6. A Pascal nyelv utasításai

Írjunk programot, amely beolvas két valós számot és a két szám közül a kisebbikkel osztja a nagyobbikat! (*felt0*)

```
program felt0;
var
  szam1, szam2, eredmeny : real;
begin
  writeln('Kérek két valós számot!');
  write('1. adat: '); readln(szam1);
  write('2. adat: '); readln(szam2);
  if szam1 > szam2 then eredmeny:= szam1/szam2
  else eredmeny:= szam2/szam1;
  writeln('Hányados: ',eredmeny:6:2);
  readln;
end.
```

A program futásainak eredményei:

```
Kérek két valós számot!
1. adat: 2
2. adat: 5
Hányados:   2.50

Kérek két valós számot!
1. adat: 5.6
2. adat: 2.3
Hányados:   2.43
```

Írjunk programot, amely beolvas egy valós számot és a számról eldönti, hogy pozitív, negatív vagy zérus! (*felt1*)

```
program felt1;
var
  szam: real;
begin
  write('Adat: '); readln(szam);
  if szam > 0 then writeln(szam:6:2,' -> pozitív')
  else if szam < 0 then writeln(szam:6:2, ' -> negatív')
  else writeln(szam:6:2,' -> zérus');
  readln;
end.
```

A program futásainak eredménye:

```
Adat: 4
  4.00 -> pozitív
```

```
Adat: -2
 -2.00 -> negatív

Adat: 0
  0.0  -> zérus
```

Írjunk programot, amely beolvas két egész számot és a számokról eldönti, hogy melyik a nagyobb, illetve egyelők-e! (*felt2*)

```
program felt2;
var
  szam1, szam2, min: integer;
begin
  writeln('Kérek két egész számot');
  write('1. adat: '); readln(szam1);
  write('2. adat: '); readln(szam2);
  if szam1 = szam2 then
    writeln('A két szám azonos, mindkettő: ', szam1)
  else if szam1 < szam2 then writeln('Az 1. adat a kisebb: ',szam1)
  else writeln('A 2. adat a kisebb: ',szam2);
  readln;
end.
```

A program futásainak eredménye::

```
Kérek két egész számot
1. adat: 3
2. adat: 4
Az 1. adat a kisebb: 3

Kérek két egész számot
1. adat: 5
2. adat: 2
A 2. adat a kisebb: 2

Kérek két egész számot
1. adat: 7
2. adat: 7
A két szám azonos, mindkettő: 7
```

Írjunk programot, amely beolvas két egész számot és megvizsgálja, hogy a második szám osztója-e az első számnak! (*felt3*)

Megjegyzés: Az oszthatóság vizsgálata a **mod** használatával nagyon kényelmes, hiszen:

```
if szam1 mod szam2 = 0 then
```

a feltétel teljesülése, azaz az **if** utasítás igaz eredménye a maradék nélküli osztást jelenti, vagyis a *szam1* osztható *szam2*-vel.

```
program felt3;
var
  szam1, szam2, min: integer;
begin
  writeln('Kérek két egész számot!');
  write('1. adat: '); readln(szam1);
  write('2. adat: '); readln(szam2);
  if szam1 mod szam2 = 0 then
     writeln(szam1, ' adatnak osztója ', szam2)
  else writeln(szam1,' adatnak nem osztója ',szam2);
  readln;
end.
```

A program futásainak eredménye:

```
Kérek két egész számot!
1. adat: 6
2. adat: 3
6 adatnak osztója 3

Kérek két egész számot!
1. adat: 12
2. adat: 5
12 adatnak nem osztója 5
```

Írjunk programot, amely beolvas két egész számot és kiszámítja, hogy az első számban a második szám hányszor van meg egészként és mennyi az osztás maradéka! (*felt4*)

```
program felt4;
var
  szam1, szam2, egesz, maradek: integer;
begin
  writeln('Kérek két egész számot!');
  write('1. adat: '); readln(szam1);
  write('2. adat: '); readln(szam2);
  if szam1 > szam2  then
  begin
    egesz:= szam1 div szam2;
    maradek:= szam1 mod szam2;
    writeln('Egész osztás eredménye: ',egesz,' maradék: ',maradek);
  end
  else
  begin
    egesz:= szam2 div szam1;
    maradek:= szam2 mod szam1;
    writeln('Osztás eredménye: ',egesz,' maradék: ',maradek);
  end;
  readln;
end.
```

A program futásainak eredménye:

```
Kérek két egész számot!
1. adat: 13
2. adat: 5
Egész osztás eredménye: 2 maradék: 3

Kérek két egész számot!
1. adat: 12
2. adat: 6
Egész osztás eredménye: 2 maradék: 0
```

Írjunk programot, amely beolvassa a víz hőmérsékletét, és szövegesen írja vissza a hőmérsékletnek megfelelő halmazállapotot! (*felt5*)

```
program felt5;
var
  fok : real;
begin
  write('A víz hőmerséklete: '); readln(fok);
  if fok <= 0 then writeln('  szilárd (jég)')
  else if (fok > 0) and (fok < 100) then writeln('  folyékony (víz)')
  else if fok >= 100 then writeln(' légnemű (gőz)');
  readln;
end.
```

