

2.6. Utasítások használata

1. Két pont távolsága: [Rekord1](#)
2. A kör területének és kerületének számítása: [Rekord2](#)
3. A körgyűrűk területének számítása és a maximális terület megkeresése: [Rekord3](#)
4. Milliméterben megadott adat átváltása: [Mm_m](#)
5. Másodpercben megadott adat átváltása: [Ora](#)
6. A víz halmazállapotának meghatározása: [Feltetel1](#)
7. Pontszámból osztályzat megállapítása: [Feltetel2](#)
8. Max, min és átlag számítása: [Feltetel3](#)
9. Három adat helyén való sorbarendezése: [Feltetel4](#)
10. Pozitív, negatív vagy zérus adat meghatározása: [Feltetel5](#)
11. Egy gyümölcs magánhangzóval vagy mássalhangzóval kezdődik: [Feltetel6](#)
12. Számjeggyel megadott osztályzatot szövegesen adja meg: [Case1](#)
13. Kalkulátor készítése (+, -, *, /): [Case2](#)
14. Elért pontszámból osztályzat megállapítása: [Case3](#)
15. Adott határ közé eső adatok számlálása: [Case4](#)
16. 15-255 közötti karakterkódok kiírása: [Ciklus1](#)
17. Az első tíz szám összege **for** ciklussal: [Ciklus2](#)
18. Az első tíz szám összege **while** ciklussal: [Ciklus3](#)
19. Az első tíz szám összege **repeat-until** ciklussal: [Ciklus4](#)
20. Faktoriális számítása: [Ciklus5](#)
21. $\sin(x)/x$ tabellázása: [Ciklus6](#)
22. Kamatos kamat számítása: [Ciklus7](#)
23. 10-500 közötti páros számok, a tízes helyen páratlanok kiírása: [Ciklus8](#)
24. Valós tömb pozitív, negatív és zérus elemeinek számlálása, összeg, átlag számítása: [Tomb1](#)
25. A egész típusú tömb páros és páratlan elemeinek számlálása : [Tomb2](#)
26. Két vektor skalárszorzata: [Tomb3](#)
27. A tömb minimális és maximális elemének keresése a hozzátartozó indexszel: [Tomb4](#)
28. Bűvös négyzet vizsgálata: [Buvos](#)
29. Sztring kerektereiről előfordulási statisztika: [Chstat](#)
30. 0-100 között gondolt szám kitalálása: [Jatek](#)
31. Kockadobás statisztikája: [Kocka](#)
32. Negyjegyű számadat fordított sorrendben történő visszaírása: [Fordit](#)
33. Születési dátumról megmondja melyik napra esett: [Oroknep](#)



Tervezzünk egy olyan programot, amely kiszámítja két pont távolságát! (*Rekord1*)

A program beolvas két koordinátpontot és kiszámítja a két pont távolságát. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program rekord1;
{$APPTYPE CONSOLE}
uses
  SysUtils;

type
  koord = record
    x1,y1,x2,y2: integer;
  end;
var
  d : koord;
  tav : real;
begin
  writeln('Ket pont tavolsaganak szamitasa');
  writeln;
  write('1. pont x koordinataja: '); readln(d.x1);
  write('1. pont y koordinataja: '); readln(d.y1);
  writeln;
  write('2. pont x koordinataja: '); readln(d.x2);
  write('2. pont y koordinataja: '); readln(d.y2);
  tav := sqrt(sqr(d.x2-d.x1)+sqr(d.y2-d.y1));
  writeln;
  writeln('A ket pont tavolsaga: ',tav:6:2);
  readln;
end.
```



Készítsünk egy olyan programot, amely beolvassa a kör sugarát és kiszámítja a kör területét és kerületét! (*Rekord2*)

A program a *kor* típusú rekord adatmezőin végez műveleteket. Beolvassa a *sugar* mező tartalmát és kiszámítja a *kerulet* és *terulet* adatmezők értékét. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáit. A program csak számokat olvas be.

