8. Eljárások és függvények

8.1 Egyszerű paraméterek

Írjunk programot, amely beolvas három valós számot, kiszámítja az összegüket és átlagukat! Használjunk a feladat megoldására alprogramokat! (*szumma*)

```
program szumma;
var
    a1,a2,a3,atl,sum: real;
    procedure olvas(var x1,x2,x3:real);
    begin
      write('1. szám: '); readln(x1);
      write('2. szám: '); readln(x2);
      write('3. szám: '); readln(x3);
    end;
    procedure szamol(x1,x2,x3:real; var atlag,ossz:real);
    begin
      ossz :=x1+x2+x3;
      atlag:=ossz/3;
    end;
    function osszeg(x1,x2,x3:real):real;
    begin
      osszeg:=x1+x2+x3;
    end;
    function fatlag(x1,x2,x3:real):real;
    begin
      fatlag:=osszeg(x1,x2,x3)/3;
    end;
begin
    olvas(a1,a2,a3);               { olvas eljárás aktiválása }
    szamol(a1,a2,a3,atl,sum);      { szamol eljárás aktiválása }
    writeln;writeln('eljárással'); { eredmények kiírása }
    writeln('összeg: ',sum:6:2);
    writeln('átlag : ',atl:6:2);
    writeln;writeln('függvénnyel');{ függvények hívása  }
    sum:=osszeg(a1,a2,a3);
    atl:=fatlag(a1,a2,a3);
    writeln('összeg: ',sum:6:2);   { eredmények kiírása }
    writeln('átlag : ',atl:6:2);
    readln;
end.
```

A program futásainak eredménye:

```
1. szám: 1
2. szám: 2
3. szám: 3
```

```
eljárással
összeg:   6.00
átlag :   2.00

függvénnyel
összeg:   6.00
átlag :   2.00

1. szám: 1.5
2. szám: 2.3
3. szám: 3.2

eljárással
összeg:   7.00
átlag :   2.33

függvénnyel
összeg:   7.00
átlag :   2.33
```

Írjunk programot, amely beolvassa a körgyűrű külső és belső sugarát! Vizsgálja meg, hogy a külső sugár nagyobb legyen a belső sugárnál! Számítsa ki a körgyűrű területét és a kerületét! A feladat megoldásához használjunk alprogramokat! (*korgyuru*)

```
program korgyuru;
var
  Rkulso,Rbelso, ter, ker: real;
procedure Olvas(var Rk, Rb: real);
begin
  write('Belso sugar: '); readln(Rb);
  repeat
    write('Kulso sugar: '); readln(Rk);
  until (Rk > Rb);
end;
procedure KorgyuruTerulet(Rk, Rb: real; var t:real);
begin
  t := (sqr(Rk)-sqr(Rb))*pi;
end;

procedure KorgyuruKerulet(Rk, Rb: real; var k:real);
begin
  k := 2*pi*(Rk+Rb);
end;
begin
  Olvas(Rkulso,Rbelso);
  KorgyuruTerulet(Rkulso, Rbelso,ter);
  writeln('Körgyűrű területe: ',ter:9:3);
  KorgyuruKerulet(Rkulso, Rbelso,ker);
  writeln('Körgyűrű kerülete: ',ker:9:3);
  readln;
end.
```

A program futásának eredménye:

```
Belso sugar: 1
Kulso sugar: 2
Körgyűrű területe:     9.425
Körgyűrű kerülete:    18.850
```

Írjunk programot, amely beolvassa a kocka oldalélét és kiszámítja a testátlóját, lapátlóját, térfogatát, felszínét! A feladatot alprogramokkal oldjuk meg! (*kocka*)

```
program kocka;
var
  oldal, terfogat, felszin, testatlo, lapatlo : real;
procedure olvas(var a: real);
begin
  write('A kocka oldala: '); readln(a);
end;
function kocka_testatlo(a:real): real;
begin
  kocka_testatlo:= a*sqrt(3);
end;
function kocka_lapatlo(a:real): real;
begin
  kocka_lapatlo:= a*sqrt(2);
end;
procedure Szamol(a:real; var terf, felsz: real);
begin
  terf:= a*sqr(a);
  felsz:= 6*sqr(a);
end;
begin
  olvas(oldal);
  testatlo:= kocka_testatlo(oldal);
  lapatlo:= kocka_lapatlo(oldal);
  szamol(oldal, terfogat, felszin);
  writeln('A kocka testátlója: ', testatlo:6:2);
  writeln('A kocka lapátlója : ', lapatlo:6:2);
  writeln('A kocka térfogata : ', terfogat:6:2);
  writeln('A kocka felszíne  : ', felszin:6:2);
  readln;
end.
```

A program futásának eredménye:

```
A kocka oldala: 1
A kocka testátlója:   1.73
A kocka lapátlója :   1.41
A kocka térfogata :   1.00
A kocka felszíne  :   6.00
```

Írjunk programot, amely beolvassa a téglatest három oldalát és kiszámítja a térfogatát és a felszínét kétféle módon! A feladatot alprogramokkal oldjuk meg! (*teglatest*)

```
program teglatest;
var
   a_old, b_old, c_old, f1,f2,t1,t2: real;
procedure TeglatestOlvas(var a,b,c:real);
begin
   writeln('A téglatest oldalai ');
   write('A oldal : '); readln(a);
   write('B oldal : '); readln(b);
   write('C oldal : '); readln(c);
end;
procedure Felszin1(a,b,c: real; var felszin:real);
begin
  felszin:= 2*(a*b + c*b + a*c);
end;
function Felszin2(a,b,c: real):real;
begin
  Felszin2:= 2*(a*b + c*b + a*c);
end;
procedure Terfogat1(a,b,c: real; var terfogat:real);
begin
  terfogat:= a*b*c;
end;
function Terfogat2(a,b,c: real):real;
begin
  Terfogat2:= a*b*c;
end;
begin
  TeglatestOlvas(a_old, b_old, c_old);
  Felszin1(a_old, b_old, c_old, f1);
  f2:= Felszin2(a_old, b_old, c_old);
  writeln('Felszíne 1 : ',f1:8:2);
  writeln('Felszíne 2 : ',f2:8:2);
  Terfogat1(a_old, b_old, c_old, t1);
  t2:=Terfogat2(a_old, b_old, c_old);
  writeln('Térfogata 1: ',t1:8:2);
  writeln('Térfogata 2: ',t2:8:2);
  readln;
end.
```

A program futásának eredménye:

```
A téglatest oldalai
A oldal : 1
B oldal : 2
C oldal : 3
Felszíne 1 :    22.00
Felszíne 2 :    22.00
Térfogata 1:     6.00
Térfogata 2:     6.00
```

Írjunk programot, amely beolvassa a gömb sugarát és kiszámítja a gömb felszínét és térfogatát! A feladatot alprogramokkal oldjuk meg! (*gomb*)

```
program gomb;
var
  sugar, g_terf, g_felsz : real;
procedure olvas(var r:real);
begin
  write('A gömb sugara: '); readln(r);
end;
procedure térfogat(r:real; var terf: real);
begin
  terf:= 4*sqr(r)*r*pi/3;
end;
procedure felszin(r:real; var felsz: real);
begin
  felsz:=4*sqr(r)*pi;
end;
begin
  olvas(sugar);
  terfogat(sugar, g_terf);
  felszin(sugar, g_felsz);
  writeln('A gömb térfogata: ', g_terf:6:2);
  writeln('A gömb felszíne : ', g_felsz:6:2);
  readln;
end.
```

A program futásának eredménye:

```
A gömb sugara: 1
A gömb térfogata:   4.19
A gömb felszíne :  12.57
```

Írjunk programot, amely beolvassa a gömb sugarát és kiszámítja kiválaszthatóan a gömb felszínét vagy térfogatát! A feladatot alprogramokkal oldjuk meg! (*gomb2*)

```
program gomb2;
var
  sugar, g_terf, g_felsz : real;
procedure olvas(var r:real);
begin
  write('A gömb sugara: '); readln(r);
end;
procedure szamitasok(r:real; jel: integer; var eredmeny: real);
begin
  case jel of
   1: eredmeny:= 4*sqr(r)*r*pi/3;
   2: eredmeny:= 4*sqr(r)*pi;
  end;
end;
```

```
begin
  olvas(sugar);
  szamitasok(sugar, 1, g_terf);
  szamitasok(sugar, 2, g_felsz);
  writeln('A gömb térfogata: ', g_terf:6:2);
  writeln('A gömb felszíne : ', g_felsz:6:2);
  readln;
end.
```

A program futásának eredménye:

```
A gömb sugara: 1
A gömb térfogata:   4.19
A gömb felszíne :  12.57
```

Írjunk programot másodfokú egyenlet kiszámítására! A feladat megoldására használjunk alprogramokat! (*masod_f2*)

```
program masod_f2;
var
    a1,b1,c1,gy1,gy2: real;
    j: integer;

procedure olvas(var a,b,c :real);
begin
   writeln('Másodfokú egyenlet megoldása');
   writeln('Ax^2+Bx+C=0 egyenlet együtthatóinak megadása');
   write('A: '); readln(a);
   write('B: '); readln(b);
   write('C: '); readln(c);
end;
procedure eredmeny_kiir(x1,x2:real; jelez:integer);
begin
   case jelez of
   1: writeLn('Az egyenletnek bármelyik szám megoldása');
   2: writeLn('Ellentmondás, nincs megoldás');
   3: writeLn('Az egyenlet elsőfoku, x=',x1:8:4);
   4: begin
         writeln('Az egyenletnek komplex megoldása van');
         writeln('X1=',x1:8:4,'+',x2:8:4,'i');
         writeln('X2=',x1:8:4,'-',x2:8:4,'i');
      end;

   5: begin
         write('Az egyenletnek csak egy');
         writeln('(két egybeeső) megoldása van');
         writeln('X=',x1:8:4);
      end;
   6: begin
         writeLn('Az egyenlet megoldásai');
         writeLn('X1=',x1:8:4);
```

```
          writeLn('X2=',x2:8:4);
      end;
  end;
end;
procedure masod(a,b,c: real; var x1,x2:real;
                var jelez: integer);
var
   d  : real;
  sign: integer;
begin
  if a=0 then
         begin
           if b=0 then
                  begin
                     if c=0 then jelez:=1
                            else jelez:=2;

                  end
                  else begin
                          x1:=-c/b;
                          jelez:=3;
                       end;
         end
         else
         begin
            d:=sqr(b)-4*a*c;
            if d=0 then sign:=0 else sign:=round(d/abs(d));
            case sign of
                -1 : begin
                       x1:=-b/2/a;
                       x2:=sqrt(abs(d))/2/a;
                       jelez:=4;
                     end;
                 0 : begin
                       x1:=-b/2/a;
                       jelez:=5;
                     end;
                 1 : begin
                       x1:=(-b+sqrt(d))/2/a;
                       x2:=(-b-sqrt(d))/2/a;
                       jelez:=6;
                     end;
            end;
         end;
  end;
begin
   olvas(a1,b1,c1);
   masod(a1,b1,c1,gy1,gy2,j);
   eredmeny_kiir(gy1,gy2,j);
   readln;
end.
```

