7. Strukturált típusok

Mintafeladat a különböző tömbtípusok konstanssal való feltöltésére és kiíratására! (*minta7_1*)

```
program minta7_1;
const
  fejlec:array[1..8] of char = 'Eredmény';
  adatok:array[1..4] of integer=(1,2,3,4);
  matrix:array[1..2,1..3] of real=((1,20,12),(3,4,7));
var
   i, j: integer;
begin
  writeln('fejléc tartalma');
  writeln(fejlec);
  writeln('adatok tömb tartalma');
  for i:=1 to 4 do
    writeln(adatok[i]);
  writeln('matrix tömb tartalma');
  for i:=1 to 2 do
  begin
    for j:=1 to 3 do
      write(matrix[i,j]:8:2);
    writeln;
  end;
  readln;
end.
```

A program futásának eredménye:

```
fejléc tartalma
Eredmény
adatok tömb tartalma
1
2
3
4
matrix tömb tartalma
    1.00   20.00   12.00
    3.00    4.00    7.00
```

Mintaprogram a különböző típusú tömbök egy elemének értékadására! (*minta7_2*)

```
program minta7_2;
const
    ar_db = 10;
var
   darab: array[1..10] of integer;
```

```
   ara  : array[1..ar_db] of real;
   logikai: array[0..5] of boolean;
   karakterek : array[1..3] of char;
begin
  darab[1]:= 12;
  ara[1]:= 12.56;
  logikai[1]:= true;
  karakterek[1]:='a';
  writeln(darab[1]:3,ara[1]:8:2,karakterek[1]:4);
  readln;
end.
```

A program futásának eredménye:

```
 12   12.56   a
```

Mintafeladat különböző típusú kétdimenziós tömb első elemének értékadására! (*minta7_3*)

```
program minta7_3;
var
   x  : array[1..3,1..3] of integer;
   y  : array[1..2,1..2] of real;
begin
  x[1,1]:= 122;
  y[1,1]:= 12.56;
  writeln(x[1,1]);
  writeln(y[1,1]);
  readln;
end.
```

A program futásának eredménye:

```
122
 1.256000000000000E+001
```

Mintafeladat a tömb típusának definálására, valamint ezzel a típussal deklarált tömbök értékadására! (*minta7_4*)

```
program minta7_4;
const
    ar_db = 10;
type
   tomb1 = array[1..10] of integer;
   tomb2 = array[1..ar_db] of real;
var
   darab: tomb1;
   ara  : tomb2;
   logikai: array[0..5] of boolean;
   karakterek : array[1..3] of char;
```

```
begin
  darab[1]:= 12;
  ara[1]:= 12.56;
  logikai[1]:= true;
  karakterek[1]:='a';
  writeln(darab[1]:3,ara[1]:8:2,karakterek[1]:4);
  readln;
end.
```

A program futásának eredménye:

```
 12   12.56   a
```

Mintafeladat a **const**, **type** használatára, valamint a konstansként deklarált tömbök feltöltésére! (*minta7_5*)

```
program minta7_5;
const
    ar_db = 4;
type
   tomb1 = array[1..4] of integer;
const
   darab : tomb1 =  (5,2,7,4);
   ara   : array[1..ar_db] of real = (1.2,2.3,4.1,5.3);
   a     : array[1..3,1..2] of integer = ((1,2),(3,4),(5,6));
var
   logikai: array[0..5] of boolean;
   karakterek : array[1..3] of char;
begin
  writeln(darab[1]:4,ara[1]:6:2);
  logikai[1]:= true;
  karakterek[1]:='a';
  writeln(a[1,1]:3,a[1,2]:3);
  writeln(a[2,1]:3,a[2,2]:3);
  writeln(a[3,1]:3,a[3,2]:3);
  readln;
end.
```

A program futásának eredménye:

```
   5  1.20
  1  2
  3  4
  5  6
```

Írjunk programot, amely beolvassa a tömb elemszámát és ellenőrzi, hogy a megengedett indexhatárba esik-e! Olvassuk be a tömb adatait, majd a páros adatok összegét és átlagát, a páratlan adatok szorzatát számítsuk ki! Az eredményeket jelenítsük meg! (*tomb01*)

```
program tomb01;
var
   i, db : integer;
   y : array[1..15] of integer;
   osszeg, szorzat, ydb: integer;
   atlag : real;
begin
  repeat
    write('Elemek száma: '); readln(db);
  until (db >= 1 ) and (db <=15);
  for i:=1 to db do
  begin
    write('y[',i:2,'] = '); readln(y[i]);
  end;
  osszeg := 0;  ydb := 0;
  szorzat := 1;
  for i:=1 to db do
  begin
    if ( y[i] mod 2 = 0) then
    begin
     osszeg := osszeg + y[i]; ydb := ydb + 1;
    end
    else
     szorzat := szorzat * y[i];
  end;
  atlag := osszeg/ydb;
  writeln('Páros adatainak összege: ',osszeg);
  writeln('Páros adatainak átlaga:  ',atlag:6:2);
  writeln('Páratlan adatainak szorzata: ',szorzat);
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 5
y[ 1] = 2
y[ 2] = 7
y[ 3] = 4
y[ 4] = 3
y[ 5] = 6
Páros adatainak összege: 12
Páros adatainak átlaga:    4.00
Páratlan adatainak szorzata: 21
```

Írjunk programot, amely beolvassa a tömb elemszámát és ellenőrzi, hogy a megengedett indexhatárba esik-e! Olvassuk be a tömb adatait. Keressük meg a tömb legkisebb és a legnagyobb elemét, majd számítsuk a legnagyobb és a legkisebb elem átlagát! (*tomb02*)

```
program tomb02;
var
   i, db : integer;
   y : array[1..30] of integer;
   max, min: integer;
   maxmin_atlag : real;
begin
  repeat
    write('Elemek száma: '); readln(db);
  until (db >= 1 ) and (db <=30);
  for i:=1 to db do
  begin
    write('y[',i:2,'] = '); readln(y[i]);
  end;
  max := y[1]; min:= y[1];
  for i:=2 to db do
  begin
    if max < y[i] then max := y[i];
    if min > y[i] then min := y[i];
  end;
  maxmin_atlag:= (max+min)/2;
  writeln('Legnagyobb eleme: ',max);
  writeln('Legkisebb eleme : ',min);
  writeln('Legnagyobb és a legkisebb elem átlaga: ',maxmin_atlag:6:2);
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 4
y[ 1] = 3
y[ 2] = 67
y[ 3] = 12
y[ 4] = 3
Legnagyobb eleme: 67
Legkisebb eleme : 3
Legnagyobb és a legkisebb elem átlaga:  35.00
```

Írjunk programot, amely beolvassa a tömb elemszámát és ellenőrzi, hogy a megengedett indexhatárba esik-e! Olvassuk be a tömb adatait. Számláljuk meg a tömb páros és páratlan adatainak számát! (*tomb03*)

```
program tomb03;
var
   i, db : integer;
   adat: array[1..20] of integer;
   pt, paros : integer;
begin
  repeat
    write('Elemek száma: '); readln(db);
  until (db >= 1 ) and (db <=20);
  for i:=1 to db do
  begin
    write('adat[',i:2,'] = '); readln(adat[i]);
  end;
  pt:=0; paros:=0;
  for i:=1 to db do
  begin
    if adat[i] mod 2 = 0 then paros := paros+1
    else
      pt:= pt+1;
  end;
  writeln('Páratlan elemek száma: ',pt);
  writeln('Páros elemek száma   : ',paros);
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 5
adat[ 1] = 2
adat[ 2] = 67
adat[ 3] = 4
adat[ 4] = 3
adat[ 5] = 6
Páratlan elemek száma: 2
Páros elemek száma   : 3
```