```
program rekord2;
{$APPTYPE CONSOLE}
uses
  SysUtils;
type
  kor = record
    sugar, kerulet, terulet : real;
  end;
var
  k : kor;
begin
  write('A kor sugara: '); readln(k.sugar);
  writeln;
  k.terulet := sqr(k.sugar) * pi;
  k.kerulet := 2 * k.sugar * pi;
  writeln('A kor terulete: ', k.terulet:5:2);
  writeln('A kor kerulete: ', k.kerulet:5:2);
  readln;
end.
```



Írjunk egy olyan programot, amely beolvassa a körgyűrűk külső és belső sugarát, kiszámítja a területüket és megkeresi a maximális területű körgyűrűt! (*Rekord3*)

A program rákérdez a beolvasandó körgyűrűk számára, beolvassa a külső és belső sugarukat, kiszámítja a területüket. Megkeresi a maximális területű körgyűrűt, valamint a hozzá tartozó tömbindex értékét. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program rekord3;
{$APPTYPE CONSOLE}
uses
  SysUtils;

type
  korgyuru = record
    Kulso_sugar, Belso_sugar, terület: real;
  end;
  tomb = array[1..15] of korgyuru;
var
  n,i,maxindex : integer;
  maxter: real;
  kgy : tomb;
begin
  write('A korgyuruk szama: '); readln(n);
  for i:=1 to n do
    begin
      writeln(i:2, '. korgyuru:');
      write('Kulso sugar: '); readln(kgy[i].Kulso_sugar);
      write('Belso sugar: '); readln(kgy[i].Belso_sugar);
      writeln;
      kgy[i].terulet := (sqr(kgy[i].Kulso_sugar) - sqr(kgy[i].Belso_sugar))*pi;
    end;
  maxter:= kgy[1].terulet;
  maxindex := 1;
  for i:=2 to n do
    if kgy[i].terulet > maxter then
      begin
        maxter := kgy[i].terulet;
        maxindex:= i;
      end;
  writeln(maxindex:3, '. korgyurunek van maximalis terulete');
  writeln('Kulso sugara: ', kgy[maxindex].Kulso_sugar:5:2);
  writeln('Belso sugara: ', kgy[maxindex].Belso_sugar:5:2);
  writeln('A maximalis terület : ', maxter:6:2);
  readln;
end.
```



Tervezzünk egy olyan programot, amely milliméterben beolvasott távolságot átalakítja m, cm és mm-re! (*Mm_m*)

A program beolvassa a távolságot milliméterben és **div**, **mod** műveletek segítségével átalakítja m, cm és mm-re. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program mm_m;
{$APPTYPE CONSOLE}
uses
    SysUtils;

var    m,cm,mm,ertek :longint;
begin
    writeln('mm-ben adott tavolsag szetbontasa m cm es mm egysegekre');
    writeln;
    write('Adat mm-ben : '); readln(ertek);
    m:=ertek div 1000;
    mm:= ertek mod 10;
    cm:= (ertek mod 1000) div 10;
    writeln( ertek,' mm = ',m,' m ', cm ,' cm ', mm ,' mm');
    readln;
end.
```



Készítsünk egy olyan programot, amely másodpercben megadott időadatot átszámítja óra, perc, másodpercre! (*Oraprogram*)

A program beolvassa az időadatot másodpercben és **div**, **mod** műveletek segítségével átalakítja óra, perc és másodpercre. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program oraprogram;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  ora,perc,mp,ertek :longint;
begin
  writeln('Ido atszamitasa masodpercbol ora perc es mp egysegekre ');
  writeln;
  write('Az idoadat masodpercben : '); readln(ertek);
  ora:=ertek div 3600;
  perc:= (ertek mod 3600) div 60;
  mp:= ertek mod 60;
  writeln( ertek,' masodperc = ',ora,' ora ',
          perc,' perc ', mp,' masodperc' );
  readln;
end.
```



Tervezzünk egy olyan programot, amely a víz halmazállapotát határozza meg a víz hőmérsékletéből!
(*Feltétel*)

A program bekéri a víz hőmérsékletét és láncolt **if** utasítással megállapítja a víz halmazállapotát és ezt írja vissza. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program feltetel1;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  t : real;
begin
  writeln('A víz halmazállapotának vizsgálata');
  writeln;
  write('hőmérséklet: '); readln(t);
  write('halmazállapota: ');
  if t > 0 then
    begin
      if t >= 100 then
        writeln('gőz')
      else
        writeln('víz');
      end
    else writeln('jég');
    readln;
  end.
```



Tegyük fel, hogy a dolgozat megírásakor elérhető maximális pontszám 100, és az érdemjegyek alsó határa rendre 50, 60, 70 és 80! A program bekéri az elért pontszámot és láncolt **if** utasítással szövegesen kiírja a dolgozatra kapott jegyet. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Feltetel2;
{$APPTYPE CONSOLE}
uses
  SysUtils;
var
  pont: 0..100;
begin
  writeln('0 es 100 kozotti pontok osztalyzata');
  writeln;
  write('Az elert pontszam: '); readln(pont);
  write('A dolgozat osztalyzata: ');
  if pont = 0 then writeln('Ervenytelen')
  else if pont < 50 then writeln('elegtelen')
  else if pont < 60 then writeln('elegseges')
  else if pont < 70 then writeln('kozepes')
  else if pont < 80 then writeln('jo')
  else writeln('jeles');
  readln;
end.
```