A program futásának eredménye:

```
Másodfokú egyenlet megoldása
Ax^2+Bx+C=0 egyenlet együtthatóinak megadása
A: 2
B: 3
C: 1
Az egyenlet megoldásai
X1= -0.5000
X2= -1.0000
```

Írjunk programot másodfokú egyenlet kiszámítására! A feladat megoldására használjunk alprogramokat! (*masod_fg*)

```
program masod_fg;
var a1,b1,c1,gy1,gy2 : real;

procedure olvas(var a,b,c :real);
begin
   writeln('Másodfokú egyenlet megoldása');
   writeln('Ax^2+Bx+C=0 egyenlet együtthatóinak megadása');
   write('A: '); readln(a);
   write('B: '); readln(b);
   write('C: '); readln(c);
end;

function diszkriminans(a,b,c:real):real;
begin
   diszkriminans:=sqr(b)-4*a*c;
end;
function sign(d:real):integer;
begin
  if d=0 then sign:=0 else sign:=round(d/abs(d));
end;
procedure masod(a,b,c: real; var x1,x2:real);
var
  d      : real;
  elojel : integer;

begin
 if a=0 then
      begin
         if b=0 then
           begin
             if c=0
               then
                 writeln('Az egyenletnek bármelyik szám megoldása')
               else
                 writeln('Ellentmondás, nincs megoldás');
           end
           else
```

```
            begin
               x1:=-c/b;
               writeln('Az egyenlet elsőfokú, x=',x1:8:4);
            end;
      end
      else
      begin
         d:=diszkriminans(a,b,c);
         elojel:=sign(d);
         case elojel of
             -1 : begin
                   x1:=-b/2/a;
                   x2:=sqrt(abs(d))/2/a;
                   writeln('Az egyenletnek komplex megoldása van');
                   writeln('X1=',x1:8:4,'+',x2:8:4,'i');
                   writeln('X2=',x1:8:4,'-',x2:8:4,'i');
                  end;
              0 : begin
                   x1:=-b/2/a;
                   write('Az egyenletnek csak egy');
                   writeln('(két egybeeső) megoldása van');
                   writeln('X=',x1:8:4);
                  end;
              1 : begin
                   x1:=(-b+sqrt(d))/2/a;
                   x2:=(-b-sqrt(d))/2/a;
                   writeLn('Az egyenlet megoldásai');
                   writeLn('X1=',x1:8:4);
                   writeLn('X2=',x2:8:4);
                  end;
         end;
       end;
end;
begin
   olvas(a1,b1,c1);
   masod(a1,b1,c1,gy1,gy2);
   readln;
end.
```

A program futásainak eredménye:

```
Másodfokú egyenlet megoldása
Ax^2+Bx+C=0 egyenlet együtthatóinak megadása
A: 2
B: 3
C: 1
Az egyenlet megoldásai
X1= -0.5000
X2= -1.0000
```

```
Másodfokú egyenlet megoldása
Ax^2+Bx+C=0 egyenlet együtthatóinak megadása
A: 1
B: 1
C: 1
Az egyenletnek komplex megoldása van
X1= -0.5000+  0.8660i
X2= -0.5000-  0.8660i

Másodfokú egyenlet megoldása
Ax^2+Bx+C=0 egyenlet együtthatóinak megadása
A: 0
B: 2
C: 1
Az egyenlet elsőfoku, x= -0.5000

Másodfokú egyenlet megoldása
Ax^2+Bx+C=0 egyenlet együtthatóinak megadása
A: 0
B: 0
C: 2
Ellentmondás, nincs megoldás
```

Írjunk programot, amely két hatjegyű oktális számot ad össze! A feladat megoldásához használjunk alprogramokat! (*oktalis1*)

```
program oktalis1;
type
     num = string[6];
     t   = array[1..6] of 0..7;
var
     i        : integer;
     okt1,okt2: num;
     osszeg   : t;

function oct_add(sz1,sz2: num; var eredm: t): integer;
var
  carry: 0..1;
  ered : integer;
begin
  carry:=0;
  for i:= 6 downto 1 do
  begin
    ered:= carry+ord(sz1[i])-ord('0')+ord(sz2[i])-ord('0');
    if ered > 7 then carry:=1 else carry:=0;
    eredm[i]:=ered and 7;
  end;
  oct_add:=carry;
end;
```

```
begin
  okt1:='123456';
  okt2:='234567';
  if oct_add(okt1,okt2,osszeg) = 1 then writeln('túlcsordult')
  else
  begin
    write(okt1,' + ',okt2,' = ');
    for i:=1 to 6 do
      write(osszeg[i]);
    writeln;
  end;
  readln;
end.
```

A program futásának eredménye:

```
123456 + 234567 = 360245
```

Írjunk programot, amely beolvas két hatjegyű oktális számot és összeadja! A feladat megoldásához használjunk alprogramokat! (*oktalis2*)

```
program oktalis2;
type
     num = string[10];
var
     i        : integer;
     okt1,okt2,okt3: num;

function oct_add2(sz1,sz2: num; var sz3: num): integer;
var
  carry: 0..1;
  ered : integer;
begin
  carry:=0;
  for i:= 6 downto 1 do
  begin
    ered:= carry+ord(sz1[i])-ord('0')+ord(sz2[i])-ord('0');
    if ered > 7 then carry:=1 else carry:=0;
    sz3[i]:=char(ered and 7 +ord('0'));
  end;
  sz3[0]:=chr(6);
  oct_add2:=carry;
end;

function oct_read:num;
var
   hiba,i,h:integer;
   okt     :num;
begin
   repeat
    hiba:=0;
```

```
    write('Oktális szám [6 jegyű: '); readln(okt);
    h:=length(okt);
    if h = 6 then
    begin
     for i:=1 to h  do
     begin
       if ord(okt[i])-ord('0') > 7 then
        begin
          writeln(i,'. digit: ',okt[i],' hibas!');
          hiba:=hiba+1;
       end;
     end;
    end
    else begin writeln('A szám 6 jegyű lehet!'); hiba:=1; end;
  until hiba = 0;
  oct_read:=okt;
end;

begin
  okt1:=oct_read;
  okt2:=oct_read;
  if oct_add2(okt1,okt2,okt3) = 1 then writeln('túlcsordult')
  else
    writeln(okt1,' + ', okt2 ,' = ',okt3 );
  readln;
end.
```

A program futásainak eredménye:

```
Oktális szám [6 jegyű]: 123456
Oktális szám [6 jegyű]: 654321
123456 + 654321 = 777777

Oktális szám [6 jegyű: 321456
Oktális szám [6 jegyű: 123456
321456 + 123456 = 445134
```

Írjunk programot, amely választhatóan beolvassa a kör vagy a gömb sugarát és kiszámítja választhatóan a kör kerületét vagy a gömb felszínét, és a kör területét ill. a gömb térfogatát! A feladat megoldásához használjunk alprogramokat! (*korgomb*)

```
program korgomb;
var
  r1, r2, ter1, ker1, felsz2, terf2: real;
procedure Olvas (valaszt: integer; var r: real);
begin
  case valaszt of
   1: begin  write('Kör sugara : '); readln(r); end;
   2: begin write('Gömb sugara: '); readln(r); end;
  end;
end;
```

```
procedure TeruletTerfogat(r: real; valaszt: integer; var t:real);
begin
  case valaszt of
   1:  t := sqr(r)*pi;
   2:  t := 4*sqr(r)*r*pi/3;
 end;
end;
procedure KeruletFelszin(r: real; valaszt:integer; var k:real);
begin
  case valaszt of
   1: k := 2*r*pi;
   2: k := 4*sqr(r)*pi;
  end;
end;
begin
  Olvas(1,r1);
  TeruletTerfogat(r1,1,ter1);
  KeruletFelszin(r1,1,ker1);
  writeln('Kör kerülete : ',ker1:9:3);
  writeln('Kör területe : ',ter1:9:3);
  writeln;
  Olvas(2,r2);
  TeruletTerfogat(r2,2,terf2);
  Keruletfelszin(r2,2,felsz2);

  writeln('Gömb felszine : ',felsz2:9:3);
  writeln('Gömb térfogata: ',terf2:9:3);
  readln;
end.
```

A program futásainak eredménye:

```
Kör sugara : 1
Kör kerülete :     6.283
Kör területe :     3.142

Gömb sugara: 1
Gömb felszine :    12.566
Gömb térfogata:     4.189
```

8.2 Rekord paraméterek

Írjunk programot, amely beolvassa a gömb sugarát és kiszámítja a gömb felszínét és térfogatát! A feladatot rekord használatával és alprogramokkal oldjuk meg! (*rekpar1*)

```
program rekpar1;
type gomb = record
              sugar: real;
              terf, felsz: real;
            end;
var
  g : gomb;
  procedure Olvas(var gr : gomb);
  begin
    write('A gömb sugara: '); readln(gr.sugar);
  end;
  procedure Szamol(var gr: gomb);
  begin
    gr.terf:= 4 * sqr(gr.sugar) * gr.sugar * pi/3;
    gr.felsz:= 4 * sqr(gr.sugar) * pi;
  end;
  procedure Kiir(gr: gomb);
  begin
     writeln('A gömb felszíne : ',gr.felsz:6:3);
     writeln('A gömb térfogata: ',gr.terf:6:3);
  end;
  begin
    Olvas(g);
    Szamol(g);
    Kiir(g);
    readln;
  end.
```