Írjunk programot, amely beolvassa a tömb elemszámát és ellenőrzi, hogy a megengedett indexhatárba esik-e! Olvassuk be a tömb adatait. Külön tömbbe tároljuk a 4-gyel és a 3-mal osztható adatokat! Írassuk ki az eredmény tömbök tartalmát! (*tomb04*)

```
program tomb04;
var
   i, n : integer;
   z: array[1..20] of integer;
   oszt4, oszt3: array[1..20] of integer;
   db4, db3 : integer;
```

```
begin
  repeat
    write('Elemek száma: '); readln(n);
  until (n >= 1 ) and (n <=20);
  for i:=1 to n do
  begin
    write('z[',i:2,'] = '); readln(z[i]);
  end;
  db4:=0; db3:=0;
  for i:=1 to n do
  begin
    if z[i] mod 4 = 0 then
    begin
      db4 := db4+1;
      oszt4[db4] := z[i];
    end;
    if z[i] mod 3 = 0 then
    begin
      db3 := db3+1;
      oszt3[db3] := z[i];
    end;
  end;
  writeln('4-gyel oszthatók száma: ',db4);
  if db4 > 0 then
  begin
   for i:=1 to db4 do
     writeln(oszt4[i]);
  end;
  writeln('3-mal oszthatók száma: ',db3);
  if db3 > 0 then
  begin
   for i:=1 to db3 do
     writeln(oszt3[i]);
  end;
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 5
z[ 1] = 12
z[ 2] = 6
z[ 3] = 3
z[ 4] = 20
z[ 5] = 4
4-gyel oszthatók száma: 3
12
20
4
3-mal oszthatók száma: 3
12
6
3
```

Írjunk programot, amely beolvassa a tömb elemszámát és ellenőrzi, hogy a megengedett indexhatárba esik-e! Olvassuk be a tömb adatait! Számláljuk meg a tömb pozitív, negatív és zérus elemeinek számát! (*tomb05*)

```
program tomb05;
var
   i, n : integer;
   x: array[1..30] of integer;
   poz_db, neg_db, zerus_db : integer;
begin
  repeat
    write('Elemek száma: '); readln(n);
  until (n >= 1 ) and (n <=30);
  for i:=1 to n do
  begin
    write('x[',i:2,'] = '); readln(x[i]);
  end;
  poz_db:=0; neg_db:=0; zerus_db := 0;
  for i:=1 to n do
  begin
    if x[i] < 0 then neg_db := neg_db+1
    else if x[i] > 0 then poz_db := poz_db+1
    else  zerus_db := zerus_db+1;
  end;
  writeln('Pozitív elemek száma: ',poz_db);
  writeln('Negatív elemek száma: ',neg_db);
  writeln('Zérus elemek száma  : ',zerus_db);
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 5
x[ 1] = 12
x[ 2] = -3
x[ 3] = 0
x[ 4] = 2
x[ 5] = -8
Pozitív elemek száma: 2
Negatív elemek száma: 2
Zérus elemek száma  : 1
```

Írjunk programot, amely beolvas egy szöveget és számláljuk meg, hogy magánhangzónként hány betűt tartalmaz! (*tomb06*)

```
program tomb06;
var
   i, db : integer;
   szoveg: string;
   adb, edb, odb, udb : integer;
```

```
begin
  write('Szöveg: '); readln(szoveg);
  db := length(szoveg);
  for i:=1 to db do
  begin
    case szoveg[i] of
    'a': adb := adb+1;
    'e': edb := edb+1;
    'o': odb := odb+1;
    'u': udb := udb+1;
    end;
  end;
  writeln('a betűk száma: ',adb);
  writeln('e betűk száma: ',edb);
  writeln('o betűk száma: ',odb);
  writeln('u betűk száma: ',udb);
  readln;
end.
```

A program futásának eredménye:

```
Szöveg: Konnyu Pascal nyelven programozni.
a betűk száma: 3
e betűk száma: 2
o betűk száma: 3
u betűk száma: 1
```

Írjunk programot, amely beolvas egy szöveget! Bontsuk a szöveget szavakra! A szavak elválasztásánál vegyük figyelembe a vesszőt, a szóközt és a mondatzáró pontot! (*tomb07*)

```
program tomb07;
var
   i, j, db : integer;
   szoveg, szo: string;
   adb, edb, odb, udb : integer;
begin
  write('Szöveg (vége . ): '); readln(szoveg);
  db := length(szoveg);
  for i:=1 to db do
  begin
    if (szoveg[i] = ',') then continue;
    if (szoveg[i] = ' ') or (szoveg[i] = '.') then
    begin
         szo[0] := chr(j); j:= 0;
         writeln(szo);
    end
    else
      begin
        j:= j+1;
        szo[j]:= szoveg[i];
```

```
      end;
  end;
  readln;
end.
```

A program futásának eredménye:

```
Szöveg (vége . ): Ma szep az ido, de holnap mar megvaltozik.
Ma
szep
az
ido
de
holnap
mar
megvaltozik
```

Írjunk programot, amely karakterenként tölti fel a sztringet! A záró karakter a . (pont). A beolvasott mondatot szavanként írjuk vissza! (*tomb08*)

```
program tomb08;
var
   i, j, db : integer;
   szoveg, szo: string;
   c : char;
   adb, edb, odb, udb : integer;
begin
  write('Szöveg (vége . ): ');
  i:=0;
  read( c);
  while (c <> '.') do
  begin
     i:=i+1;
     szoveg[i]:= c;
     read(c);
  end;
  i:=i+1;
  szoveg[i] := '.';
  szoveg[0]:= chr(i);
  db := length(szoveg);
  writeln('A beolvasott szöveg: ',szoveg);
  for i:=1 to db do
  begin
    if (szoveg[i] = ',') then continue;
    if (szoveg[i] = ' ') or (szoveg[i] = '.') then
    begin
         szo[0] := chr(j); j:= 0;
         writeln(szo);
    end
    else
      begin
        j:= j+1;
        szo[j]:= szoveg[i];
```

```
      end;
  end;
  readln; readln;
end.
```

A program futásának eredménye:

```
Szöveg (vége . ): Ha nem esne az eso, elmennenk kirandulni.
A beolvasott szöveg: Ha nem esne az eso, elmennenk kirandulni.
Ha
nem
esne
az
eso
elmennenk
kirandulni
```

Írjunk programot, amely beolvassa a konstansként megadott elemszámú elem adatait, és az *index* tömbben tárolt index alapján számítsuk ki az adatokkal feltöltött tömb *index*-edik adatának összegét! (*tomb09*)

```
program tomb09;
const
   index_db = 4;
   db = 10;
   index : array[1..4] of integer = (2,5,6,9);
var
   i: integer;
   x: array[1..15] of integer;
   osszeg: integer;
begin
  writeln('10 darab egész szám beolvasása');
  for i:=1 to db do
  begin
    write('x[',i:2,'] = ');
    readln(x[i]);
  end;
  osszeg:= 0;
  for i:=1 to index_db do
  begin
    osszeg := osszeg +  x[index[i]];
  end;
  writeln('Összeg: ',osszeg);
  readln;
end.
```

A program futásának eredménye:

```
10 darab egész szám beolvasása
x[ 1] = 1
x[ 2] = 3
x[ 3] = 2
x[ 4] = 5
```

```
x[ 5] = 6
x[ 6] = 4
x[ 7] = 8
x[ 8] = 7
x[ 9] = 9
x[10] = 12
Összeg: 22
```

Írjunk programot, amely a konstansként megadott mátrixot feltölti a sor és az oszlop indexek összegével! Írassuk ki az így feltöltött mátrixot! (*tomb10*)

```
program tomb10;
const
   n = 5;
   m = 8;
var
   i, j: integer;
   tomb: array[1..n,1..m] of integer;
begin
  for i:=1 to n do
    for j:=1 to m do
      tomb[i,j]:= i+j;
  writeln('A tömb elemei: ');
  writeln;
  for i:=1 to n do
    begin
      for j:=1 to m do
        write(tomb[i,j]:5);
      writeln;
    end;
  readln;
end.
```