Tervezzünk egy olyan programot, amely három egész típusú adatból megkeresi a maximális és a minimális értéket, valamint kiszámítja az átlagukat! (*Feltétel3*)

A program bekér három egész típusú adatot. Az adatok közül **if** utasításokkal kiválasztja a minimális és a maximális értéket, majd kiszámítja a maximális és a minimális adatok átlagát. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Feltetel3;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  a,b,c,max,min : integer;
  atlag : real;
begin
  write('a: '); readln(a);
  write('b: '); readln(b);
  write('c: '); readln(c);

  max := a;
  min := a;
  if b > max then max := b;
  if c > max then max := c;
  if b < min then min := b;
  if c < min then min := c;
  writeln;
  writeln('Min ertek: ',min);
  writeln('Max ertek: ',max);
  atlag := (max+min)/2;
  writeln('Max es Min atlaga: ',atlag:6:2);
  readln;
end.
```



Írjunk egy olyan programot, amely három változó tartalma alapján növekvő sorrendbe írja vissza az adatokat! (*Feltétel4*)

A program bekér a három egész típusú adatot és három **if** utasítás felhasználásával úgy cseréli a változók tartalmát, hogy végül a változók növekvő sorrendben fogják tartalmazni az adatokat. Két változó tartalmának a cseréjéhez munkarekeszt kell használnunk. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Feltetel4;
{$APPTYPE CONSOLE}
uses
  SysUtils;
var
  x, x1,x2,x3 : integer;
begin
  write('1. szam : '); readln(x1);
  write('2. szam : '); readln(x2);
  write('3. szam : '); readln(x3);
  if x1 > x2 then
    begin
      x := x1;
      x1 := x2;
      x2 := x;
    end;
  if x1 > x3 then
    begin
      x := x1;
      x1 := x3;
      x3 := x;
    end;
  if x2 > x3 then
    begin
      x := x2;
      x2 := x3;
      x3 := x;
    end;
  writeln;
  writeln('A szamok novekvo sorrendben: ',x1:4,x2:4,x3:4);
  readln;
end.
```



Tervezzünk egy olyan programot, amely a beolvasott adatról megállapítja, hogy pozitív, negatív vagy zérus! (*Feltétel5*)

A program bekéri az adatot, és feltételek kiértékelésével szövegesen írja ki, hogy az adat pozitív, negatív vagy zérus. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Feltetel5;
{$APPTYPE CONSOLE}
uses
  SysUtils;
var
  a : real;
begin
  write('Adat: ');
  readln(a);
  if a > 0 then writeln('Az adat pozitív');
  if a < 0 then writeln('Az adat negatív');
  if a = 0 then writeln('Az adat zérus');
  readln;
end.
```



Készítsünk egy olyan programot, amely beolvassa egy gyümölcs nevét és megállapítja, hogy az magánhangzóval vagy mássalhangzóval kezdődik-e! (*Feltétel6*)

A program bekéri a gyümölcs nevét. A neveket csak kisbetűvel írhatjuk. Képezi a kisbetűs magánhangzók és a mássalhangzók halmazát, majd a gyümölcs első karakterét vizsgálja, és annak megfelelően, hogy a kezdőbetű melyik halmazba tartozik, írja ki az eredményt. A *length* függvénnyel meghatározza a gyümölcs karaktereinek számát.

```
program Feltetel6;
{$APPTYPE CONSOLE}
uses
  SysUtils;

type
  abc = set of 'a'..'z';
var
  magan, massalh : abc;
  gyum : string[50];
  hossza : integer;

begin
  write('Gyumolcs (kisbetuvel):');
  readln(gyum);
  magan:= ['a','e','i','o','u'];
  massalh:= ['a'..'z'] - magan;
  hossza := length(gyum);
  writeln;
  if gyum[1] in magan then writeln('Maganhangzoval kezdodik.');
```

if gyum[1] in massalh then writeln('Massalhangzoval kezdodik.');

writeln('A szo hossza: ',hossza);

readln;

end.



Tervezzünk egy olyan programot, amely beolvassa az osztályzat értékét, és szövegesen írja vissza!
(Case1)

A program bekéri a számjegyes osztályzatot és **case** utasítást használva, a kiválasztott jegynek megfelelő szöveges osztályzatot írja vissza. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Case1;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  osztalyzat : integer;
  jegy : string;
begin
  write('Az osztalyzat: ');
  readln(osztalyzat);
  case osztalyzat of
    1: jegy := 'elegtelen';
    2: jegy := 'elegseges';
    3: jegy := 'kozepes';
    4: jegy := 'jo';
    5: jegy := 'jeles';
    else jegy := 'Hibas adat.';
  end;
  writeln('A jegy: ', jegy);
  readln;
end.
```



