CIKLUSOK

Do-While

While

készítette: Vastag Attila

2017

A program egyik legfontosabb tulajdonsága, hogy ismétléseket képes végrehajtani.

A **ciklus** (**iteráció**) **ismétlést** jelent, ilyenkor egy vagy több utasítás újra és újra végrehajtódik.

Vannak olyan esetek amikor az iterációk számát előre eltudjuk dönteni, de van azonban amikor a futási körülményektől függ. Ha az ismétlés feltételtől függően történik, vagy előre nem meghatározhatott ismétlésről van szó, két féle ciklust alkalmazhatunk:

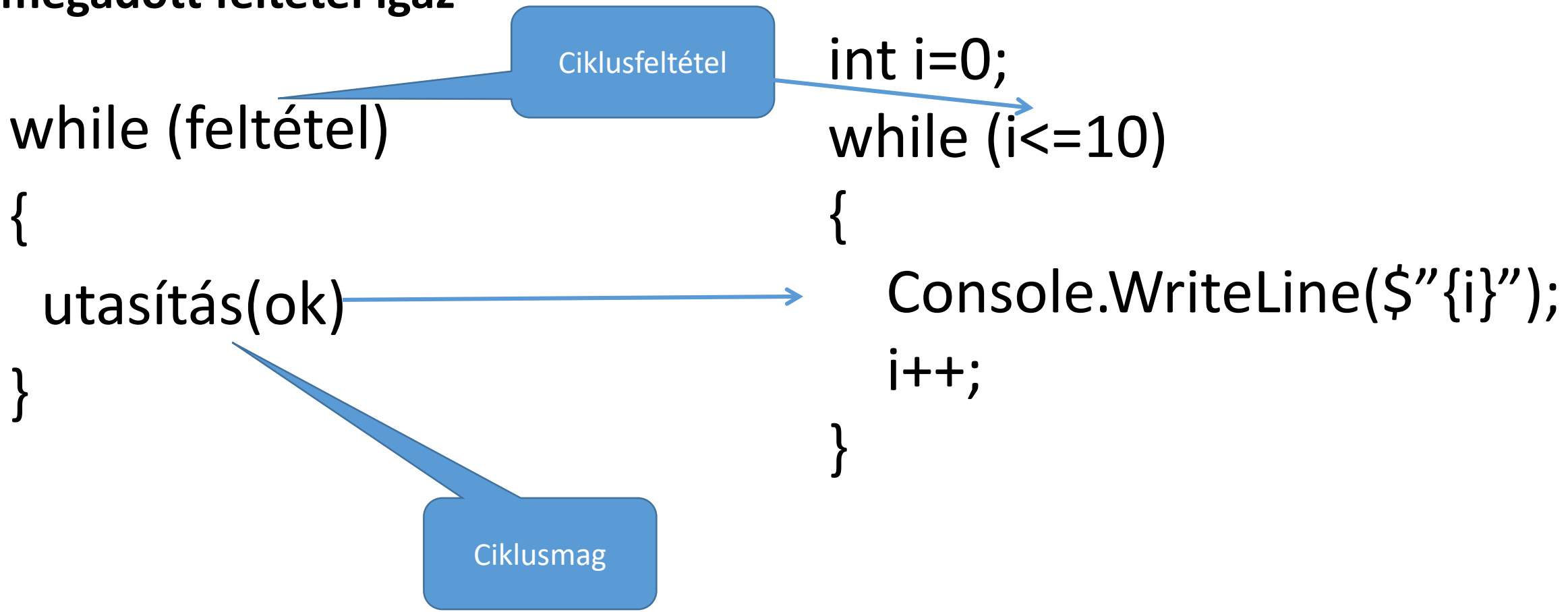
- **Előtesztelő ciklust** (*while*)
- **Hátultesztelő ciklust** (*do - while*)