A program futásának eredménye:

```
A tömb elemei:

    2    3    4    5    6    7    8    9
    3    4    5    6    7    8    9   10
    4    5    6    7    8    9   10   11
    5    6    7    8    9   10   11   12
    6    7    8    9   10   11   12   13
```

Írjunk programot, amely beolvassa a 10 elemű egész típusú tömb adatait és egy másik tömböt úgy tölti fel a beolvasott adatokkal, hogy a páros elemeket a tömb első indexétől, a páratlan elemeit az utolsó (10) indexétől tárolja! (*tomb11*)

```
program tomb11;
var
   i, j, paros, paratlan: integer;
   x,y: array[1..10] of integer;
begin
  writeln('Kérem a tömb 10 darab elemét: ');
  for i:=1 to 10 do
  begin
    write('a[',i,']= ');
    readln(x[i]);
  end;
  paros:=1;
  paratlan:=10;
  for i:=1 to 10 do
  begin
    if not odd(i) then
    begin
      y[paros]:=x[i];
      paros:=paros+1;
    end
    else
    begin
      y[paratlan]:=x[i];
      paratlan:=paratlan-1;
    end;
  end;
  writeln('Az eredeti tömb:');
  for i:=1 to 10 do
    write(x[i]:6);
  writeln;
  writeln('A rendezett tömb:');
  for i:=1 to 10 do
    write(y[i]:6);
  writeln;
  readln;
end.
```

A program futásának eredménye:

```
Kérem a tömb 10 darab elemét:
a[1]= 1
a[2]= 2
a[3]= 3
a[4]= 4
a[5]= 5
a[6]= 6
a[7]= 7
a[8]= 8
a[9]= 9
```

```
a[10]= 10
Az eredeti tömb:
     1     2     3     4     5     6     7     8     9    10
A rendezett tömb:
     2     4     6     8    10     9     7     5     3     1
```

Írjunk programot, amely beolvassa a megadott elemszámú tömb adatait! Vizsgáljuk meg, hogy az adatok számtani sorozatot alkotnak-e vagy nem! (*tomb12*)

```
program tomb12;
var
   i, n, diff, szam: integer;
   tomb: array[1..10] of integer;
   szamtani :boolean;
begin
   writeln('Elemek száma(2-10) ');
   repeat
     write('Elemszám: '); readln(n);
   until n in[2..10];
   for i:=1 to n do
   begin
     write(i,'. elem: ');
     readln(tomb[i]);
   end;
   szamtani:=true;
   diff:=tomb[2]-tomb[1];
   for i:=2 to n-1 do
     if (tomb[i+1]-tomb[i]) <> diff then
        szamtani:=false;
   if szamtani then
   begin
     writeln('A tömb elemei számtani sorozatot alkotnak.');
     writeln('A differencia: ',diff);
   end
   else writeln('A tömb elemei nem alkotnak számtani sorozatot.');
   readln;
end.
```

A program futásának eredménye:

```
Elemek száma(2-10)
Elemszám: 4
1. elem: 2
2. elem: 5
3. elem: 8
4. elem: 11
A tömb elemei számtani sorozatot alkotnak.
A differencia: 3
```

```
Elemek száma(2-10)
Elemszám: 4
1. elem: 2
2. elem: 5
3. elem: 9
4. elem: 12
A tömb elemei nem alkotnak számtani sorozatot.
```

Írjunk programot, amely beolvassa a megadott elemszámú tömb adatait! Számítsuk ki a páros adatok szorzatát és a páratlan adatok összegét! (*tomb13*)

```
program tomb13;
var
   k: array[1..20] of integer;
   i, db, ptosszeg, ptdb, parosdb, szorzat: integer;
begin
  repeat
    write('Elemek száma: '); readln(db);
  until (db >= 1 ) and (db <=20);
  for i:=1 to db do
  begin
    write('k[',i:2,'] = '); readln(k[i]);
  end;
  ptosszeg:=0; ptdb:= 0;
  szorzat:=1; parosdb:=0;
  for i:=1 to db do
  begin
    if k[i] mod 2 = 0 then
    begin
      parosdb := parosdb+1;
      szorzat := szorzat*k[i];
    end
    else
    begin
      ptdb:= ptdb+1;
      ptosszeg:= ptosszeg+k[i];
    end;
  end;
  writeln('Páratlan adatok száma  : ',ptdb);
  writeln('Páratlan adatok összege: ',ptosszeg);
  writeln('Páros adatok darabszáma: ',parosdb);
  writeln('Páros adatok szorzata  : ',szorzat);
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 6
k[ 1] = 1
k[ 2] = 2
k[ 3] = 3
k[ 4] = 4
```

```
k[ 5] = 5
k[ 6] = 6
Páratlan adatok száma  : 3
Páratlan adatok összege: 9
Páros adatok darabszáma: 3
Páros adatok szorzata  : 48
```

Írjunk programot, amely beolvassa a megadott elemszámú tömb adatait! Keressük ki a tömb legnagyobb és legkisebb adatát úgy, hogy jegyezzük meg az adathoz tartozó indexet is! (*tomb14*)

```
program tomb14;
var
   x: array[1..30] of real;
   i, db, max_index, min_index: integer;
   min, max: real;
begin
  repeat
    write('Elemek száma: '); readln(db);
  until (db >= 1 ) and (db <=30);
  for i:=1 to db do
  begin
    write('x[',i:2,'] = '); readln(x[i]);
  end;
  max := x[1]; max_index :=1;
  min := x[1]; min_index :=1;
  for i:=2 to db do
  begin
    if x[i] < min then
    begin
      min := x[i];
      min_index := i;
    end;
    if x[i] > max then
    begin
      max := x[i];
      max_index:= i;
    end;
  end;
  writeln('Legnagyobb adat       : ',max:6:2);
  writeln('Legnagyobb adat indexe: ',max_index);
  writeln('Legkisebb adat        : ',min:6:2);
  writeln('Legkisebb adat indexe : ',min_index);
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 6
x[ 1] = 2
x[ 2] = 17
x[ 3] = 3
x[ 4] = -1
x[ 5] = 6
x[ 6] = 12
Legnagyobb adat       :  17.00
Legnagyobb adat indexe: 2
Legkisebb adat        :  -1.00
Legkisebb adat indexe : 4
```

Írjunk programot, amely beolvassa egy adott elemszámú tömb adatait! Keressük meg a tömb legnagyobb adatát és ezzel az adattal osszuk végig az összes elemét! A normált adatokat jelenítsük meg! (*tomb15*)

```
program tomb15;
var
   i, db : integer;
   y : array[1..30] of real;
   max : real;
begin
  repeat
    write('Elemek száma: '); readln(db);
  until (db >= 1 ) and (db <=30);
  for i:=1 to db do
  begin
    write('y[',i:2,'] = '); readln(y[i]);
  end;
  max := y[1];
  for i:=2 to db do
    if max < y[i] then max := y[i];
  writeln('Legnagyobb eleme: ',max:6:2);
  writeln('Normált elemek');
  for i:=1 to db do
  begin
   y[i] := y[i]/max;
   writeln(y[i]:6:2);
  end;
  readln;
end.
```

A program futásának eredménye:

```
Elemek száma: 5
y[ 1] = 3
y[ 2] = 2
y[ 3] = 10
y[ 4] = 4
y[ 5] = 7
Legnagyobb eleme:  10.00
```

```
Normált elemek
  0.30
  0.20
  1.00
  0.40
  0.70
```