Írjunk egy olyan programot, amely kalkulátorként működik, összeadás, kivonás, szorzás és az osztás művelet végrehajtására alkalmas! (*Case2*)

A program bekéri az első operandust, a műveleti jelet majd a másik operandust. A műveleti jelnek megfelelően elvégzi a műveletet. Ha nullával oszt, vagy hibás a műveleti jel, akkor hibajelzést ad. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program az operandusok helyén csak számokat olvas be.

```
program Case2;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  a,b,eredmeny : real;
  muveletijel   : char;
  jo : boolean;
begin
  write('1. operandus: '); readln(a);
  write('Muveleti jel (+,-,/,*): ');
  readln(muveletijel);
  write('2. operandus: '); readln(b);
  writeln;

  jo := true;
  case muveletijel of
    '+': eredmeny := a + b;
    '-': eredmeny := a - b;
    '/': begin
      if b > 0 then
        eredmeny := a / b
      else
        begin
          eredmeny := 0;
          write(' 0-val valo osztas miatt hibas muvelet');
          jo := false;
        end;
      end;
    '*': eredmeny := a * b;
  else begin
    writeln('Hibas a muveleti jel');
    jo := false;
  end;
end;
if jo then
  writeln(a:6:2, ' ', muveletijel, ' ', b:6:2, ' = ', eredmeny:6:2);
readln;
end.
```



A program bekéri az elért pontszámot. A **case** utasítással felállított ponthatárok közé eső pontról szöveges osztályzatot ad eredményként. Tegyük fel, hogy a dolgozat megírásakor elérhető maximális pontszám 100 és az érdemjegyek 50, 60, 70 és 90! Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Case3;

{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  pont : 0 .. 100;
begin
  writeln('0 - 100 között elért pontok osztályzata: ');
  write('Pontszám: '); readln(pont);
  writeln;
  write('A dolgozat osztályzata: ');
  case pont of
    90..100: writeln('jeles');
    70.. 89: writeln('jo');
    60.. 69: writeln('kozepes');
    50.. 59: writeln('elegseges');
    0.. 49: writeln('elegtelen');
    else writeln('Hibas adat!');
  end;
  readln;
end.
```



Készítsünk egy olyan programot, amely beolvasott adatokból statisztikát készít, mennyi esik a megadott határok közé! (*Case4*)

A program bekéri az adatok számát (legfeljebb 30), majd az adatok számának megfelelő darab nem negatív egész adatot ciklusban beolvas. A ciklus futása alatt vizsgáljuk, hogy a szám melyik 0,1,10,100, és 1000 határokkal jellemezhető intervallumba esik és a különböző szakaszok elemeit megszámláljuk. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Case4;
{$APPTYPE CONSOLE}
uses
  SysUtils;

const m=30;
var
  n,db0,db1,db2,db3,db4,i:integer;
  adat:array[1..m] of word;
begin
  write('Adatok szama (max. 30): ');
  readln(n);
  for i:=1 to n do
    begin
      write('adat[' ,i:2,']: ');
      readln(adat[i]);
    end;
  writeln;
  db0:=0;db1:=0; db2:=0; db3:=0; db4:=0;
  for i:=1 to n do
    begin
      case adat[i] of
        0      : inc(db0);
        1..10   : inc(db1);
        11..100 : inc(db2);
        101..1000 : inc(db3);
        else    : inc(db4);
      end;
    end;
  writeln(n:4,' adatbol: ');
  writeln('0      ',db0:4);
  writeln('1..10   ',db1:4);
  writeln('11..100   ',db2:4);
  writeln('101..1000 ',db3:4);
  writeln('      >1000 ',db4:4);
  readln;
end.
```




A program egy **for** ciklus ciklusváltozóját 15-255 közötti értékeken futtatja végig, kiírja a kód (ciklusváltozó) értékét és a hozzá tartozó karaktert, valamint minden tíz adat után új sort kezd.

```
program Ciklus9;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  i,j : integer;
begin
  writeln('15-255 kozotti karakterek es a kodjuk');
  writeln;
  j := 0;
  for i := 15 to 255 do
    begin
      j := j+1;
      write(i:4, char(i):2);
      if j mod 11 = 0 then
        begin
          writeln;
        end;
    end;
  readln;
end.
```