A program futásának eredménye:

```
A gömb sugara: 1
A gömb felszíne : 12.566
A gömb térfogata:  4.189
```

Írjunk programot, amely beolvassa a henger sugarát, és magasságát, kiszámítja a henger felszínét és térfogatát! A feladatot rekord használatával és alprogramokkal oldjuk meg! (*rekpar2*)

```
program rekpar2;
type  henger = record
                 sugar, magassag: real;
                 terf, felsz: real;
               end;
var
  h : henger;
```

```
  procedure Olvas(var hg : henger);
  begin
    writeln('A henger adatai');
    write('Sugara   : '); readln(hg.sugar);
    write('Magassága: '); readln(hg.magassag);
  end;
  procedure Szamol(var hg: henger);
  begin
    hg.terf:= sqr(hg.sugar) * pi * hg.magassag;
    hg.felsz:= 2 * hg.sugar * pi * hg.magassag + 2 * sqr(hg.sugar)*pi;
  end;
  procedure Kiir(hg: henger);
  begin
     writeln('A henger felszíne : ',hg.felsz:6:3);
     writeln('A henger térfogata: ',hg.terf:6:3);
  end;
  begin
    Olvas(h);
    Szamol(h);
    Kiir(h);
    readln;
  end.
```

A program futásának eredménye:

```
A henger adatai
Sugara   : 1
Magassága: 1
A henger felszíne : 12.566
A henger térfogata:  3.142
```

Írjunk programot, amely beolvassa a körgyűrű belső és külső sugarát, kiszámítja a körgyűrű kerületét és területét! A feladatot rekord használatával és alprogramokkal oldjuk meg! (*rekpar3*)

```
program rekpar3;
type
     korgyuru = record
                  Rkulso,
                  Rbelso: real;
             end;
var
  k1: korgyuru;
  ter1, ker1: real;

procedure Olvas(var k: korgyuru);
begin
   write('Belso sugar: '); readln(k.Rbelso);
   repeat
    write('Kulso sugar: '); readln(k.Rkulso);
  until (k.Rkulso > k.Rbelso);
end;
```

```
procedure KorgyuruTerulet(k:korgyuru; var ter:real);
begin
  ter := (sqr(k.Rkulso)-sqr(k.Rbelso))*pi;
end;

procedure KorgyuruKerulet(k:korgyuru; var ker:real);
begin
  ker := 2*pi*(k.Rkulso+k.Rbelso);
end;
begin
  Olvas(k1);
  KorgyuruTerulet(k1,ter1);
  writeln('Körgyűrű területe: ',ter1:9:3);
  KorgyuruKerulet(k1,ker1);
  writeln('Körgyűrű kerülete: ',ker1:9:3);
  readln;
end.
```

A program futásának eredménye:

```
Külső sugár: 2
Belső sugár: 1
Körgyűrű területe:     9.425
Körgyűrű kerülete:    18.850
```

Írjunk programot, amely két komplex számot összead, kivon, szoroz és oszt, valamint kiírja a művelet eredményét! A feladat megoldásához használjunk alprogramokat! (*kompln*)

```
program kompln;
 type
      komplex = record
         re,im: real;
      end;

  procedure osszead(x,y:komplex; var z:komplex);
  begin
     z.re :=  x.re + y.re;
     z.im :=  x.im + y.im;
  end;
  procedure kivon(x,y:komplex; var z:komplex);
  begin
     z.re :=  x.re - y.re;
     z.im :=  x.im - y.im;
  end;
  procedure szoroz(x,y:komplex; var z:komplex);
  begin
     z.re := (x.re * y.re) +(- x.im * y.im);
     z.im := (x.re * y.im) + (x.im * y.re);
  end;
```

```
  procedure osztas(x,y:komplex; var z:komplex);
  begin
     z.re:=((x.re*y.re)+(x.im*y.im))/
              ((y.re*y.re)+(y.im*y.im));
     z.im:=((x.im*y.re)+(-x.re*y.im))/
              ((y.re*y.re)+(y.im*y.im));
  end;
  procedure kiir(x,y:komplex;muvjel: char;z:komplex);
  begin
    writeln('(',x.re:6:1,'+',x.im:6:1,'i) ',muvjel,
          '(',y.re:6:1,'+',y.im:6:1,'i) = ',
          '(',z.re:6:1,'+',z.im:6:1,'i)');
  end;

var
   a,b,c1,c2,c3,c4: komplex;
begin
  writeln;
  write('1. komplex szám valós része   : ');readln(a.re);
  write('               képzetes része: ');readln(a.im);
  writeln;
  write('2. komplex szám valós része   : ');readln(b.re);
  write('               képzetes része: ');readln(b.im);

  writeln;
  writeln('Műveletek');
  writeln('Az összeadás eredménye: ');
  osszead(a,b,c1);
  kiir(a,b,'+',c1);
  writeln('A kivonás eredménye: ');
  kivon(a,b,c2);
  kiir(a,b,'-',c2);
  writeln('A szorzás eredménye: ');
  Szoroz(a,b,c3);
  kiir(a,b,'*',c3);
  writeln('Az osztás eredménye: ');
  osztas(a,b,c4);
  kiir(a,b,'/',c4);
  readln;
end.
```

A program futásának eredménye:

```
1. komplex szám valós része   : 1
               képzetes része: 2

2. komplex szám valós része   : 3
               képzetes része: 4

Műveletek
Az összeadás eredménye:
(   1.0+   2.0i) +(   3.0+   4.0i) = (   4.0+   6.0i)
A kivonás eredménye:
(   1.0+   2.0i) -(   3.0+   4.0i) = (  -2.0+  -2.0i)
```

```
A szorzás eredménye:
(   1.0+   2.0i) *(   3.0+   4.0i) = (  -5.0+  10.0i)
Az osztás eredménye:
(   1.0+   2.0i) /(   3.0+   4.0i) = (   0.4+   0.1i)
```

Írjunk programot, amely beolvas két koordinátapontot és kiszámítja a két pont távolságát! A feladat megoldásához használjunk alprogramokat! (*koord*)

```
program koord;
  type
    KoordPont=record
      x:integer;
      y:integer;
    end;
    adat=array[1..2]of KoordPont;
  var
    w,a1:adat;
    tav:real;

  procedure olvas(var a:adat);
    var i:integer;
    begin
      writeln('Adja meg a koordiátapontokat: ');
        for i:=1 to 2 do
        begin
          writeln;
          write(i:2 ,' elem x koordinátája: ');readln(a[i].x);
          write(i:2 ,' elem y koordinátája: ');readln(a[i].y);
        end;
    end;

  function Tavolsag(a:adat):real;
    begin
      Tavolsag:=sqrt(sqr(a[1].x-a[2].x)+sqr(a[1].y-a[2].y));
    end;

begin
  olvas(a1);
  tav:=Tavolsag(a1);
  writeln;
  writeln('A két pont távolsága: ',tav:5:2);
  readln;
end.
```

A program futásának eredménye:

```
 1 elem x koordinátája: 1
 1 elem y koordinátája: 1

 2 elem x koordinátája: 2
 2 elem y koordinátája: 2

A két pont távolsága:  1.41
```

Írjunk programot, amely beolvas egy mondatot (max 255 karakter). A beolvasott mondatot bontsuk szavakra és készítsünk statisztikát táblázatosan kiírva a szó hossza, magánhangzók száma, mássalhangzók száma! A feladat megoldására használjunk alprogramokat! (*mondat_r*)

```
program mondat_r;
type
       szotip = record
                  szo:string;
                  db:integer;
                  mgh_db:integer;
                  msh_db:integer;
                end;
       vizsgal = record
                  mondat: string;
                  szavak: array[1..50] of szotip;
                  szo_db: integer;
                end;

      procedure Olvas(var m:vizsgal);
      begin
          writeln('Mondat: '); readln(m.mondat);
      end;
      procedure Szo_bont(var m:vizsgal);
      var
          i,j,k:integer;
      begin
           j:=1; k:=0;
           for i:=1 to length(m.mondat) do
           begin
             if (m.mondat[i] <> ' ') and (m.mondat[i] <> '.')
                                  then
                                    begin
                                     k:=k+1;
                                     m.szavak[j].szo[k]:=m.mondat[i];
                                    end;
             if(m.mondat[i] = ' ') or (m.mondat[i] = '.') then
                                     begin
                                      m.szavak[j].szo[0]:=chr(k);
                                      k:=0; j:=j+1;
                                     end;
          end;
          m.szo_db:=j-1;
      end;
      procedure Kiir(m:vizsgal);
      var i:integer;
      begin
          for i:=1 to m.szo_db do
            writeln(m.szavak[i].szo);
      end;
```

```
      procedure Statisztika(var m:vizsgal);
      var
           i,j,ma,ms     :integer;
           abc,mgh,msh   :set of 'a'..'z';
           Nabc,Nmgh,Nmsh: set of 'A'..'Z';
      begin
           abc:=['a'..'z'];
           Nabc:=['A'..'Z'];
           mgh:=['a','e','u','i','o'];
           Nmgh:=['A','E','U','I','O'];
           Nmsh:=Nabc-Nmgh;
           msh:=abc-mgh;
           for i:=1 to m.szo_db do
           begin
              m.szavak[i].db:= length(m.szavak[i].szo);
              ma:=0; ms:=0;
              for j:=1 to m.szavak[i].db do
              begin
                if  m.szavak[i].szo[j] in mgh then ma:=ma+1;
                if  m.szavak[i].szo[j] in msh then ms:=ms+1;
                if  m.szavak[i].szo[j] in Nmgh then ma:=ma+1;
                if  m.szavak[i].szo[j] in Nmsh then ms:=ms+1;
              end;
              m.szavak[i].mgh_db:=ma;
              m.szavak[i].msh_db:=ms;
           end;
        end;
        procedure Mindent_kiir(m:vizsgal);
        var i:integer;
        begin
           writeln('szavak':10,'szó hossza':15,
                   'mgh száma':12,'msg száma':15);
           for i:=1 to m.szo_db do
            writeln(m.szavak[i].szo:10,m.szavak[i].db:10,
                    m.szavak[i].mgh_db:14,m.szavak[i].msh_db:15);
      end;
var
  v: vizsgal;
begin
   Olvas(v);
   Szo_bont(v);
   Kiir(v);
   Statisztika(v);
   Mindent_kiir(v);
   readln;
end.
```