Írjunk programot, amely beolvassa egy 3x3-as mátrix adatait! Számítsuk ki az alsó háromszög mátrix elemeinek összegét, a főátlóbeli elemek szorzatát, valamint az alsó háromszög elemeinek szorzatát! (*tomb16*)

```
program tomb16;
var
   i, j: integer;
   w : array[1..3,1..3] of integer;
   felso_osszeg, atlo_szorzat, also_szorzat: integer;
begin
  for i:=1 to 3 do
    begin
      for j:=1 to 3 do
        begin
          write('w[',i:2,',',j:2,'] = '); readln(w[i,j]);
        end;
        writeln;
    end;
  felso_osszeg:=0; atlo_szorzat:=1; also_szorzat:=1;
  for i:=1 to 3 do
    for j:=1 to 3 do
     begin
       if i < j then felso_osszeg := felso_osszeg+w[i,j];
       if i = j then atlo_szorzat:= atlo_szorzat*w[i,j];
       if i > j then also_szorzat:= also_szorzat*w[i,j];
     end;
   writeln('Felső háromszög elemeinek összege: ',felso_osszeg);
   writeln('Átló elemeinek szorzata: ',atlo_szorzat);
   writeln('Alsó háromszög elemeinek szorzata: ',also_szorzat);
   readln;
end.
```

A program futásának eredménye:

```
w[ 1, 1] = 1
w[ 1, 2] = 2
w[ 1, 3] = 3

w[ 2, 1] = 4
w[ 2, 2] = 5
w[ 2, 3] = 6

w[ 3, 1] = 7
w[ 3, 2] = 8
w[ 3, 3] = 9
```

```
Felső háromszög elemeinek összege: 11
Átló elemeinek szorzata: 45
Alsó háromszög elemeinek szorzata: 224
```

Írjunk programot, amely egy 3x3-as mátrix főátlójába 2, felső háromszög elemeibe 1-et, és az alsó háromszög elemeibe 3-at tesz! Számítsuk ki a háromszög sorainak összegét és szorzatát! (*tomb17*)

```
program tomb17;
var
   i, j: integer;
   xx : array[1..3,1..3] of integer;
   szorzat:array[1..3] of integer;
   sor1_osszeg, sor2_osszeg, sor3_osszeg: integer;
begin
  writeln('A mátrix elemei');
  for i:=1 to 3 do
    begin
      for j:=1 to 3 do
        begin
          if i < j then xx[i,j]:=1;
          if i = j then xx[i,j]:=2;
          if i > j then xx[i,j]:=3;
          write(xx[i,j]:3);
        end;
        writeln;
    end;
  sor1_osszeg:=0; sor2_osszeg:=0; sor3_osszeg:=0;
  szorzat[1]:=1; szorzat[2]:=1; szorzat[3]:=1;
  for i:=1 to 3 do
    for j:=1 to 3 do
     begin
       case i of
        1: sor1_osszeg:= sor1_osszeg+xx[i,j];
        2: sor2_osszeg:= sor2_osszeg+xx[i,j];
        3: sor3_osszeg:= sor3_osszeg+xx[i,j];
      end;
      szorzat[i]:=szorzat[i]*xx[i,j];
    end;
   writeln('1. sor összege: ',sor1_osszeg);
   writeln('2. sor összege: ',sor2_osszeg);
   writeln('3. sor összege: ',sor3_osszeg);
   writeln;
   for i:=1 to 3 do
     writeln(i:1,'. sor szorzata: ',szorzat[i]);
   readln;
end.
```

A program futásának eredménye:

```
A mátrix elemei
  2  1  1
  3  2  1
  3  3  2
1. sor összege: 4
2. sor összege: 6
3. sor összege: 8
1. sor szorzata: 2
2. sor szorzata: 6
3. sor szorzata: 18
```

Írjunk programot, amely beolvas max. 80 karakter hosszú szöveget! Számláljuk meg a szövegben lévő magánhangzók és egyéb karakterek számát! (*tomb18*)

```
program tomb18;
var
   buffer: array[1..80] of char;
   c: char;
   i, maganh_db, egyeb_db: integer;
begin
  write('Szöveg 79 karakter (vége .): ');
  i:=0;
  read(c);
  while (c <> '.') do
  begin
     i:=i+1;
     buffer[i]:= c;
     if i = 80 then
     begin
       if buffer[80] <> '.' then
       begin
         writeln('80. karakter nem .'); i:=79;
         break;
       end;
     end;
     read(c);
  end;
  i:=i+1;
  buffer[i] := '.';
  egyeb_db:= 0; maganh_db:= 0;
  i:=1;
  while (buffer[i] <> '.') do
  begin
    if buffer[i] in ['a','e','i','o','u'] then
       maganh_db:=maganh_db+1
    else egyeb_db:= egyeb_db+1;
    i:=i+1;
  end;
```

```
  egyeb_db:=egyeb_db+1;
  writeln('A magánhangzók száma: ',maganh_db);
  writeln('Egyéb karakterek száma: ',egyeb_db);
  readln; readln;
end.
```

A program futásának eredménye:

```
Szöveg 79 karakter (vege .): Ismetles a tudas anyja.
A magánhangzók száma: 7
Egyéb karakterek száma: 16
```

Írjunk programot, amely a *teglalap* rekordban definiált téglalap két oldalát olvassuk be! Számítsuk ki a téglalap területét és kerületét! (*rekord01*)

```
program rekord01;
var
  teglalap : record
               a_oldal, b_oldal: real;
               ter, ker: real;
             end;
begin
  write('Téglalap a oldala: '); readln(teglalap.a_oldal);
  write('Téglalap b oldala: '); readln(teglalap.b_oldal);
  teglalap.ter:=teglalap.a_oldal*teglalap.b_oldal;
  teglalap.ker:=2*(teglalap.a_oldal+teglalap.b_oldal);
  writeln('A téglalap területe: ', teglalap.ter:6:2);
  writeln('A téglalap kerülete: ', teglalap.ker:6:2);
  readln;
end.
```

A program futásának eredménye:

```
Téglalap a oldala: 2
Téglalap b oldala: 3
A téglalap területe:   6.00
A téglalap kerülete:  10.00
```

Írjunk programot, amely beolvassa ***muvelet*** típusú rekord két operandusát és a műveleti jelét! Számítsuk ki a megadott adatokkal a kijelölt műveletet! (*rekord02*)

```
program rekord02;
type
     muvelet = record
               a, b, eredmeny: real;
               muvjel: char;
             end;
var
     m1: muvelet;
begin
   write('1. adat: '); readln(m1.a);
   write('2. adat: '); readln(m1.b);
```

```
   repeat
      write('művelet: '); readln(m1.muvjel);
   until m1.muvjel in ['+','-','*'];
   case m1.muvjel of
     '+' : m1.eredmeny:= m1.a + m1.b;
     '-' : m1.eredmeny:= m1.a - m1.b;
     '*' : m1.eredmeny:= m1.a * m1.b;
   end;
   writeln(m1.a:6:2,' ',m1.muvjel,' ',m1.b:6:2,' = ',m1.eredmeny:6:2);
   readln;
end.
```

A program futásának eredménye:

```
1. adat: 4
2. adat: 5
művelet: +
  4.00 +   5.00 =   9.00
```

Írjunk programot, amely beolvassa a *tavolsag* típusú rekord koordinátapontjainak *x* és *y* koordinátáit! Számítsuk ki a két koordinátapont közötti távolságot! (*rekord03*)

```
program rekord03;
type
     pont = record
               x, y: integer;
             end;
     tavolsag = record
             p1, p2 : pont;
             tav: real;
           end;
var  t: tavolsag;
begin
    writeln('1. koordinátapont ');
    write('   x: '); readln(t.p1.x);
    write('   y: '); readln(t.p1.y);
    writeln('2. koordinátapont ');
    write('   x: '); readln(t.p2.x);
    write('   y: '); readln(t.p2.y);
    t.tav:= sqrt(sqr(t.p1.x-t.p2.x)+ sqr(t.p1.y-t.p2.y));
    writeln('Távolság: ',t.tav:8:4);
    readln;
end.
```