Tervezzünk egy olyan programot, amely az első tíz egész számot összeadja **for** ciklus használatával!
(*Ciklus2*)

A program **for** ciklussal adja össze az első tíz számot.

```
program ciklus2;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  szam, osszeg : integer;
begin
  osszeg := 0;
  for szam := 1 to 10 do
    osszeg := osszeg + szam;
  writeln('Az osszeg for ciklussal : ', osszeg);
  readln;
end.
```



Készítsünk egy olyan programot, amely az első tíz egész számot összeadja **while** ciklus használatával!
(*Ciklus3*)

A program **while** ciklussal adja össze az első tíz számot.

```
program ciklus3;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  szam, osszeg : integer;
begin
  szam := 1;
  osszeg := 0;
  while szam < 11 do
  begin
    osszeg := osszeg + szam;
    szam := szam+1;
  end;
  writeln('Az osszeg while ciklussal : ',osszeg);
  readln;
end.
```



Írjunk egy olyan programot, amely az első tíz egész számot összeadja **repeat-until** ciklus használatával! (*Ciklus4*)

A program **repeat-until** ciklussal adja össze az első tíz számot.

```
program ciklus4;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  szam, osszeg : integer;
begin
  szam := 1;
  osszeg := 0;
  repeat
    osszeg := osszeg + szam;
    szam := szam+1;
  until szam >= 11;
  writeln('Az osszeg repeat-until ciklussal : ',osszeg);
  readln;

end.
```



A program **while** ciklussal kiszámítja az adatként beolvasott faktoriális értékét. A program célja a ciklusutasítás használatának bemutatása. Nem foglalkozunk azzal, hogy a faktoriális kiszámításakor túlléphetünk a számábrázolási határokon.

```
program Ciklus5;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  f, n : integer;
begin
  writeln('Faktoriális számítása');
  write('n : '); readln(n);
  write(n:3, ' ! = ');
  f := 1;
  while( n > 0 ) do
  begin
    f := f*n;
    dec(n);
  end;
  writeln(f:10);
  readln;
end.
```



A program -1-től 1-ig, 0.1 tizedes léptékkal táblázatot készít a $\sin(x)/x$ függvényről **while** ciklus használatával. A 0 helyen a függvény nem értelmezett, így ezt kihagyjuk a táblázatból.

```
program ciklus6;
{$APPTYPE CONSOLE}
uses
  SysUtils;

const
  a = -1.0; b = 1.0; lepes = 0.1;
var
  x:real;
begin
  writeln('x':7, 'sin(x)/x':18);
  writeln('=====');
  x := a-lepes;
  while x <= b do
  begin
    x := x + lepes;
    if abs(x) < 1e-11 then continue;
    writeln(x:10:5, sin(x)/x: 15:6);
  end;
  readln;
end.
```



A program beolvassa az alaptőkét, a lekötés évét, a kamatlábat és kiszámítja a tőke értékét kamatos-kamat számítással. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Ciklus7;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  alap,t,kamat : real;
  ev,i : integer;
begin
  writeln('Kamatos kamat szamitasa');
  writeln;
  write('Alap toke [Ft] : '); readln(alap);
  write('Lekotes eve: '); readln(ev);
  write('Kamat [%] : '); readln(kamat);
  t := alap;
  kamat := kamat/100;
  for i := 1 to ev do
    t := t + t*kamat;
  writeln;
  writeln(ev, ' ev mulva ', trunc(t), ' Ft lesz a betet');
  readln;
end.
```



Tervezzünk egy olyan programot, amely kiírja a 10-500 közötti páros számokat, melyek a tízes helyen páratlanok! (*Ciklus8*)

A program **for** ciklust használva a ciklusváltozót (*i*) 5-250.-ig futtatja. A ciklusban a vizsgált szám (*x*) a ciklusváltozó kétszerese, így csak a páros számokat vizsgáljuk. Az *x* vizsgált számot tízzel osztva és az eredmény egész részét véve (*trunc*) az *odd* függvénnyel eldönthetjük, hogy a szám páratlan-e. A *j* változó segítségével minden tíz kiírt szám után új sort kezdünk.

