11. Objektum-orientált programozás

A 10 minta feladatban bemutatjuk az egyszerű pascal programból kiindulva az objektum-orientált programozás minden tulajdonságát.

Írjunk programot, amely két valós adatnak kiszámítja a mértani közepét! (*MINTA_01*)

```
{ 1. feladat: normal Pascal program }
program minta1;
var
    x,y,z:real;
begin
  writeln('1. feladat: normal Pascal program');
  x:= 1;
  y:= 2;
  z:=sqrt(x*y);
  writeln('Az ',x:5:1,' és',y:5:1,' mértani közepe: ',z:6:2);
  readln;
end.
```

A program futásának eredménye:

```
1. feladat: normal Pascal program
Az   1.0 és  2.0 mértani közepe:   1.41
```

Írjunk programot, amely **rekord** használatával számítja ki két valós szám mértani közepét! (*MINTA_02*)

```
{ 2. feladat: record használata }
program minta2;
type
   adat = record
      x,y,z :real;
     end;
var
    w: adat;
begin
  writeln('2. feladat: record használata  ');
  w.x:= 1;
  w.y:= 2;
  w.z:=sqrt(w.x*w.y);
  writeln('Az ',w.x:5:1,' és',w.y:5:1,' mértani közepe: ',w.z:6:2);
  readln;
end.
```

A program futásának eredménye:

```
2. feladat: record használata
Az   1.0 és  2.0 mértani közepe:   1.41
```

Írjunk objektum-orientált programot, amely két valós szám mértani közepét számítja ki modul használatával! (\MINTA_03)

```
unit minta3u;
INTERFACE
type
   adat = object
    protected
      x,y,z :real;
    public
     procedure init(x1,y1:real);
     procedure szamol;
     procedure kiir;
     end;
IMPLEMENTATION
  procedure adat.init(x1,y1:real);
  begin
     x:= x1;
     y:= y1;
     z:= 0;
  end;
  procedure adat.szamol;
  begin
     z:=sqrt(x*y);
  end;
  procedure adat.kiir;
  begin
    writeln('Az ',x:5:1,' és',y:5:1,' mértani közepe: ',z:6:2);
  end;
begin
end.

{ 3. feladat: objektum használata }
program minta3pr;
uses minta3u;
var
    w: adat;
begin
  writeln('3. feladat: objektum használata ');
  w.init(1,2);
  w.szamol;
  w.kiir;
  readln;
end.
```

A program futásának eredménye:

```
3. feladat: objektum használata
Az   1.0 és  2.0 mértani közepe:   1.41
```

Módosítsuk a *MINTA_03* könyvtárban lévő programot, hogy a főprogram dinamikus objektumpéldányt használjon! (*MINTA_04*)

```
unit minta4u;
INTERFACE
type
   padat = ^adat;
   adat = object
    protected
      x,y,z :real;
    public
     procedure init(x1,y1:real);
     procedure szamol;
     procedure kiir;
     end;
IMPLEMENTATION
  procedure adat.init(x1,y1:real);
  begin
     x:= x1;
     y:= y1;
     z:= 0;
  end;
  procedure adat.szamol;
  begin
     z:=sqrt(x*y);
  end;
  procedure adat.kiir;
  begin
    writeln('Az ',x:5:1,' és',y:5:1,' mértani közepe: ',z:6:2);
  end;
begin
end.
{ 4. feladat: dinamikus objektumpéldány }
program minta4pr;
uses minta4u;
var
    pw: padat;
begin
  writeln('4. feladat: dinamikus objektumpéldány');
  pw := new(padat);
  pw^.init(1,2);
  pw^.szamol;
  pw^.kiir;
  dispose(pw);
  readln;
end.
```

A program futásának eredménye:

```
4. feladat: dinamikus objektumpéldány
Az   1.0 és  2.0 mértani közepe:   1.41
```

Módosítsuk a *MINTA_04* könyvtárban lévő programot, hogy az objektumnak adatmezői dinamikusak legyenek, főprogram dinamikus objektumpéldányt használjon! (*MINTA_05*)

```
unit minta5u;
INTERFACE
type
   preal = ^real;
   padat = ^adat;
   adat = object
    protected
      px,py,pz : preal;
    public
     constructor init(x1,y1:real);
     destructor Done;
     procedure szamol;
     procedure kiir;
     end;
IMPLEMENTATION
  constructor adat.init(x1,y1:real);
  begin
     px:= new(preal);
     py:= new(preal);
     pz:= new(preal);
     px^:= x1;
     py^:= y1;
     pz^:= 0;
  end;
  procedure adat.szamol;
  begin
     pz^:=sqrt(px^ * py^);
  end;
  procedure adat.kiir;
  begin
    writeln('Az ',px^:5:1,' és',py^:5:1,' mértani közepe: ',pz^:6:2);
  end;
  destructor adat.Done;
  begin
   dispose(px);
   dispose(py);
   dispose(pz);
  end;
begin
end.
```

```
{ 5. feladat: dinamikus objektumpéldány
    dinamikus adatmezők, new kibővitett hívása }

program minta5pr;
uses minta5u;
var
    pw: padat;
begin
  writeln('5. feladat: dinamikus adatmezők, new kibővitett hívása');
  pw := new(padat,init(1,2));
  pw^.szamol;
  pw^.kiir;
  dispose(pw,Done);
  readln;
end.
```

A program futásának eredménye:

```
5. feladat: dinamikus adatmezők, new kibővitett hívása
Az   1.0 és  2.0 mértani közepe:   1.41
```

