9. Modulok

Írjunk modult (*komplexm*), amely alkalmas komplex számok között az összeadás, kivonás, szorzás és osztás műveletének végrehajtására! (*KOMPLEX*)

```
unit komplexm;
interface
 type
      Komplex = record
                re,im: real;
       end;
      procedure Szamol( x,y:Komplex; muv_jel:char; var ered:Komplex);
      procedure Kiir(x,y:Komplex;muv_jel:char;ered:Komplex);
implementation
  procedure Szamol( x,y:Komplex; muv_jel:char; var ered:Komplex);
  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 Kiir(x,y:Komplex;muv_jel:char;ered:Komplex);
  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;
end.
```

Írjunk menüvezérelt programot, amely használja a *komplexm* modult a komplex számok között az összeadás, kivonás, szorzás és osztás műveletének végrehajtására! (*KOMPLEX*)

```
program komplpr;
uses komplexm;
var
   x1,x2,ered0:Komplex;
   m          :char;
begin
    write('1. komplex szám valós része   : '); readln(x1.re);
    write('1. komplex szám képzetes része: '); readln(x1.im);
    writeln;
    write('2. komplex szám valós része   : '); readln(x2.re);
    write('2. komplex szám képzetes része: '); readln(x2.im);
   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
                        Szamol(x1,x2,m,ered0);
                        Kiir(x1,x2,m,ered0);
                        writeln;
                        writeln('További művelet ''Enter'' leütése');
                        readln;
                      end;
     'a','A':
         begin
          write('1. komplex szám valós része   : '); readln(x1.re);
          write('1. komplex szám képzetes része: '); readln(x1.im);
          write('2. komplex szám valós része   : '); readln(x2.re);
          write('2. komplex szám képzetes része: '); readln(x2.im);
         end;
     end;
   until (m = 'k') or (m = 'K');
   readln;
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) = (  -1.0+  -1.0i)

További művelet 'Enter' leütése


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

művelet: k
```

Írjunk modult (*szumma*), amely max. 50 elemű valós tömböt beolvas és az elemek összegét és átlagát függvényekkel és eljárással is kiszámítja! (*SZUMMA*)

```
unit szumma;
interface
const M = 50;
type
     tomb = array[1..M] of real;
      procedure olvas(var x:tomb; var db:integer);
      procedure szamol(x:tomb;db:integer;
                       var atlag,ossz:real);
      function osszeg(x:tomb; db:integer):real;
      function fatlag(x:tomb;db:integer):real;
implementation
    procedure olvas(var x:tomb; var db:integer);
    var
       i,kod:integer;
       szam : string;
    begin
       repeat
         write('elemek száma: '); readln(db);
       until (db>0) and (db<=M);
       for i:=1 to db do
       begin
        repeat
         write(i:2,'. adat: '); readln(szam);
         val(szam,x[i],kod);
        until kod = 0;
       end;
    end;
    procedure szamol(x:tomb;db:integer;
                     var atlag,ossz:real);
    var
         i:integer;
    begin
       ossz:=0;
       for i:=1 to db do
         ossz:=ossz+ x[i];
       atlag:=ossz/db;
    end;
   function osszeg(x:tomb; db:integer):real;
   var
     i:integer;
     s:real;
   begin
     s:=0;
     for i:=1 to db do
      s:=s+x[i];
    osszeg:=s;
  end;
```

```
  function fatlag(x:tomb;db:integer):real;
  begin
   fatlag:=osszeg(x,db)/db;
  end;
begin
end.
```

Írjunk programot, amely használja a *szumma* modult! ((*SZUMMA*)

```
program szummapr;
uses szumma;
var
    a: tomb;
    n: integer;
    sum,atl:real;
begin
    olvas(a,n);
    szamol(a,n,atl,sum);
    writeln; writeln('eljárással');
    writeln('összeg: ',sum:6:2);
    writeln('átlag : ',atl:6:2);
    writeln; writeln('függvénnyel');
    sum:=osszeg(a,n);
    atl:=fatlag(a,n);
    writeln('összeg: ',sum:6:2);
    writeln('átlag : ',atl:6:2);
    readln;
end.
```