```
program ciklus8;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  i,j,x: integer;
begin
  writeln('10 es 500 kozotti paros szamok, melyek a ' );
  writeln('tizes helyen paratlan szamot tartalmaznak');
  writeln;
  j := 1;
  for i:=5 to 250 do
  begin
    x := i*2;
    if odd(trunc(x/10)) then
    begin
      write(x:4);
      if j mod 10 = 0 then
      begin
        writeln;
        j := 0;
      end;
      j := j+1;
    end;
  end;
  readln;
end.
```




Készítsünk egy olyan programot, amely a megszámolja a tömb pozitív, negatív és zérus elemeit, kiszámítja a tömb elemeinek összegét és átlagát! (*Tomb1*)

A program beolvassa a tömb elemszámát, majd feltölti a tömböt. A *poz*, *neg* és *zero* változókba megszámolja a tömb pozitív, negatív és zérus elemeinek számát. Az *osszeg* változóba kiszámítja a tömb elemeinek összegét, az *atlag* változóba pedig az átlagot. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program tomb1;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  i,db,poz,neg,zero : integer;
  osszeg,atlag : real;
  y : array[1..20] of real;
begin
  write('A tomb elemeinek szama: ');
  readln(db);
  for i :=1 to db do
    begin
      write(i:2, '.elem: ');
      readln(y[i]);
    end;
  poz := 0; neg := 0; zero := 0;
  i:=1;
  while i<=db do
    begin
      if y[i] < 0 then neg := neg+1;
      if y[i] > 0 then poz := poz+1;
      if y[i] = 0 then zero := zero+1;
      i:=i+1;
    end;
  osszeg := 0;
  i:= 0;
  repeat
    i:=i+1;
    osszeg := osszeg+y[i];
  until i = db;
  atlag := osszeg/db;
  writeln;
  writeln('Pozitiv elemek szama: ',poz);
  writeln('Negativ elemek szama: ',neg);
  writeln('Zerus elemek szama : ',zero);
  writeln;
  writeln('A tomb elemeinek osszege: ',osszeg:8:2);
  writeln('A tomb elemeinek atlaga : ',atlag:8:2);
  readln;
end.
```



Írjunk egy olyan programot, amely az egész típusú elemeket tartalmazó tömb páros és páratlan elemeit számlálja meg. (*Tomb2*)

A program beolvassa a tömb elemeinek számát, majd feltölti a tömböt. A *paros* változóba a tömb páros elemeinek számát, a *paratlan* változóba pedig a páratlan elemeinek számát tölti. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program tomb2;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  i,db,paros,paratlan : integer;
  x : array[1..20] of integer;
begin
  write('A tomb elemeinek szama: ');
  readln(db);
  for i :=1 to db do
    begin
      write(i:2, '.elem: ');
      readln(x[i]);
    end;
  paros := 0; paratlan := 0;
  i:=1;
  while i<=db do
    begin
      if odd(x[i]) then paratlan := paratlan+1
        else paros := paros+1;
      i:=i+1;
    end;
  writeln;
  writeln('Paratlan elemek szama: ',paratlan);
  writeln('Paros elemek szama   : ',paros);
  readln;
end.
```



Tervezzünk egy olyan programot, amely két vektor skalárszorzatát számítja ki! (*Tomb3*)

Megadjuk, hogy hány dimenzióban dolgozunk (n maximum 20). A program feltölti a két tömböt és az *skszorzat* változóba kiszámítja a két vektor skaláris szorzatát. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program tomb3;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  i,n : integer;
  x1,x2: array[1..20] of real;
  skszorzat: real;
begin
  writeln('Ket vektor skalar szorzata');
  writeln;
  write('A vektor elemeinek szama: ');
  readln(n);
  for i:=1 to n do
    begin
      write('x1[' ,i:2,']= '); readln(x1[i]);
      write('x2[' ,i:2,']= '); readln(x2[i]);
      writeln;
    end;
  skszorzat:=0;
  for i:=1 to n do
    skszorzat:= skszorzat+x1[i]*x2[i];
  writeln('A ket vektor skalar szorzata: ',skszorzat:8:2);
  readln;
end.
```



Írjunk egy olyan programot, amely a tömb legnagyobb és legkisebb értékét, valamint a hozzá tartozó indexet keresi meg! (*Tomb4*)

