

## 2.7 Függvények és eljárások

1. Köbre emelő függvény: [Fg1](#)
2. Négyzetátaló függvény: [Fg2](#)
3. Primszám ellenőrző függvény: [Primszam](#)
4. Barátfüggvény: [Barátságos](#)
5. Téglatest felszínének, térfogatának számítása: [Eljaras1](#)
6. Vektor koordinátáinak összege és átlaga: [Eljaras2](#)
7. Tankör max. krdetipontjának és évfolyamátlag számítása: [Eljaras3](#)
8. Egy szó magánhangzóinak statisztikája: [Eljaras4](#)



A program beolvassa a valós adatot, aktiválja a *kob* függvényt, és kiírja az eredményt. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

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

function kob(x : real): real;
begin
  kob := sqr(x)*x;
end;
var
  a, ered: real;
begin
  write('adat: '); readln(a);
  ered := kob(a);
  writeln(a:6:2, ' kobe = ',ered:6:2);
  readln;
end.
```



A program beolvassa a négyzet oldalát és aktiválja a *NegyzetAtlo* függvényt, amely kiszámítja a négyzet átlóját, majd megjeleníti az eredményt. Megjegyezzük, hogy az áttekinthetőség kedvéért nem kezeltük a felhasználó esetleges hibáját. A program csak számokat olvas be.

```
program fg2;

{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  a_oldal, atlo : real;
function NegyzetAtlo(a:real):real;
begin
  NegyzetAtlo := a*sqrt(2);
end;
begin
  write('A oldal: '); readln(a_oldal);
  atlo := NegyzetAtlo(a_oldal);
  writeln('A negyzet atloja: ',atlo:6:2);
  readln;
end.
```



## Írjunk egy olyan programot, amely egy számról megállapítja, hogy prímszám-e! (*Primszam*)

A program beolvassa a prímszám vizsgálatára megadott számot, aktiválja a *Prim* függvényt, és kiírja a prímvizsgálat eredményét.

A *Prim* logikai típusú függvény, amely igaz értékkel tér vissza, ha a szám prímszám. A prímszámok jellegzetessége, hogy pontosan két osztójuk van (az 1 és maga a szám). A *Prim* függvény sorra megvizsgálja a számokat, míg négyzetük meg nem haladja a paraméterként megadott számot, hogy osztói-e a paraméternek. Ha találunk osztót a *prm* – kezdetben igazra állított - logikai változó hamisra vált. Ha nincs osztója a paraméternek, akkor a *prm* igaz marad. A függvény visszatérési értéke a *prm* lesz.

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

```
program PrimSzam;

{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  a : integer;

function prim(x:integer): boolean;
var
  p : integer;
  prm: boolean;
begin
  p := 2; prm := true;
  while (p*p < x) and prm do
    begin
      prm := not(x mod p = 0);
      p := p+1;
    end;
  if prm then prm := true
    else prm := false;
end;
begin
  writeln('Prim szam vizsgalata'); writeln;
  write('A szam: '); readln(a);
  if prim(a) then writeln(a, ' prim szam')
    else writeln(a, ' nem prim szam');
  readln;
end.
```



Két számról akkor mondjuk, hogy barátságosak, ha mindkét számra igaz, hogy osztóinak összege megegyezik a másik számmal.

A program beolvassa a két számot, aktiválja a *barat* függvényt, amely megvizsgálja a számok osztóit és összegzi azokat (*ossz1*, *ossz2*). A függvény *true* értéket ad vissza, ha a két számról elmondhatjuk, hogy barátságosak. Az eredmény szöveges formában íródik ki.

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

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

var
  szam1,szam2 : integer;
function barat(sz1,sz2: integer): boolean;
var
  ossz1,ossz2,i: integer;
begin
  ossz1 := 0;
  writeln('Első szám: ',sz1);
  for i:=1 to sz1 div 2 do
    if sz1 mod i = 0 then
      begin
        writeln(i);
        ossz1 := ossz1+i;
      end;
  writeln('összeg: ',ossz1);
  writeln;
  ossz2 := 0;
  writeln('Második szám: ',sz2);
  for i:=1 to sz2 div 2 do
    if sz2 mod i = 0 then
      begin
        writeln(i);
        ossz2 := ossz2+i;
      end;
  writeln('összeg: ',ossz2);
  if (ossz1 = sz2) and (ossz2 = sz1) then barat := true
    else barat := false;
end;

begin
  writeln('Két szám barátságának vizsgálata');
  write('1. szám: '); readln(szam1);
  write('2. szám: '); readln(szam2);
  writeln;
  if barat(szam1,szam2) then writeln('baratsagos')
    else writeln('nem baratsagos');
  readln;
end.
```