A program futásának eredménye:

```
Mondat:
Ma szep az ido.
Ma
szep
az
ido
```

```
    szavak      szó hossza   mgh száma      msg száma
        Ma          2             1             1
      szep          4             1             3
        az          2             1             1
       ido          3             2             1
```

Írjunk programot, amely beolvas egy mondatot és megállapítja, hogy kérdő, felkiáltó vagy kijelentő! Keressük meg a mondat legrövidebb és a leghosszabb szavát! A feladat megoldására használjunk alprogramokat! (*mon_tip*)

```
program mon_tip;
type
   tszo=record
        mondat:string;
        legrov:string;
        rdb   :integer;
        leghos:string;
        hdb   :integer;
        tipus :char;
        end;
var
   szoveg: tszo;

procedure olvas(var m:tszo);
var
  mondat_vege:set of char;
  i          :integer;
  c          :char;
begin
  mondat_vege:=['.','?','!'];
  i:=0;
  writeln('A mondat (max.80 karakter): ');
  repeat
     read(c);
     i:=i+1;
     m.mondat[i]:=c;
  until (c in mondat_vege) and (i< 80);
  m.mondat[i]:=' ';
  m.tipus:=c;    { mondat záró karaktere }
  m.mondat[0]:=chr(i); { a mondat hossza }
end;
procedure keres(var m:tszo);
var
   i,j,db:integer;
   szo :string;
begin
   m.rdb   :=255;
   m.hdb   :=0;
   db:=length(m.mondat); j:=0;
```

```
   for i:=1 to db do
   begin
      if  not (m.mondat[i] in [' ',',']) then
      begin
         j:=j+1;
         szo[j]:=m.mondat[i];
     end
     else
     begin
       if j<> 0 then
       begin
         szo[0]:=chr(j);
         if j>m.hdb then
                    begin
                      m.hdb      :=j;
                      m.leghos   :=szo;
                    end;
         if j<m.rdb then
                    begin
                      m.rdb      :=j;
                      m.legrov   :=szo;
                    end;
         j:=0;
       end;
     end;
  end;
end;

procedure kiir(m:tszo);
begin
   writeln;
   case m.tipus of
    '!' : write('Felkiáltó ');
    '?' : write('Kérdő ');
    '.' : write('Kijelentő ');
    end;
   writeln('mondat.');
   writeln('A legrövidebb szava: ',m.legrov);
   writeln('A leghosszabb szava: ',m.leghos);
end;
begin
   olvas(szoveg);
   keres(szoveg);
   kiir(szoveg);
   readln;readln;
end.
```

A program futásának eredménye:

```
A mondat (max.80 karakter):
Ma szep az ido kirandulni megyunk.

Kijelentő mondat.
A legrövidebb szava: Ma
A leghosszabb szava: kirandulni
```

Írjunk programot, amely rákérdez az adatok számára és beolvassa a valós adatokat. Számítsuk ki az adatok összegét, átlagát és ellenőrizzük, hogy számtani vagy mértani sorozatot alkotnak-e! A feladat megoldására használjunk alprogramokat! (*vizsgal*)

```
program vizsgal;
type
   tomb = array [1.40] of real;
   stat = record
        x : tomb;
        db: integer;
        osszeg, atlag: real;
        szamtani_sorozat: boolean;
     end;

var
    Xosszeg: real;
    t:stat;
procedure TombOlvas(var st:stat);
var
   i: integer;
begin
  write('Elemek száma: ');
  readln(st.db);

  for i:= 1 to st.db do
  begin
    write(i:2,'. elem: '); readln(st.x[i]);
  end;
end;
function Vizsgal(var st:stat):boolean;
var
     dx: real;
     i,jel: integer;
begin
    dx:= st.x[2]-st.x[1];
    Vizsgal:= true;
    st.szamtani_sorozat:= true;
    for i:= 3 to st.db do
       if st.x[i]-st.x[i-1] <> dx then
               begin
                 Vizsgal:= false;
                 st.szamtani_sorozat := false;
                 break;
               end;
   Vizsgal:=st.szamtani_sorozat;
end;
function StOsszeg(var st:stat):real;
var
   i: integer;
begin
   st.Osszeg:= 0;
```

```
  for i:=1 to st.db do
      st.Osszeg:= st.Osszeg+ st.x[i];
  StOsszeg:= st.Osszeg;
end;
procedure Statlag(var st:stat);
begin
  st.Atlag:= stOsszeg(st)/st.db;
end;

begin
  TombOlvas(t);
  Xosszeg:= StOsszeg(t);
  Statlag(t);
  writeln('Átlag : ',t.atlag:8:2);
  writeln('Összeg: ',Xosszeg:8:2);
  if Vizsgal(t) then  writeln('számtani sorozat')
                else  writeln('nem számtani sorozat');
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 4
 1. elem: 2
 2. elem: 4
 3. elem: 6
 4. elem: 8
Átlag :     5.00
Összeg:    20.00
számtani sorozat
```

8.3 Tömb paraméterek

Írjunk programot, amely beolvassa az adatok számát, majd beolvassa az egész típusú adatokat. Számítsuk ki a páros adatok összegét és a páratlan adatok szorzatát! A feladat megoldásához használjunk alprogramokat! (*tombtgy1*)

```
program tombtgy1;
const N=50;
  type
    tomb=array[1..N] of integer;

  var
    x,y1:tomb;
    db,n1:integer;
    ossz,szorz:integer;
  procedure olvas(var y:tomb; var n:integer);
    var i:integer;
    begin
      write('Az elemek száma: ');readln(n);
        for i:=1 to n do
          begin
            write('x[',i:2,']= ');readln(y[i]);
          end;
    end;

  function ParosOsszeg(y:tomb; n:integer):integer;
    var szum, i:integer;
      begin
        szum:=0;
        for i:=1 to n do
        begin
         if y[i] mod 2 = 0 then
          szum:=szum+y[i];
        end;
      ParosOsszeg:=szum;
    end;

  function ParatlanSzorzat(y:tomb; n:integer):integer;
    var szor, i:integer;
      begin
        szor:=1;
        for i:=1 to n do
        begin
          if y[i] mod 2 = 1 then
          szor:=szor*y[i];
        end;
      ParatlanSzorzat:=szor;
    end;
begin
  olvas(y1,n1);
  ossz:= ParosOsszeg(y1,n1);
```

```
  writeln('A páros elemek összege  : ',ossz);
  szorz:= ParatlanSzorzat(y1,n1);
  write('A páratlan elemek szorzata: ',szorz);
  readln;
end.
```

A program futásának eredménye:

```
Az elemek száma: 5
x[ 1]= 1
x[ 2]= 2
x[ 3]= 3
x[ 4]= 4
x[ 5]= 5
A páros elemek összege  : 6
A páratlan elemek szorzata: 15
```

Írjunk programot, amely rákérdez az elemek számára, beolvassa az egész típusú elemeket. Kérdezzük rá az oszthatóság vizsgálatához szükséges adatra, és számláljuk meg, hogy a tömbnek hány eleme osztható vele! A feladat megoldására használjunk alprogramokat! (*tombtgy2*)

```
program tombtgy2;
type
     tomb = array[1..10] of integer;
var
    x  : tomb;
    m,oszthat,oszt_db : integer;

function szamol(n:integer; v: tomb; oszt:integer):integer;
var
   i,db  : integer;
begin
   db:= 0;
   for i:=1 to n do
     if (v[i] mod oszt) = 0 then db:=db+1;
   szamol:= db;
end;

procedure olvas(var n: integer; var v:tomb);
var i:integer;
begin
    write('Elemek száma: '); readln(n);
    for i:=1 to n do
    begin
     write(i,'. elem : '); readln(v[i]);
   end;
end;

begin
    olvas(m,x);
    writeln;
```

```
    write('Oszthatóság: ');
    readln(oszthat);
    oszt_db := szamol(m,x,oszthat);
    writeln('A tömb elemei közül ',oszt_db,' elem osztható: ',oszthat);
    readln;
end.
```

A program futásainak eredménye:

```
Elemek száma: 5
1. elem : 2
2. elem : 6
3. elem : 4
4. elem : 12
5. elem : 8

Oszthatóság: 4
A tömb elemei közül 3 elem osztható: 4


Elemek száma: 5
1. elem : 1
2. elem : 2
3. elem : 3
4. elem : 4
5. elem : 5

Oszthatóság: 3
A tömb elemei közül 1 elem osztható: 3
```

Írjunk programot, amely beolvassa az elemek számát, és feltölt 2 vektort valós adatokkal! Számítsuk ki a két vektor skalár szorzatát és keressük meg mind a két tömb legkisebb elemét! A feladat megoldására használjunk alprogramokat! (*tombtgy3*)

```
program tombtgy3;
type
    vekt=array[1..10] of real;
var
  v1,v2:vekt;
  db   : integer;
  v1_m, v2_m:real;
procedure VektOlvas(var n:integer; var x,y: vekt);
var
   i:integer;
begin
  write('Elemek száma: '); readln(n);
  for i:=1 to n do
  begin
    write('x ',i:2,'.elem: '); readln(x[i]);
    write('y ',i:2,'.elem: '); readln(y[i]);
    writeln;
  end;
end;
```

```
function SkalarSzorzat(n:integer; x,y:vekt):real;
var
   i:integer;
   s:real;
begin
    s:= 0;
    for i:=1 to n do
      s:= s+x[i]*y[i];
    SkalarSzorzat:= s;
end;

procedure Minimum_kereses(n:integer; x:vekt; var min:real);
var
  i:integer;
begin
   min:= x[1];
   for i:=1 to n do
     if x[i]<min then min:= x[i];
end;
begin
  VektOlvas(db,v1,v2);
  writeln('Skalár szorzat: ', SkalarSzorzat(db, v1,v2):6:2);
  Minimum_kereses(db, v1,v1_m);
  writeln('v1 minimuma: ',v1_m:6:2);
  Minimum_kereses(db,v2,v2_m);
  writeln('v2 minimuma: ',v2_m:6:2);
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 3
x  1.elem: 2
y  1.elem: 4

x  2.elem: 6
y  2.elem: 7

x  3.elem: 2
y  3.elem: 6

Skalár szorzat:  62.00
v1 minimuma:   2.00
v2 minimuma:   4.00
```