A program beolvassa a tömb elemeinek számát, majd feltölti a tömböt. Az első elemre állítja a legkisebb és a legnagyobb kezdőértéket, valamint az index 1 lesz. A ciklust csak kettőtől indul, ha talál olyan elemet, amely kisebb a minimumnál, akkor cserél, illetve ha talál nagyobb a maximumnál, akkor cserél és az aktuális indexet eltárolja. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program tomb4;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  i,db, min_index, max_index : integer;
  min,max: real;
  x : array[1..20] of real;
begin
  write('A tomb elemeinek szama: ');
  readln(db);
  for i :=1 to db do
    begin
      write(i:2, '.elem: ');
      readln(x[i]);
    end;
  min_index := 1;
  min := x[1];
  max_index := 1;
  max := x[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;
  writeln('A tomb maximalis eleme: ',max:6:2);
  writeln('A maximalis elem indexe: ',max_index);
  writeln('A tomb minimalis eleme: ',min:6:2);
  writeln('A minimalis elem indexe: ',min_index);
  readln;
end.
```



Készítsünk egy olyan programot, amely egy maximálisan 10x10-es véletlen-számokkal feltöltött mátrixról megállapítja, hogy bűvös négyzet-e! (Egy mátrix akkor bűvös négyzet, ha sorainak, oszlopinak és átlóinak elemösszege megegyezik.) (*Buvos*)

A program bekéri a bűvös négyzet méretét, amely 2 és 10 között fogadható el.

A program *{feltoltes}* része véletlen-számokkal feltölti a mátrixot és visszaírja a képernyőre. A **Randomize** eljárás inicializálja a véletlen számok generálását. A **Random(paraméter)** függvény véletlen számokat állít elő $[0, paraméter)$ intervallumban.

A *{kiiras}* programrészlet visszaírja a DOS ablakba a számokat.

Ezek után a program az *{ellenorzes}* részben képezi a sorok összegét, az oszlopok összegét és az átló összegét, majd az algoritmus szerint ellenőrzésre kerül az összegek azonossága. Bármelyik összeg nem egyezik a vizsgálat során az előzőkkel, a mátrix már nem bűvös négyzet.

Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Buvos;
{$APPTYPE CONSOLE}
uses
  SysUtils;

const
  maxn = 10;

var
  bn      : array [1..maxn+1, 1..maxn+1] of byte;
  i,j,n   : byte;
  sum     : integer;
  bnok    : boolean;

begin
  write('Kerem a buvos negyzet meretet [2..10]: ');
  readln(n);
  writeln;
  if not (n in [2..10]) then halt;

  {feltöltés}
  randomize;
  for i:=1 to n do
    for j:=1 to n do
      bn[i,j] := random(10);

  { kiiras }
  for i:=1 to n do
    begin
      for j:=1 to n do
        write(bn[i,j] :5);
      writeln;
    end;

  { ellenorzes }
  { sorok osszege }
  for i:=1 to n do
    begin
      bn[i,n+1]:=0;
      for j:=1 to n do
        bn[i,n+1]:=bn[i,n+1]+bn[i,j];
      end;

  { oszlopok osszege }
  for j:=1 to n do
    begin
      bn[n+1,j]:=0;
      for i:=1 to n do
```

```

        bn[n+1,j]:=bn[n+1,j]+bn[i,j];
    end;

    { floatlo osszege }
    bn[n+1,n+1]:=0;
    for i:=1 to n do
        bn[n+1,n+1]:=bn[n+1,n+1]+bn[i,i];

    { mellekatlo osszege }
    sum:=0;
    for i:=1 to n do
        sum:=sum+bn[n+1-i,i];

    bnok:=true;
    for i:=1 to n+1 do
        if sum<>bn[i,n+1] then bnok:=false;

    for j:=1 to n do
        if sum<>bn[n+1,j] then bnok:=false;
    writeln;
    if bnok then writeln('Buvos negyzet')
        else writeln('Nem buvos negyzet');

    readln;
end.

```



Írjunk egy olyan programot, amely egy beolvasott sztring karaktereiről előfordulási statisztikát készít!
(*Chstat*)

A program bekéri egy szöveget, majd nullázza a statisztikát tároló tömböt (*chnt*). A szöveg hosszáig számláló ciklusban a szöveg karaktere a statisztika tömb indexe, melynek megfelelő eleme minden egyes karakter előforduláskor eggyel növekszik. Eredményként a karakter és előfordulási adata kerül kiírásra.

```
program Chstat;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  s      : string;
  ch     : char;
  chcnt  : array[#0..#255] of byte;
  i      : byte;
begin
  write('Kerek egy szoveget: '); readln(s);
  writeln;
  for ch:=#0 to #255 do
    chcnt[ch]:=0;

  for i:=1 to length(s) do
    inc(chcnt[s[i]]);

  for ch:=#0 to #255 do
    begin
      if chcnt[ch]<>0 then
        begin
          write ('|':4,ch:1,'|':', chcnt[ch]:3);
          end;
        end;
      readln;
    end.
end.
```



Tervezzünk egy olyan programot, amely kitalál 0-100 között egy számot és aztán a felhasználóval kitalálós játékot játszik! (*Jatek*)

A program generál egy véletlen-számot 0-100 között. A **Randomize** eljárás inicializálja a véletlen számok generálását. A **Random(paraméter)** függvény véletlen számokat állít elő $[0, paraméter)$ intervallumban.

