Übungsblock 9

Lösungen Übungsblatt 8:

Aufgabe 1:

```
program aufgabe1;
type
   sachen= (hut, hemd, schirm);
   geschenke = set of sachen;
   geschenke2 = set of sachen;
   typ1 = record
          feld: integer
   end;
   typ2  = record
          feld: integer
   end;
```

Übungsblock 9

```
var
    fueropa: geschenke;
    fueroma: geschenke2;
    str1 : typ1;
    str2 : typ2;
begin
  str1.feld:=1;
{ str2 := str1;  geht in Psacal naemlich nicht!}
  fueropa := [hut];
  fueroma := fueropa;
  if (fueroma = fueropa) then writeln('namentliche
  Typgleichheit ');
end.
```

Übungsblock 9

Aufgabe 2.1: siehe kfz.pas

```
type
  jahr = 1980..2010;
  kraftstoffe = (Gas, Diesel, Benzin, Elektro, Hybrid,
  Sonstiges);
  ziffer = '0'..'9';

  kfz = record
     marke: string;
     modell: string;
     kraftstoffart: kraftstoffe;
     erstzulassung: jahr;
     kilometer: real;
     preis: real;
     plz : array[1..5] of ziffer
  end;
```

Übungsblock 9

Aufgabe 2.2:

```
    erlmerkmale = (Servo, Radio, CD, AHK, Airbags);
    merkmale = set of erlmerkmale;
    kfz = record
            . . .
            ausstattung: merkmale;
    end;
**************************************************************
    merkmal = packed array[1..25] of char;//oder: string
    merkmale = file of merkmal; {kurz: file of string;
                                          nicht in fpc   }

    kfz = record
            . . .
            ausstattung: merkmale;

    end;
```

Übungsblock 9

Blöcke in Pascal :

- Inline-Blöcke
- Prozeduren
- Funktionen

- Anfangs- und Endmarkierung: BEGIN ….END

- Inline-Blöcke:
  - nur für logische Zusammenfassung
  - keine lokalen Variablen oder Typen

```
var
   i,j,k: integer;
begin
   i := 4; j:= 2*i;
   begin
            k := i;
            i := 10
   end;
   writeln(´Hier i, j, k: ´,i:3,j:3,k:3)
end.
```

Übungsblock 9

Funktionen in Pascal :

- berechnen immer ein Ergebnis
- Funktionsdeklarationen nach const-type-var-Vereinbarungen im Programm
- Syntax:

  FUNCTION *name* (*formale Parameterliste*): *ergebnistyp;*

  Deklarationsteil

  BEGIN

  …{mind. eine Anweisung: *name* := …}

  END;

- *Parameterliste:* (*id1* : *typ1*, *id2,id3* : *typ2, …)*
- Deklarationsteil: const-, type-, var-, Funktions- und Prozedurdeklarationen

Übungsblock 9

- Beispiel: Binomialkoeffizient mittels Fakultät

```
FUNCTION bin (n, k : integer) : integer;
    var   zaehler, nenner : integer;
    FUNCTION fak (n : integer) : integer;
            var         i, hilf : integer;
            begin
                        hilf := 1;
                        for i := 1 to n do hilf := hilf * i;
                        fak := hilf
            end;
begin
  zaehler := fak(n);
  nenner := fak(k) * fak(n-k);
  bin := zaehler DIV nenner
end.
```

Übungsblock 9

Prozeduren in Pascal:

- selbständig ausführbare Anweisung („Unterprogramm“)
- ohne Rückgabewert
- Prozedurdeklarationen nach Const-Type-Var-Vereinbarungen im Programm (bel. mischbar mit Funktionen)
- Syntax:

  PROCEDURE *name* (*formale Parameterliste*)*;*

  Deklarationsteil
  BEGIN
  …
  END;

- Deklarationsteil: const-, type-, var- , Funktions- und Prozedurdeklarationen

Übungsblock 9

- *Parameterliste:* (*id1* : *typ1*, *id2,id3* : *typ2, …)*
- Parameter können *var*iable Parameter sein
  (var id1:typ1,id2: typ2, var id3: typ3, ….)
  Wertänderung von var-Parametern im Prozedurrumpf erlaubt, wird an übergeordneten Block durchgereicht
  Prinzip: call-by-reference
- Beispiel:

```
procedure einlesen(var x,y: integer);
        begin
                writeln(´Bitte geben Sie zwei Int-Zahlen an´);
                readln(x ,y);
                writeln(´Danke´)
        end;
```

Übungsblock 9

Rekursion:

- bei Funktionen und Prozeduren erlaubt
- bei Prozeduren Auswirkung bei var-Parametern beachten!
- Beispiel:

```
program test;
var z,n:integer;
procedure p(z: integer);
      var s: integer;
      begin
                  s := 2;
                  if z <= n then
                    begin
                              z := z+s;
                              p(z)
                    end;
                  write(z,´ ´)
    end;
begin {test}
    readln(n);
    z:=2;
   p(z);
   write(z)
end.
```

1. Ausgabe für n=4?
2. Ausgabe für n=4,
   wenn z var-Parameter?

Übungsblock 9

Funktionen/Prozeduren als Parameter:

- bei manchen Implementierungen nur selbstdefinierte Funktionen oder Prozeduren als Parameter erlaubt
- aktueller Parameter beim Aufruf muss bzgl. aller Parameter (und Rückgabe) in den Typen mit formalem Parameter übereinstimmen
- Freepascal (fpc) erlaubt sie so nicht!
- Beispiel: Summenfunktion

```
FUNCTION summe (n: integer,
                          FUNCTION f(i:integer):real) :real;
begin
  if n=1
  then summe := f(1)
  else summe := f(n) + summe (n-1, f);
end;
```