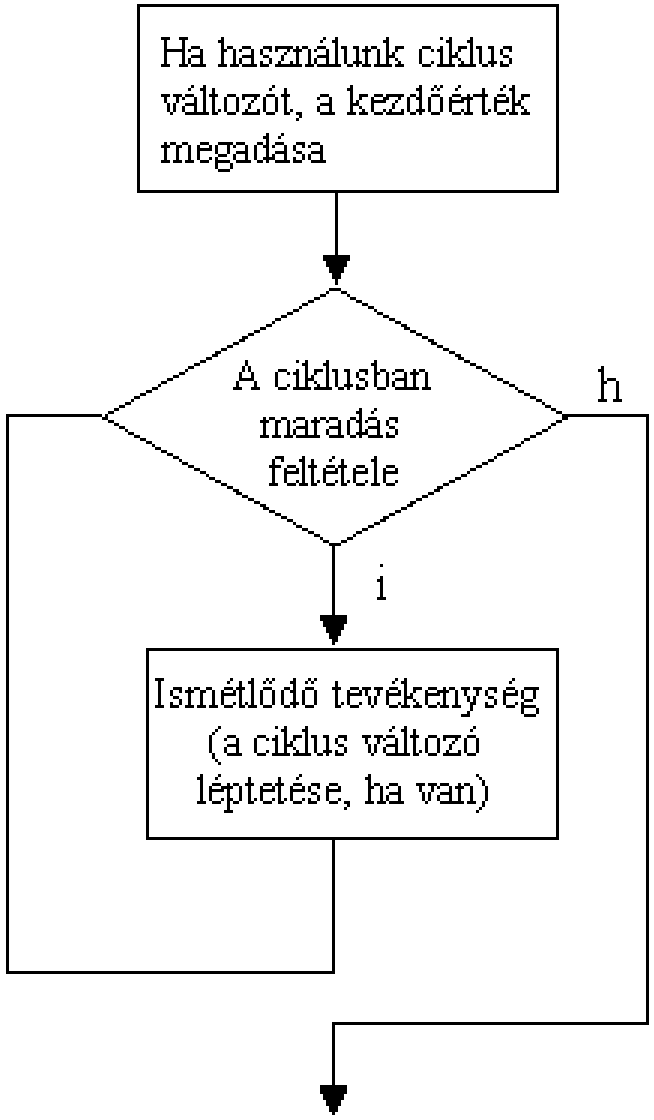
Ilyen problémák megoldására használni tudjuk a **for** végtelen ciklust is a **break** kulcsó segítségével, amely a ciklust megszakítja egy definiált feltétel teljesítése után!

While

előltesztelő ciklus

- A while ciklussal egy utasításblokkot mindaddig ismételhetünk, **míg egy megadott feltétel igaz**





A folyamatábra a *while* ciklus logikai modellje. Az első feldolgozást szimbolizáló téglalap csak akkor szükséges, ha a ciklusban használunk ciklus változót, esetleg azt a ciklus bennmaradási feltételeként alkalmazzuk. A ciklus feltétel része tartalmazza a ciklusban maradás feltételét. A második feldolgozás blokk tartalmazza a ciklus utasításait, valamint a ciklus változójának léptetését, természetesen csak abban az esetben, ha van ciklusváltozó. Ha megértjük a folyamatábra és az általános forma működési elvét, akkor azok alapján bármilyen *while* ciklust el tudunk készíteni.

A *while*-t használó megoldásnál ügyeljünk arra, hogy a ciklus változóját léptetni kell, vagyis az értékét minden lefutáskor növeljük meg eggyel, mert a *for* ciklussal ellentétben itt a rendszer erről nem gondoskodik. (Emlékezzünk vissza a *for* ciklusra! Ott a ciklus változó növelése automatikus és nem is célszerű azt külön utasítással növelni).

Ügyeljünk arra is, hogy a ciklus változójának mindig adjunk kezdő értéket, ellenkező esetben az ismétlések száma nem lesz megfelelő.

Természetesen a *while* ciklust leginkább a logikai értékek vizsgálatakor használjuk, s nem számláló ciklusként (arra ott van a *for*).

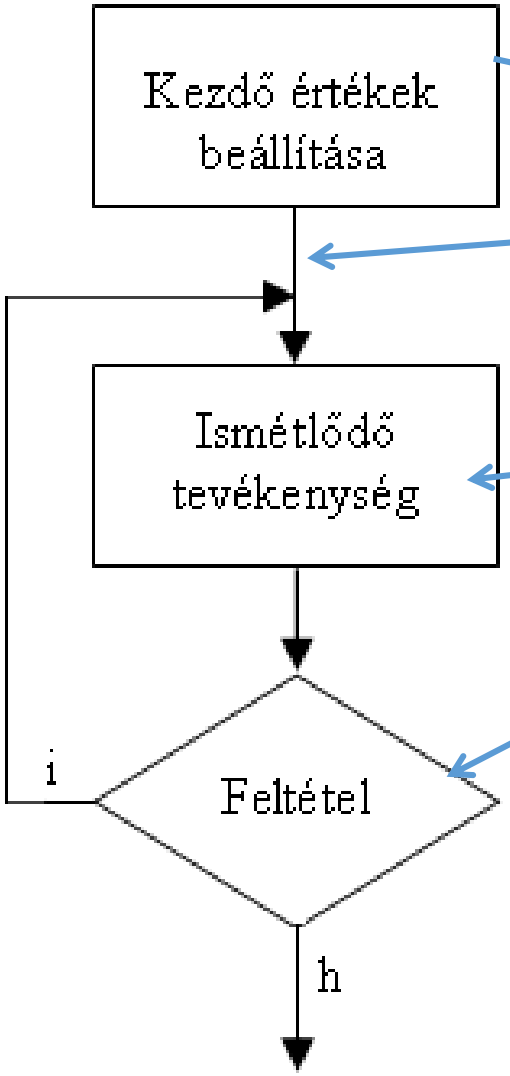
Vegyük sorra, hogy mikor célszerű logikai, vagy más néven feltételes ciklusokat alkalmazni.

- Amennyiben nem tudjuk meghatározni a ciklus ismétlésinek a számát (futás közben dől el).
- Ha a leállás valamilyen logikai értéket szolgáltató kifejezés eredményétől függ.
- Több leállási feltétel együttes használatakor.
- Az előző három pont együttes fennállása esetén.