A program futásának eredménye:

```
1. koordinátapont
   x: 1
   y: 1
2. koordinátapont
   x: 2
   y: 2
Távolság:   1.4142
```

Írjunk programot, amely beolvassa *teglatest_típus* típusú rekord három oldalát. Számítsuk ki a téglatest térfogatát és felszínét! (*rekord04*)

```
program rekord04;
type
  teglatest_tipus = record
                     a_oldal, b_oldal, c_oldal: real;
                     terfogat, felszin: real;
                    end;
var
     tg: teglatest_tipus;
begin
  write('Téglatest a oldala: '); readln(tg.a_oldal);
  write('Téglatest b oldala: '); readln(tg.b_oldal);
  write('Téglatest c oldala: '); readln(tg.c_oldal);

  tg.terfogat:= tg.a_oldal*tg.b_oldal*tg.c_oldal;
  tg.felszin:=2*(tg.a_oldal*tg.b_oldal+ tg.a_oldal*tg.c_oldal+
                 tg.b_oldal*tg.c_oldal);
  writeln('A téglatest térfogata: ', tg.terfogat:6:2);
  writeln('A téglatest felszíne : ', tg.felszin:6:2);
  readln;
end.
```

A program futásának eredménye:

```
Téglatest a oldala: 1
Téglatest b oldala: 2
Téglatest c oldala: 3
A téglatest térfogata:   6.00
A téglatest felszíne :  22.00
```

Írjunk programot, amely beolvassa a megadott elemszámú tömb *butor* típusú elemeinek adatait! Keressük meg a legdrágább és a legolcsóbb bútor azonosítóját és árát! Számláljuk meg az első osztályú és a másodosztályú butorok számát! (*rekordt01*)

```
program rekordt01;
const butor_db = 25;
type
     butor = record
               azonosito: string;
               db: integer;
               ar: real;
               minoseg: integer;
             end;
     raktar = array[1..butor_db] of butor;
var
     db, i: integer;
     r: raktar;
     legdragabb: butor;
```

```
     legolcsobb: butor;
     minoseg1_szama: integer;
     minoseg2_szama: integer;
begin
  repeat
      write('Bútortípusok száma: '); readln(db);
  until db in [1..20];
for i:=1 to db do
  begin
      writeln;
      write('Azonosító     : '); readln(r[i].azonosito);
      write(' áruk száma   : '); readln(r[i].db);
      write('        ára   : '); readln(r[i].ar);
      write('minőség(1,2)  : '); readln(r[i].minoseg);
  end;
  legdragabb:=r[1];
  legolcsobb:=r[1];
  for i:=1 to db do
  begin
    if legolcsobb.ar > r[i].ar then
       legolcsobb:= r[i];
    if legdragabb.ar < r[i].ar then
       legdragabb:= r[i];
    if r[i].minoseg = 1 then
       minoseg1_szama:= minoseg1_szama+r[i].db;
    if r[i].minoseg = 2 then
       minoseg2_szama:= minoseg2_szama+r[i].db;
  end;
  writeln;
  writeln('Legolcsóbb bútor: ',
          legolcsobb.azonosito,' á    ra: ',legolcsobb.ar:10:2);
  writeln('Legdrágább bútor: ',
          legdragabb.azonosito,' ára: ',legdragabb.ar:10:2);
  writeln('Selejtes bútorok száma: ',minoseg2_szama);
  writeln('Első osztályú bútorok száma: ',minoseg1_szama);
  readln;
end.
```

A program futásának eredménye:

```
Bútortípusok száma: 3

Azonosító     : asztal
 áruk száma   : 4
        ára   : 4500
minőség(1,2)  : 1

Azonosító     : szek
 áruk száma   : 12
        ára   : 2100
minőség(1,2)  : 2

Azonosító     : szekreny
 áruk száma   : 6
        ára   : 6000
minőség(1,2)  : 1
```

```
Legolcsóbb bútor: szek ára:    2100.00
Legdrágább bútor: szekreny ára:    6000.00
Selejtes bútorok száma: 12
Első osztályú bútorok száma: 10
```

Írjunk programot, amely beolvassa a megadott elemszámú tömb szemely rekord elemeinek adatait! Keressük meg a legnagyobb fizetésű, a legkevesebb szabadságú, legidősebb, a legfiatalabb rekordját! Számítsuk ki a fizetések összegét és az átlagfizetést! (*rekordt02*)

```
program rekordt02;
const max_letszam = 25;
type
     szemely = record
               nev: string[20];
               kora: integer;
               lakhely: string[20];
               fizetes: real;
               szabadsag: integer;
             end;
     munkahely = array[1..max_letszam] of szemely;
var
  letszam, i: integer;
  iroda: munkahely;
  legnagyobb_fizetes: szemely;
  legkevesebb_szabadsag: szemely;
  legidosebb: integer;
  legfiatalabb: integer;
  osszfizetes, atlagfizetes: real;
  bfegin
    repeat
       write('Alkalmazottak száma: '); readln(letszam);
    until letszam in [1..25];
    for i:=1 to letszam do
    begin
      writeln;
      write('Név      : '); readln(iroda[i].nev);
      write('Kora     : '); readln(iroda[i].kora);
      write('Lakhelye : '); readln(iroda[i].lakhely);
      write('Fizetés  : '); readln(iroda[i].fizetes);
      write('Szabadság: '); readln(iroda[i].szabadsag);
    end;
    legnagyobb_fizetes:= iroda[1];
    legkevesebb_szabadsag:= iroda[1];
    legidosebb:= iroda[1].kora;
    legfiatalabb:= iroda[1].kora;
    osszfizetes:= iroda[1].fizetes;
    for i:=2 to letszam do
    begin
      if legnagyobb_fizetes.fizetes < iroda[i].fizetes then
            legnagyobb_fizetes:= iroda[i];
```

```
      if legkevesebb_szabadsag.szabadsag > iroda[i].szabadsag then
            legkevesebb_szabadsag:= iroda[i];
      if legidosebb < iroda[i].kora then
             legidosebb:= iroda[i].kora;
      if legfiatalabb > iroda[i].kora then
             legfiatalabb:= iroda[i].kora;
      osszfizetes:= osszfizetes+iroda[i].fizetes;
    end;
    atlagfizetes:=osszfizetes/letszam;
    writeln;
    writeln('Legnagyobb fizetés: ',legnagyobb_fizetes.fizetes:8:1,
             ' neve: ',
             legnagyobb_fizetes.nev);
    writeln('Legkevesebb szabadság: ',
             legkevesebb_szabadsag.szabadsag,' neve:
             legkevesebb_szabadsag.nev);
   writeln('Legidősebb  : ',legidosebb);
   writeln('Legfiatalabb: ',legfiatalabb);
   writeln('Fizetések összege: ',osszfizetes:8:1);
   writeln('Átlagfizetés     : ',atlagfizetes:8:1);
   legkevesebb_szabadsag:= iroda[i];
      if legidosebb < iroda[i].kora then
             legidosebb:= iroda[i].kora;
      if legfiatalabb > iroda[i].kora then
             legfiatalabb:= iroda[i].kora;
      osszfizetes:= osszfizetes+iroda[i].fizetes;
    end;
    atlagfizetes:=osszfizetes/letszam;
    writeln;
    writeln('Legnagyobb fizetés: ',legnagyobb_fizetes.fizetes:8:1,
            ' neve: ', legnagyobb_fizetes.nev);
    writeln('Legkevesebb szabadság: ',
             legkevesebb_szabadsag.szabadsag,' neve:
             legkevesebb_szabadsag.nev);
   writeln('Legidősebb  : ',legidosebb);
   writeln('Legfiatalabb: ',legfiatalabb);
   writeln('Fizetések összege: ',osszfizetes:8:1);
   writeln('Átlagfizetés     : ',atlagfizetes:8:1);
   readln;
end.
```

