10. Dinamikus memóriahasználat

Írjunk programot, amely beolvas egy egész számot és kiszámítja a négyzetét! (*mut1*)

```
program mut1;
var
   a: ^integer;
   ered : integer;

begin
   new(a);
   write('Kérek egy egész számot: ');
   readln(a^);
   ered := sqr(a^);
   writeln('Négyzete: ',ered);
   dispose(a);
   readln;
end.
```

A program futásának eredménye:

```
Kérek egy egész számot: 2
Négyzete: 4
```

Készítsünk programot, amely adott darabszámú, valós elemű, dinamikus helyfoglalású tömböt feltölt adatokkal! Számítsuk ki a tömb elemeinek összegét, és keressük meg a legkisebb és a legnagyobb elemét! (*mut2*)

```
program mut2;
type t = array[1..30] of real;
var
   pt : ^t;
   db, i: integer;
   max,min, osszeg: ^real;

begin
   new(pt);
   new(max); new(min); new(osszeg);
   write('Adatok száma: ');
   readln(db);
   for i:=1 to db do
   begin
     write(i:2,'. adat: '); readln(pt^[i]);
   end;
   osszeg^ := 0;
   min^:=pt^[1]; max^:= pt^[1];
   for i:=1 to db do
   begin
      osszeg^ := osszeg^ + pt^[i];
```

```
      if max^ < pt^[i] then max^:= pt^[i];
      if min^ > pt^[i] then min^:= pt^[i];
   end;
   writeln;
   writeln('Az adatok összege: ',osszeg^:6:2);
   writeln('Legkisebb adat   : ',min^:6:2);
   writeln('Legnagyobb adat  : ',max^:6:2);
   dispose(min); dispose(max); dispose(osszeg);
   dispose(pt);
   readln;
end.
```

A program futásának eredménye:

```
Adatok száma: 4
 1. adat: 23
 2. adat: 1
 3. adat: 45
 4. adat: 3

Az adatok összege:  72.00
Legkisebb adat   :   1.00
Legnagyobb adat  :  45.00
```

Írjunk programot, amely egy karaktertömbben megszámlálja a mássalhangzó-magánhangzó párokat! A feladat megoldásához használjunk alprogramot! (*keresm*)

```
program keresm;
type  Pstring = ^string;
var
    Ps :Pstring;
    db:integer;
    function mghmsh(x:Pstring):integer;
    type
         abc= set of 'a'..'z';
    var
          t,mgh,msh:abc;
          i,k,m,van:integer;
    begin
      t :=['a'..'z'];
      mgh:=['a','o','e','i','u'];
      msh:=t-mgh;
      k:=ord(x^[0]);      { karakterek száma }
      van:=0; m:=0;
      for i:=1 to k do
      begin
        if x^[i] in msh then van:=1;
        if (x^[i] in mgh) and (van = 1)  then
                  begin  m:=m+1; van:=0; end;
      end;
      mghmsh:=m;
    end;
```

```
begin
   new(Ps);
   write('A szó: '); readln(Ps^);
   writeln('msh-mgh párok száma: ',mghmsh(Ps));
   dispose(Ps);
   readln;
end.
```

A program futásának eredménye:

```
A szó: szalmakalap
msh-mgh párok száma: 4
```

Írjunk programot, amely adott darabszámú dinamikus egész elemű tömböt tölt fel adatokkal. Számláljuk meg, hogy a tömb hány eleme páros, ill. páratlan, valamint hány eleme osztható hárommal. A feladat megoldásához használjunk alprogramokat! (*tombmut1*)

Használjuk fel az alábbi típusdefiníciót és deklarációt:

```
type tomb = array[1..30] of integer;
     ptomb = ^tomb;
     pinteger = ^integer;

var
   py : ptomb;
   y_db, oszt3: integer;
   par_db, pt_db: pinteger;
```

Az ***Olvas*** eljárásban olvassuk be a tömb elemeinek számát és töltsük fel a dinamikus tömböt adatokkal.

```
procedure Olvas(var x: ptomb; var n:integer);
```

A ***Keres*** eljárásban számláljuk meg, hogy a tömnek hány páros és hány páratlan eleme van.

```
procedure Keres(x:ptomb; n: integer;
                var parosak, paratlanok:pinteger);
```

Az ***Osztható3*** eljárásban számláljuk meg, hogy a tömbnek hány eleme osztható 3-mal.

```
function Oszthato3(x:ptomb; n:integer):integer;
```