Írjunk programot, amely adott darabszámú valós típusú adatottal tölti fel a tömböt. Keressük meg a tömb legnagyobb elemét, a legkisebb elemét és annak indexét. A legnagyobb elemmel készítsünk normálást, melyet egy eredmény tömbben végezzük el, valamint jelenítsük meg a normált tömb tartalmát. A feladatot alprogramokkal oldjuk meg! (*tpar1*)

```
program tpar1;
type rtomb = array[1..20] of real;
var
  x , normx: rtomb;
  x_db, xmin_index: integer;
  xmax, xmin: real;

  procedure Olvas(var y : rtomb; var n: integer);
  var
    i: integer;
  begin
    write('Adatok száma: '); readln(n);
    for i:=1 to n do
    begin
      write(i:2,'. adat: '); readln(y[i]);
    end;
  end;

  function MaxAdatKeres(y: rtomb; n: integer):real;
  var
    i: integer;
    max: real;
  begin
       max:= y[1];
    for i:=1 to n do
      if y[i] > max then max:= y[i];
    MaxAdatKeres:= max;
  end;

  procedure MinAdatKeres(y: rtomb; n: integer; var min: real;
                         var min_index: integer);
  var
    i: integer;
  begin
    min:= y[1];
    min_index:= 1;

    for i:=2 to n do
    begin
      if y[i] < min then
      begin
       min:= y[i];
       min_index:= i;
      end;
    end;
end;
```

```
procedure AdatokNormalasa(y: rtomb; var ny: rtomb; n: integer;
                          max: real);
 var
   i: integer;
 begin
   for i:=1 to n do
     ny[i] := y[i]/max;
 end;

 procedure Kiir(y: rtomb; ny: rtomb; n: integer);
 var
   i: integer;
 begin
   writeln;
   writeln('Index   Adatok Normált adatok ');
   writeln('----------------------------');
   for i:=1 to n do
     writeln(i:2,y[i]:10:1,ny[i]:10:1);
   writeln;
 end;
 begin
    Olvas(x,x_db);
    xmax:= MaxAdatKeres(x, x_db);
    MinAdatKeres(x, x_db, xmin, xmin_index);
    AdatokNormalasa(x,normx,x_db,xmax);
    Kiir(x,normx, x_db);
    writeln('A legnagyobb adat: ',xmax:6:2);
    writeln('A legkisebb adat : ',xmin:6:2);
    writeln('A legkisebb adat indexe: ',xmin_index);
    readln;
  end.
```

A program futásának eredménye:

```
Adatok száma: 4
 1. adat: 2
 2. adat: 6
 3. adat: 3
 4. adat: 12

Index   Adatok Normált adatok
----------------------------
 1       2.0       0.2
 2       6.0       0.5
 3       3.0       0.3
 4      12.0       1.0

A legnagyobb adat:  12.00
A legkisebb adat :   2.00
A legkisebb adat indexe: 1
```

Írjunk programot, amely adott darabszámú egész típusú adatottal tölti fel a tömböt. Számítsuk ki a pozitív adatok átlagát, és negatív adatok összegét! A feladatot alprogramokkal oldjuk meg! (*tpar2*)

```
program tpar2;
type itomb = array[1..40] of integer;
var
  a: itomb;
  a_db: integer;
  Poz_atlag: real;
  Neg_osszeg, Neg_db: integer;
  procedure TOlvas(var x : itomb; var n: integer);
  var
    i: integer;
  begin
    write('Adatok száma: '); readln(n);
    for i:=1 to n do
    begin
      write(i:2,'. adat: '); readln(x[i]);
    end;
  end;
  function PozAtlagSzamol(x: itomb; n: integer):real;
  var
    i,p_db: integer;
    p_atlag: real;
  begin
    p_atlag:= 0; p_db:= 0;
    PozAtlagSzamol:= 0;
    for i:=1 to n do
      if x[i] > 0 then
      begin
        p_atlag:= p_atlag+x[i];
        p_db:= p_db+1;
      end;
      if p_db = 0 then writeln('A pozitív adatok száma: ',p_db)
      else
      PozAtlagSzamol:= p_atlag/p_db;
  end;
  procedure NegOsszeg(x: itomb; n: integer;
                      var negossz, negdb: integer);
  var
    i: integer;
  begin
    negossz:= 0;
    negdb:= 0;
    for i:=1 to n do
    begin
      if x[i] < 0 then
      begin
        negossz:= negossz+x[i];
```

```
        negdb:= negdb+1;
      end;
    end;
  end;
  begin
    TOlvas(a,a_db);
    Poz_atlag:= PozAtlagSzamol(a, a_db);
    NegOsszeg(a, a_db, Neg_osszeg, Neg_db);
    if Poz_atlag > 0 then
      writeln('A pozitív adatok átlaga: ',Poz_atlag:6:2);
    writeln('Negatív adatok összege : ',Neg_osszeg);
    writeln('Negatív adatok darabszáma: ',Neg_db);
    readln;
  end.
```

A program futásának eredménye:

```
Adatok száma: 5
 1. adat: 12
 2. adat: -3
 3. adat: 5
 4. adat: -4
 5. adat: 7
A pozitív adatok átlaga:   8.00
Negatív adatok összege : -7
Negatív adatok darabszáma: 2
```

Írjunk programot, amely adott darabszámú egész típusú adatokkal tölti fel a tömböt! Számítsuk ki a tömb páratlan adatainak összegét, és a páros adatainak átlagát! A feladatot alprogramokkal oldjuk meg! (*tpar3*)

```
program tpar3;
type tomb = array[1..30] of integer;
var
  w: tomb;
  w_db: integer;
  Paratlan_osszeg: integer;
  Paros_atlag: real;
  procedure Olvas(var y : tomb; var n: integer);
  var
    i: integer;
  begin
    write('Adatok száma: '); readln(n);
    for i:=1 to n do
    begin
      write(i:2,'. adat: '); readln(y[i]);
    end;
  end;
  function ParatlanOsszegSzamol(y: tomb; n: integer):integer;
  var
    i,ptosszeg, ptdb: integer;
```

```
  begin
    ptosszeg:= 0;   ptdb:= 0;
    for i:=1 to n do
      if y[i] mod 2 = 1 then
      begin
        ptosszeg:= ptosszeg+y[i];
        ptdb:= ptdb+1;
      end;
      if ptdb = 0 then
        writeln('Nincs az adatok között páratlan szám!');
      ParatlanOsszegSzamol:= ptosszeg;
  end;
  procedure ParosAtlagSzamol(y: tomb; n: integer; var patlag: real);
  var
    i, pdb: integer;
  begin
    patlag:= 0; pdb:= 0;
    for i:=1 to n do
    begin
      if y[i] mod 2 = 0 then
      begin
        patlag:= patlag+y[i];
        pdb:= pdb+1;
      end;
    end;
    if pdb = 0 then writeln('Nincs az adatok között páros szám!')
    else
     patlag:= patlag/pdb;
  end;
  begin
    Olvas(w,w_db);
    Paratlan_osszeg:= ParatlanOsszegSzamol(w, w_db);
    ParosAtlagSzamol(w, w_db, Paros_atlag);
    if Paros_atlag > 0 then
       writeln('A páros adatok átlaga    : ',Paros_atlag:6:2);
    writeln('A páratlan adatok összege: ',Paratlan_osszeg);
    readln;
  end.
```

A program futásának eredménye:

```
Adatok száma: 6
 1. adat: 1
 2. adat: 2
 3. adat: 3
 4. adat: 4
 5. adat: 5
 6. adat: 6
A páros adatok átlaga    :   4.00
A páratlan adatok összege: 9
```

Írjunk programot, amely beolvas max 80 karakteres szöveget, megszámlálja a magánhangzókat és a magánhangzókat egy megadott karakterre cseréli! A feladatot alprogramokkal oldjuk meg! (*tpar4*)

```
program tpar4;
type st80 = string[80];
var
  szoveg: st80;
  s_db: integer;
  mgh_db: integer;
  kar_csere: char;
  procedure SzovegOlvas(var s: st80; var sdb: integer);
  var
     i: integer;
  begin
     write('szöveg: '); readln(s); sdb:=ord(s[0]);
  end;
  function MaganHSzamol(s: st80; sdb: integer):integer;
  var
    i,mgh_db: integer;
  begin
    mgh_db:= 0;
    for i:=1 to sdb do
      if s[i] in ['a','e','i','o','u'] then
        mgh_db:= mgh_db+1;
    MaganHSzamol:= mgh_db;
  end;
  procedure Cserel(var s: st80; sdb: integer; c: char);
  var
    i: integer;
  begin
    for i:=1 to sdb do
    begin
      if s[i] in ['a','e','i','o','u'] then s[i]:= c;
    end;
  end;
  begin
    SzovegOlvas(szoveg,s_db);
    mgh_db:= MaganHSzamol(szoveg, s_db);
    writeln('Magánhangzók száma: ',mgh_db);
    write('Cserekarakter: '); readln(kar_csere);
    Cserel(szoveg, s_db,kar_csere);
    writeln('Az átalakított szöveg: ',szoveg);
    readln;
  end.
```

A feladat megoldása:

```
szöveg: ma szep az ido.
Magánhangzók száma: 5
Cserekarakter: *
Az átalakított szöveg: m* sz*p *z *d*.
```