A program futásainak eredménye:

```
A víz hőmérséklete: 21
  folyékony (víz)

A víz hőmérséklete: -1
  szilárd (jég)

A víz hőmérséklete: 100
 légnemű (gőz)
```

Írjunk programot, amely beolvass két egész számot, majd *m* vagy *d* műveleti kódot. A *m* kód a két szám modulusának, a *d* kód a két szám egész osztásának műveletét jelöli meg! (*case1*)

```
program case1;
var
  adat1, adat2, eredmeny, hiba: integer;
  muvjel: char;
begin
   hiba:= 0;
   writeln('Kérek két egész számot!');
   write('1. adat: '); readln(adat1);
```

```
   write('2. adat: '); readln(adat2);
   write('Muvelet (mod - m, div - d ):');
   readln(muvjel);

     case muvjel of
     'm': eredmeny := adat1 mod adat2;
     'M': eredmeny := adat1 mod adat2;
     'd': eredmeny := adat1 div adat2;
     'D': eredmeny := adat1 div adat2;
     else begin
            writeln('Hibás műveleti kód!');
            hiba:= 1;
          end;
    end;
    if hiba = 0 then
    begin
      writeln('A művelet eredménye:');
      case muvjel of
        'm': writeln(adat1,' mod ',adat2, ' = ',eredmeny);
        'M': writeln(adat1,' mod ',adat2, ' = ',eredmeny);
        'd': writeln(adat1,' div ',adat2, ' = ',eredmeny);
        'D': writeln(adat1,' div ',adat2, ' = ',eredmeny);
      end;
    end;
   readln;
end.
```

A program futásainak eredménye:

```
Kérek két egész számot!
Művelet (mod - m, div - d ):m
A művelet eredménye:
12 mod 6 = 0

Kérek két egész számot!
1. adat: 15
2. adat: 2
Művelet (mod - m, div - d ):M
A művelet eredménye:
15 mod 2 = 1

Kérek két egész számot!
1. adat: 6
2. adat: 2
Művelet (mod - m, div - d ):d
A művelet eredménye:
6 div 2 = 3

Kérek két egész számot!
1. adat: 15
2. adat: 7
Művelet (mod - m, div - d ):D
A művelet eredménye:
15 div 7 = 2
```

Írjunk programot, amely beolvas két pozitív egész számot, majd *s* vagy *m* műveleti kódot! Végezzünk ellenőrzést, azaz ne olvassunk be 0 vagy negatív számot! Az *s* kód a két szám számtani közepének, az *m* kód a két szám mértani közepének műveletét jelöli meg! (*case2*)

```
program case2;
var
  adat1, adat2, eredmeny: real;
  muvjel: char;
begin
  writeln('Kérek két pozitív számot!');
  repeat
     write('1. adat: '); readln(adat1);
  until adat1 > 0;
  repeat
    write('2. adat: '); readln(adat2);
  until adat2 > 0;
  repeat
    write('Művelet (számtani közép - s,S, mértani közép - m,M): ');
    readln(muvjel);
  until muvjel in ['s','S','m','M'];
  case muvjel of
     's': eredmeny := (adat1 + adat2)/2;
     'S': eredmeny := (adat1 + adat2)/2;
     'm': eredmeny := sqrt(adat1 * adat2);
     'M': eredmeny := sqrt(adat1 * adat2);
   end;
   writeln('A művelet eredménye:');
   case muvjel of
     's': writeln('számtani közepe(',adat1:6:2,',',adat2:6:2,
                  ') => ',eredmeny:6:2);
     'S': writeln('számtani közepe(',adat1:6:2,',',adat2:6:2,
                  ') => ',eredmeny: 6:2);

     'm': writeln('mértani közepe(',adat1:6:2,',',adat2:6:2,
                  ') => ',eredmeny:8:3);
     'M': writeln('mértani közepe(',adat1:6:2,',',adat2:6:2,
                  ') => ',eredmeny:8:3);
   end;
   readln;
end.
```

A program futásainak eredménye:

```
Kérek két pozitív számot!
1. adat: 1
2. adat: 2
Művelet (számtani közép - s,S, mértani közép - m,M): s
A művelet eredménye:
számtani közepe(  1.00,  2.00) =>   1.50
```

```
Kérek két pozitív számot!
1. adat: 1
2. adat: 2
Művelet (számtani közép - s,S, mértani közép - m,M): m
A művelet eredménye:
mértani közepe(  1.00,  2.00) =>    1.414

Kérek két pozitív számot!
1. adat: 0
1. adat: -1
1. adat: 2
2. adat: 0
2. adat: 3
Művelet (számtani közép - s,S, mértani közép - m,M): S
A művelet eredménye:
számtani közepe(  2.00,  3.00) =>   2.50
```