A feladat megoldása:

```
program tombmut1;
type tomb = array[1..30] of integer;
     ptomb = ^tomb;
     pinteger = ^integer;
procedure Olvas(var x: ptomb; var n:integer);
var
   i:integer;
begin
   write('Elemek száma: '); readln(n);
   new(x);
   for i:=1 to n do
   begin
      write(i:2,'. adat: '); readln(x^[i]);
   end;
end;
procedure Keres(x:ptomb; n: integer;
                var parosak, paratlanok:pinteger);
var
   i:integer;
begin
   parosak^:= 0; paratlanok^:= 0;
for i:= 1 to n do
   begin
     if x^[i] mod 2 = 0 then parosak^:= parosak^+1
     else paratlanok^:= paratlanok^+1;
   end;
end;
function Oszthato3(x:ptomb; n:integer):integer;
var
   i, db_3:integer;
begin
  db_3:= 0;
  for i:=1 to n do
   if x^[i] mod 3 = 0 then db_3:=db_3+1;
  Oszthato3:= db_3;
end;
var
   py : ptomb;
   y_db, oszt3: integer;
   par_db, pt_db: pinteger;
begin
   new(par_db); new(pt_db);
   Olvas(py,y_db);
   Keres(py,y_db,par_db,pt_db);
   oszt3:= Oszthato3(py,y_db);
   writeln('Párosak száma        : ',par_db^);
   writeln('Páratlanok száma     : ',pt_db^);
   writeln('3-mal oszthatók száma: ',oszt3);
   dispose(par_db); dispose(pt_db);
   dispose(py);
   readln;
 end.
```

A program futásának eredménye:

```
Elemek száma: 6
 1. adat: 2
 2. adat: 6
 3. adat: 12
 4. adat: 7
 5. adat: 21
 6. adat: 8
Párosak száma        : 4
Páratlanok száma     : 2
3-mal oszthatók száma: 3
```

Írjunk programot, amely adott darabszámú dinamikus egész elemű tömböt tölt fel adatokkal. Számláljuk meg, hogy a tömb hány eleme páros, ill. páratlan, valamint hány eleme osztható hárommal. A feladat megoldásához használjunk alprogramokat! (*tombmut2*)

```
program tombmut2;
type tomb = array[1..30] of integer;
     ptomb = ^tomb;
     pinteger = ^integer;
procedure olvas(var x: ptomb; var n:integer);
var
   i:integer;
begin
   write('Elemek száma: '); readln(n);
   new(x);
   for i:=1 to n do
   begin
      write(i:2,'. adat: '); readln(x^[i]);
   end;
end;
procedure Keres(x:ptomb; n: integer; var poz, neg, zero:pinteger);
var
   i:integer;
begin
   poz^:= 0; neg^:= 0; zero^:=0;
for i:= 1 to n do
   begin
     if x^[i] > 0 then poz^:= poz^+1;
     if x^[i] < 0 then neg^:= neg^+1;
     if x^[i] = 0 then zero^:= zero^+1;
   end;
end;
```

```
function PozOsszeg(x:ptomb; n:integer):integer;
var
   i, ossz:integer;
begin
  ossz:= 0;
  for i:=1 to n do
   if x^[i] >  0 then ossz:=ossz+x^[i];

  PozOsszeg:= ossz;
end;
var
   pw : ptomb;
   w_db, pozOssz: integer;
   poz_db, neg_db, zero_db: pinteger;
begin
   new(poz_db); new(neg_db); new(zero_db);
   olvas(pw,w_db);
   Keres(pw,w_db,poz_db,neg_db,zero_db);
   pozOssz:= PozOsszeg(pw,w_db);
   writeln('Pozitívok száma  : ',poz_db^);
   writeln('Negatívok száma  : ',neg_db^);
   writeln('Zérusok száma    : ',zero_db^);
   writeln('Pozitívok összege: ',pozOssz);
   dispose(poz_db); dispose(neg_db); dispose(zero_db);
   dispose(pw);
   readln;
 end.
```

A program futásának eredménye:

```
Elemek száma: 6
 1. adat: 2
 2. adat: -4
 3. adat: 0
 4. adat: 6
 5. adat: 7
 6. adat: -2
Pozitívok száma  : 3
Negatívok száma  : 2
Zérusok száma    : 1
Pozitívok összege: 15
```