A program futásának eredménye:

```
elemek száma: 5
 1. adat: 1
 2. adat: 2
 3. adat: 3
 4. adat: 4
 5. adat: 5

eljárással
összeg:  15.00
átlag :   3.00

függvénnyel
összeg:  15.00
átlag :   3.00
```

Írjunk modult (*tavm*), amely maximálisan 20 koordinátapontot tárol és kiszámítja a két legtávolabbi pont távolságát! (\T*AV*)

```
unit tavm;
INTERFACE
type
      pont = record
         x:real;
         y:real;
      end;
      Pontok = record
         t:array[1..20] of pont;
         db: integer;
         tav: real;
         p1,p2: pont;
      end;
      procedure Olvas(var v:Pontok);
      procedure Tavolsag(var v:Pontok);

IMPLEMENTATION

      procedure Olvas(var v:Pontok);
      var i,x0,y0:integer;
      begin
       repeat
        write('Az adatok száma (min.2): '); readln(v.db);
       until (v.db>1) and (v.db<=20);
        for i:=1 to v.db do
        begin
          writeln;
          write(i:2,'. adat x koord: '); readln(v.t[i].x);
          write('         y koord: ');   readln(v.t[i].y);
        end;
        v.tav:=0; v.p1.x:=0; v.p1.y:=0; v.p2.x:=0; v.p2.y:=0;
      end;
      procedure Tavolsag(var v:Pontok);
      var
         tavmax     : real;
         hely1,hely2: integer;
         cikl1,cikl2: integer;
      begin
         hely1:=1; hely2:=2;
         tavmax:=sqrt(sqr(v.t[hely1].x-v.t[hely1].x)+
                  sqr(v.t[hely2].y-v.t[hely2].y));
         for cikl1:=1 to v.db-1 do
           for cikl2:=2 to v.db do
           begin
            v.tav:=sqrt(sqr(v.t[cikl1].x-v.t[cikl2].x)+
                  sqr(v.t[cikl1].y-v.t[cikl2].y));
```

```
                                if v.tav>tavmax then begin
                                   tavmax:=v.tav;
                                   hely1:=cikl1;
                                   hely2:=cikl2;
                                  end;
            end;
          v.tav:=tavmax;
          v.p1.x:=v.t[hely1].x; v.p1.y:=v.t[hely1].y;
          v.p2.x:=v.t[hely2].x; v.p2.y:=v.t[hely2].y;
        end;
begin
end.
```

Írjunk programot, amely a *tavm* unitot használja! (*TAV*)

```
program tavpr;
uses tavm;
var
  w: Pontok;
begin
   Olvas(w);
   Tavolsag(w);
   writeln;
   writeln('Két legtávolabbi pont x1: ',
            w.p1.x:4:0,'  y1: ',w.p1.y:4:0);
   writeln('                        x2: ',
            w.p2.x:4:0,'  y2: ',w.p2.y:4:0);
   writeln;
   writeln('A két pont távolsága: ',w.tav:10:2);
   readln;
end.
```

A program futásának eredménye:

```
Az adatok száma (min.2): 3

 1. adat x koord: 1
         y koord: 1

 2. adat x koord: 2
         y koord: 2

 3. adat x koord: 3
         y koord: 6

Két legtávolabbi pont x1:    1  y1:    1
                      x2:    3  y2:    6

A két pont távolsága:       5.39
```