Írjunk programot, amely beolvassa az adatok darabszámát! Majd az adott darabszámú egész típusú adatról eldönti, hogy mennyi pozitív, negatív és zérus adatot olvasott be! (*for1*)

```
program for1;
var
  i, db: integer;
  pozitiv_db, negativ_db, zerus_db, adat: integer;
begin
  pozitiv_db:=0; negativ_db:=0; zerus_db:=0;
  write('Adatok száma: '); readln(db);
  for i:=1 to db do
  begin
    write('adat: '); readln(adat);
    if adat < 0 then negativ_db:= negativ_db+1;
    if adat > 0 then pozitiv_db:= pozitiv_db+1;
    if adat = 0 then zerus_db:= zerus_db+1;
  end;
  writeln('Pozitív adatok száma: ',pozitiv_db);
  writeln('Negatív adatok száma: ',negativ_db);
  writeln('Zérus adatok száma  : ',zerus_db);
  readln;
end.
```

A program futásának eredménye:

```
Adatok száma: 6
adat: 3
adat: -4
adat: 0
adat: -2
adat: 0
adat: 3
Pozitív adatok száma: 2
Negatív adatok száma: 2
Zérus adatok száma  : 2
```

Írjunk programot, amely 1 és 16 közötti számokról eldönti, hogy mennyi osztható 4-gyel, illetve páratlan szám! (*for2*)

```
program for2;
var
  i: integer;
  paratlan, oszt4: integer;
begin
  for i:=1 to 16 do
  begin
    if i mod 4 = 0 then oszt4:=oszt4+1;
    if i mod 2 = 1 then paratlan:=paratlan+1;
  end;
  writeln('Páratlanok száma: ',paratlan);
  writeln('4-gyel oszthatók száma: ',oszt4);
  readln;
end.
```

A program futásának eredménye:

```
Páratlanok száma: 8
4-gyel oszthatók száma: 4
```

Írjunk programot, amely beolvassa az adatok darabszámát és az adott darabszámú adatokról eldönti, hogy mennyi osztható 5-gyel, illetve mennyi páros, egész típusú adatot olvasott! (*for3*)

```
program for3;
var
  i, db: integer;
  paros, oszt5, adat: integer;
begin
  write('Adatok száma: '); readln(db);
  paros:=0; oszt5:=0;
  for i:=1 to db do
  begin
    write('adat: '); readln(adat);
    if adat mod 2 = 0 then paros:=paros+1;
    if adat mod 5 = 0 then oszt5:=oszt5+1;
  end;
  writeln('Párosak száma: ',paros);
  writeln('5-tel oszthatók száma: ',oszt5);
  readln;
end.
```

A program futásának eredménye:

```
Adatok száma: 6
adat: 12
adat: 15
adat: 32
adat: 40
```

```
adat: 2
adat: 5
Párosak száma: 4
5-tel oszthatók száma: 3
```

Írjunk programot, amely 1 és 16 közötti egész számokból megszámlálja a 4-gyel oszthatókat és a páratlanokat **while** ciklussal! (*while1*)

```
program while1;
var
  i: integer;
  paratlan, oszt4: integer;
begin
  i:=0;
  while i<16 do
  begin
    i:=i+1;
    if i mod 4 = 0 then oszt4:=oszt4+1;
    if i mod 2 = 1 then paratlan:=paratlan+1;
  end;
  writeln('Páratlanok száma: ',paratlan);
  writeln('4-gyel oszthatók száma: ',oszt4);
  readln;
end.
```

A program futásának eredménye:

```
Páratlanok száma: 8
4-gyel oszthatók száma: 4
```

Írjunk programot, amely egész számokat olvas mindaddig, míg két egymás után következő adat zérus! Az adatokat számláljuk meg, a két egymás után következő nullából csak egyet vegyünk figyelembe. Negatív adat esetén a ciklus a következő lépésnél folytassa a működését! A pozitív adatok szorzatát számítsuk ki, a nulla adatokat számláljuk meg! Ha két egymás után következő adat nulla, az össz darabszámot csökkentsük eggyel és lépjünk ki a ciklusból! (*while2*)

```
program while2;
var
  i: integer;
  osszes_db, szorzat, nulla, adat: integer;
begin
  osszes_db:=0; szorzat:=1; nulla:=0;
  i:=0;
  writeln('Két egymás után következő zérus zárja a beolvasást!');
  while i< 1 do
  begin
    osszes_db:=osszes_db+1;
```

```
    write('adat: '); readln(adat);
    if adat < 0 then continue;
    if adat > 0 then szorzat:=szorzat*adat;
    if adat = 0 then nulla:=nulla+1;
    if nulla = 2 then begin osszes_db:=osszes_db-1; break; end;
    if adat <> 0 then nulla:=0;
  end;
  writeln('Adatok száma: ',osszes_db);
  writeln('Pozitív adatok szorzata: ',szorzat);
  readln;
  readln;
end.
```

A program futásának eredménye:

```
Két egymás után következő zérus zárja a beolvasást!
adat: 3
adat: -4
adat: 5
adat: 0
adat: 6
adat: 0
adat: 0
Adatok száma: 6
Pozitív adatok szorzata: 90
```

