

3. Gyakorlat

Rövid elméleti összefoglaló

• Ciklusutasítások

- **for** ciklus: a ciklusmagban az utasítás a megadott számsorozat mentén kerül végrehajtásra.
for (kezdőérték_kif; feltétel_kif; léptető_kif)
utasítás;

Ha több kezdőértéket adunk a ciklus fejében, vesszővel kell azokat elválasztanunk.

<pre>//MINTA3_01 #include <iostream> #include <iomanip> using namespace std; int main() { int i, szam, hatvany; cout.setf(ios::right); for (i = 1, szam = 2, hatvany = 1; i<=6; i++) { hatvany = hatvany * szam; cout << setw(4) << hatvany << endl; } cin.get(); return 0; }</pre>	<p>A <i>setw()</i> használatához szükséges az alábbi fejlécmagy:</p> <pre>#include <iomanip></pre> <p>Írassuk ki 2 hatványait 1-6 között, jobbra tömörítve és 4 pozícióban!</p> <p>A program futásának eredménye:</p> <pre> 2 4 8 16 32 64</pre>
--	--

- **while** ciklus: a ciklusmag mindaddig végrehajtódik, míg a kifejezés értéke igaz (**true**, nem nulla)
while (kifejezés)
{
utasítás(ok);
}

<pre>//MINTA3_02 //. . . int i, szam = 2, hatvany; hatvany = 1; cout.setf(ios::right); i = 1; while (i <= 6) { hatvany = hatvany * 2; cout << setw(4) << hatvany << endl; i++; } //. . .</pre>	<p>A 2 hatványainak kiírását while ciklussal oldjuk meg.</p> <p>A while ciklusnál a ciklusváltozót az adott kezdőértékre állítjuk be (i = 1) a ciklus utasítása előtt. Úgy adjuk meg a feltételt (i<=6), hogy a kezdőérték mellett és a kívánt végértékig igaz legyen, mert csak akkor fogja a ciklus törzsében lévő utasításokat végrehajtani.</p> <p>A ciklus belsejében a ciklusváltozó értékét változtatnunk kell (i++), oly módon, hogy a ciklus elérhesse a végértékét, különben végtelen ciklust kapunk. A ciklus akkor áll le, amikor a feltétel hamis lesz.</p>
---	---

- **do - while** ciklus : legalább egyszer végrehajtja az utasítást, majd a továbbiakban addig addig ismétli az utasítást, míg a kifejezés értéke igaz (**true**, nem nulla)
do
{
utasítás;
}**while** (kifejezés);

<pre>//MINTA3_03 //. . . int i, szam = 2, hatvany; hatvany = 2; cout.setf(ios::right); i = 1; do { hatvany = hatvany * 2; cout << setw(4) << hatvany << endl; i++; }while (i<=6); . . .</pre>	<p>A 2 hatványainak kiírását do-while ciklussal oldjuk meg. Ez a ciklus is addig működik, míg a feltétel igaz.</p> <p>Beállítjuk a ciklusváltozó kezdőértékét:</p> <pre>i = 1;</pre> <p>A ciklus törzse egyszer feltétel nélkül lefut, mert a feltételt a ciklus végén vizsgáljuk. Ugyanakkor a ciklusváltozó értékét (i++) változtatnunk kell, különben végtelen ciklust kapunk.</p> <p>A ciklus a feltétel hamisra válásával áll le. A feltételt úgy kell beállítanunk, hogy a kívánt kezdő- és végérték között menjen végbe a ciklus törzse.</p>
--	--

- A **continue** utasítás
A **continue** utasítás a ciklusok soron következő iterációját indítja a mögötte álló utasítások átlépésével.
- A **break** utasítással kiléphetünk a ciklusból.
- A **goto** utasítással a megadott címkéjű sorra adjuk át a vezérlést:
 goto *címke*;
 utasítások
 címke;
- **Definíciók bevitele az utasításokba**
Például a **for** ciklus ciklusváltozóját definiálhatjuk:

```
for (int i = 0; i < 10; i++)
{
    ...
}
```

Megjegyezzük, hogy az így definiált változó csak a **for** utasításon belül használható. A ciklusból való kilépés után az *i* változó nem használható fel.

```
//MINTA3_04
#include <iostream>
using namespace std;
#include <iomanip>
int main()
{
    double atlag = 0;
    for (int i=1, ossz = 0, db = 0; i<15; i++)
    {
        if (i == 12) break;
        if (i == 10) continue;
        ossz += i; db++;
        cout << "i: " << setw(2) << i << "    osszeg: "
              << setw(4) << ossz
              << "    db: " << setw(4) << db << endl;
        atlag = (double)ossz/db;
    }
    cout << endl << "atlag: " << atlag << endl;
    cin.get();
    return 0;
}
```

Feladat: Adjuk össze a ciklus változójának értékét és számláljuk meg az adatokat! A ciklusváltozó 10-es értékénél a **continue** kihagyja az iterációt és a 11-esnél folytatja. A ciklusból a 12. lépésnél lépünk ki **break** utasítással.