A program futásának eredménye:

```
Alkalmazottak száma: 3

Név      : Kiss Katalin
Kora     : 19
Lakhelye : Bp
Fizetés  : 18000
Szabadság: 15

Név      : Nagy Marton
Kora     : 23
Lakhelye : Bp
Fizetés  : 25000
Szabadság: 12
```

```
Név      : Toth Andras
Kora     : 32
Lakhelye : Bp
Fizetés  : 45000
Szabadság: 6

Legnagyobb fizetés:  45000.0 neve: Toth Andras
Legkevesebb szabadság: 6 neve: Toth Andras
Legidősebb  : 32
Legfiatalabb: 19
Fizetések összege:  88000.0
Átlagfizetés     :  29333.3
```

Írjunk programot, amely beolvassa a tomb típusú struktúra adatait, melyben egy 5 elemű tömböt kell feltölteni adatokkal! Számítsuk ki az adatok összegét, átlagát és szorzatát! (*rekordt03*)

```
program rekordt03;
const db = 5;
type tomb = record
              x : array[1..db] of integer;
              osszeg, szorzat: integer;
              atlag: real;
            end;
var
   t: tomb;
   i: integer;
begin
   for i:=1 to db do
   begin
     write(i:2,'. elem: '); readln(t.x[i]);
   end;
   t.osszeg:= 0; t.szorzat:=1;
   for i:=1 to db do
   begin
     t.osszeg:= t.osszeg+t.x[i];
     t.szorzat:= t.szorzat*t.x[i];
   end;
   t.atlag:= t.osszeg/db;
   writeln;
   writeln('Összeg: ',t.osszeg);
   writeln('Átlag : ',t.atlag:6:2);
   writeln('Szorzat: ',t.szorzat);
   readln;
end.
```

A program futásának eredménye:

```
 1. elem: 3
 2. elem: 2
 3. elem: 5
 4. elem: 7
 5. elem: 4
```

```
Összeg: 21
Átlag :   4.20
Szorzat: 840
```

Írjunk programot, amely beolvassa a megadott elemszámú tömb *tavolsag* rekord típusú elemeinek adatait, azaz két koordinátapontnak az x és y koordinátáit! Számítsuk ki a két koordinátapont távolságát és keressük meg a legkisebb és a legnagyobb távolságot! (*rekordt04*)

```
program rekordt04;
type
     pont = record
               x, y: integer;
             end;
     tavolsag = record
             p1, p2 : pont;
             tav: real;
           end;
     tavTomb = array[1..10] of tavolsag;
var
     t: tavTomb;
     i, n: integer;
     legkisebb_tav, legnagyobb_tav: real;
begin
  repeat
    write('távolsagok száma: '); readln(n);
  until n in [2..10];
  for i:=1 to n do
  begin
    writeln;
    writeln('1. koordinátapont ');
    write('   x: '); readln(t[i].p1.x);
    write('   y: '); readln(t[i].p1.y);
    writeln('2. koordinátapont ');
    write('   x: '); readln(t[i].p2.x);
    write('   y: '); readln(t[i].p2.y);
    t[i].tav:= sqrt(sqr(t[i].p1.x-t[i].p2.x)+
               sqr(t[i].p1.y-t[i].p2.y));
    writeln('Tavolsag: ',t[i].tav:8:4);
  end;
  legkisebb_tav:= t[1].tav;
  legnagyobb_tav:= t[1].tav;
  for i:=1 to n do
  begin
    if legkisebb_tav > t[i].tav then legkisebb_tav:= t[i].tav;
    if legnagyobb_tav < t[i].tav then legnagyobb_tav:= t[i].tav;
  end;
  writeln;
  writeln('Legkisebb távolság : ',legkisebb_tav:6:2);
  writeln('Legnagyobb távolság: ',legnagyobb_tav:6:2);
  readln;
end.
```

A program futásának eredménye:

```
távolságok száma: 3

1. koordinátapont
   x: 2
   y: 2
2. koordinátapont
   x: 1
   y: 1
Távolság:   1.4142
1. koordinátapont
   x: 4
   y: 5
2. koordinátapont
   x: 2
   y: 3
Távolság:   2.8284

1. koordinátapont
   x: 3
   y: 3
2. koordinátapont
   x: 5
   y: 6
Távolság:   3.6056

Legkisebb távolság :   1.41
Legnagyobb távolság:   3.61
```

Írjunk programot, amely beolvassa a megadott elemszámú *muhely* típusú tömb *auto* típusú rekordjainak mezeit! Keressük meg a legnagyobb javítási költségű autót, a legrégebbi és a legfiatalabb kocsi gyártási évét! Számítsuk ki az össz javítási és az össz vizsga költséget! (*rekordt05*)

```
program rekordt05;
const max_autoszam = 12;
type
     auto  = record
               tipus: string[15];
               gyartasiev: integer;
               rendszam: string[7];
               javitas_ara: real;
               vizsgakoltseg: real;
             end;
     muhely = array[1..max_autoszam] of auto;
var
     auto_db, i: integer;
     szerviz: muhely;
     legnagyobb_javkoltseg: auto;
     ossz_vizsgakoltseg: real;
     legregebbi_kocsi: integer;
     legfiatalabb_kocsi: integer;
```

```
     ossz_javitasiar: real;
begin
  repeat
    write('Autók száma: '); readln(auto_db);
  until auto_db in [1..12];
  for i:=1 to auto_db do
  begin
    writeln;
    write('Autó típusa  : '); readln(szerviz[i].tipus);
    write('Gyártási éve : '); readln(szerviz[i].gyartasiev);
    write('Rendszáma    : '); readln(szerviz[i].rendszam);
    write('Javítás ára  : '); readln(szerviz[i].javitas_ara);
    write('Vizsgaköltség: '); readln(szerviz[i].vizsgakoltseg);
  end;
  legnagyobb_javkoltseg:= szerviz[1];
  ossz_vizsgakoltseg:= szerviz[1].vizsgakoltseg;
  legregebbi_kocsi:= szerviz[1].gyartasiev;
  legfiatalabb_kocsi:= szerviz[1].gyartasiev;
  ossz_javitasiar:= szerviz[1].javitas_ara;
  for i:=2 to auto_db do
  begin
    if legnagyobb_javkoltseg.javitas_ara < szerviz[i].javitas_ara then
       legnagyobb_javkoltseg:= szerviz[i];
    if legregebbi_kocsi > szerviz[i].gyartasiev then
       legregebbi_kocsi:= szerviz[i].gyartasiev;
    if legfiatalabb_kocsi < szerviz[i].gyartasiev then
       legfiatalabb_kocsi:= szerviz[i].gyartasiev;
    ossz_javitasiar:= ossz_javitasiar+szerviz[i].javitas_ara;
    ossz_vizsgakoltseg:=ossz_vizsgakoltseg+szerviz[i].vizsgakoltseg;
  end;
  writeln;
  writeln('Legnagyobb javítási költség    : ',
          legnagyobb_javkoltseg.javitas_ara:8:1,' neve: ',
          legnagyobb_javkoltseg.tipus);
  writeln('Legrégebbi kocsi gyártási éve  : ',legregebbi_kocsi);
  writeln('Legfiatalabb kocsi gyártási éve: ',legfiatalabb_kocsi);
  writeln('Össz javítási költség: ',ossz_javitasiar:8:1);
  writeln('Össz vizsgaköltség   : ',ossz_vizsgakoltseg:8:1);
  readln;
end.
```