Do - While

hátultesztelő ciklus

- A *while* utasítás feltétele az első ellenőrzéskor ha hamis, a ciklusmag egyszer sem fut le
- Amennyiben szükséges hogy a ciklus utasításai legalább egyszer lefussanak, úgy a *Do - While* parancsot, azaz hátul tesztelő ciklust használunk.
- A *Do* utáni utasítások végrehajtódnak, majd a *while* utáni feltétel igaz esetén ismétlésre kerül a ciklusmag, hamis feltétel esetén a ciklus után folytatjuk az utasításokat




```
do
{
utasítás(ok)
}
while (feltétel);
```

```
char c;
Console.WriteLine(”Kilépés : v”);
do
{
   c = Console.Read();
   Console.WriteLine($”A beolvasott változó : {c}”);
} while (c != ’v’);
```

•A *while* után zárójelekben megadott feltétel teljesülése esetén ismétli az utasításokat, ugyanúgy, mint az elől tesztelő ciklusnál, csak itt már a ciklusmag biztos, hogy lefut egyszer.

•A ciklusmagban törekedni kell arra, hogy a feltétel véges lépés után hamissá váljon, különben végtelen ciklust kapunk.

break

ciklusok megszakítása

break kulcsszó segítségével megszakíthatjuk a ciklust bármelyik pillanatban ha egy bizonyos feltétel teljesül.

A jobb oldali példában a *while* ciklus mindaddig futni fog még az *i* értéke el nem éri a 7-et. Mikor az *i* értéke 7 lesz, az *if* feltétele igaz lesz, végrehajtódik az *if* igaz ága és a **break** utasítás hatására a ciklusból kiugrik a program.

```
int i=0;
while (true)
{
    Console.WriteLine($”{i}”);
    if( i == 7)
    {
        break;
    }
    i++;
}
```

FELADATOK

megoldani DO és DO-WHILE ciklusokkal

1 – Kérjük meg a felhasználót, hogy adjon meg egy 0 – 9 közötti számot. Addig ismételjük amíg nem lesz jó a bevitel! Jó bevitel után írjuk ki ezt a számot a képernyőre.

2 – Írjunk programot, amely bekéri a nevünket, és ha azt megadtuk, akkor üdvözlő szöveggel üdvözli a felhasználót.

3 – A számítógép random generáljon egy számot 0 és 9 között. Majd a felhasználónak ki kell találni ezt a számot 5 nekifutásból.

4 – Írjunk programot, amely a felhasználótól addig kér be számokat, még azok összege meghaladja a 100-at. Minden bekérés után jelezni a felhasználónak a jelenlegi összeget és a hányadik bevitelnél tart.

5 – Írjunk programot, amelyben a felhasználó megad egy határértéket (min 100), majd a felhasználótól addig kér be számokat, még azok összege nem haladja meg a határértéket. Minden bekérés után jelezni a felhasználónak a jelenlegi összeget. Ha az összeg eléri a hatarerteket akkor kiírni, hány lépésben érte el a felhasználó a hatarerteket.

FELADATOK

megoldani DO és DO-WHILE ciklusokkal

6 - Olvassunk be egy életkort 0-99 között. Addig ismételjük amíg nem lesz jó a bevitel! Adjuk meg hogy melyik korosztályba esik az illető!
(0-6: gyerek, 7-18: iskolás, 19-65: dolgozó, 65- nyugdíjas)

7 – Kérjünk be egy számot és egy másikat úgy, hogy nagyobb legyen az elsőnél. Számoljunk visszafelé a nagyobbik számtól a kisebbik felé. (A feladat kiegészíthető azzal, hogy bekérjük a lépésközt is, ami kisebb kell legyen a két szám különbségénél.)

8 – Keszítsünk automata menüt ahol egy szám egy üdítőt jelent. A felhasználónak választania kell üdítőt, de csak azokból amelyek a kínálatban vannak. Amennyiben nem azt választjuk nem kapunk üdítőt.

9 – Kérjünk be egy 3 jegyű számot és állapítsuk meg, hogy osztható e 7-el. Addig ismételjük a bekérést, amíg nem 3 jegyű a megadott szám.

10- Kérjünk be egy max 2 jegyű, pozitív **n** számot:

- Írjuk ki 0 és **n** közt a páros számokat
- Adjuk össze 0 és **n** közt az 5-el osztható számokat
- Számoljuk meg, hány szám osztható 0 és **n** közt 11-el
- Írjuk ki azon számokat 0 és **n** közt amelyek 7-el osztva 3-at adnak maradékul

11 – Írjunk programot amely a felhasználótól bekér egy páros majd egy tőlle nagyobb páratlan számot. A következő lépésben generáljunk egy véletlen számot e két érték közt és határozzuk meg mely szám (páros vagy páratlan) van messzebb a véletlen számunktól.
Számítsuk ki a két bekért érték közti átlagot is.
Számoljuk meg a 4-el osztható számok számát is.