Módosítsuk a *MINTA_05* könyvtárban tárolt programot, hogy a *badat* objektumot az *adat* objektumból származtassuk és a *csokken* metódusa az *x* és *y* adatmezőit felére csökkentse! (*MINTA_06*)

```
unit minta6u1;
INTERFACE
type
   preal = ^real;
   padat = ^adat;
   adat = object
    protected
      px,py,pz : preal;
    public
     constructor init(x1,y1:real);
     destructor Done;
     procedure szamol;
     procedure kiir;
     end;
IMPLEMENTATION
  constructor adat.init(x1,y1:real);
  begin
     px:= new(preal);
     py:= new(preal);
     pz:= new(preal);
     px^:= x1;
     py^:= y1;
     pz^:= 0;
  end;
```

```
  procedure adat.szamol;
  begin
     pz^:=sqrt(px^ * py^);
  end;
  procedure adat.kiir;
  begin
    writeln('Az ',px^:5:1,' és',py^:5:1,' mértani közepe: ',pz^:6:2);
  end;
  destructor adat.Done;
  begin
   dispose(px);
   dispose(py);
   dispose(pz);
  end;
begin
end.

unit minta6u2;
INTERFACE
uses minta6u1;
type
    pbadat = ^badat;
    badat = object(adat)
    public
     procedure csokkent;
    end;
IMPLEMENTATION
    procedure badat.csokkent;
    begin
        px^:= px^ /2;
        py^:= py^/2;
    end;
begin
end.

{ 6. feladat: orokoles }
program minta6pr;
uses minta6u1, minta6u2;
var
  pw: padat;
  pw1: pbadat;
begin
  writeln('6. feladat: öröklődés ');
  writeln('Az ős objektum');
  pw := new(pbadat,init(1,2));
  pw^.szamol;
  pw^.kiir;
  writeln;
  writeln('A származtatott objektum');
  pw1 := new(pbadat,init(1,2));
  pw1^.szamol;
  pw1^.kiir;
  writeln('Felére csökkentett adattal');
```

```
  pw1^.csokkent;
  pw1^.szamol;
  pw1^.kiir;
  dispose(pw,Done); dispose(pw1,Done);
  readln;
end.
```

A program futásának eredménye:

```
6. feladat: öröklődés
Az ős objektum
Az   1.0 és  2.0 mértani közepe:   1.41

A származtatott objektum
Az   1.0 és  2.0 mértani közepe:   1.41
Felére csökkentett adattal
Az   0.5 és  1.0 mértani közepe:   0.71
```

Módosítsuk a *\MINTA_06* könyvtárban tárolt programot, úgy hogy a *badat* objektumnak legyen egy *Szamol* metódusa, amely az ős objektum azonos nevű metódusát felülbírálja, a feladata számtani közép számítása! (*\MINTA_07*)

```
unit minta7u1;
INTtERFACE
type
   preal = ^real;
   padat = ^adat;
   adat = object
   protected
      px,py,pz : preal;
   public
     constructor init(x1,y1:real);
     destructor Done;
     procedure szamol;
     procedure kiir;
     end;
IMPLEMENTATION
  constructor adat.init(x1,y1:real);
  begin
     px:= new(preal);
     py:= new(preal);
     pz:= new(preal);
     px^:= x1;
     py^:= y1;
     pz^:= 0;
  end;
  procedure adat.szamol;
  begin
     pz^:=sqrt(px^ * py^);
     writeln('Mértani közép: ',pz^:5:2);
  end;
```

```
  procedure adat.kiir;
  begin
    writeln('Az ',px^:5:1,' és',py^:5:1,' mértani közepe: ',pz^:6:2);
  end;
  destructor adat.Done;
  begin
   dispose(px);
   dispose(py);
   dispose(pz);
  end;
begin
end.
unit minta7u2;
INTERFACE
uses minta7u1;
type
    pbadat = ^badat;
    badat = object(adat)
   public
    procedure szamol;
    procedure csokkent;
    end;
IMPLEMENTATION
    procedure badat.szamol;
    begin
       pz^ :=(px^+py^)/2;
       writeln('Számtani közép: ',pz^:5:2);
    end;
    procedure badat.csokkent;
    begin
        px^:= px^ /2;
        py^:= py^/2;
    end;
begin
end.

{ 7. feladat: Szamol metódusának felülbirálása }
program minta7pr;
uses minta7u1, minta7u2;
var
    pw: padat;
    pw1: pbadat;
begin
  writeln('7. feladat: az ős Szamol metódusának felülbírálása ');
  writeln('Az ős objektum');
  pw:=new(padat,init(1,2));
  pw^.szamol;
  writeln;
  write('A származtatott objektum ');
  writeln('Szamol metódusa számtani közepet számol ');
  pw1 := new(pbadat,init(1,2));
  pw1^.szamol;
```

```
  dispose(pw,Done);
  dispose(pw1,Done);
  readln;
end.
```

A program futásának eredménye:

```
7. feladat: az ős Szamol metódusának felülbírálása
Az ős objektum
Mértani közép:  1.41

A származtatott objektum Szamol metódusa számtani közepet számol
Számtani közép:  1.50
```

Módosítsuk a *\MINTA_07* könyvtárban tárolt programot, úgy hogy az adat objektum *vegrehajt* metódusa hívja a saját *szamol* metódusát és ezt a metódust hívja a főprogramban a két objektumpéldány! Mit tapasztalunk? (*\MINTA_08*)

```
unit minta8u1;
INTERFACE
type
   preal = ^real;
   padat = ^adat;
   adat = object
   protected
      px,py,pz : preal;
   public
     constructor init(x1,y1:real);
     destructor Done;
     procedure szamol;
     procedure vegrehajt;
     procedure kiir;
     end;
IMPLEMENTATION
  constructor adat.init(x1,y1:real);
  begin
     px:= new(preal);
     py:= new(preal);
     pz:= new(preal);
     px^:= x1;
     py^:= y1;
     pz^:= 0;
  end;
  procedure adat.szamol;
  begin
     pz^:=sqrt(px^ * py^);
     writeln('Mértani közép: ',pz^:5:2);
  end;
```

```
  procedure adat.vegrehajt;
  begin
    szamol;
  end;

  procedure adat.kiir;
  begtin
    writeln('Az ',px^:5:1,' és',py^:5:1,' mértani közepe: ',pz^:6:2);
  end;
  destructor adat.Done;
  begin
   dispose(px);
   dispose(py);
   dispose(pz);
  end;
begin
end.

unit minta8u2;
INTERFACE
uses minta8u1;
type
    pbadat = ^badat;
    badat = object(adat)
    public
      procedure szamol;
      procedure csokkent;
    end;
IMPLEMENTATION
    procedure badat.szamol;
    begin
       pz^ :=(px^+py^)/2;
       writeln('Számtani közép: ',pz^:5:2);
    end;
    procedure badat.csokkent;
    begin
        px^:= px^ /2;
        py^:= py^/2;
    end;
begin
end.

{ 8. feladat: Szamol metódus bezáródása }
program minta08;
uses minta8u1, minta8u2;
var
    pw: padat;
    pw1: pbadat;
begin
  writeln('8. feladat: Szamol metódus bezáródása');
  writeln('Az ős objektum');
  pw:=new(padat,init(1,2));
  pw^.vegrehajt;
```

```
  writeln;
  writeln('A származtatott objektum');
  write('Metódus bezáródása: ');
  writeln(' az ős objektum Szamol metódusa hivódik meg ');
  pw1 := new(pbadat,init(1,2));
  pw1^.vegrehajt;
  dispose(pw,Done);
  dispose(pw1,Done);
  readln;
end.
```