Készítsünk egy olyan programot, amely kiszámítja a téglatest felszínét és térfogatát függvényekkel és eljárásokkal! (*Eljaras1*)

A program eljárásai és függvényei:

- A *TeglatestAdatai* eljárás az *a,b,c* kimenő paraméterben adja vissza a téglatest három oldalát.
- A *FelszinElj* eljárás az *a,b,c* bemenő paraméterben kapja meg a téglatest három oldalát és a *felszin* kimenő paraméterben adja vissza a téglatest felszínét.
- A *TerfogatElj* eljárás az *a,b,c* bemenő paraméterben kapja meg a téglatest három oldalát és *terfogat* kimenő paraméterben adja vissza a téglatest térfogatát.
- A *FelszinFuggv* függvény az *a,b,c* bemenő paraméterben kapja meg a téglatest három oldalát és a téglatest felszínét a függvényérték adja vissza.
- A *TerfogatFuggv* függvény az *a,b,c* bemenő paraméterben kapja meg a téglatest három oldalát és a téglatest térfogatát a függvényérték adja vissza.

A program aktiválja az eljárásokat és a függvényeket, és az eredményeket pedig kiírja.

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

```
program eljaras1;
{$APPTYPE CONSOLE}
uses
  SysUtils;

var
  a_old, b_old, c_old : real;
  felsz1, terf1, felsz2, terf2: real;
  procedure TeglatestAdatai(var a,b,c:real);
  begin
    writeln('A teglatest oldalai');
    write('A oldal : '); readln(a);
    write('B oldal : '); readln(b);
    write('C oldal : '); readln(c);
  end;
  procedure FelszinElj(a,b,c:real; var felszin: real);
  begin
    felszin := 2*(a*b+b*c+a*c);
  end;
  function FelszinFuggv(a,b,c:real):real;
  begin
    FelszinFuggv := 2*(a*b+b*c+a*c);
  end;
  procedure TerfogatElj(a,b,c:real; var terfogat: real);
  begin
    terfogat := a*b*c;
  end;
  function TerfogatFuggv(a,b,c:real):real;
  begin
    TerfogatFuggv := a*b*c;
  end;
begin
  TeglatestAdatai(a_old,b_old,c_old);
  FelszinElj(a_old,b_old,c_old,felsz1);
  felsz2:= FelszinFuggv(a_old,b_old,c_old);
  TerfogatElj(a_old,b_old,c_old,terf1);
  terf2:= TerfogatFuggv(a_old,b_old,c_old);
  writeln;
  writeln('A teglatest felszine eljarasssal : ',felsz1:6:2);
  writeln('A teglatest felszine fuggvennyel : ',felsz2:6:2);
  writeln('A teglatest terfogata eljarasssal: ',terf1:6:2);
  writeln('A teglatest terfogata fuggvennyel: ',terf2:6:2);
  readln;
end.
```



A program az alábbi eljárásokat és függvényeket tartalmazza:

- A *VektOlvas* eljárás a kimenő *tomb* típusú *x* paraméterében adja vissza vektor tartalmát, az *n* paraméterben pedig a vektor elemeinek számát.
- A *VektOsszeg* függvény bemenő paraméterekben kapja meg vektor tartalmát (*x*), az elemek számát (*n*) és a vektor összegét adja vissza.
- A *VektAtlag* eljárás a bemenő paraméterekben kapja meg vektor tartalmát (*x*), az elemek számát (*n*) és a vektor átlagát az *atl* kimenő paraméterben adja vissza.