Írjunk programot, amely 1 és 16 közötti egész számokból megszámlálja a 4-gyel oszthatókat és a páratlanokat **repeat** ciklussal! (*repeat1*)

```
program repeat1;
var
  i: integer;
  paratlan, oszt4: integer;
begin
  i:=0;
  repeat
    i:=i+1;
    if i mod 4 = 0 then oszt4:=oszt4+1;
    if i mod 2 = 1 then paratlan:=paratlan+1;
  until i = 16;
  writeln('Páratlanok száma: ',paratlan);
  writeln('4-gyel oszthatók száma: ',oszt4);
  readln;
end.
```

A program futásának eredménye:

```
Páratlanok száma: 8
4-gyel oszthatók száma: 4
```

Írjunk programot, amely Celsius fokból Fahrenheit, ill. Reaumur fokra számol át és táblázatot készít, a hőmérséklet -30 Celsius foktól +50 Celsius fokig 5 Celsius fokos lépésekben változzon! (*celsiuspr*)

```
program celsiuspr;
var celsius:integer;
    fahrenheit,reaumur:real;
begin
  writeln('Celsius':15,'Fahrenheit':20,'Reaumur':20);
  writeln;
  celsius:=-30;
  while celsius<=50 do
  begin
    fahrenheit:=1.8*celsius+32;
    reaumur:=0.8*celsius;
    writeln(celsius:13,fahrenheit:20:2,reaumur:20:2);
    celsius:=celsius+5;
  end;
  readln;
end.
```

A program futásának eredménye:

```
       Celsius          Fahrenheit             Reaumur

           -30              -22.00              -24.00
           -25              -13.00              -20.00
           -20               -4.00              -16.00
           -15                5.00              -12.00
           -10               14.00               -8.00
            -5               23.00               -4.00
             0               32.00                0.00
             5               41.00                4.00
            10               50.00                8.00
            15               59.00               12.00
            20               68.00               16.00
            25               77.00               20.00
            30               86.00               24.00
            35               95.00               28.00
            40              104.00               32.00
            45              113.00               36.00
            50              122.00               40.00
```

Írjunk programot, amely beolvas egy szöveget és kiírja, hogy magánhangzóval, vagy mássalhangzóval kezdődik és hány karakter a hossza! (*szoveg*)

```
program szoveg;
type
    abc = set of 'a'..'z';

var
  magan, massalhangzo: abc;
  sz : string;
  szam, hossza : integer;

begin
  magan:=[];
  magan:= ['a','o','i','u','e'];
  massalhangzo := ['a'..'z']-magan;

  write('Szöveg: '); readln(sz);
  hossza := ord(sz[0]);
  if sz[1] in magan then
       writeln(sz,' magánhangzóval kezdődik: ',sz[1]);
  if sz[1] in massalhangzo then
       writeln(sz,' mássalhangzóval kezdődik: ',sz[1]);
  write('A beolvasott szöveg hossza: ', hossza, ' karakter');
  readln;
end.
```

A program futásainak eredménye:

```
Szöveg: alma
alma magánhangzóval kezdődik: a
A beolvasott szöveg hossza: 4 karakter

Szöveg: korte
korte mássalhangzóval kezdődik: k
A beolvasott szöveg hossza: 5 karakter
```

Írjunk programot, amely beolvas egy egész számot 1 és 3 között. A számot szövegesen írja vissza. Ha a szám kisebb 1-nél, vagy nagyobb 3-nál, akkor

```
          A szám nem esik 1-3 közé!
```

szöveget írja ki! (*numsz*)

```
program numsz;
var
  szam: integer;
begin
  write('Szam (1..3): '); readln(szam);
  write('A beolvasott szám: ',szam);
  case szam of
    1: writeln(' - Egy');
    2: writeln(' - Ketto');
```

```
    3: writeln(' - Harom');
    else writeln(' - A szám nem esik 1-3 közé!');
  end;
  readln;
end.
```

A program futásainak eredménye:

```
Szam (1..3): 1
A beolvasott szám: 1 - Egy

Szam (1..3): 0
A beolvasott szám: 0 - A szám nem esik 1-3 közé!

Szam (1..3): 5
A beolvasott szám: 5 - A szám nem esik 1-3 közé!

Szam (1..3): 3
A beolvasott szám: 3 – Harom
```

Írjunk programot, amely beolvas egy mondatot . tal zárva. Készítsünk statisztikát a mondatban lévő magánhangzók darabszámáról! (*statiszt*)

```
program statiszt;
var a,e,i,o,u:integer;
    ch:char;
begin
  write('A program a . végjelig olvas.');
  writeln;
  a:=0;
  e:=0;
  i:=0;
  o:=0;
  u:=0;
  read(ch);
  while ch <> '.' do
    begin
    case ch of
       'a':a:=a+1;
       'e':e:=e+1;
       'i':i:=i+1;
       'o':o:=o+1;
       'u':u:=u+1;
    end; {case}
    read(ch);
    end; {while}
    writeln('Magánhangzók statisztikája:');
    writeln('a: ',a:5,' db');
    writeln('e: ',e:5,' db');
    writeln('i: ',i:5,' db');
    writeln('o: ',o:5,' db');
    writeln('u: ',u:5,' db');
    readln;  readln;
end.
```