A program futásának eredménye:

```
8. feladat: Szamol metódus bezáródása
Az ős objektum
Mértani közép:  1.41

A származtatott objektum
Metódus bezáródása:  az ős objektum Szamol metódusa hivódik meg
Mértani közép:  1.41
```

Módosítsuk a *MINTA_08* könyvtárban lévő programot, úgy hogy az *adat* objektum *szamol* metódusa és a *badat* objektum *szamol* metódusa legyen **virtual**. Mit tapasztalunk? (*MINTA_09*)

```
unit minta9u1;
INTERFACE
type
   preal = ^real;
   padat = ^adat;
   adat = object
   protected
      px,py,pz : preal;
   public
     constructor init(x1,y1:real);
     destructor Done;
     procedure szamol; virtual;
     procedure vegrehajt;
     procedure kiir;
     end;
IMPLEMENTATION
  constructor adat.init(x1,y1:real);
  begin
     px:= new(preal);
     py:= new(preal);
     pz:= new(preal);
     px^:= x1;
     py^:= y1;
     pz^:= 0;
  end;
```

```
  procedure adat.szamol;
  begin
     pz^:=sqrt(px^ * py^);
     writeln('Mértani közép: ',pz^:5:2);
  end;

  procedure adat.vegrehajt;
  begin
    szamol;
  end;
  procedure adat.kiir;
  begin
    writeln('Az ',px^:5:1,' és',py^:5:1,' mértani közepe: ',pz^:6:2);
  end;
  destructor adat.Done;
  begin
   dispose(px);
   dispose(py);
   dispose(pz);
  end;
begin
end.
```

```
unit minta9u2;
INTERFACE
uses minta9u1;
type
    pbadat = ^badat;
    badat = object(adat)
    public
      procedure szamol; virtual;
      procedure csokkent;
    end;
IMPLEMENTATION
    procedure badat.szamol;
    begin
       pz^ :=(px^+py^)/2;
       writeln('Számtani közép: ',pz^:5:2);
    end;
    procedure badat.csokkent;
    begin
        px^:= px^ /2;
        py^:= py^/2;
    end;
begin
end.
```

```
{ 9. feladat: Szamol metódus virtuális, bezáródás megszűnik }
program minta09;
uses minta9u1, minta9u2;
var
    pw: padat;
    pw1: pbadat;
begin
  writeln('9. feladat: Szamol metódus virtuális, bezáródás megszűnik');
  writeln('Az ős objektum');
  pw:=new(padat,init(1,2));
  pw^.vegrehajt;
  writeln;
  writeln('A származtatott objektum');
  writeln('A virtuális Metódus bezáródása megszűnik ');
  pw1 := new(pbadat,init(1,2));
  pw1^.vegrehajt;
  dispose(pw,Done);
  dispose(pw1,Done);
  readln;
end.
```

A program futásának eredménye:

```
9. feladat: Szamol metódus virtuális, bezáródás megszűnik
Az ős objektum
Mértani közép:  1.41

A származtatott objektum
A virtuális Metódus bezáródása megszűnik
Számtani közép:  1.50
```

Írjunk objektum-orientált programot, amely komplex számokkal elvégzi a négy alapműveletet! (KOMPLEX*kompl_o*)

```
(* kompl_o.pas *)
program kompl_o;
 type
   Komplex = object
   protected
       re,im: real;
   public
  procedure Init(re0, im0: real);
  end;
  procedure Komplex.Init( re0,im0:real);
  begin
    re:= re0;
    im:= im0;
  end;
```

```
type
    KomplexMuv = object
    private
        x,y,ered: Komplex;
    public
    procedure Init(re0,im0,re1,im1:real);
    procedure Osszead;
    procedure Kivon;
    procedure Szoroz;
    procedure Oszt;
    procedure Kiir(m:char);
end;
procedure KomplexMuv.Init(re0,im0,re1,im1:real);
begin
   x.Init(re0,im0);
   y.Init(re1,im1);
   ered.Init(0,0);
end;
procedure KomplexMuv.Osszead;
begin
   ered.re :=  x.re + y.re;
   ered.im :=  x.im + y.im;
end;
procedure KomplexMuv.Kivon;
begin
   ered.re :=  x.re - y.re;
   ered.im :=  x.im - y.im;
end;
procedure KomplexMuv.Szoroz;
begin
   ered.re := x.re * y.re - x.im * y.im;
   ered.im := x.re * y.im + x.im * y.re;
end;
procedure KomplexMuv.Oszt;
begin
   ered.re:=((x.re*y.re)+(x.im*y.im))/
            ((y.re*y.re)+(y.im*y.im));
   ered.im:=((x.im*y.re)+(-x.re*y.im))/
            ((y.re*y.re)+(y.im*y.im));
end;
procedure KomplexMuv.Kiir(m: char);
begin
  write(x.re:3:1,'+',x.im:3:1,'i');
  write(' ',m,' ');
  write(y.re:3:1,'+',y.im:3:1,'i = ');
  writeln(ered.re:5:1,'+',ered.im:5:1,'i');
end;
```

```
var
   re1,im1,re2,im2: real;
   a: KomplexMuv;
begin

  writeln;
  write('1. komplex szám valós része   : ');readln(re1);
  write('                képzetes része: ');readln(im1);

  write('2. komplex szám valós része   : ');readln(re2);
  write('                képzetes része: ');readln(im2);

  a.Init(re1,im1,re2,im2);
  writeln;
  writeln('Műveletek');
  writeln('Az összeadás eredménye: ');
  a.Osszead;
  a.Kiir('+');
  writeln('A kivonás eredménye: ');
  a.Kivon;
  a.Kiir('-');
  writeln('A szorzás eredménye: ');
  a.Szoroz;
  a.Kiir('*');
  writeln('Az osztás eredménye: ');
  a.Oszt;
  a.Kiir('/');
  readln;
end.
```