Írjunk modult (*rendezm*), amely tanulók adatait olvassa be és rendezi névsor és születési dátum szerint! (*RENDEZ*)

```
unit rendezm;
INTERFACE
type
        str20 = string[20];
        tanulo = record
                    nev : str20;
                    kora: integer;
                 end;
        tanulok= array[1..20] of tanulo;
        procedure olvas(var m:integer; var t:tanulok);
        procedure abcrendez(m:integer; var t:tanulok);
        procedure abckora(m: integer; var t:tanulok);
        procedure kiir(m:integer; t:tanulok);

IMPLEMENTATION

procedure olvas(var m:integer; var t:tanulok);
var
   i:integer;
begin
   write('A tanulók száma: '); readln(m);
   for i:=1 to m do
   begin
      writeln;
      write('Neve         : '); readln(t[i].nev);
      write('Születési éve: '); readln(t[i].kora);
   end;
end;

procedure abcrendez(m:integer; var t:tanulok);
var
   i,j: integer;
   s  : tanulo;
begin
   for i:=1 to m-1 do
     for j:=i+1 to m dto
     begin
        if(t[i].nev > t[j].nev) then begin
                                        s   :=t[i];
                                        t[i]:=t[j];
                                        t[j]:=s;
                                     end;
    end;
end;
```

```
procedure abckora(m: integer; var t:tanulok);
var
   i,j: integer;
   s  : tanulo;
begin
   for i:=1 to m-1 do
     for j:=i+1 to m 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 kiir(m:integer; t:tanulok);
var
    i,k:integer;
begin
   for i:=1 to m do
   begin
      k:= length(t[i].nev);
      writeln(t[i].nev,' ':20-k,t[i].kora:3);
   end;
end;

begin
end.
```

Írjunk programot, amely a *rendezm* modult használja! (*RENDEZ*)

```
program rendezpr;
uses rendezm;
var
   evf: tanulok;
   n: integer;

begin
   olvas(n,evf);
   writeln;
   writeln('Alapadatok');
   kiir(n,evf);
   abcrendez(n,evf);
   writeln;
   writeln('Név szerint rendezett adatok');
   kiir(n,evf); writeln;
   abckora(n,evf);
   writeln('Születési év szerint rendezett adatok');
   kiir(n,evf);
   readln;
end.
```

A program futásának eredménye:

```
A tanulók száma: 3

Neve         : Nagy Istvan
Születési éve: 1992

Neve         : Kiss Katalin
Születési éve: 1989

Neve         : Toth Marton
Születési éve: 1995

Alapadatok
Nagy Istvan         1992
Kiss Katalin        1989
Toth Marton         1995

Név szerint rendezett adatok
Kiss Katalin        1989
Nagy Istvan         1992
Toth Marton         1995

Születési év szerint rendezett adatok
Kiss Katalin        1989
Nagy Istvan         1992
Toth Marton         1995
```

Írjunk modult (*sport_u*), amely *sport* rekord használatával a sportágak adatait tartja nyilván! Számítsuk ki az össz létszámot, keressük meg a legtöbb létszámú sportágat, számláljuk meg a díjak számát, valamint az első díjak számát, a díj lehet 1,2,3. (*SPORT*)

```
UNIT sport_u;
INTERFACE
type   sport= record
                sportag: string;
                letszam: integer;
                dij: integer;
            end;
    sportklub = array[1..10] of sport;
    procedure SportKlubOlvas(var sp: sportklub; var n: integer);
    function ElsoDijakSzama( sp: sportklub; n: integer):integer;
    function OsszLetszam( sp: sportklub; n: integer):integer;
    function DijakSzama( sp: sportklub; n: integer):integer;
    procedure MaxLetszamKeres( sp: sportklub; n:integer;
                        var Maxletszam:sport);
    procedure Kiir( sp: sportklub; n: integer);
    procedure MaxKiir( mp: sport);
```

```
IMPLEMENTATION
  procedure SportKlubOlvas(var sp: sportklub; var n: integer);
  var
     i: integer;
  begin
    write('A sport szakosztáyok 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

  end.
```

Írjunk programot, amely használja a *sport_u* modult! (*SPORT*

SPORT_PR.PAS program használja a **SPORT_U.PPU** unitot.

```
program sport_pr;
uses sport_u;
var
  s : sportklub;
  s_db: integer;
  max_letszam: sport;
```

```
    ossz_letszam, dijak_szama, elso_dij: integer;
  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 díjak 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 : kajak
létszám : 4
díj     : 2

sportág : kenu
létszám : 2
díj     : 1

sportág : vitorlás
létszám : 5
díj     : 1


A sportklub adatainak megjelenítése

   Sportág   létszám   díj
-------------------------------
     kajak       4      2
      kenu       2      1
  vitorlás       5      1

A legnagyobb létszámú szakosztály adatainak megjelenítése

   Sporáag   létszám   díj
-------------------------------
  vitorlás       5      1

A díjak darabszáma: 3
Első díjak száma  : 2
Össz létszám      : 11
```