A program futásának eredménye:

```
A program a . végjelig olvas.
Ma szep az ido, de holnap eshet az eso.
Magánhangzók statisztikája:
a:      4 db
e:      5 db
i:      1 db
o:      3 db
u:      0 db
```

Olvassunk be egy pénzösszeget (legyen egész szám). Határozzuk meg, hogy a legkevesebb bankjegy és érme felhasználása mellett az egyes címletekből hányat kell használni az összeg pontos kifizetéséhez!
A rendelkezésre álló címletek: 5000 Ft, 1000 Ft, 500 Ft, 200 Ft, 100 Ft, 50 Ft, 20 Ft, 10 Ft, 5 Ft, 2 Ft, 1 Ft.
(*penzvalt*)

```
program penzvalt;
var  ertek:longint;
     cimlet:integer;
     i:integer;
begin
   repeat
      writeln('Kérem az összeget (1...60000)');
      write('Összeg: ');
      readln(ertek);
   until (ertek>=1) and (ertek<=60000);
   writeln;
   writeln('cimlet':8,'darab':15);
   { 5000-es és 1000-es cimletek cikluson kívül }
   cimlet:=ertek div 5000;
   ertek:=ertek mod 5000;
   if cimlet<>0 then writeln(5000:7,cimlet:15);
   cimlet:=ertek div 1000;
   ertek:=ertek mod 1000;
   if cimlet<>0 then writeln(1000:7,cimlet:15);
   { a többi cimlet ciklusban }
   for i:=9 downto 1 do
   begin
      if (i mod 3)=0 then
      begin
         cimlet:=ertek div (5*round(exp(ln(10)*((i div 3)-1))));
         ertek:=ertek mod (5*round(exp(ln(10)*((i div 3)-1))));
         if cimlet<>0 then
            writeln(5*round(exp(ln(10)*((i div 3)-1))):7,cimlet:15);
      end;
      if (i mod 3)=2 then
      begin
         cimlet:=ertek div (2*round(exp(ln(10)*(i div 3))));
         ertek:=ertek mod (2*round(exp(ln(10)*(i div 3))));
```

```
         if cimlet<>0 then
            writeln(2*round(exp(ln(10)*(i div 3))):7,cimlet:15);
      end;
      if (i mod 3)=1 then
      begin
         cimlet:=ertek div (round(exp(ln(10)*(i div 3))));
         ertek:=ertek mod (round(exp(ln(10)*(i div 3))));
         if cimlet<>0 then
            writeln(round(exp(ln(10)*(i div 3))):7,cimlet:15);
      end;
   end;
   readln;
end.
```

A program futásának eredménye:

```
Kérem az összeget (1...60000)
Összeg: 12342

  cimlet          darab
   5000              2
   1000              2
    200              1
    100              1
     20              2
      2              1
```

Olvassunk be egy egész számot 0-3999 között és alakítsuk át római számmá! (*romai*)

```
program romai;
var szam:0..3999;
    egy,ot,tiz:char;
    szamok:array[1..4] of 0..9;
    i,j:integer;
begin
  writeln('Kérek egy számot 0..3999 között');
  repeat
    write('Szám: ');
    readln(szam);
  until (szam>=0) and (szam<=3999);
  for i:=3 downto 0 do      { digitekre bontás }
    begin
     szamok[i+1]:=szam div round(exp(i*ln(10)));
     szam:=szam mod round(exp(i*ln(10)));
    end;
  write('A szám római alakban: ');
  for i:=4 downto 1 do
    begin
     case i of
      4:egy:='M';
      3:begin egy:='C';ot:='D';tiz:='M'; end;
      2:begin egy:='X';ot:='L';tiz:='C'; end;
```

```
      1:begin egy:='I';ot:='V';tiz:='X'; end;
     end;
     case szamok[i] of
      1..3:for j:=1 to szamok[i] do write(egy);
         4:write(egy,ot);
         5:write(ot);
      6..8:begin write(ot);for j:=1 to szamok[i]-5 do write(egy); end;
         9:write(egy,tiz);
     end;
  end;   {for}
  readln;
end.
```

A program futásainak eredménye:

```
Kérek egy számot 0..3999 között
Szám: 2543
A szám római alakban: MMDXLIII

Kérek egy számot 0..3999 között
Szám: 2006
A szám római alakban: MMVI
```

Írjunk programot, amely beolvas két egész számot, a nagyobbat elosztja a kisebbel, majd a hányadost és kiírja a maradékot! (*osztas1*, *osztas2*)

A feladat megoldása **if** utasítással.

```
program osztas1;
var osztando,oszto:integer;
    hanyados, maradek:integer;
begin
   writeln('Kérek két egész számot!');
   write('Az első: ');
   readln(osztando);
   write('A második: ');
   readln(oszto);
   if osztando>=oszto then
      begin
         hanyados:=osztando div oszto;
         maradek:=osztando mod oszto;
         writeln('Az első szám volt nagyobb')
      end
    else
      begin
         hanyados:=oszto div osztando;
         maradek:=oszto mod osztando;
         writeln('A második szám volt nagyobb')
       end;
   writeln('Hányados: ',hanyados);
   writeln('Maradék : ',maradek);
   readln;
end.
```