A program futásának eredménye:

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

Műveletek
Az összeadás eredménye:
1.0+1.0i + 2.0+2.0i =   3.0+  3.0i
A kivonás eredménye:
1.0+1.0i - 2.0+2.0i =  -1.0+ -1.0i
A szorzás eredménye:
1.0+1.0i * 2.0+2.0i =   0.0+  4.0i
Az osztás eredménye:
1.0+1.0i / 2.0+2.0i =   0.5+  0.0i
```

Írjunk objektum-orientált, menüvezérelt programot, amely komplex számokkal elvégzi a négy alapműveletet! (*KOMPLEX \komplo_m*)

```
(* komplo_m.pas *)
program komplo_m;
 type
   Komplex = object
   protected
       re,im: real;
   public
    procedure Init(re0, im0: real);
  end;
  procedure Komplex.Init( re0,im0:real);
  begin
    re:= re0;
    im:= im0;
  end;
  type
      KomplexMuv = object
      protected
          x,y,ered: Komplex;
          muv_jel : char;
      public
      procedure Init(re0,im0,re1,im1:real;muv_jel0:char);
      procedure Szamol;
      procedure Kiir;
      function Ered_re:real;
      function Ered_im:real;
  end;
  procedure KomplexMuv.Init(re0,im0,re1,im1:real;muv_jel0:char);
  begin
     x.Init(re0,im0);
     y.Init(re1,im1);
     ered.Init(0,0);
     muv_jel:=muv_jel0;
  end;
  procedure KomplexMuv.Szamol;
  begin
    case muv_jel of
    '+': begin
           ered.re :=  x.re + y.re;
           ered.im :=  x.im + y.im;
         end;
    '-': begin
           ered.re :=  x.re - y.re;
           ered.im :=  x.im - y.im;
         end;
    '*': begin
           ered.re := x.re * y.re - x.im * y.im;
           ered.im := x.re * y.im + x.im * y.re;
         end;
```

```
     '/':begin
           ered.re:=((x.re*y.re)+(x.im*y.im))/
                    ((y.re*y.re)+(y.im*y.im));
           ered.im:=((x.im*y.re)+(-x.re*y.im))/
                    ((y.re*y.re)+(y.im*y.im));
         end;
     end;
  end;
  procedure KomplexMuv.Kiir;
  begin
    writeln;
    write('(',x.re:6:1,'+',x.im:6:1,'i)');
    write(' ',muv_jel,' ');
    write('(',y.re:6:1,'+',y.im:6:1,'i) = ');
    writeln('(',Ered.re:6:1,'+',Ered.im:6:1,'i)');
  end;
  function KomplexMuv.Ered_re:real;
  begin
    Ered_re:= ered.re;
  end;
  function KomplexMuv.Ered_im:real;
  begin
    Ered_im:= ered.im;
  end;
var
   re1,im1,re2,im2:real;
   m: char;
   a: KomplexMuv;
begin

    write('1. komplex szám valós része   : '); readln(re1);
    write('1. komplex szám képzetes része: '); readln(im1);
    writeln;
    write('2. komplex szám valós része   : '); readln(re2);
    write('2. komplex szám képzetes része: '); readln(im2);
   repeat
     writeln;
     writeln('Komplex műveletek');
     writeln('a: új adat beolvasása');
     writeln('+: összeadás');
     writeln('-: kivonás');
     writeln('*: szorzás');
     writeln('/: osztás');
     writeln('k: kilépés');
     writeln;
     repeat
       write('művelet: '); readln(m);
     until m in['a','A','+','-','*','/','k','K'];
     case m of
     '+','-','*','/': begin
                        a.Init(re1,im1,re2,im2,m);
                        a.Szamol;
                        a.Kiir;
```

```
                        writeln;
                        writeln('További művelet ''Enter'' leütése');
                       readln;
                     end;
     'a','A':
         begin
          write('1. komplex szám valós része   : '); readln(re1);
          write('1. komplex szám képzetes része: '); readln(im1);
          writeln;
          write('2. komplex szám valós része   : '); readln(re2);
          write('2. komplex szám képzetes része: '); readln(im2);
         end;
     end;
   until (m = 'k') or (m = 'K');
end.
```

A program futásának eredménye:

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

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

Komplex műveletek
a: új adat beolvasása
+: összeadás
-: kivonás
*: szorzás
/: osztás
k: kilépés

művelet: +

(   1.0+   1.0i) + (   2.0+   2.0i) = (   3.0+   3.0i)

További művelet 'Enter' leütése

Komplex műveletek
a: új adat beolvasása
+: összeadás
-: kivonás
*: szorzás
/: osztás
k: kilépés

művelet: *

(   1.0+   1.0i) * (   2.0+   2.0i) = (   0.0+   4.0i)

További művelet 'Enter' leütése
```

```
Komplex műveletek
a: új adat beolvasása
+: összeadás
-: kivonás
*: szorzás
/: osztás
k: kilépés

művelet: /

(   1.0+   1.0i) / (   2.0+   2.0i) = (   0.5+   0.0i)

További művelet 'Enter' leütése

Komplex műveletek
a: új adat beolvasása
+: összeadás
-: kivonás
*: szorzás
/: osztás
k: kilépés
8. feladat: Szamol metódus bezáródása
```