A program futásának eredménye:

```
Autók száma: 3

Autó típusa  : mercedes
Gyártási éve : 2002
Rendszáma    : ABC-123
Javítás ára  : 12000
Vizsgaköltség: 3000

Autó típusa  : Skoda
Gyártási éve : 2003
Rendszáma    : ACD-456
Javítás ára  : 5000
Vizsgaköltség: 2000
```

```
Autó típusa  : Fiat
Gyártási éve : 2005
Rendszáma    : ACF-354
Javítás ára  : 2000
Vizsgaköltség: 1500

Legnagyobb javítási költség    :  12000.0 neve: mercedes
Legrégebbi kocsi gyártási éve  : 2002
Legfiatalabb kocsi gyártási éve: 2005
Össz javítási költség:  19000.0
Össz vizsgaköltség   :   6500.0
```

Írjunk programot, amely beolvassa a megadott elemszámú *raktar* típusú tömb *aru* típusú rekord elemeit! Számítsuk ki az össz értéket, átlag árat és az össz darabszámot! (*rekordt06*)

```
program rekordt06;
type
     aru = record
               azonosito: string[20];
               cikkszam: string[7];
               ar: real;
               db: integer;
             end;
     raktar = array[1..40] of aru;
var
     arufajta, i: integer;
     bolt : raktar;
     atlagar, osszertek:real;
     osszdarab: integer;
begin
  repeat
    write('Árufajták száma: '); readln(arufajta);
  until arufajta in [1..40];
  for i:=1 to arufajta do
  begin
    writeln;
    write('azonosító : '); readln(bolt[i].azonosito);
    write('cikkszám  : '); readln(bolt[i].cikkszam);
    write('ára       : '); readln(bolt[i].ar);
    write('darabszáma: '); readln(bolt[i].db);
  end;
  atlagar := 0; osszertek:=0; osszdarab:=0;
  for i:=1 to arufajta do
  begin
    atlagar:= atlagar+bolt[i].ar;
    osszertek:= osszertek+ bolt[i].ar*bolt[i].db;
    osszdarab:= osszdarab+bolt[i].db;
  end;
  atlagar:= atlagar/arufajta;
  writeln;
  writeln('Átlagár: ',atlagar:6:2);
```

```
  writeln('Összérték: ',osszertek:6:2);
  writeln('Összdarab: ',osszdarab);
  readln;
end.
```

A program futásának eredménye:

```
Árufajták száma: 3

azonosító : kalapacs
cikkszám  : 01
ára       : 500
darabszáma: 3

azonosító : veso
cikkszám  : 02
ára       : 200
darabszáma: 4

azonosító : fogo
cikkszám  : 03
ára       : 350
darabszáma: 5

Átlagár: 350.00
Összérték: 4050.00
Összdarab: 12
```

Írjunk programot, amely beolvassa a megadott elemszámú *adat_tomb* típusú tömb *adat* típusú rekordjainak *x, y* elemeit! Számítsuk ki a két adatának számtani és mértani közepét, a számtani közepek átlagát és a mértani közepek összegét! (*rekordt07*)

```
program rekordt07;
type
     adat = record
               x, y : real;
               szk, mk: real;
            end;
     adat_tomb = array[1..20] of adat;
var
     db, i: integer;
     w: adat_tomb;
     szk_atlag, mk_osszeg: real;
begin
  repeat
      write('Adatok száma: '); readln(db)
  until db in [1..20];
  for i:=1 to db do
  begin
    writeln;
    write('1. adat: '); readln(w[i].x);
    write('2. adat: '); readln(w[i].y);
```

```
    w[i].szk := (w[i].x + w[i].y)/2;
    w[i].mk := sqrt(w[i].x *w[i].y);
    writeln('szk: ',w[i].szk:6:2,'  mk: ',w[i].mk:6:2);
  end;
  szk_atlag:= 0; mk_osszeg:=0;
  for i:=1 to db do
  begin
    szk_atlag:= szk_atlag+w[i].szk;
    mk_osszeg:= mk_osszeg+w[i].mk;
  end;
  szk_atlag:=szk_atlag/db; writeln;
  writeln('Számtani közepek átlaga: ',szk_atlag:6:2);
  writeln('Mértani közepek összege: ',mk_osszeg:6:2);
  readln;
end.
```

A program futásának eredménye:

```
Adatok száma: 3

1. adat: 1
2. adat: 2
szk:   1.50  mk:   1.41

1. adat: 3
2. adat: 4
szk:   3.50  mk:   3.46

1. adat: 5
2. adat: 6
szk:   5.50  mk:   5.48

Számtani közepek átlaga:   3.50
Mértani közepek összege:  10.36
```

Írjunk programot, amely beolvassa a megadott elemszámú *tankor* típusú tömb *diak* típusu rekord elemeit! Keressük meg a legjobb átlagot és a legkisebb kreditpontot! (*rekordt08*)

```
program rekordt08;
type
     diak = record
               nev: string;
               kreditpont: integer;
               atlag: real;
            end;
     tankor = array[1..20] of diak;
var
     letszam, i: integer;
     tk1: tankor;
     minkredit: integer;
     maxatlag: real;
```

```
begin
  repeat
      write('Diákok száma: '); readln(letszam);
  until letszam in [1..20];

  for i:=1 to letszam do
  begin
    writeln;
    write('Diák neve  : '); readln(tk1[i].nev);
    write(' kreditpont: '); readln(tk1[i].kreditpont);
    write(' átlag     : '); readln(tk1[i].atlag);
  end;
  minkredit:=tk1[1].kreditpont;
  maxatlag:=tk1[1].atlag;
  for i:=2 to letszam do
  begin
    if minkredit > tk1[i].kreditpont then
       minkredit:= tk1[i].kreditpont;
    if maxatlag < tk1[i].atlag then
       maxatlag:= tk1[i].atlag;
  end;
  writeln;
  writeln('Legkisebb kreditpont: ',minkredit);
  writeln('Legjobb átlag: ',maxatlag:6:2);
  readln;
end.
```

A program futásának eredménye:

```
Diákok száma: 3

Diák neve  : Kiss Andras
 kreditpont: 120
 átlag     : 4.5

Diák neve  : Nagy Istvan
 kreditpont: 100
 átlag     : 3

Diák neve  : Toth Zoltan
 kreditpont: 180
 átlag     : 5

Legkisebb kreditpont: 100
Legjobb átlag:   5.00
```

Írjunk programot, amely halmazokon végez különféle halmazműveleteket és az eredményt megjeleníti! (*halmaz1*)

```
program halmaz1;
type
   ABC = (A,B,C,D,E,F,G,H);
   halmaz = set of ABC;
var
   h1, h2, h3, h4, h5: halmaz;
   hmuv1, hmuv2, hmuv3, hmuv4: halmaz;
   i: ABC;

begin
  h1:= [A,B,D,E];
  h2:= [A,D,E,G];
  h3:= [B,D,E];
  h4:= [F,G];
  hmuv1:= h1 + h2;
  hmuv2:= h1 * h2;
  hmuv3:= (h3 + h4) * h2;
  hmuv4:= h1 - h3;
  write('hmuv1 halmaz elemei: ');
  for i:=A to H do
  begin
    if i in hmuv1 then
    begin
      case i of
        A: write('A');
        B: write('B');
        C: write('C');
        D: write('D');
        E: write('E');
        F: write('F');
        G: write('G');
        H: write('H');
      end;
    end;
  end;
  writeln;
  write('hmuv2 halmaz elemei: ');
  for i:=A to H do
  begin
    if i in hmuv2 then
    begin
      case i of
        A: write('A');
        B: write('B');
        C: write('C');
        D: write('D');
        E: write('E');
        F: write('F');
        G: write('G');
        H: write('H');
```

```
      end;
    end;
  end;
  writeln;
  write('hmuv3 halmaz elemei: ')
  for i:=A to H do
  begin
    if i in hmuv3 then
    begin
      case i of
        A: write('A');
        B: write('B');
        C: write('C');
        D: write('D');
        E: write('E');
        F: write('F');
        G: write('G');
        H: write('H');
      end;
    end;
  end;
  writeln;
  write('hmuv4 halmaz elemei: ');
  for i:=A to H do
  begin
    if i in hmuv4 then
    begin
      case i of
        A: write('A');
        B: write('B');
        C: write('C');
        D: write('D');
        E: write('E');
        F: write('F');
        G: write('G');
        H: write('H');
      end;
    end;
  end;
  writeln;
  readln;
end.
```