Írjunk programot, amely beolvas egy mondatot és kihagyja a kis betűs magánhangzókat, majd megszámlálja a szóközöket! (*tpar5*)

```
program tpar5;
var
  mondat, ujmondat: string;
  m_db: integer;
  szokoz_db: integer;
  procedure MondatOlvas(var s: string; var sdb: integer);
  var
     i: integer;
  begin
     write('szöveg: '); readln(s);
     sdb:=ord(s[0]);
  end;
  function KihagyMgh(s: string; sdb: integer):string;
  var
    i, j: integer;
    sk: string;
  begin
    j:= 0;
    for i:=1 to sdb do
      if not( s[i] in ['a','e','i','o','u']) then
      begin
        j:= j+1;
        sk[j]:=s[i];
      end;
      sk[0]:= chr(j);
     KihagyMgh:= sk;
  end;
  procedure SzokozSzamlal(var s: string; sdb: integer;
                          var szokozdb: integer);
  var
    i: integer;
  begin
    szokozdb:=0;
    for i:=1 to sdb do
    begin
      if s[i] = ' ' then szokozdb:= szokozdb+1;
    end;
  end;
  begin
    MondatOlvas(mondat,m_db);
    ujmondat:= KihagyMgh(mondat, m_db);
    writeln('új mondat: ',ujmondat);
    SzokozSzamlal(mondat, m_db,szokoz_db);
    writeln('A szóközök száma: ',szokoz_db);
    readln;
  end.
```

A program futásának eredménye:

```
szöveg: ma szep az ido.
új mondat: m szp z d.
A szóközök száma: 3
```

Írjunk programot, amely beolvas egy mondatot és a kisbetűs magánhangzókat megtoldja a v és a magánhangzó ismétlésével! A feladatot alprogramokkal oldjuk meg! (*tpar6*)

```
program tpar6;
var
  mondat, ujmondat: string;
  m_db: integer;
  procedure SzOlvas(var sz: string; var szdb: integer);
  var
     i: integer;
  begin
     write('szöveg: '); readln(sz);
     szdb:=ord(sz[0]);
  end;
  function Atalakit(sz: string; szdb: integer):string;
  var
    i, j: integer;
    sk: string;
  begin
    j:= 0;
    for i:=1 to szdb do
    begin
      if sz[i] in ['a','e','i','o','u',
                   'A','E','I','O','U'] then
      begin
        j:= j+1;
        sk[j]:= sz[i];
        j:= j+1;
        sk[j]:= 'v';
        j:= j+1;
        sk[j]:= sz[i];
      end
      else

      begin
        j:= j+1;
        sk[j]:= sz[i];
      end;
     end;
     sk[0]:= chr(j);
     Atalakit:= sk;
  end;
  begin
    SzOlvas(mondat,m_db);
    ujmondat:= Atalakit(mondat, m_db);
```

```
    writeln('új mondat: ',ujmondat);
    readln;
  end.
```

A program futásának eredménye:

```
szöveg: Tudsz igy beszelni?
új mondat: Tuvudsz ivigy beveszevelnivi?
```

Írjunk programot, amely adott darabszámú valós típusú adatokkal tölti fel a tömböt. Számláljuk meg a tömb azon elemeit, melyek a korlát fölé esnek. A feladatot alprogramokkal oldjuk meg! (*tpar7*)

```
program tpar7;
type  tomb = array[1..10] of real;
var
   X       : tomb;
   X_db    : integer;
   X_korlat: real;
   X_talal : integer;
procedure TombOlvas(var n: integer; var y : tomb);
var i: integer;
begin
  write('Elemek száma: ');
  readln(n);
  for i:= 1 to n do
  begin
    write(i:2,'.elem: ');
    readln(y[i]);
  end;
end;
procedure Keres( n: integer; y:tomb;
                 korlat: real;
                 var talal: integer);
var
   i: integer;
begin
  talal:=0;
  for i:=1 to n do
    if korlat < y[i] then inc(talal);
end;
begin
  TombOlvas(X_db,X);
  write('Korlát értéke: ');
  readln(X_korlat);
  Keres(X_db,X,X_korlat,X_talal);
  writeln(X_korlat:8:2,
          ' korlát feletti elemek száma: ',
          X_talal);
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 5
 1.elem: 1
 2.elem: 2
 3.elem: 3
 4.elem: 4
 5.elem: 5
Korlát értéke: 2
    2.00 korlát feletti elemek száma: 3
```

Írjunk programot, amely beolvassa az elemek számát, és feltölti a *t_tomb* típusú tömböt adatokkal. Az adatok a téglalap bal felső koordinátája, valamint a szélessége és magassága. Írjunk olyan alprogramot, amely egy megadott koordináta pontról eldönti, hogy melyik téglalap tartalmazza, azt jelenítsük meg a képernyőn! A feladat megoldására használjunk alprogramokat! (*rekpart1*)

```
program rekpart1;
type
        teglalap = record
                    x,y,a,b: integer;
                  end;
        t_tomb = array[1..10] of teglalap;

var
    w     : t_tomb;
    m     : integer;
    Xkoord,Ykoord: integer;

procedure olvas_teglalap(var n: integer;var z: t_tomb);
var
    i:integer;
begin
    write('Elemek száma: '); readln(n);
    for i:=1 to n do
    begin
      writeln('A téglalap bel felső sarkának koordinátája');
      write(i,'. x: '); readln(z[i].x);
      write(i,'. y: '); readln(z[i].y);
      writeln('A téglalap két oldala');
      write(i,'. a: '); readln(z[i].a);
      write(i,'. b: '); readln(z[i].b);
      writeln;
   end;
end;
procedure tartalmaz(x_koord,y_koord:integer;
                    n: integer; z:t_tomb);
var
    i, van:integer;
```

```
begin
   van:=0;
   writeln;
   writeln('Az alábbi téglalapok tartalmzzák a koordinátapontot.');
   writeln;
   for i:=1 to n do
     if( x_koord >= z[i].x) and (x_koord <= z[i].x+z[i].a)
       and ( y_koord >= z[i].y) and (y_koord <= z[i].x+z[i].b)
       then
       begin
          writeln(i:2,'. téglalap tartalmazza:');
          writeln('A téglalap adatai:');
          writeln('X koord: ',w[i].x);
          writeln('Y koord: ',w[i].y);
          writeln(' szélessége: ',w[i].a);
          writeln(' magassága : ',w[i].b);
          writeln;   van:=1;
      end;
      if van = 0 then writeln('Egy téglalap sincs',
                              ' amely tartalmazná a koordinátapontot!');
end;
begin
    olvas_teglalap(m,w);
    write('A pont x koordinátája: '); readln(Xkoord);
    write('A pont y koordinátája: '); readln(Ykoord);
    tartalmaz(Xkoord,Ykoord,m,w);
    readln;
end.
```

A program futásainak eredménye:

```
Elemek száma: 3
A téglalap bel felső sarkának koordinátája
1. x: 1
1. y: 1
A téglalap két oldala
1. a: 5
1. b: 5

A téglalap bel felső sarkának koordinátája
2. x: 2
2. y: 2
A téglalap két oldala
2. a: 6
2. b: 6

A téglalap bel felső sarkának koordinátája
3. x: 5
3. y: 5
A téglalap két oldala
3. a: 8
3. b: 8

A pont x koordinataja: 3
A pont y koordinataja: 3
```

```
Az alábbi téglalapok tartalmzzák a koordinátapontot.

 1. téglalap tartalmazza:
A téglalap adatai:
X koord: 1
Y koord: 1
 szélessége: 5
 magassága : 5

 2. téglalap tartalmazza:
A téglalap adatai:
X koord: 2
Y koord: 2
 szélessége: 6
 magassága : 6
```

Írjunk programot, amely beolvassa az elemek számát, és feltölti a *t_tomb* típusú tömböt adatokkal. Az adatok a téglalap bal felső koordinátája, valamint a szélessége és magassága. Írjunk olyan alprogramot, amely egy megadott koordináta pontról eldönti, hogy melyik téglalap tartalmazza, és keressük meg az adott koordinátapont és a téglalap bal felső sarkának koordinátapontja közötti legkisebb távolságot! A feladat megoldására használjunk alprogramokat! (*rekpart2*)

```
program rekpart2;
type
         teglalap = record
                      x,y,a,b: integer;
                    end;
         t_tomb = array[1..10] of teglalap;

var
    w     : t_tomb;
    m, j     : integer;
    Xkoord,Ykoord: integer;
    tavolsag:real;

procedure olvas_teglalap(var n: integer;var z: t_tomb);
var
    i:integer;
begin
    write('Elemek száma: '); readln(n);
    for i:=1 to n do
    begin
      writeln(i:2,'. téglalap');
      write('  x: '); readln(z[i].x);
      write('  y: '); readln(z[i].y);
      write('  a: '); readln(z[i].a);
      write('  b: '); readln(z[i].b);
      writeln;
   end;
end;
```

```
procedure tartalmaz(x_koord,y_koord:integer;
                    n: integer; z:t_tomb; var tavmin:real;
                    var index :integer);
var
    i: integer;
    tav:real;
begin
   tavmin:=-1; index:=-1;
   for i:=1 to n do
     if( x_koord >= z[i].x) and (x_koord <= z[i].x+z[i].a)
       and ( y_koord >= z[i].y) and (y_koord <= z[i].x+z[i].b)
       then
         begin
            tav:=sqrt(sqr(x_koord-z[i].x)+sqr(y_koord-z[i].y));
            if tavmin = -1 then begin tavmin :=tav;  index:=i; end;
            if tav < tavmin then begin tavmin:= tav; index:=i; end;
         end;
end;

begin
    olvas_teglalap(m,w);
     write('A pont x koordinátája: '); readln(Xkoord);
    write('A pont y koordinátája: '); readln(Ykoord);
    tartalmaz(Xkoord,Ykoord,m,w,tavolsag, j);
    writeln;
    if j = -1 then writeln('Nincs belső pont.')
    else
    begin
      write(j:2,'. téglalapnál van ');
      writeln('a legkisebb távolság: ',tavolsag:6:2);
    end;
    readln;
end.
```