Módosítsuk a *komplo_m* programot, úgy hogy az objektumokat helyezzük *kompl_u* könyvtárba és próbáljuk ki a *kompl_pr* főprogramban! (*KOMPLEX\ kompl_u, kompl_pr*)

```
(* kompl_u.pas *)
unit kompl_u;
INTERFACE
 type
   Komplex = object
   protected
       re,im: real;
   public
     procedure Init(re0, im0: real);
  end;
  type
    KomplexMuv = object
    protected
      x,y,ered: Komplex;
      muv_jel : char;
    public
      procedure Init(re0,im0,re1,im1:real;muv_jel0:char);
      procedure Szamol;
      procedure Kiir;
      function Ered_re:real;
      function Ered_im:real;
  end;
```

```
IMPLEMENTATION
  procedure Komplex.Init( re0,im0:real);
  begin
    re:= re0;
    im:= im0;
  end;
  procedure KomplexMuv.Init(re0,im0,re1,im1:real;muv_jel0:char);
  begin
     x.Init(re0,im0);
     y.Init(re1,im1);
     ered.Init(0,0);
     muv_jel:=muv_jel0;
  end;
  procedure KomplexMuv.Szamol;
  begin
    case muv_jel of
    '+': begin
           ered.re :=  x.re + y.re;
           ered.im :=  x.im + y.im;
         end;
    '-': begin
           ered.re :=  x.re - y.re;
           ered.im :=  x.im - y.im;
         end;
    '*': begin
           ered.re := x.re * y.re - x.im * y.im;
           ered.im := x.re * y.im + x.im * y.re;
         end;
     '/':begin
           ered.re:=((x.re*y.re)+(x.im*y.im))/
                    ((y.re*y.re)+(y.im*y.im));
           ered.im:=((x.im*y.re)+(-x.re*y.im))/
                    ((y.re*y.re)+(y.im*y.im));
         end;
     end;
  end;
  procedure KomplexMuv.Kiir;
  begin
    writeln;
    write('(',x.re:6:1,'+',x.im:6:1,'i)');
    write(' ',muv_jel,' ');
    write('(',y.re:6:1,'+',y.im:6:1,'i) = ');
    writeln('(',Ered.re:6:1,'+',Ered.im:6:1,'i)');
  end;
  function KomplexMuv.Ered_re:real;
  begin
    Ered_re:= ered.re;
  end;
  function KomplexMuv.Ered_im:real;
  begin
    Ered_im:= ered.im;
  end;
end.
```

```
(* kompl_pr.pas *)
program kompl_pr;
uses kompl_u;
var
   re1,im1,re2,im2:real;
   m: char;
   a: KomplexMuv;
begin

    write('1. komplex szám valós része   : '); readln(re1);
    write('1. komplex szám képzetes része: '); readln(im1);
    writeln;
    write('2. komplex szám valós része   : '); readln(re2);
    write('2. komplex szám képzetes része: '); readln(im2);
   repeat

     writeln;
     writeln('Komplex műveletek');
     writeln('a: új adat beolvasása');
     writeln('+: összeadás');
     writeln('-: kivonás');
     writeln('*: szorzás');
     writeln('/: osztás');
     writeln('k: kilépés');
     writeln;

     repeat
       write('művelet: '); readln(m);
     until m in['a','A','+','-','*','/','k','K'];
     case m of
     '+','-','*','/': begin
                        a.Init(re1,im1,re2,im2,m);
                        a.Szamol;
                        a.Kiir;
                        writeln;
                        writeln('További művelet ''Enter'' leütése');
                       readln;
                      end;
     'a','A':
         begin
          write('1. komplex szám valós része   : '); readln(re1);
          write('1. komplex szám képzetes része: '); readln(im1);
          writeln;
          write('2. komplex szám valós része   : '); readln(re2);
          write('2. komplex szám képzetes része: '); readln(im2);
         end;
     end;
   until (m = 'k') or (m = 'K');
end.
```

A program futásának eredménye:

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

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

Komplex műveletek
a: új adat beolvasása
+: összeadás
-: kivonás
*: szorzás
/: osztás
k: kilépés

művelet: +

(   2.0+   2.0i) + (   3.0+   3.0i) = (   5.0+   5.0i)

További művelet 'Enter' leütése

Komplex műveletek
a: új adat beolvasása
+: összeadás
-: kivonás
*: szorzás
/: osztás
k: kilépés

művelet: a
1. komplex szám valós része   : 1
1. komplex szám képzetes része: 2

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

Komplex műveletek
a: új adat beolvasása
+: összeadás
-: kivonás
*: szorzás
/: osztás
k: kilépés

művelet: -

(   1.0+   2.0i) - (   3.0+   4.0i) = (  -2.0+  -2.0i)

További művelet 'Enter' leütése
```

```
Komplex műveletek
a: új adat beolvasása
+: összeadás
-: kivonás
*: szorzás
/: osztás
k: kilépés

művelet: k
```

Írjunk objektum-orientált programot tanulók nevét névsorban rendezi, a születési dátumukat pedig csökkenő sorrendbe rendezi! A főprogramban statikus objektum-példányt használjunk! (*RENDEZ\rendezp1*)

```
(* rendezp1.pas *)
program rendezp1;
type
        str20 = string[20];
        tanulo = object
        protected
                   nev : str20;
                   kora: integer;
        public
        procedure Init(nev0:str20; kora0:integer);
        end;
        procedure tanulo.Init(nev0:str20; kora0:integer);
        begin
           nev:=nev0;
           kora:=kora0;
        end;
type    tomb = array[1..20] of tanulo;
        osztaly = object
        protected
            t : tomb;
            db: integer;
        public
        procedure Init;
        procedure Abc_rendez;
        procedure Kora_rendez;
        procedure Kiir;
        end;
        procedure osztaly.Init;
        var
           i:integer;
           nev0 : str20;
           kora0: integer;
        begin
          repeat
            write('A tanulók száma: '); readln(db);
          until db <= 20;
```

```
   for i:=1 to db do
   begin
       write('Neve         : '); readln(nev0);
       write('Születési éve: '); readln(kora0);
       writeln;
      t[i].Init(nev0,kora0);
  end;
end;
procedure osztaly.Abc_rendez;
var
   i,j: integer;
   s  : tanulo;
begin
   for i:=1 to db-1 do
     for j:=i+1 to db do
     begin
        if(t[i].nev > t[j].nev) then
            begin
               s:=t[i];
               t[i] := t[j];
               t[j]:=s;
            end;
    end;
end;
procedure osztaly.Kora_rendez;
var
   i,j: integer;
   s  : tanulo;
begin
   for i:=1 to db-1 do
     for j:=i+1 to db do
     begin
        if(t[i].kora < t[j].kora) then
          begin
               s:=t[i];
               t[i]:= t[j];
               t[j]:= s;
           end;
    end;
end;
procedure osztaly.kiir;
var
    i,n:integer;
begin
   for i:=1 to db do
   begin
      n:= length(t[i].nev);
      writeln(t[i].nev,' ':20-n, t[i].kora:3);
   end;
end;
```

