte Datentypen

1) **Felder (Arrays)**
   Ein Feld, auch Array genannt, besteht aus einer zu Vereinbarenden festen Anzahl von Komponenten gleichen Typs. Die einzelnen Komponenten werden durch Indizes gekennzeichnet, die als Werte von sogenannten Indexausdrücken berechenbar sind. Die Typdefinition eines Feldes muss also die Indextypen und den Komponententyp enthalten:
   array [Indextypliste] of Komponententyp
2) **Dateien (Files)**
   Ein File besteht aus einer Folge von beliebig vielen Komponenten gleichen Types. In der Typdefinition genügt also die Angabe des Komponententyps:

   file of Komponententyp

   Der Komponententyp darf ein beliebiger Typ sein, ausgenommen ein File Type bzw. ein dynamischer Feldtyp.
3) **Verbunde (records)**
   Ein Record besteht aus einer festen Anzahl von Komponenten von jeweils beliebigem Typ. Er wird in der Form:

   Record Datensatzliste end

4) **Mengen (sets)**
   Der Wertebereich eines Mengentyps besteht aus allen Teilmengen eines vorgegebenen Grundbereichs. Die Typdefinition einer Menge muss deshalb nur den Typ des Grundbereichs enthalten:

   set of Grundbereichstyp
5) **Typdeklaration (Type)**

6) **Geschützte Typen (Restricted)**
Type name= Restricted Bezugsdatentyp
Typdeklaration und Geschützte Type sind neu im Extended Pascal.

Beispiel

1) Felder (Arrays)
Array [1…10] of array [`a`…`z`] of boolean
Array [(rot, gelb, blau, schwarz)]

2) Dateien (Files)
Reset, Rewrite, Put, Get

3) Verbunde (Records)

```
record  Monat:( Jan…Dez );
          tag:     1…31
          Jahr:    integer;
END;
```

4) Mengen (Sets)
Type menge = Set of 1…3;
Vereinbarte Mengentyp enthält als werte die Teilmenge
[],[1],[2],[3],[1,2],[1,3],[2,3],[1,2,3]

Zusammenfassung

- Pascal, wenigstens in seinem Standardformat, hat den Ruf, ein Safe zu sein, aber begrenzte Sprache. Der Zweck von dieser Einführung zu den Eigenschaften von Extended Pascal soll zeigen, dass der Bereich von der Sprache großartig vermehrt worden ist ,ohne seine Sicherheit zu beeinträchtigen.
- Pascal war ursprünglich als reine Lernsprache gedacht und ist dafür sehr gut geeignet. Inzwischen in vielen Verschiedenen Erweiterungen (Turbo-Pascal, Prospero-Pascal, UCSP-Pascal) Verbreitet.
- Die Erweiterung sind meist sehr fähig, aber die dadurch entstehenden Dialekte zu unterschiedlich, um Programme übertragen zu können.

Literatur

[1] An introduction to the Extended Pascal Language
By Tony Hetherington
prospero Software Ltd ., 190 Castelnau, London SW 13 9 DH , England

[2] İntro to Pascal , Brian Brown/Peter Henry, 1988-1999)

[3] wikipedia