A program futásának eredménye:

```
Kérek két egész számot!
Az első   : 7
A második : 12
A második szám volt nagyobb
Hányados  : 1
Maradék   : 5
```

A feladat megoldása **case** utasítással.

```
program osztas2;
var osztando,oszto:integer;
    hanyados, maradek:integer;
    muvelet: integer;
begin
     writeln('Kérek két egész számot!');
     write('Az első   : ');
     readln(osztando);
     write('A második : ');
     readln(oszto);
     if osztando>=oszto then  muvelet:=1 else muvelet:=2;
     case muvelet of
     1:   begin
             hanyados:=osztando div oszto;
             maradek:=osztando mod oszto;
             writeln('Az első szám volt nagyobb');
          end;
     2:   begin
             hanyados:=oszto div osztando;
             maradek:=oszto mod osztando;
             writeln('A második szám volt nagyobb');
         end;
     end;
     writeln('Hányados  : ',hanyados);
     writeln('Maradék   : ',maradek);
     readln;
end.
```

A program futásainak eredménye:

```
Kérek két egész számot!
Az első   : 12
A második : 25
A második szám volt nagyobb
Hányados  : 2
Maradék   : 1

Kérek két egész számot!
Az első   : 25
A második : 13
Az első szám volt nagyobb
Hányados  : 1
Maradék   : 12
```

Írjunk programot, amely beolvas egy egész számot és kiírja ki az összes osztóját, illetve jelzi, ha prím szám! (*oszto0, oszto1*)

A feladat megoldása **for** ciklussal:

```
program oszto0;
var szam, oszto: integer;
    van: boolean;
begin
   van:=false;
   writeln('Összes osztó keresése');
   write('Szám: '); readln(szam);
   writeln('Osztók ');
   oszto:=2;
   for oszto:=2 to szam div 2 do
     if szam mod oszto = 0 then
     begin
       write(oszto,' ');
       van:=true;
    end;
    if not van then writeln('   prím szám');
   readln;
end.
```

A program futásainak eredménye:

```
Összes osztó keresése
Szám: 13
Osztók
   prím szám

Összes osztó keresése
Szám: 255
Osztók
3 5 15 17 51 85
```

A feladat megoldása **while** ciklussal:

```
program oszto1;
var szam, oszto: integer;
    van : boolean;
begin
   van:= false;
   writeln('Összes osztó keresése');
   write('Szám: '); readln(szam);
   writeln('Osztók ');
   oszto:=2;
   while (oszto <= szam div 2) do
   begin
     if szam mod oszto = 0 then
     begin
       write(oszto,' ');
```

```
        van:= true;
      end;
      oszto:=oszto+1;
   end;
   if not van then writeln('  prím szám');
   readln;
end.
```

A program futásának eredménye:

```
Összes osztó keresése
Szám: 155
Osztók
5 31
```

Módosítsuk az *oszto1* programot, hogy az osztók kiírása egy sorban tízesével és 6 mezőszélességgel történjen! A feladatot **while** ciklussal oldjuk meg! (*oszto2*)

```
program oszto2;
var szam,oszto,szamlalo: integer;
    van:boolean;
begin
   van:=false;
   writeln('Összes osztó keresése');
   write('Szám: '); readln(szam);
   writeln('Osztók');
   oszto:=2;
   szamlalo:=0;
   while (oszto <= szam div 2) do
   begin
     if szam mod oszto = 0 then
       begin
         write(oszto:6); van:= true;
         szamlalo:=szamlalo+1;
         if szamlalo = 10 then
                         begin
                           writeln; szamlalo:=0;
                         end;
       end;
      oszto:=oszto+1;
   end;
   if not van then writeln('  prím szám');
   readln;
end.
```

A program futásának eredménye:

```
Összes osztó keresése
Szám: 12444
Osztók
     2     3     4     6    12    17    34    51    61    68
   102   122   183   204   244   366   732  1037  2074  3111
  4148  6222
```

Módosítsuk az *oszto1* programot, hogy az osztók kiírása egy sorban tízesével és 6 mezőszélességgel történjen! A feladatot **repeat until** ciklussal oldjuk meg! (*oszto3*)!

```
program oszto3;
var szam, oszto: integer;
    szamlalo: integer;
    van: boolean;
begin
   van:= true;
   writeln('Összes osztó keresése');
   write('Szám: '); readln(szam);
   writeln('Osztók');
   oszto:=2;
   szamlalo:=0;
   repeat
     if szam mod oszto = 0 then
     begin
       write(oszto:6); van:= true;
       szamlalo:=szamlalo+1;
       if szamlalo = 10 then
                        begin
                           writeln; szamlalo:=0;
                        end;
     end;
     oszto:=oszto+1;
   until oszto > szam div 2;
   if not van then writeln('  prím szám');
   readln;
end.
```

A program futásának eredménye:

```
Szám: 12444
Osztók
     2     3     4     6    12    17    34    51    61    68
   102   122   183   204   244   366   732  1037  2074  3111
  4148  6222
```