```
var
   evf: osztaly;
begin
   evf.Init;
   writeln; writeln('Alapadatok'); evf.kiir;
   evf.Abc_rendez;
   writeln; writeln('Név szerint rendezett adatok');
   evf.kiir; writeln;
   evf.Kora_rendez;
   writeln('Születési év szerint rendezett adatok');
   evf.Kiir;
   readln;
end.
```

A program futásának eredménye:

```
A tanulók száma: 3
Neve         : Kiss Istvan
Születési éve: 1989

Neve         : Nagy Ilona
Születési éve: 1992

Neve         : Toth Katalin
Születési éve: 1972

Alapadatok
Kiss Istvan         1989
Nagy Ilona          1992
Toth Katalin        1972

Név szerint rendezett adatok
Kiss Istvan         1989
Nagy Ilona          1992
Toth Katalin        1972

Születési év szerint rendezett adatok
Nagy Ilona          1992
Kiss Istvan         1989
Toth Katalin        1972
```

Írjunk objektum-orientált programot tanulók nevét névsorban rendezi, a születési dátumukat pedig csökkenő sorrendbe rendezi! Az objektumban dinamikus adatmezőt és a főprogramban dinamikus objektum-példányt használjunk! (RENDEZ*rendezp2*)

```
(* rendezp2.pas *)
program rendezp2;
type
        str20 = string[20];
        Ptanulo = ^tanulo;
        tanulo = object
```

```
        protected
                  nev : str20;
                  kora: integer;
        public
        constructor Init(nev0:str20; kora0:integer);
        end;
        constructor tanulo.Init(nev0:str20; kora0:integer);
        begin
           nev:=nev0;
           kora:=kora0;
        end;
type
        Posztaly = ^osztaly;
        osztaly = object
        protected
            t : array[1..20] of Ptanulo;
            db: integer;
        public
        constructor Init;
        destructor Done;
        procedure Abc_rendez;
        procedure Kora_rendez;
        procedure Kiir;
        end;
        constructor osztaly.Init;
        var
           i:integer;
           nev0 : str20;
           kora0: integer;
        begin
          repeat
            write('A tanulók száma: '); readln(db);
          until db <= 20;
          for i:=1 to db do
          begin
              write('Neve         : '); readln(nev0);
              write('Születési éve: '); readln(kora0);
             t[i]:=new(Ptanulo,Init(nev0,kora0));
         end;
       end;
       destructor osztaly.Done;
       var
          i:integer;
       begin
          for i:=1 to db do
           dispose(t[i]);
       end;
       procedure osztaly.Abc_rendez;
       var
          i,j: integer;
          s  : str20;
          k  : integer;
```

```
      begin
         for i:=1 to db-1 do
           for j:=i+1 to db do
           begin
              if(t[i]^.nev > t[j]^.nev) then
                  begin
                     s:=t[i]^.nev; k:=t[i]^.kora;
                     t[i]^.nev := t[j]^.nev;
                     t[i]^.kora:= t[j]^.kora;
                     t[j]^.nev:=s;t[j]^.kora:=k;
                  end;
          end;
      end;
      procedure osztaly.Kora_rendez;
      var
         i,j: integer;
         s  : str20;
         k  : integer;
      begin
         for i:=1 to db-1 do
           for j:=i+1 to db do
           begin
              if(t[i]^.kora < t[j]^.kora) then
                begin
                     s:=t[i]^.nev; k:=t[i]^.kora;
                     t[i]^.nev := t[j]^.nev;
                     t[i]^.kora:= t[j]^.kora;
                     t[j]^.nev:=s;t[j]^.kora:=k;
                 end;
          end;
      end;
      procedure osztaly.kiir;
      var
          i,n:integer;
      begin
         for i:=1 to db do
         begin
            n:= length(t[i]^.nev);
            writeln(t[i]^.nev,' ':20-n, t[i]^.kora:3);
         end;
      end;
var
   pevf: Posztaly;
begin
   pevf:=new(Posztaly,Init);
   writeln; writeln('Alapadatok');
   pevf^.kiir;
   pevf^.Abc_rendez;
   writeln; writeln('Név szerint rendezett adatok');
   pevf^.kiir; writeln;
   pevf^.Kora_rendez;
   writeln('Születési év szerint rendezett adatok');
   pevf^.Kiir;
```

```
   dispose(pevf,Done);
   readln;
end.
```

A program futásának eredménye:

```
A tanulók száma: 3
Neve         : Toth Katalin
Születési éve: 1972
Neve         : Nagy Istvan
Születési éve: 1989
Neve         : Kiss Nora
Születési éve: 1967

Alapadatok
Toth Katalin        1972
Nagy Istvan         1989
Kiss Nora           1967

Név szerint rendezett adatok
Kiss Nora           1967
Nagy Istvan         1989
Toth Katalin        1972

Születési év szerint rendezett adatok
Nagy Istvan         1989
Toth Katalin        1972
Kiss Nora           1967
```

Írjunk objektum-orientált programot, amely 20 valós adatoknak összegét és átlagát képezi, megkeresi a legnagyobb és a legkisebb elemét!

Az objektumban statikus adatmezőket és a főprogramban statikus objektum-példányt használjunk! (*VEKTOR\vektor_s*)

```
(* vektor_s.pas *)
program vektor_s;
type
     rtip = object
     protected
        x: real;
     public
     procedure Init(x0:real);
     end;
     procedure rtip.Init(x0:real);
     begin
       x:= x0;
     end;
```