A program futásának eredménye:

```
hmuv1 halmaz elemei: ABDEG
hmuv2 halmaz elemei: ADE
hmuv3 halmaz elemei: DEG
hmuv4 halmaz elemei: A
```

Írjunk programot, amely egy tíz elemű egész típusú tömb minden elemét elhelyezi a csere irányától függően eggyel nagyobb, vagy eggyel kisebb indexű helyre, (a tizedik elemet az első, ill. az első elemet a tizedik helyére), majd írassuk ki a keletkezett tömböt (*csere*)

```
program csere;
var tomb:array[1..10] of integer;
    i,temp:integer;
    irany:char;
begin
  writeln('Kérem a tömb elemeit!');
  for i:= 1 to 10 do
  begin
       write('tömb[',i,']= ');
       readln(tomb[i]);
  end;
  writeln('A tömb:');
  for i:=1 to 10 do
     write(tomb[i]:6);
  writeln;
  writeln('A csere iránya: felfelé vagy lefelé (f/l)');
  repeat
     write('Irány: ');
     readln(irany);
  until irany in ['f','F','l','L'];
  writeln('A tömb a ciklikus csere után:');
  if (irany='f') or (irany='F') then
  begin
    temp:=tomb[10];
    for i:=9 downto 1 do
      tomb[i+1]:=tomb[i];
    tomb[1]:=temp;
  end
  else
  begin
    temp:=tomb[1];
    for i:=1 to 9 do
      tomb[i]:=tomb[i+1];
    tomb[10]:=temp;
  end;
  for i:= 1 to 10 do
     write(tomb[i]:6);
  writeln;
  readln ;
end.
```

A program futásának eredménye:

```
Kérem a tömb elemeit!
tömb[1]= 1
tömb[2]= 2
tömb[3]= 3
tömb[4]= 4
```

```
tömb[5]= 5
tömb[6]= 6
tömb[7]= 7
tömb[8]= 8
tömb[9]= 9
tömb[10]= 10
A tömb:
     1     2     3     4     5     6     7     8     9    10
A csere iránya: felfelé vagy lefelé (f/l)
Irány: f
A tömb a ciklikus csere után:
    10     1     2     3     4     5     6     7     8     9
```

Írjunk programot, amelyben hozzunk létre és írassunk vissza egy 5*7-es kétdimenziós tömböt úgy, hogy a tömb elemei a két index összegét vegyék fel! (*ketdim*)

```
program ketdim;
var i,j:integer;
    tomb:array[1..5,1..7] of integer;
begin
  for i:=1 to 5 do
    for j:=1 to 7 do
       tomb[i,j]:=i+j;
  writeln('A tömb elemei: ');   { Kiíratás }
  writeln;
  for i:=1 to 5 do
    begin
     for j:=1 to 7 do
        write(tomb[i,j]:5);
     writeln;writeln;
    end;
  readln;
 end.
```

A program futásának eredménye:

```
A tömb elemei:
    2    3    4    5    6    7    8
    3    4    5    6    7    8    9
    4    5    6    7    8    9   10
    5    6    7    8    9   10   11
    6    7    8    9   10   11   12
```

Írjunk programot, amelyben beolvasunk egy max. 40 karakter hosszúságú karaktersorozatot, és jelenítsük meg a képernyőn, a sor közepére illesztve. Egy képernyősorban 80 karakter helyezhető el! (*kozepre*)

```
program kozepre;
var buff:string[40];
    k:integer;
begin
   writeln('Kérek egy max. 40 karakter hosszúságú szöveget!');
   write('Szöveg: ');
   readln(buff);
   writeln;
   k:=(80-ord(buff[0])) div 2+ord(buff[0]);
   writeln(buff:k);
   readln;
end.
```

A program futásának eredménye:

```
Kérek egy max. 40 karakter hosszúságú szöveget!
Szöveg: ma szep az ido.

                                ma szep az ido.
```

Írjunk programot, amely beolvas egy max 80 karakter hosszúságú sztringet! Vizsgáljuk meg, hogy a sztringben hol fordulnak elő numerikus karakterek és a pozíciójukat is jelenítsük meg! (*numerikus*)

```
program numerikus;
var sor:string[80];
    i:integer;
    ch:char;
    num:boolean;
begin
  writeln('Kérek egy max. 80 karakter hosszúságú karaktersorozatot!');
  write('sor: ');
  readln(sor);
  num:=true;
  for i:=1 to ord(sor[0]) do
  begin
    if sor[i] in ['0'..'9'] then
    begin
    num:=false;
    writeln(i,'. elem: ',sor[i]:6);
    end;
  end;
  if num then writeln('Nincsenek numerikus karakterek');
  readln;
end.
```

A program futásainak eredménye:

```
Kérek egy max. 80 karakter hosszúságú karaktersorozatot!
sor: Ma augusztus 6. van, este 8 ora.
14. elem:      6
27. elem:      8

Kérek egy max. 80 karakter hosszúságú karaktersorozatot!
sor: Szep az ido.
Nincsenek numerikus karakterek
```

Írjunk programot, amely egy n*n-es kétdimenziós tömböt listáz sorfolytonosan, oszlopfolytonosan, csak a főátlóban lévő elemeket! (*tomblist*)

```
program tomblist;
var tomb:array[1..10,1..10]of integer;
    szam,i,j:integer;
begin
  writeln('Kérem a tömb sor és oszlopszámát (max 10)!');
  repeat
    write('Elemszám: ');
    readln(szam);
  until szam in [1..10];
  writeln('Kérem a tömb elemeket');
  for i:= 1 to szam do
  begin
   for j:= 1 to szam do
    begin
      write('tomb[',i,',',j,']= ');
      readln(tomb[i,j]);
    end;
    writeln;
  end;
  writeln('A tömb: ');
  for i:=1 to szam do
   begin
     for j:=1 to szam do
       write(tomb[i,j]:6);
     writeln;
   end;
  writeln('A tömb elemei sorfolytonosan:');
  for i:=1 to szam do
    for j:=1 to szam do
      write(tomb[i,j]:6);
    writeln;
  writeln('A tömb elemei oszlopfolytonosan:');
  for i:=1 to szam do
    for j:=1 to szam do
       write(tomb[j,i]:6);
    writeln;
```

```
  writeln('A főátlóban lévő elemek:');
  for i:=1 to szam do
    for j:= 1 to szam do
     if i=j then writeln('tomb[',i,',',j,']= ',tomb[i,j]);
  readln;
end.
```

A program futásának eredménye:

```
Kérem a tömb sor és oszlopszámát (max 10)!
Elemszám: 3
Kérem a tömb elemeket
tomb[1,1]= 1
tomb[1,2]= 2
tomb[1,3]= 3

tomb[2,1]= 4
tomb[2,2]= 5
tomb[2,3]= 6

tomb[3,1]= 7
tomb[3,2]= 8
tomb[3,3]= 9

A tömb:
     1     2     3
     4     5     6
     7     8     9
A tömb elemei sorfolytonosan:
     1     2     3     4     5     6     7     8     9
A tömb elemei oszlopfolytonosan:
     1     4     7     2     5     8     3     6     9
A főátlóban lévő elemek:
tomb[1,1]= 1
tomb[2,2]= 5
tomb[3,3]= 9
```