A program futásainak eredménye:

```
Elemek száma: 3
 1. téglalap
  x: 2
  y: 2
  a: 4
  b: 4
 2. téglalap
  x: 5
  y: 5
  a: 3
  b: 3
 3. téglalap
  x: 6
  y: 6
  a: 7
  b: 7
```

```
A pont x koordinátája: 3
A pont y koordinátája: 3
 1. téglalapnál van a legkisebb távolság:   1.41
```

Második futtatás:

```
Elemek száma: 3
 1. téglalap
  x: 1
  y: 1
  a: 4
  b: 4
 2. téglalap
  x: 2
  y: 2
  a: 6
  b: 6
 3. téglalap
  x: 5
  y: 5
  a: 7
  b: 7
A pont x koordinátája: 15
A pont y koordinátája: 15
Nincs belső pont.
```

Írjunk programot, amely beolvassa a koordinátapontok számát, és feltölti a *koordinatak* típusú tömböt, melynek minden eleme *koord* típusú struktúra. Olvassuk be elemenként a két koordinátapont adatát és számítsuk ki a távolságát. Számítsuk ki a pontok közötti össz távolságot! Keressük meg a legnagyobb távolságot! Írassuk ki az adatokat! A feladat megoldására használjunk alprogramokat! (*rekpart3*)

```
program rekpart3;
type  koord = record
                x1,y1,x2,y2: integer;
                tav: real;
            end;
    koordinatak = array[1..30] of koord;
var
  k : koordinatak;
  k_db: integer;
  ossz_tav: real;
  max_koord: koord;
  procedure KoordOlvas(var kd: koordinatak; var n: integer);
  var
     i: integer;
  begin
    write('A koordináták darabszáma: '); readln(n);
```

```
  writeln('A koordináták adatai');
  for i:=1 to n do
  begin
    write('1.koord  x1 : '); readln(kd[i].x1);
    write('         y1 : '); readln(kd[i].y1);
    write('2.koord  x2 : '); readln(kd[i].x2);
    write('         y2 : '); readln(kd[i].y2);
    kd[i].tav :=sqrt(sqr(kd[i].x1-kd[i].x2)+sqr(kd[i].y1-kd[i].y2));
    writeln;
  end;
end;
function Ossztav( kd: koordinatak; n: integer):real;
var
   i : integer;
   tavolsag: real;
begin
  tavolsag:=0;
  for i:=1 to n do
  begin
    tavolsag:= tavolsag+ kd[i].tav
  end;
  Ossztav:= tavolsag;
end;
procedure MaxTavKeres( kd: koordinatak; n:integer;
                       var MaxTav:koord);
var
  i: integer;
begin
  MaxTav:= kd[1];
  for i:=2 to n do
  begin
    if MaxTav.tav < kd[i].tav then
       MaxTav:= kd[i];
  end;
end;
procedure Kiir( kd: koordinatak; n: integer);
var
   i: integer;
begin
   writeln;
   writeln('A koordináták adatainak megjelenítése');
   writeln;
   writeln('   x1   y1    x2   y2     tav  ');
   writeln('-------------------------------');
   for i:=1 to n do
   begin
      write(kd[i].x1:5);
      write(kd[i].y1:5);
      write(kd[i].x2:6);
      write(kd[i].y2:5);
      writeln(kd[i].tav:9:1);
   end;
end;
```

```
  procedure MaxKiir( mk: koord);
  begin
     writeln;
    writeln('A max távolságú koordinátapont adatainak megjelenítése');
     writeln;
     writeln('   x1   y1    x2   y2     tav  ');
     writeln('--------------------------------');
     write(mk.x1:5);
     write(mk.y1:5);
     write(mk.x2:6);
     write(mk.y2:5);
     writeln(mk.tav:9:1);
  end;
  begin
    KoordOlvas(k,k_db);
    Kiir(k,k_db);
    ossz_tav:= Ossztav(k,k_db);
    MaxTavKeres(k,k_db,max_koord);
    writeln;
    writeln('A távolságok összege: ',ossz_tav:6:2);
    MaxKiir(max_koord);
    readln;
  end.
```

A program futásának eredménye:

```
A koordináták darabszáma: 3
A koordináták adatai
1.koord  x1 : 1
         y1 : 1
2.koord  x2 : 2
         y2 : 2

1.koord  x1 : 5
         y1 : 5
2.koord  x2 : 7
         y2 : 7

1.koord  x1 : 3
         y1 : 4
2.koord  x2 : 6
         y2 : 7

A koordináták adatainak megjelenítése
   x1   y1    x2   y2     tav
--------------------------------
    1    1     2    2      1.4
    5    5     7    7      2.8
    3    4     6    7      4.2

A távolságok összege:   8.49

A max távolságú koordinátapont adatainak megjelenítése
   x1   y1    x2   y2     tav
--------------------------------
    3    4     6    7      4.2
```

Írjunk programot, amely beolvassa a szakosztályok számát, és feltölti a *sportklub* típusú tömböt, melynek minden eleme *sport* típusú struktúra. Olvassuk be elemenként a sportágat, létszámot és a helyezési díjat (1,2,3). Számláljuk meg a díjakat és külön az első díjakat, számítsuk ki az össz létszámot! Keressük meg a legnagyobb létszámú szakosztályt! A feladat megoldására használjunk alprogramokat! (*rekpart4*)

```
program rekpart4;
type   sport= record
                sportag: string;
                letszam: integer;
                dij: integer;
            end;
    sportklub = array[1..10] of sport;
var
  s : sportklub;
  s_db: integer;
  max_letszam: sport;
  ossz_letszam, dijak_szama, elso_dij: integer;
  procedure SportKlubOlvas(var sp: sportklub; var n: integer);
  var
     i: integer;
  begin
    write('A sport szakosztályok száma: '); readln(n);
    writeln('A szakosztályok adatai');
    for i:=1 to n do
    begin
      write('sportág : '); readln(sp[i].sportag);
      write('létszám : '); readln(sp[i].letszam);
      write('díj     : '); readln(sp[i].dij);
      writeln;
    end;
  end;
  function ElsoDijakSzama( sp: sportklub; n: integer):integer;
  var
     i : integer;
     db: integer;
  begin
    db:=0;
    for i:=1 to n do
    begin
      if sp[i].dij = 1 then db:= db + sp[i].dij;
    end;
    ElsoDijakSzama:= db;
  end;
```

```
function OsszLetszam( sp: sportklub; n: integer):integer;
var
   i : integer;
   letszamdb: integer;
begin
  letszamdb:=0;
  for i:=1 to n do
  begin
    letszamdb:= letszamdb + sp[i].letszam;
  end;
  OsszLetszam:= letszamdb;
end;
function DijakSzama( sp: sportklub; n: integer):integer;
var
   i : integer;
   db: integer;
begin
  db:=0;
  for i:=1 to n do
  begin
    if sp[i].dij > 0 then db:= db + 1;
  end;
  DijakSzama:= db;
end;
procedure MaxLetszamKeres( sp: sportklub; n:integer;
                           var Maxletszam:sport);
var
  i: integer;
begin
  Maxletszam:= sp[1];
  for i:=2 to n do
  begin
    if Maxletszam.dij < sp[i].letszam then
       Maxletszam:= sp[i];
  end;
end;
procedure Kiir( sp: sportklub; n: integer);
var
   i: integer;
begin
   writeln;
   writeln('A sportklub adatainak megjelenítése');
   writeln;
   writeln('   Sportág   létszám   díj  ');
   writeln('------------------------------');
   for i:=1 to n do
   begin
      write(sp[i].sportag:10);
      write(sp[i].letszam:8);
      writeln(sp[i].dij:7);
   end;
end;
```

```
  procedure MaxKiir( mp: sport);
  begin
     writeln;
 writeln('A legnagyobb létszámú szakosztály adatainak megjelenítése');
     writeln;
     writeln('   Sportág   létszám   díj  ');
     writeln('------------------------------');
     write(mp.sportag:10);
     write(mp.letszam:8);
     writeln(mp.dij:7);
  end;
  begin
    SportKlubOlvas(s,s_db);
    Kiir(s,s_db);
    dijak_szama:= DijakSzama(s,s_db);
    elso_dij:=ElsoDijakSzama(s, s_db);
    ossz_letszam:= OsszLetszam(s, s_db);
    MaxLetszamKeres(s,s_db,max_letszam);
    MaxKiir(max_letszam);
    writeln;
    writeln('A dijak darabszáma: ',dijak_szama);
    writeln('Első díjak száma: ',elso_dij);
    writeln('Össz létszám: ',ossz_letszam);
    readln;
  end.
```

A program futásának eredménye:

```
A sport szakosztályok száma: 3
A szakosztályok adatai
sportág : Evezos
létszám : 12
díj     : 2

sportág : Kenus
létszám : 8
díj     : 1

sportág : Vitorlas
létszám : 15
díj     : 1


A sportklub adatainak megjelenítése

   Sportág   létszám   díj
------------------------------
    Evezos      12      2
     Kenus       8      1
  Vitorlas      15      1
```

```
A legnagyobb létszámú szakosztály adatainak megjelenítése

   Sportág   létszám   díj
------------------------------
  Vitorlas      15      1

A dijak darabszáma: 3
Első díjak száma: 2
Össz létszám: 35
```

