1.

a. Här är ett sätt att skriva `WHILE`-satsen mha `LOOP` (det finns fler sätt).

```
LOOP
  IF villkor THEN
    satser
  ELSE
    EXIT
  END
END
```

b. Andra exempel på skalära typer är typen `BOOLEAN`, `CARDINAL`, `REAL` och uppräkningstyper. Skalära typer (utom `REAL`) kan

- användas för styrvariabeln i `FOR`-loopar,
- undersökas i `CASE`-satser,
- vara element i mängdtyper (`SET OF` ...),
- konverteras till och från typen `CARDINAL` med funktionerna `ORD` och `VAL`.

De kan jämföras med alla jämförelseoperatorerna (även `REAL`).

2.

a. Exempel på implementering av proceduren `ReadBool`:

```
FROM InOut IMPORT ReadString,Done;
FROM String IMPORT Compare,CompareResult;

PROCEDURE ReadBool(VAR b:BOOLEAN; VAR success:BOOLEAN);
VAR s:ARRAY [1..6] OF CHAR;
BEGIN
  ReadString(s);
  success:=Done;
  IF Done THEN
    IF Compare(s,"TRUE")=equal THEN b:=TRUE
    ELSIF Compare(s,"FALSE")=equal THEN b:=FALSE
    ELSE success:=FALSE
    END
  END
END ReadBool;
```

b. Antalet stjärnor som skrivs ut blir:

$$2 + 4 + 6 + \ldots + 60 = \frac{2 + 60}{2} \times 30 = 31 \times 30 = 930$$

3.

a. Exempel på implementering av proceduren `Concat`.

```
FROM String IMPORT Assign,Length;

PROCEDURE Concat(source1,source2: ARRAY OF CHAR;
                 VAR dest: ARRAY OF CHAR;
                 VAR success: BOOLEAN);
VAR i,l1,l2,l:CARDINAL;
BEGIN
  l1:=Length(source1);
  l2:=Length(source2);
  l:=l1+l2;
  success:= l<=HIGH(dest)+1;
  IF success THEN
    Assign(dest,source1,success); (* lyckas alltid *)
    FOR i:=0 TO l1-1 DO
      dest[l1+i]:=source2[i]
    END;
    IF l<=HIGH(dest) THEN
      dest[l]:=0C
    END
  END
END Concat;
```

b. Eftersom `RECORD`-typen `Point` inte kan användas som resultat-typ gör vi om proceduren `add` så att den tar en `VAR`-parameter istället.

```
PROCEDURE add(p1,p2:Point; VAR p:Point);
BEGIN
  p.x:=p1.x+p2.x;
  p.y:=p1.y+p2.y;
END add2;
.
.
.
  add2(p1,p2,p4);
  add2(p3,p4,p4);
```

4. Implementeringsdel till modulen `BokListModul`:

```
IMPLEMENTATION MODULE BokListModul;
(* En modul för länkade listor av böcker. *)

FROM Storage IMPORT ALLOCATE;
FROM BokModul IMPORT Bok;

PROCEDURE cons(b:Bok; bs:BokLista):BokLista;
(* Lägger till boken b först i listan bs. *)
VAR ny:BokLista;
BEGIN
  NEW(ny);
  ny^.head:=b;
  ny^.tail:=bs;
  RETURN ny
END cons;

PROCEDURE null(bs:BokLista):BOOLEAN;
(* ger TRUE om listan bs är tom, annars FALSE *)
BEGIN
  RETURN bs=Nil
END null;

PROCEDURE hd(bs:BokLista):Bok;
(* hd ger den första boken i listan bs.
   bs får inte vara tom *)
BEGIN
  RETURN bs^.head
END hd;

END BokListModul.
```

5. Proceduren `allaBockerAv`:

```
FROM BokModul IMPORT Bok,Namn;
FROM BokListModul IMPORT BokLista,cons,Nil;
FROM String IMPORT Compare,CompareResult;

PROCEDURE allaBockerAv(forf:Namn; r:BokRegister):BokLista;
VAR i:[1..max]; bs:BokLista;
BEGIN
  bs:=Nil;
  FOR i:=1 TO r.antal DO
    IF Compare(r.bok[i].forfattare,forf)=equal THEN
      bs:=cons(r.bok[i],bs)
    END
  END;
  RETURN bs
END allaBockerAv;
```


6. Definitionsdel och implementeringsdel till modulen `FunktionsModul`.

```
DEFINITION MODULE FunktionsModul;

CONST N=10;

TYPE Funktion = RECORD A,B:REAL; k:ARRAY [0..N] OF REAL END;

PROCEDURE eval(f:Funktion; x:REAL):REAL;

PROCEDURE addera(f1,f2:Funktion):Funktion;

PROCEDURE derivera(f:Funktion):Funktion;

END FunktionsModul.
```

Implementeringsdelen:

```
IMPLEMENTATION MODULE FunktionsModul;
FROM MathLib IMPORT Sin,Cos;

PROCEDURE eval(f:Funktion; x:REAL):REAL;
VAR val:REAL; i:[0..N];
BEGIN
  val:=0.0;
  FOR i:=N TO 0 BY -1 DO
    val:=val*x+f.k[i]
  END;
  val:=val+f.A*Sin(x)+f.B*Cos(x);
  RETURN val
END eval;

PROCEDURE addera(f1,f2:Funktion):Funktion;
BEGIN
 (* ... *)
END addera;

PROCEDURE derivera(f:Funktion):Funktion;
BEGIN
 (* ... *)
END derivera;

END FunktionsModul.
```