```
type
  vector = object
  protected
     t:array[1..20] of rtip;
     db    : integer;
     szum  : real;
     atl   : real;
     max_elem: real;
     min_elem: real;
  public
  procedure Init;
  function Szumma: real;
  function Atlag: real;
  function Maximum:real;
  function Minimum:real;
  end;
  procedure vector.Init;
  var i :integer;
      x0:real;
  begin
      repeat
        write('Elemek száma: '); readln(db);
     until db<=20;
     for i:=1 to db do
     begin
      write(i:2,'.elem: '); readln(x0);
      t[i].Init(x0);
     end;
  end;
  function vector.Szumma:real;
  var i:integer;
  begin
    szum:=0;
    for i:=1 to db do
      szum:=szum+t[i].x;
    Szumma:=szum;
  end;
  function vector.Atlag:real;
  begin
    Atlag:=Szumma/db;
  end;
  function vector.Maximum:real;
  var i:integer;
      m: real;
  begin
     max_elem:= t[1].x;
     for i:=2 to db do
      if max_elem < t[i].x then begin
                                  m:=max_elem;
                                  max_elem:=t[i].x;
                                  t[i].x:= m;
                                end;
     Maximum:=max_elem;
  end;
```

```
     function vector.Minimum:real;
     var i:integer;
         m: real;
     begin
        min_elem:= t[1].x;
        for i:=2 to db do
         if min_elem > t[i].x then begin
                                     m:=min_elem;
                                     min_elem:=t[i].x;
                                     t[i].x:= m;
                                   end;
       Minimum:=min_elem;
     end;
var
   v:vector;
begin
   v.Init;
   writeln('Szumma: ',v.Szumma:5:1);
   writeln('Átlag : ',v.Atlag:5:1);
   writeln('Maximális elem: ',v.Maximum:5:1);
   writeln('Minimális elem: ',v.Minimum:5:1);
   readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 4
 1.elem: -2
 2.elem: 12
 3.elem: 3
 4.elem: -1
Szumma:  12.0
Átlag :   3.0
Maximális elem:  12.0
Minimális elem:  -2.0
```

Írjunk objektum-orientált programot, amely 20 valós adatoknak összegét és átlagát képezi, megkeresi a legnagyobb és a legkisebb elemét!

Az objektumban dinamikus adatmezőt és a főprogramban dinamikus objektum-példányt használjunk! (*VEKTOR\vektor_p*)

```
(* vektor_p.pas *)
program vektor_p;
uses crt;
type
     Prtip = ^rtip;
     rtip = object
        x: real;
     constructor Init(x0:real);
     end;
```

```
     constructor rtip.Init(x0:real);
     begin
       x:= x0;
     end;
type
     Pvector = ^vector;
     vector = object
        t:array[1..20] of Prtip;
        db    : integer;
        szum  : real;
        atl   : real;
        max_elem: real;
        min_elem: real;
     constructor Init;
     destructor Done;
     function Szumma: real;
     function Atlag: real;
     function Maximum:real;
     function Minimum:real;
     end;
     constructor vector.Init;
     var i :integer;
         x0:real;
     begin
        repeat
           write('Elemek száma: '); readln(db);
        until db<=20;
        for i:=1 to db do
        begin
         write(i:2,'.elem: '); readln(x0);
         t[i]:=new(Prtip,Init(x0));
        end;
     end;
     destructor vector.Done;
     var i: integer;
     begin
       for i:=1 to db do
         dispose(t[i]);
     end;
     function vector.Szumma:real;
     var i:integer;
     begin
       szum:=0;
       for i:=1 to db do
         szum:=szum+t[i]^.x;
       Szumma:=szum;
     end;
     function vector.Atlag:real;
     var i:integer;
     begin
       atl:=0;
```

```
       for i:=1 to db do
         atl:= atl+t[i]^.x;
       atl:=atl/db;
       Atlag:=atl;
     end;
     function vector.MAximum:real;
     var i:integer;
         m: real;
     begin
        max_elem:= t[1]^.x;
        for i:=2 to db do
         if max_elem < t[i]^.x then begin
                                      m:=max_elem;
                                      max_elem:=t[i]^.x;
                                      t[i]^.x:= m;
                                    end;
        Maximum:=max_elem;
     end;
     function vector.Minimum:real;
     var i:integer;
         m: real;
     begin
        min_elem:= t[1]^.x;
        for i:=2 to db do
         if min_elem > t[i]^.x then begin
                                      m:=min_elem;
                                      min_elem:=t[i]^.x;
                                      t[i]^.x:= m;
                                    end;
       Minimum:=min_elem;
     end;
var
   pv:Pvector;
begin
   pv:=new(Pvector,Init);
   writeln('Szumma: ',pv^.Szumma:5:1);
   writeln('Átlag : ',pv^.Atlag:5:1);
   writeln('Maximális elem: ',pv^.Maximum:5:1);
   writeln('Minimális elem: ',pv^.Minimum:5:1);
   dispose(pv,Done);
   readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 4
 1.elem: 2
 2.elem: 4
 3.elem: 3
 4.elem: 1
Szumma:  10.0
Átlag :   2.5
Maximális elem:   4.0
Minimális elem:   1.0
```

Írjunk objektum-orientált programot, amely beolvas egy mondatot, felbontja szavakra, kiírja a szó hosszát, a magánhangzók és a mássalhangzók számát táblázatosan. (*MONDAT\mondat_o*)

```
(* mondat_o *)
{$R-}
program mondat_o;
type
       szotip = object
                  szo:string;
                  db:integer;
                  mgh_db:integer;
                  msh_db:integer;
                end;
       vizsgal = object
                  mondat: string;
                  szavak: array[1..50] of szotip;
                  szo_db: integer;
                procedure Init;
                procedure Szo_bont;
                procedure Kiir;
                procedure Statisztika;
                procedure Mindent_kiir;
                end;
      procedure vizsgal.Init;
      begin
          write('mondat: '); readln(mondat);
      end;
      procedure vizsgal.Szo_bont;
      var
          i,j,k:integer;
      begin
           j:=1; k:=0;
           for i:=1 to length(mondat) do
           begin
             if (mondat[i] <> ' ') and (mondat[i] <> '.')
                                  then
                                    begin
                                     k:=k+1;
                                     szavak[j].szo[k]:=mondat[i];
                                    end;
             if(mondat[i] = ' ') or (mondat[i] = '.') then
                                     begin
                                      szavak[j].szo[0]:=chr(k);
                                      k:=0; j:=j+1;
                                     end;
          end;
          szo_db:=j-1;
      end;
```

