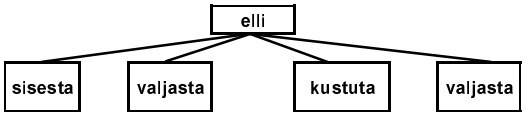
3. Näide (protseduuride kasutamise peale)
Ülesanne
Ellimineeri reaalarvulises ruutmaatriksis peadiagonaal, nihutades kõiki temast paremal olevaid elemente 1 (indeksi) võrra vasakule

i \ j	1	2	3	4
1	1,0	2,0	-1,0	-2,5
2	3,5	0,0	2,4	4,3
3	3,7	-3,0	2,1	5,8
4	6,8	7,5	-4,0	-1,0

i \ j	1	2	3	4
1	2,0	-1,0	-2,5	
2	3,5	2,4	4,3	
3	3,7	-3,0	5,8	
4	6,8	7,5	-4,0	



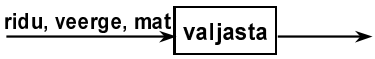
sisesta - sisestab RAM-i maatriksi

maatriks - $mat_{i\,j}$ programmis mat[i,j]

- identifikaator (asukoht), ridade ja veergude arv

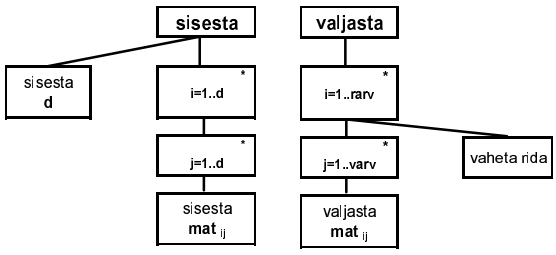
sisesta (var d: i10; var mat: maatriks);

valjasta - väljastab (ekraanile) RAM-ist maatriksi



valjasta (ridu, veerge: i10; mat: maatriks);

kustuta(d: i10; var mat: maatriks);



Märkus:
Pascal´is on lubatud muutujaid kirjeldada 2-l moel:
• kasutades tüübikirjeldust

```
type maatriks = array [1..10, 1..10] of real;
var mat: maatriks;
```

• ühendades muutujakirjelduse tüübikirjeldusega

```
var mat: array [1..10, 1..10] of real;
```

Reegel: Parameetrite kirjeldamisel peab kasutama tüübikirjeldust

```
type      maatriks = array [1..10, 1..10] of real;
          i10 = 0..10;

procedure sisesta(var d: i10; var mat: maatriks);
      { Protseduur sisestab reaalarvulise ruutmaatriksi järgu d ja
        maatriksi mat ridade kaupa }

var   i,j: i10;
begin
  writeln('Sisesta maatriksi järk 1..10');
  read(d);
  for i:=1 to d do
    begin
      writeln('Sisesta ',i:1,'-e rea ',d:1,' elementi');
      for j:=1 to d do read(mat[i,j])
    end
end;
```

programmi ***elli*** tekst on ´catpascal´-is **elli.p**
seal on ka **matprot.p** - 1. maatriksi ülesanne protseduuridega

4. Funktsioonid

4.1. Üldist
Pascal´is saab kasutada funktsioone (f)
Standardf-d. SQRT(x), SIN(x), …
nn. standard-Pascal´is oli neid 17, turbo-des palju rohkem
kasutajal on võimalik neid ise teha

4.2. Kirjeldamine

function *nimi* (formaalsete parameetrite loetelu) : tüüp;
kirjeldused;
begin
lause1;
lause2;
.
.
***nimi* := ...**
end;

f tüüp

NB! Peab olema vähemalt 1 lause, mis omistab väärtuse f-i identifikaatorile (nimele)

4.3. Näide
Koosta f nimega ***max***, mis leiab 2-st reaalarvust suurema

```
function max (x,y: real): real;
    begin
        if  x > y
            then max := x
            else  max := y
    end;
```

suur := max(pikkus, laius)
suur := 2*max(r, 1.5)
suurim := max(a, max(b,c))

rekursioon - pöördumine enda poole

4.4. Võrdlus protseduuridega
• f töö tulemus - vähemalt 1 väärtus, mis omistatakse f nimele
prots. töö tulemus - 0 või rohkem väärtust
• pöördumine f poole - kordaja erikuju
pöördumine prots. poole - lause erikuju
• kirjeldamine f - algab ***function*** - ga, funktsiooni tüüp
prots. - algab ***procedure*** - ga

4.5. Kõrvaleffekt

```
function fo (x: integer): real;
    begin
        v := v*x;
        f  := sqrt(x) + 1
    end;
```

peaprogrammis lause z := fo(x1) + v
mitteaditiivne
Ära omista funktsioonis globaalsetele muutujatele väärtusi!
Ära kasuta funktsioonis globaalseid muutujaid!!
Ära kasuta protseduurides ja funktsioonides globaalseid muutujaid!!

5. Rekursioon

Ülesande rekursiivne lahendus:
• ülesande lahendamine asendatakse uue ülesande lahendamisega, mis on eelmisega sarnane, kuid lihtsam
• selline asendus toimub seni, kuni ülesande lahendus muutub triviaalseks

Näited

• x^n = 1 kui n = 0
= $x*x^{n-1}$ kui n > 0

```
function aste(x:real; n:integer):real;
   begin
      if  n = 0
          then aste := 1.0
          else  aste := x*aste(x, n-1)
   end;
```

• n! = 1 kui n = 0
= n*(n-1)! kui n > 0

```
function factor(n:integer): integer;
   begin
      if  n = 0
          then factor := 1
          else factor := n* factor(n-1)
   end;
```

• A(m,n)=n+1 kui m = 0
= A(m-1,1) kui n = 0
= A(m-1, A(m,n-1)) kui m,n <> 0