Ezek után kezdődhet a játék. A program játékostól kér egy számot, majd tájékoztatja arról, hogy a megadott szám sok, kevés vagy éppen megegyezik a gondolt számmal, ebben az esetben a próbálkozások számát is kiírja.

A játék egészen addig ismétlődik, míg a játékos nem akar tovább játszani. Az **Uppcase** függvény nagybetűssé konvertálja a paramétereként megadott karaktert.

Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Jatek;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  szam,n,f:integer;
  c:char;
begin
repeat
  n:=0;
  randomize; { veletlenszeruen inditja a veletlenszam-generatort }
  f:=random(101);
  writeln('Talalja ki a 0 - 100 kozotti szamot !');
  repeat
    write('szam = '); readln(szam);
    n:=n+1;
    if szam<f then writeln('          keves')
    else
      if szam>f then writeln('          sok ')
      else writeln('Talalt ',n:5,' probalkozassal');
  until f=szam;
  write('Akar meg játszani? (i/n): ');readln(c);
until Uppcase(c) <>'I';
end.
```




A program beolvassa a kockadobások számát, majd véletlenszám-generátorral szimulálva a kockadobást a megadott számszor dob és statisztikát készít a dobásokról. (A **Randomize** eljárás inicializálja a véletlen számok generálását. A **Random(paraméter)** függvény véletlen számokat állít elő $[0, paraméter)$ intervallumban.)

Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Kocka;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  hany_dobas, i, dobas : integer;
const
  hany : array [1..6] of integer = (0, 0, 0, 0, 0, 0);

begin
  randomize;

  write('Hany dobas legyen: ');
  readln(hany_dobas);
  writeln;
  for i:=1 to hany_dobas do
    begin
      dobas := random(6) + 1;
      inc(hany[dobas]);
    end;
  writeln(hany[1], ' egyes, ', hany[2], ' kettes, ', hany[3], ' harmas, ',
    hany[4], ' negyes, ', hany[5], ' otos, ', hany[6], ' hatos. ');
  writeln;
  readln;
end.
```



Írjunk egy olyan programot, amely a beolvasott 4 jegyű számjegyeket fordított sorrendben írja vissza.
(Fordit)

A program a beolvasott négyjegyű szám nagyságát ellenőrzi, majd **div** és **mod** művelettel megfordítja a jegyeket.

Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Fordit;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  x      :integer;
  w,s,r,q,p: integer;
begin
  repeat
    write('Kerek egy negyjegyű számot: ');
    readln(x);
  until (x>=0) and (x<=9999);

  p:=x div 1000;
  w:=x mod 1000;
  q:=w div 100;
  w:=w mod 100;
  r:=w div 10;
  s:=w mod 10;
  w:=1000*s + 100*r + 10*q + p;
  writeln('A szám jegyei fordított sorrendben: ',w);
  readln;
end.
```



Tervezzünk egy olyan programot, amely a születési dátumról megmondja, hogy a hét melyik napjára esett! (*Oroknap*)

A program beolvassa a születési dátumot és az évről, hónapról és a napról megmondja, hogy a hét melyik napjára esett. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program Oroknap;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  ev, ho, nap : integer;
  x, jelzoszam : integer;
  nap_rendben, szokoev : boolean;
  szokonap : integer;
const
  napok : array [0..6] of string[10] = ('vasarnap', 'hetfo',
    'kedd', 'szerda', 'csutortok', 'pentek', 'szombat');

begin
  writeln('Irjon be egy datumot, megmondom milyen napra esik!');
  ev := -1;
  while (ev < 1901) or (ev > 2099) do
    begin
      write('ev [1901-2099] : ');
      readln(ev);
    end;
  ev:=ev-1900;
  szokoev:=(ev mod 4 = 0) and (ev mod 100 <> 0) or (ev mod 400 = 0);
  szokonap := ord(szokoev);
  repeat
    write('honap: ');
    readln(ho);
  until (ho >= 1) and (ho <= 12);
  repeat
    nap_rendben := TRUE;
    write('nap: ');
    readln(nap);
    if (nap < 1) or (nap > 31) then nap_rendben := FALSE;
    if ((ho=4) or (ho=6) or (ho=9) or (ho=11)) and (nap=31)
      or ((ho=2) and (nap>28+szokonap)) then
      begin
        writeln('Ebben a hónapban nincs ilyen nap.');
```