A program futásának eredménye:

```
i: 1    osszeg:    1    db:    1
i: 2    osszeg:    3    db:    2
i: 3    osszeg:    6    db:    3
i: 4    osszeg:   10    db:    4
i: 5    osszeg:   15    db:    5
i: 6    osszeg:   21    db:    6
i: 7    osszeg:   28    db:    7
i: 8    osszeg:   36    db:    8
i: 9    osszeg:   45    db:    9
i: 11   osszeg:   56    db:   10

atlag: 5.6
```

```
//MINTA3_05
#include <iostream>
using namespace std;
int main()
{
    char c = 'Q', kod = 'q';
    int space = 0, ujsor = 0, tab = 0;
    cout << "Szoveg olvasas (q betuig) : " << endl;
    while (c != kod)
    {
        c = cin.get();
        switch (c)
        {
            case ' ': space++; break;
            case '\n': ujsor++; break;
            case '\t': tab++; break;
        }
    }
    cout << "ures helyek szama: " << space << endl;
    cout << "uj sorok szama: " << ujsor << endl;
    cout << "vizszintes tab. szama: " << tab << endl;
    cin.get(); cin.get();
    return 0;
}
```

Feladat: Olvassunk karaktereket, és számláljuk meg, hogy hány szóközt, új sort, vízszintes *tab* karaktert ütöttünk le! A kilépés a *q* karakterrel történjen!

A program futásának eredménye:

```
Szoveg olvasas (q betuig) :
Ma szep az ido.
Holnap lehet, hogy esni fog.
Nem lesz jo.
q
ures helyek szama: 7
uj sorok szama: 3
vizszintes tab. szama: 2
```

Feladatok

1. Olvassuk be az adatok darabszámát (max. 15) egy *db* változóba! A **for** ciklus használatával *db* darab valós számot olvasva, számláljuk meg a pozitív, negatív és a zérus adatokat! (GYAK3_1)

```
int db, i, neg=0, poz=0, zerus=0;
double adat;
```

A program futásának eredménye:

```
Adatok szama: 4
Adat: 3
Adat: -4
Adat: 0
Adat: 6
Pozitiv adatok szama: 2
Negativ adatok szama: 1
Zerus adatok szama : 1
```

2. Olvassunk be egy kijelentő mondatot karakterenként a pontig a **while** ciklus használatával. (A karaktereket a `c = cin.get();` utasítással olvashatjuk be.) Számláljuk meg a magánhangzókat (a,e,i,o,u) és az egyéb karaktereket! (GYAK3_2)

```
char c = '*', kod = '.';
int a = 0, e = 0, i = 0, o = 0, u = 0, egyeb=0;
```

A program futásának eredménye:

```
Mondat beolvasasa a pontig (.) :
Ma szep az ido.
a betuk szama: 2
e betuk szama: 1
i betuk szama: 1
o betuk szama: 1
u betuk szama: 0
egyeb karakterek szama: 10
```

3. Olvassunk be egy pozitív egész számot 2 - 50 között! A helyes beolvasás ellenőrzéséhez használjunk **do - while** ciklust. Írjuk ki a beolvasott szám osztóit! (GYAK3_3)

```
int szam, i;
```

A program futásának eredménye:

```
Pozitiv egesz szam (max 50): 24
Osztoi:
2
3
4
6
8
12
```

4. Olvassunk be egy szót (max. 40 karaktert), és válasszuk szét a magánhangzókat az egyéb karakterektől (mássalhangzók)! (GYAK3_4)

```
char szo[40], maganh[40], massalh[40];
int i=0, j=0, k=0;
```

A program futásának eredményei:

```
Szo: diofa
Maganhangzok: ioa
Massalhangzok: df

Szo: kertkapupant
Maganhangzok: eaua
Massalhangzok: krtkppnt
```