Írjunk programot, amely a *Calculator* rekordot felhasználva műveleteket hajt végre. A feladat megoldásához használjunk alprogramot, melynek paramétere *PCalculator* típusú legyen! (*muveletm*)

```
program muveletm;
type
    PCalculator = ^Calculator;
    Calculator = record
          a,b:real;
      muv_jel: char;
     eredmeny: real;
    end;
```

```
var
    PCal:PCalculator;
function Szamol( sz:PCalculator):real;
begin
   case sz^.muv_jel of
    '+' : sz^.eredmeny:= sz^.a+sz^.b;
    '-' : sz^.eredmeny:= sz^.a-sz^.b;
    '*' : sz^.eredmeny:= sz^.a*sz^.b;
    '/' : sz^.eredmeny:= sz^.a/sz^.b;
    '^' : begin
          if(sz^.a > 0) then  sz^.eredmeny:=exp(sz^.b*(ln(sz^.a)))
                      else   sz^.eredmeny:=0;
          end;
    end;
    Szamol:= sz^.eredmeny;
end;
begin
   new(PCal);
   write('1. adat: '); readln(PCal^.a);
   write('2. adat: '); readln(PCal^.b);
   repeat
      write('műveletijel (+,-,*,/,^): '); readln(PCal^.muv_jel);
   until PCal^.muv_jel in ['+','-','*','/','^'];
   writeln(PCal^.a:5:2,' ',PCal^.muv_jel,PCal^.b:5:2,'= ',
           Szamol(PCal):5:2);
   dispose(PCal);
   readln;
 end.
```

A program futásainak eredménye:

```
1. adat: 3
2. adat: 4
műveletijel (+,-,*,/,^): +
 3.00 + 4.00=  7.00

1. adat: 5
2. adat: 6
műveletijel (+,-,*,/,^): /
 5.00 / 6.00=  0.83

1. adat: 2
2. adat: 3
műveletijel (+,-,*,/,^): ^
 2.00 ^ 3.00=  8.00
```

Írjunk programot, amely használja a *Komplex* rekordot komplex műveletek végrehajtására. Az alprogramok paramétere a *PKomplex* mutató! (*komplm*)

A feladat megoldásához használja az alábbi definíciót:

```
type
      PKomplex = ^Komplex;
      Komplex = record
         re,im: real;
      end;
```

A feladat megoldása:

```
program komplm;
 type
      PKomplex = ^Komplex;
      Komplex = record
         re,im: real;
      end;

  procedure Osszead(x,y:PKomplex; var ered:PKomplex);
  begin
     ered^.re :=  x^.re + y^.re;
     ered^.im :=  x^.im + y^.im;
  end;
  procedure Kivon(x,y:PKomplex; var ered:PKomplex);
  begin
     ered^.re :=  x^.re - y^.re;
     ered^.im :=  x^.im - y^.im;
  end;
  procedure Szoroz(x,y:PKomplex; var ered:PKomplex);
  begin
     ered^.re := x^.re * y^.re - x^.im * y^.im;
     ered^.im := x^.re * y^.im + x^.im * y^.re;
  end;
  procedure Oszt(x,y:PKomplex; var ered:PKomplex);
  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 Kiir(x,y:PKomplex;m: char;ered:PKomplex);
  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
   Px1,Px2,Pered0: PKomplex;
begin
  new(Px1); new(Px2); new(Pered0);
  writeln;

  write('1. komplex szám valós része   : ');readln(Px1^.re);
  write('                képzetes része: ');readln(Px1^.im);

  write('2. komplex szám valós része   : ');readln(Px2^.re);
  write('                képzetes része: ');readln(Px2^.im);

  writeln;
  writeln('Műveletek');
  writeln('Az összeadás eredménye: ');
  Osszead(Px1,Px2,Pered0);
  Kiir(Px1,Px2,'+',Pered0);
  writeln('A kivonás eredménye: ');
  Kivon(Px1,Px2,Pered0);
  Kiir(Px1,Px2,'-',Pered0);
  writeln('A szorzás eredménye: ');
  Szoroz(Px1,Px2,Pered0);
  Kiir(Px1,Px2,'*',Pered0);
  writeln('A osztás eredménye: ');
  Oszt(Px1,Px2,Pered0);
  Kiir(Px1,Px2,'/',Pered0);
  dispose(Px1); dispose(Px2); dispose(Pered0);
  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
A osztás eredménye:
1.0+1.0i / 2.0+2.0i =   0.5+  0.0i
```