Írjunk programot, amely beolvas egy egész számot és meghatározza a szám törzstényezőit, például: 12 = 2*2*3! (*torzs2, torzs3, torzs4*)

```
program torzs2;
var
   szam,oszto: integer;
begin
  writeln('Törzstényezőkre bontás');
  write('Szám: '); readln(szam);
  write('Törzstényezői: ');
```

```
  while szam <> 1 do
  begin
     oszto:=2;
     while (szam mod oszto) <> 0 do
        oszto:=oszto+1;
     szam:= szam div oszto;
     if(szam = 1) then write(oszto)
                  else write(oszto,'*');
  end;
  readln;
end.
```

A program futásának eredménye:

```
Törzstényezőkre bontás
Szám: 12444
Törzstényezői: 2*2*3*17*61
```

A feladat második megoldása egy boolean változó használatával és **while** ciklussal:

```
program torzs3;
var
   szam,oszto: integer;
   elso      : boolean;
begin
  writeln('Törzstényezőkre bontás');
  write('Szám: '); readln(szam);
  write('Törzstényezői: ');
  elso := true;
  while szam <> 1 do
  begin
     oszto:=2;
     while (szam mod oszto) <> 0 do
        oszto:=oszto+1;
        if elso then begin elso:=false; write(oszto) end
                else  write('*',oszto);
     szam:= szam div oszto;
  end;
  readln;
end.
```

A program futásának eredménye:

```
Törzstényezőkre bontás
Szám: 12444
Törzstényezői: 2*2*3*17*61
```

A feladat harmadik megoldása egy boolean változó használatával és **repeat** – **until** ciklussal:

```
program torzs4;
uses crt;
var
   szam,oszto: integer;
   elso      : boolean;
begin
  clrscr;
  writeln('Törzstényezőkre bontás');
  write('Szám: '); readln(szam);
  write('Törzstényezői: ');
  elso:=true;
  if szam <> 1 then
  begin
    oszto:=2;
    repeat
      if (szam mod oszto) = 0 then
      begin
       szam:= szam div oszto;
       if elso then begin elso:=false; write(oszto) end
               else  write('*',oszto);
      end
      else oszto:=oszto+1;
    until szam=1;
  end;
end.
```

A program futásának eredménye:

```
Törzstényezőkre bontás
Szám: 12444
Törzstényezői: 2*2*3*17*61
```

Módosítsuk a TORZS3.PAS programot, hogy a kiíratása a tömörebb, hatványkitevős alakban jelenítse meg az eredményt (*torzs5*)

A törzstényezős alak kiíratásánál használható a tömörebb hatványkitevős alak.
Például:

64 = 2*2*2*2*2*2 = 2^6
576 = 2*2*2*2*2*2*3*3 = 2^6*3^2

```
program torzs5;
var
   szam,oszto,elozo,hatv: integer;
   elso: boolean;
begin
  writeln('Törzstényezőkre bontás');
```

```
  write('Szám: '); readln(szam);
  write('Törzstényezői: ');
  elozo:= 2;
  hatv:=0;
  elso:=true;
  while szam <> 1 do
  begin
     oszto:=2;
     while (szam mod oszto) <> 0 do
        oszto:=oszto+1;
     if elso then
     begin
       elozo:=oszto;
       write(oszto);
        elso:=false;
     end;
     if(elozo = oszto) then
     begin
       hatv:=hatv+1;
     end
     else
     begin

      if hatv > 1 then
      begin
        write('^',hatv,'*',oszto);
        hatv:=1;
      end
      else write('*',oszto);
   end;
   szam:= szam div oszto;

   if szam = 1 then
   begin
     if (elozo = oszto) then
     begin
       if hatv > 1 then write('^',hatv);
     end
   end;
   elozo:=oszto;
  end;
  readln;
end.
```

A program futásainak eredménye:

```
Törzstényezőkre bontás
Szám: 64
Törzstényezői: 2^6

Törzstényezőkre bontás
Szám: 576
Törzstényezői: 2^6*3^2
```

Írjunk programot, amely beolvas két egész számot és meghatározza a legnagyobb közös osztójukat! (*lkoszto1*, *lkoszto2*)

```
program lkoszto1;
var szam1,szam2, oszto,kozos: integer;
begin
   writeln('Legnagyobb közös osztó keresése');
   write('1. szám: '); readln(szam1);
   write('2. szám: '); readln(szam2);
   oszto:=2;
   kozos:=0;
   while (oszto <= szam1) and (oszto <= szam2) do
   begin
     if(szam1 mod oszto = 0) and
       (szam2 mod oszto = 0 )    then kozos:=oszto;
     oszto:=oszto+1
   end;
  if kozos = 0 then writeln('Nincs közös osztó')
               else writeln('Legnagyobb közös osztó: ',kozos);
  readln;
end.
```

A program futásának eredménye:

```
Legnagyobb közös osztó keresése
1. szám: 12
2. szám: 64
Legnagyobb közös osztó: 4
```