```
     procedure vizsgal.Kiir;
      var i:integer;
      begin
          for i:=1 to szo_db do
            writeln(szavak[i].szo);
      end;
      procedure vizsgal.Statisztika;
      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 szo_db do
           begin
              szavak[i].db:= length(szavak[i].szo);
              ma:=0; ms:=0;
              for j:=1 to szavak[i].db do
              begin
                if  szavak[i].szo[j] in mgh then ma:=ma+1;
                if  szavak[i].szo[j] in msh then ms:=ms+1;
                if  szavak[i].szo[j] in Nmgh then ma:=ma+1;
                if  szavak[i].szo[j] in Nmsh then ms:=ms+1;
              end;
              szavak[i].mgh_db:=ma;
              szavak[i].msh_db:=ms;
           end;
        end;
        procedure vizsgal.Mindent_kiir;
        var i:integer;
        begin
           writeln('szavak':10,'szó hossza':15,
                   'mgh száma':12,'msg száma':15);
           for i:=1 to szo_db do
            writeln(szavak[i].szo:10,szavak[i].db:10,
                    szavak[i].mgh_db:14,szavak[i].msh_db:15);
      end;
var
  v: vizsgal;
begin
   v.Init;
   v.Szo_bont;
   v.Kiir;
   v.Statisztika;
   v.Mindent_kiir;
   readln;
end.
```

A program futásának eredménye:

```
mondat: Pettyes ez a labda.
Pettyes
ez
a
labda
    szavak     szó hossza   mgh száma      msg száma
   Pettyes         7             2             5
        ez         2             1             1
         a         1             1             0
     labda         5             2             3
```

Írjunk objektum-orientált programot, amely a kör sugara alapján kiszámítja a kör kerületét és területét! A főprogramban statikus objektumpéldányt használjunk! (*KOR\oop2*)

```
program oop2;
type
      Kor = object
      protected
        r     : real;
        ter,ker : real;
      public
       procedure Init(r0:real);
       function terulet:real;
       function kerulet:real;
     end;
     procedure Kor.Init(r0:real);
     begin
       r       := r0;
     end;
     function Kor.terulet:real;
     begin
       ter:=r*r*pi;
       terulet:= ter;
     end;
     function Kor.kerulet:real;
     begin
       ker:=2*r*pi;
       kerulet:= ker;
     end;
var
   k1,k2: kor;
   sug: real;
begin
   write('Kör sugara: '); readln(sug);
   k1.Init(sug);
   writeln('Terület : ',k1.terulet:5:2);
   writeln('Kerület : ',k1.kerulet:5:2);
```

```
   writeln;
   k2.Init(1);
   writeln('Terület : ',k2.terulet:5:2);
   writeln('Kerület : ',k2.kerulet:5:2);
   readln;
end.
```

A program futásának eredménye:

```
Kör sugara: 3
Terület : 28.27
Kerület : 18.85

Terület :  3.14
Kerület :  6.28
```

Írjunk objektum-orientált programot, amely a kör sugara alapján kiszámítja a kör kerületét és területét! A főprogramban dinamikus objektumpéldányt használjunk! (*KOR\oop2d*)

```
program oop2d;
type
     pkor = ^kor;
     kor = object
     public
        r:real;
        ter,ker:real;
     public
       constructor init(r0:real);
       function kerulet:real;
       function terulet:real;
   end;
   constructor kor.init(r0:real);
   begin
      r:=r0;
   end;
   function kor.kerulet:real;
   begin
      ker:= 2*r*pi;
      kerulet:=ker;
   end;
   function kor.terulet:real;
   begin
      ter:=r*r*pi;
      terulet:=ter;
   end;
var
  pk1,pk2:pkor;
  t,k:real;
begin
    pk1 := new(pkor,init(1));
```

```
    writeln('Terület: ',pk1^.terulet:8:2);
    writeln('Kerület: ',pk1^.kerulet:8:2);
    writeln;
    pk2 := new(pkor,init(3));
    t:=pk2^.terulet;
    k:=pk2^.kerulet;
    writeln('Terület: ',pk2^.ter:8:2);
    writeln('Kerület: ',pk2^.ker:8:2);
    dispose(pk1);
    dispose(pk2);
    readln;
end.
```

A program futásának eredménye:

```
Terület:     3.14
Kerület:     6.28

Terület:    28.27
Kerület:    18.85
```

Írjunk objektum-orientált programot, amely a kör sugara alapján kiszámítja a kör kerületét és területét! Az objektumban dinamikus adatmezőket és a főprogramban dinamikus objektumpéldányt használjunk! (*KOR\oop2dd*)

```
(* oop2dd.pas *)
program oop2dd;
type
     pkor = ^kor;
     preal = ^real;
     kor = object
     public
        r:preal;
        ter,ker:preal;
     public
       constructor init(r0:real);
       destructor done;
       function kerulet:real;
       function terulet:real;
   end;
   constructor kor.init(r0:real);
   begin
      r := new(preal);
      ter := new(preal);
      ker := new(preal);
      r^:=r0;
   end;
   destructor kor.done;
   begin
      dispose(r);dispose(ter); dispose(ker);
   end;
```

```
   function kor.kerulet:real;
   begin
      ker^:= 2*r^*pi;
      kerulet:=ker^;
   end;
   function kor.terulet:real;
   begin
      ter^:=r^*r^*pi;
      terulet:=ter^;
   end;
var
  pk1,pk2:pkor;
  t,k:real;
begin
    pk1 := new(pkor,init(1));
    writeln('Terület: ',pk1^.terulet:8:2);
    writeln('Kerület: ',pk1^.kerulet:8:2);
    writeln;
    pk2 := new(pkor,init(3));
    t:=pk2^.terulet;
    k:=pk2^.kerulet;
    writeln('Terület: ',pk2^.ter^:8:2);
    writeln('Kerület: ',pk2^.ker^:8:2);
    dispose(pk1,done);
    dispose(pk2,done);
    readln;
end.
```

A program futásának eredménye:

```
Terület:     3.14
Kerület:     6.28

Terület:    28.27
Kerület:    18.85
```