A program aktiválja az eljárásokat és a függvényt, majd pedig kiírja az eredményt.

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

```
program eljaras2;
{$APPTYPE CONSOLE}
uses
  SysUtils;

type
  tomb = array[1..5] of real;
var
  db: integer;
  a : tomb;
  Osszeg, Atlag : real;

procedure VektOlvas(var x: tomb; var n: integer);
var
  i : integer;
begin
  write('A vektor elemeinek szama: '); readln(n);
  for i:=1 to n do
    begin
      write(i:2, '. elem: ');
      readln(x[i]);
    end;
  end;

function VektOsszeg(x: tomb; n:integer): real;
var
  i : integer;
  s : real;
begin
  s := 0;
  for i:=1 to n do
    s:=s+x[i];
  VektOsszeg := s;
end;

procedure VektAtlag(x : tomb; n : integer; var atl: real);
var
  i : integer;
begin
  atl:= VektOsszeg(x,n);
  atl := atl/n;
end;
begin
  VektOlvas(a,db);
  Osszeg := VektOsszeg(a,db);
  VektAtlag(a,db,atlag);
  writeln;
  writeln('A vektor elemeinek osszege: ',Osszeg:6:2);
  writeln('A vektor elemeinek atlaga : ',atlag:6:2);
  readln;
```

end.





Tervezzünk egy olyan programot, amely beolvassa a diákok adatait, kikeresi a maximális kreditpontot és kiszámítja az évfolyam tanulmányi átlagát! (*Eljárás3*)

A program *tanulo* és *tk* felhasználói típusai:

```
type
  tanulo = record
    neve      : string[40];
    tankor    : integer;
    kredit    : integer;
    atlag     : real;
  end;
  tk = array[1..20] of tanulo;
```

A program az alábbi eljárásokat és függvényt tartalmazza:

- A *TankorAdatok* eljárás beolvassa a diákok adatait és a *tk* típusú *x* kimenő paraméterben az adatokat, a diákok számát pedig az *n* paraméter adja vissza.
- A *KeresMaxKredit* függvény bemenő paraméterként kapja meg a diákok adatait (*x*), a diákok számát (*n*) és a maximális kreditpontot adja vissza.
- Az *EvfolyamAtlag* eljárás bemenő paraméterként kapja meg a diákok adatait (*x*), a diákok számát (*n*) és az évfolyam tanulmányi átlagát az *EvfAtl* kimenő paraméterben adja vissza.

A program aktiválja az eljárásokat és a függvényt, majd pedig kiírja az eredményt.