Írjunk programot, amely beolvassa az utasok számát, és feltölti az *utasok* típusú tömböt, melynek minden eleme *utas* típusú struktúra! Olvassuk be elemenként az utas nevét, ahová utazik, a távolságot és a jegy árát! Keressük meg a legolcsóbb jegyet, a legnagyobb távolságot, számítsuk ki az utasok számát! Számláljuk meg az adott városba utazók számát! Írassuk ki az adatokat! A feladat megoldására használjunk alprogramokat! (*rekpart5*)

```
program rekpart5;
type
     st11 = string[11];
     utas = record
                utasnev: st11;
                varos: st11;
                km: integer;
                jegyar: real;
            end;
    utasok = array[1..20] of utas;
var
  u : utasok;
  u_db: integer;
  varos:st11;
  varosba_utazok_letszama: integer;
  max_km: integer;
  min_jegyar: real;
  procedure UtasokOlvas(var ut: utasok; var n: integer);
  var
     i: integer;
  begin
    write('Az utasok száma: '); readln(n);
    writeln('Az utasok adatai');
    for i:=1 to n do
    begin
      write('utasnév     : '); readln(ut[i].utasnev);
      write('város       : '); readln(ut[i].varos);
      write('távolság(km): '); readln(ut[i].km);
      write('jegy ára    : '); readln(ut[i].jegyar);
      writeln;
    end;
  end;
```

```
function MinJegyAr( ut: utasok; n: integer):real;
var
   i : integer;
   ar: real;
begin
  ar:= ut[1].jegyar;
  for i:=2 to n do
  begin
    if ut[i].jegyar < ar then ar:= ut[i].jegyar;
  end;
  MinJegyAr:= ar;
end;
function VarosbaUtazokLetszama( ut: utasok; n: integer;
                                varos:st11):integer;
var
   i : integer;
   letszam: integer;
begin
  letszam:= 0;
  for i:=1 to n do
  begin
    if varos = ut[i].varos then letszam:= letszam + 1;
  end;
  VarosbaUtazokLetszama:= letszam;
end;
procedure MaxKmKeres( ut: utasok; n:integer; var maxkm:integer);
var
  i: integer;
begin
  maxkm:= ut[1].km;
  for i:=2 to n do
  begin
    if maxkm < ut[i].km then
       maxkm:= ut[i].km;
  end;
end;
procedure Kiir( ut: utasok; n: integer);
var
   i: integer;
begin
   writeln;
   writeln('Az utasok adatainak megjelenítése');
   writeln;
   writeln('    utasnév      város     km    jegyár ');
   writeln('-----------------------------------------');
   for i:=1 to n do
   begin
      write(ut[i].utasnev:11);
      write(ut[i].varos:11);
      write(ut[i].km:7);
      writeln(ut[i].jegyar:10:1);
   end;
end;
```

```
  begin
    UtasokOlvas(u,u_db);
    Kiir(u,u_db);
    min_jegyar:= MinJegyAr(u,u_db);
    MaxKmKeres(u,u_db,max_km);
    writeln;
    write('Város: '); readln(varos);
    varosba_utazok_letszama:= VarosbaUtazokLetszama(u, u_db,varos);
    writeln;
    writeln('A legalacsonyabb ár: ',min_jegyar:6:1);
    writeln('Legtávolabb város  : ',max_km);
    writeln(varos,'-ba utazok létszáma: ',varosba_utazok_letszama);
    readln;
  end.
```

A program futásának eredménye:

```
Az utasok száma: 3
Az utasok adatai
utasnév     : Nagy Attila
város       : Foldvar
távolság(km): 150
jegy ára    : 450

utasnév     : Kiss Maria
város       : Kaposvar
távolság(km): 200
jegy ára    : 570

utasnév     : Toth Endre
város       : Pecs
távolság(km): 350
jegy ára    : 1200


Az utasok adatainak megjelenítése

    utasnév      város     km    jegyár
----------------------------------------
Nagy Attila    Foldvar    150     450.0
 Kiss Maria   Kaposvar    200     570.0
 Toth Endre       Pecs    350    1200.0

Város: Pecs

A legalacsonyabb ár:  450.0
Legtávolabb város  : 350
Pecs-ba utazok létszáma: 1
```

Írjunk programot, amely beolvassa a diákok számát, és feltölti a *tankor* típusú tömböt, melynek minden eleme *diak* típusú struktúra. Olvassuk be elemenként a diák nevét, tankör számát, a kreditpontját! Keressük meg a legnagyobb kreditpontot eljárással és függvénnyel! A feladat megoldására használjunk alprogramokat! (*rekpart6*)

```
program rekpart6;
type
    diak = record
        neve: string;
        tk  : integer;
        kredit: integer;
      end;
   tankor= array[1..30] of diak;
var
  t1: tankor;
  db: integer;
  mk1,mk2: integer;

procedure Olvas(var n: integer;
                var t:tankor);
var i: integer;
begin
   write('A diákok száma: ');
   readln(n);
   for i:=1 to n do
   begin
     writeln;
     writeln(i:2,'. diák: ');
     write('Neve  : '); readln(t[i].neve);
     write('Tankör: '); readln(t[i].tk);
     write('Kredit: '); readln(t[i].kredit);
   end;
end;
procedure MaxKredit_P(n:integer; t:tankor; var max_kredit:integer);
var
   i: integer;
begin
  max_kredit:=t[1].kredit;
  for i:=2 to n do
    if max_kredit<t[i].kredit then max_kredit:= t[i].kredit;
end;
function MaxKredit_F(n:integer; t:tankor):integer;
var
   max_kredit,i:integer;
begin
  max_kredit:=t[1].kredit;
  for i:=2 to n do
    if max_kredit<t[i].kredit then max_kredit:= t[i].kredit;
  MaxKredit_F:=max_kredit;
end;
```

```
begin
  Olvas(db,t1);
  MaxKredit_P(db,t1,mk1); writeln;
  writeln('Max kredit eljárással : ',mk1);
  mk2:=MaxKredit_f(db,t1);
  writeln('Max kredit függvénnyel: ',mk2);
  readln;
end.
```

A program futásának eredménye:

```
A diákok száma: 3

 1. diák:
Neve  : Kiss Katalin
Tankör: 1
Kredit: 78

 2. diák:
Neve  : Nagy Attila
Tankör: 1
Kredit: 96

 3. diák:
Neve  : Toth Istvan
Tankör: 1
Kredit: 56

Max kredit eljárással : 96
Max kredit függvénnyel: 96
```

Írjunk programot, amely beolvassa a diákok számát, és feltölti a *korok* típusú tömböt, melynek minden eleme *kor* típusú struktúra! Olvassuk be elemenként a sugarat! Számítsuk ki a kör kerületét és a területét! Jelenítsük meg tömb rekordjainak tartalmát táblázatosan! A feladat megoldására használjunk alprogramokat! (*rekpart7*)

```
program rekpart7;
const  kdb = 20;
type kor = record
             sugar, ter, ker: real;
           end;
korok = array[1..kdb] of kor;
var
  k : korok;
  k_szama: integer;
procedure olvas(var kr: korok; var db: integer);
var
   i: integer;
begin
   repeat
     write('Körök száma: '); readln(db);
   until db in [1..kdb];
```

```
   for i:=1 to db do
   begin
     write(i:2,'.kör sugara: '); readln(k[i].sugar);
   end;
 end;
procedure szamitasok(var kr:korok; db: integer);
var
   i: integer;
begin
  for i:= 1 to db do
  begin
    kr[i].ter:= sqr(kr[i].sugar)*pi;
    kr[i].ker:=2*kr[i].sugar*pi;
  end;
end;
procedure eredmenytkiir(kr:korok; db: integer);
var
   i: integer;
begin
  writeln('   sugár     terület     kerület');
  for i:= 1 to db do
  begin
    writeln(kr[i].sugar:8:1,kr[i].ter:12:2,kr[i].ker:12:2);
  end;
end;

begin
  olvas(k, k_szama);
  szamitasok(k, k_szama);
  eredmenytkiir(k,k_szama);
  readln;
end.
```

A program futásának eredménye:

```
Körök száma: 3
 1.kör sugara: 1
 2.kör sugara: 4
 3.kör sugara: 7
   sugár     terület     kerület
     1.0        3.14        6.28
     4.0       50.27       25.13
     7.0      153.94       43.98
```

Írjunk programot, amely adott darabszámú tömböt tölt fel adatokkal, melynek minden eleme *aru* rekord. Számítsuk az áruk darabszámát és össz értékét! A feladatot alprogramokkal oldjuk meg! (*rekpart8*)

```
program rekpart8;
type  aru = record
              azonosito: string[12];
              db: integer;
              ar : real;
            end;
    raktar = array[1..20] of aru;
var
  r : raktar;
  r_db: integer;
  ossz_db: integer;
  ossz_ar: real;
  procedure RaktarOlvas(var rt: raktar; var n: integer);
  var
     i: integer;
  begin
    write('Az áruk darabszáma: '); readln(n);
    writeln('Az áruk adatai');
    for i:=1 to n do
    begin
      write('azonosító : '); readln(rt[i].azonosito);
      write('darabszáma: '); readln(rt[i].db);
      write('ára       : '); readln(rt[i].ar);
      writeln;
    end;
  end;
  function DbSzamol( rt: raktar; n: integer):integer;
  var
     i, osszdb: integer;
  begin
    osszdb:=0;
    for i:=1 to n do
    begin
      osszdb:= osszdb+ rt[i].db
    end;
    DbSzamol:= osszdb;
  end;
  procedure RaktarErtek( rt: raktar; n:integer; var osszertek:real);
  var
    i: integer;
  begin
    osszertek:= 0;
    for i:=1 to n do
    begin
      osszertek:= osszertek + rt[i].db * rt[i].ar;
    end;
  end;
```

```
  procedure Kiir( rt: raktar; n: integer);
  var
     i: integer;
  begin
     writeln;
     writeln('A raktár adatainak megjelenítése');
     writeln;
     writeln('  Azonosító     Db        Ár  ');
     writeln('-------------------------------');
     for i:=1 to n do
     begin
        write(rt[i].azonosito:12);
        write(rt[i].db:6);
        writeln(rt[i].ar:12:1);
     end;
  end;
  begin
    RaktarOlvas(r,r_db);
    Kiir(r,r_db);
    ossz_db:= DbSzamol(r,r_db);
    RaktarErtek(r,r_db,ossz_ar);
    writeln;
    writeln('A raktár áruinak össz darabszáma: ',ossz_db);
    writeln('A raktár áruinak összértéke     : ',ossz_ar:6:2);
    readln;
  end.
```

A program futásának eredménye:

```
Az áruk darabszáma: 3
Az áruk adatai
azonosító : asztal
darabszáma: 3
ára       : 5000

azonosító : szek
darabszáma: 12
ára       : 1500

azonosító : szekreny
darabszáma: 4
ára       : 8000


A raktár adatainak megjelenítése

  Azonosító     Db        Ár
-------------------------------
      asztal     3      5000.0
        szek    12      1500.0
    szekreny     4      8000.0

A raktár áruinak össz darabszáma: 19
A raktár áruinak összértéke     : 65000.00
```