A feladat másik megoldása **repeat until** ciklus használatával:

```
program lkoszto2;
var szam1,szam2, oszto,kozos: integer;
begin
   writeln('Legnagyobb közös osztó keresése');
   write('1. szám: '); readln(szam1);
   write('2. szám: '); readln(szam2);
   oszto:=2;
   kozos:=0;
   if(szam1 > 1) and (szam2 > 1)
   then
     repeat
       if(szam1 mod oszto = 0) and (szam2 mod oszto = 0 )
         then kozos:=oszto;
       oszto:=oszto+1
     until (oszto > szam1) or (oszto > szam2);
     if kozos = 0 then writeln('Nincs közös osztó')
                  else writeln('Legnagyobb közös osztó: ',kozos);
     readln;
end.
```

A program futásának eredménye:

```
Legnagyobb közös osztó keresése
1. szám: 12
2. szám: 64
Legnagyobb közös osztó: 4
```

Írjunk programot, amely az euklideszi algoritmus segítségével számolja ki két pozitív egész szám legnagyobb közös osztóját! (*lnko*)

Az euklideszi algoritmus szerint, ha a kisebbik szám maradék nélkül van meg a nagyobbikban, a kisebbik lesz a legnagyobb közös osztó. Ha van maradék, a legnagyobb közös osztó kisebb vagy egyenlő lesz a maradékkal. A következő próbálkozásnál a kisebbik szám lép az osztandó, a maradék pedig az osztó helyébe. Az algoritmus így mindaddig folytatódik, míg a maradék nulla nem lesz, ekkor az aktuális osztó lesz a legnagyobb közös osztó.

```
program lnko;
 var a,b,maradek,temp:integer;
begin
  writeln('Kérek két pozitív egész számot (0..30000)');
  repeat
     write('1. szám: '); readln(a);
  until (a>=0) and (a<=30000);
  repeat
     write('2. szám: '); readln(b);
  until (b>=0) and (b<=30000);
  if a<b then
  begin
    temp:=a;
    a:=b;
    b:=temp;
  end;
  maradek:=b;
  while maradek<>0 do
  begin
    maradek:=a mod b;
    a:=b;
    b:=maradek;
  end;
  writeln('A két szám legnagyobb közös osztója: ',a);
  readln;
end.
```

A program futásainak eredménye:

```
Kérek két pozitív egész számot (0..30000)
1. szám: 150
2. szám: 2500
A két szám legnagyobb közös osztója: 50
```

```
Kérek két pozitív egész számot (0..30000)
1. szám: 150
2. szám: 2250
A két szám legnagyobb közös osztója: 150
```

Írjunk programot, amely beolvas két egész számot, és megkeresi a legkisebb közös többszörösüket! (*tobbs1*, *tobbs2*)

A feladat megoldása **while** ciklussal:

```
program tobbs1;
var
   szam1,oszto1: integer;
   szam2,oszto2: integer;
   lktobb: integer;
begin
  writeln('Legkisebb közös többszörös');
  repeat
    write('1.szám: '); readln(szam1);
  until szam1 > 1;
  repeat
    write('2.szám: '); readln(szam2);
  until szam2 > 1;
  lktobb:=1;
  while (szam2 <> 1) or (szam1 <> 1) do
  begin
    while szam1 <> 1 do
    begin
      oszto1:=2;
      while (szam1 mod oszto1) <> 0 do
         oszto1:=oszto1+1;
      if szam2 mod oszto1 = 0 then
        begin
          szam2 := szam2 div oszto1;
        end;
      lktobb:=lktobb*oszto1;
      szam1:= szam1 div oszto1;
    end;
    while szam2 <> 1 do
    begin
      oszto2:=2;
      while (szam2 mod oszto2) <> 0 do
        oszto2:=oszto2+1;
      szam2:=szam2 div oszto2;
      lktobb:=lktobb*oszto2;
    etnd;
  end;
  writeln('Legkisebb közös többszörös: ',lktobb);
  readln;
end.
```

A program futásának eredménye:

```
Legkisebb közös többszörös
1.szám: 12
2.szám: 45
Legkisebb közös többszörös: 180
```

A feladat megoldása **repeat until** ciklussal:

```
program tobbs2;
var
   szam1,oszto1: integer;
   szam2,oszto2: integer;
   lktobb: integer;
begin
  writeln('Legkisebb közös többszörös');
  repeat
    write('1.szám: '); readln(szam1);
  until szam1 > 1;
  repeat
    write('2.szám: '); readln(szam2);
  until szam2 > 1;
  lktobb:=1;
  repeat
    repeat
      oszto1:=2;
      while (szam1 mod oszto1) <> 0 do
         oszto1:=oszto1+1;
      if szam2 mod oszto1 = 0 then
        begin
          szam2 := szam2 div oszto1;
        end;
      lktobb:=lktobb*oszto1;
      szam1:= szam1 div oszto1;
    until szam1 = 1;
    repeat
      oszto2:=2;
      while (szam2 mod oszto2) <> 0 do
        oszto2:=oszto2+1;
      szam2:=szam2 div oszto2;
      lktobb:=lktobb*oszto2;
    until szam2 = 1;
  until (szam1 = 1) and (szam2 = 1);
  writeln('Legkisebb közös többszörös: ',lktobb);
  readln;
end.
```

A program futásának eredménye:

```
Legkisebb közös többszörös
1.szám: 11
2.szám: 44
Legkisebb közös többszörös: 44
```