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

```
program eljaras3;
{$APPTYPE CONSOLE}
uses
  SysUtils;

type
  tanulo = record
    neve      : string[40];
    tankor    : integer;
    kredit    : integer;
    atlag     : real;
  end;
  tk = array[1..20] of tanulo;
var
  tan_db, MaxKredit: integer;
  tanulok : tk;
  TanAtlag : real;
procedure TankorAdatok(var x: tk; var n: integer);
var
  i : integer;
begin
  write('A tanulok szama: '); readln(n);
  writeln;
  for i:=1 to n do
  begin
    writeln(i:2, '. tanulo: ');
    write('Neve      : '); readln(x[i].neve);
    write('Tankor    : '); readln(x[i].tankor);
    write('Kredit    : '); readln(x[i].kredit);
    write('Atlag     : '); readln(x[i].atlag);
    writeln;
  end;
end;
function KeresMaxKredit(x: tk; n: integer): integer;
var
  i : integer;
  kr : integer;
begin
  kr := x[1].kredit;
```

```

    for i:=2 to n do
        if x[i].kredit > kr then kr := x[i].kredit;
        KeresMaxKredit := kr;
    end;
procedure EvfolyamAtlag(x : tk; n : integer; var EvfAtl: real);
var
    i : integer;
begin
    EvfAtl:= 0;
    for i:=1 to n do
        EvfAtl:= EvfAtl + x[i].atlag;
    EvfAtl := EvfAtl/n;
end;
begin
    TankorAdatok(tanulok,tan_db);
    MaxKredit := KeresMAxKredit(tanulok, tan_db);
    EvfolyamAtlag(tanulok,tan_db,TanAtlag);
    writeln;
    writeln('Az evfolyam maximalis kreditpontja: ',MaxKredit);
    writeln('Az evfolyam atlaga : ',TanAtlag:6:2);
    readln;
end.

```



A program *szo* felhasználói típusa:

```
type
  szo = record
    szoveg : string[80];
    hossza : integer;
    mgh_a: integer;
    mgh_e: integer;
    mgh_i: integer;
    mgh_o: integer;
    mgh_u: integer;
  end;
```

A program az alábbi eljárásokat és függvényt tartalmazza:

- Az *Olvas* eljárás olvassa be vizsgálandó szöveget és a *szo* típusú *sz* kimenő paraméterben kisbetűsre konvertálva adja vissza. (A ***LowerCase*** függvény a paramétereként megadott sztringet kisbetűsre konvertálja)
- A *vizsgal* függvényt bemenő paraméterként kapja a *szo* típusú *sz* szöveget, elkészíti a statisztikát, és a megszámlolt magánhangzókat a *szo* típus megfelelő mezőiben tárolja.

A program aktiválja az *Olvas* eljárást és a *vizsgal* függvényt és kiírja a szövegben található magánhangzók statisztikáját.

```
program eljaras4;
{$APPTYPE CONSOLE}
uses
  SysUtils;
type
  szo = record
    szoveg : string[80];
    hossza : integer;
    mgh_a: integer;
    mgh_e: integer;
    mgh_i: integer;
    mgh_o: integer;
    mgh_u: integer;
  end;
procedure olvas(var sz : szo);
begin
  write('Szoveg: '); readln(sz.szoveg);
  sz.szoveg:=LowerCase(sz.szoveg);
end;
function vizsgal(var sz: szo): integer;
var
  i, mgh_db : integer;
begin
  sz.hossza := length(sz.szoveg);
  sz.mgh_a := 0;
  sz.mgh_e := 0;
  sz.mgh_i := 0;
  sz.mgh_o := 0;
  sz.mgh_u := 0;
  mgh_db := 0;
  for i:=1 to sz.hossza do
    begin
      if sz.szoveg[i] in ['a','e','i','o','u']
      then
        mgh_db := mgh_db+1;

      case sz.szoveg[i] of
        'a': sz.mgh_a:= sz.mgh_a+1;
        'e': sz.mgh_e:= sz.mgh_e+1;
        'i': sz.mgh_i:= sz.mgh_i+1;
        'o': sz.mgh_o:= sz.mgh_o+1;
        'u': sz.mgh_u:= sz.mgh_u+1;
```

```

        end;
    end;
    vizsgal := mgh_db;
end;
var
    sz: szo;
begin
    olvas(sz);
    writeln;
    writeln('A szo osszes maganhangzoinak szama: ',vizsgal(sz));
    writeln('a maganhangzo szama: ',sz.mgh_a);
    writeln('e maganhangzo szama: ',sz.mgh_e);
    writeln('i maganhangzo szama: ',sz.mgh_i);
    writeln('o maganhangzo szama: ',sz.mgh_o);
    writeln('u maganhangzo szama: ',sz.mgh_u);
    readln;
end.

```