VEZÉRLŐSZERKEZETEK C# NYELVEN

Elágazások 1.

1. **Egyágú:**

```
if( feltétel)
{   utasítások; }
```

2. **Kétágú:**

```
if(feltétel)
{   utasítások; }
else
{   utasítások; }
```

3. **Többágú**

```
if(feltétel1)
{   utasítások; }
else if(feltétel2)
{   utasítások; }
else if(feltétel3)
{   utasítások; }
⋮
else
{   utasítások; }
```

Tetszőleges számú else if ág lehet

Elágazások 2.

Többirányú elágazás másképp: switch=kapcsoló

```
switch(kifejezés)
{
    case érték1:
        utasítások;
        break;
    case érték2:
        utasítások;
        break;
    case érték3:
        utasítások;
        break;
    default:
        utasítások;
        break;
}
```

A szelektor kifejezés típusa csak egész, vagy szöveges vagy logikai vagy felsorolás típus lehet.

Feltételek megfogalmazása

Összehasonlító operátorok és logikai operátorok segítségével

Összehasonlító operátorok

<, >, <=, >=, ==, !=

Példák:

a<b ,,a kisebb, mint b ?”

a<=b ,,a kisebb vagy egyenlő, mint b ?”

a>=b ,,a nagyobb vagy egyenlő, mint b ?”

a==b ,,a egyenlő b-vel?”

a!=b ,,a nem egyenlő b-vel?”

Logikai operátorok

&& A logikai ÉS operátor.

|| A logikai VAGY operátor

^ KIZÁRÓ VAGY operátor

Ciklusok

Ciklust akkor használunk, ha valamely utasítást vagy utasítások sorozatát nem csak egyszer, hanem többször is végre szeretnénk hajtani.

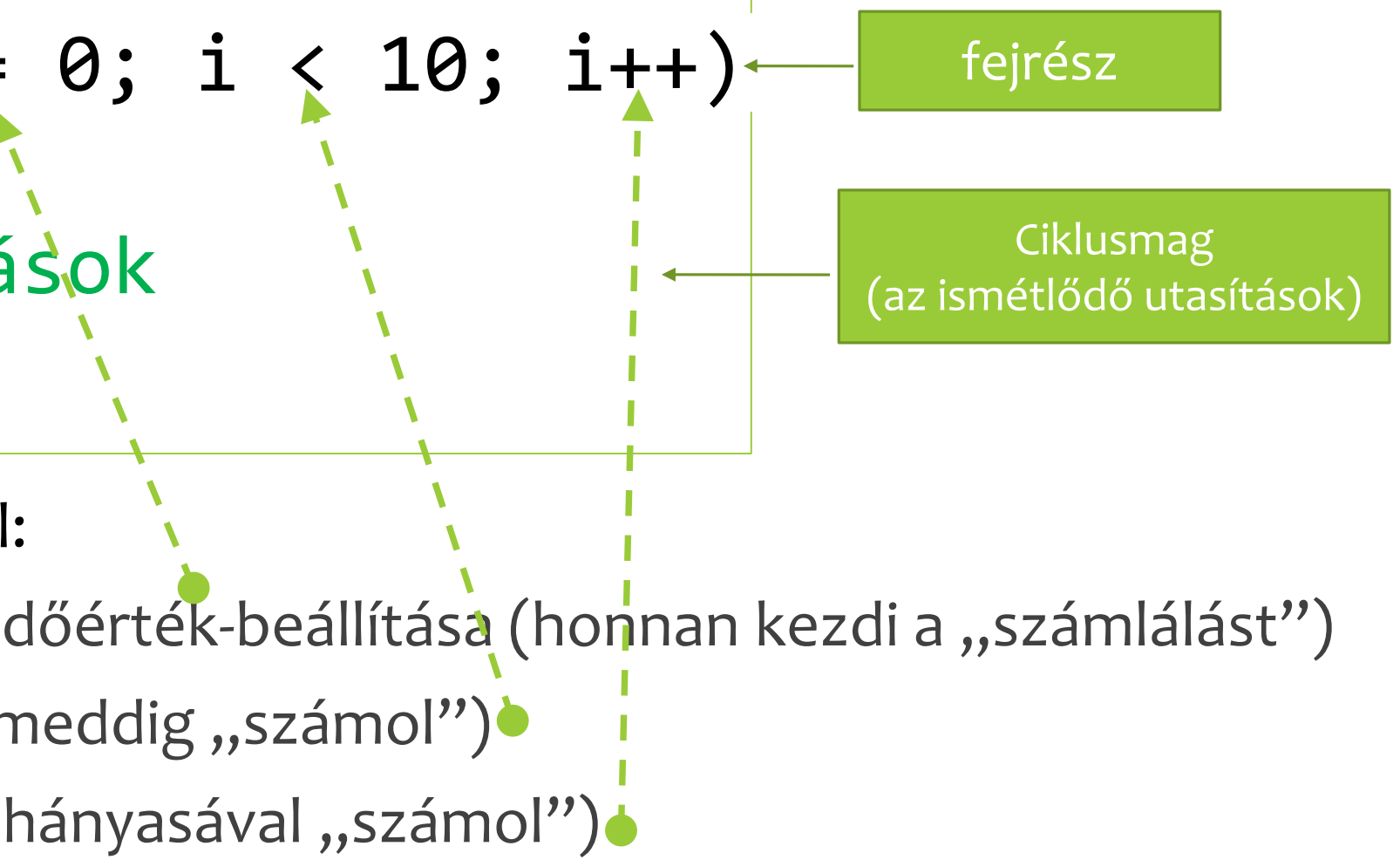
1. Számláló (meghatározott lépés számú) ciklus
2. Elől tesztelő
3. Hátul tesztelő

} feltételes ciklus

A C#-ban mindkét feltételes ciklus esetén bennmaradási feltételt adunk meg (nem kilépési feltételt).

1. for ciklus (számlálós ciklus)

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



A fejrész 3 részből áll:

1. ciklusváltozó kezdőérték-beállítása (honnan kezdi a „számlálást”)
2. vezérlő feltétel (meddig „számol”)
3. léptető utasítás (hányasával „számol”)

A for ciklus további lehetőségei:

- A fejrész bármelyik része elhagyható, de a két ;-t akkor is ki kell tenni.
- A break utasítással ki lehet „ugrani” a ciklusból.

```
for( ; ; )     //ez végtelen ciklus lesz, ha nem teszünk a
{       }      ciklusmagba break-et

for(int i=1;  ;i++)  //ez is végtelen ciklus lesz a break nélkül,
{                          mert hiányzik a bennmaradási feltétel
}
```

A for ciklus további lehetőségei:

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

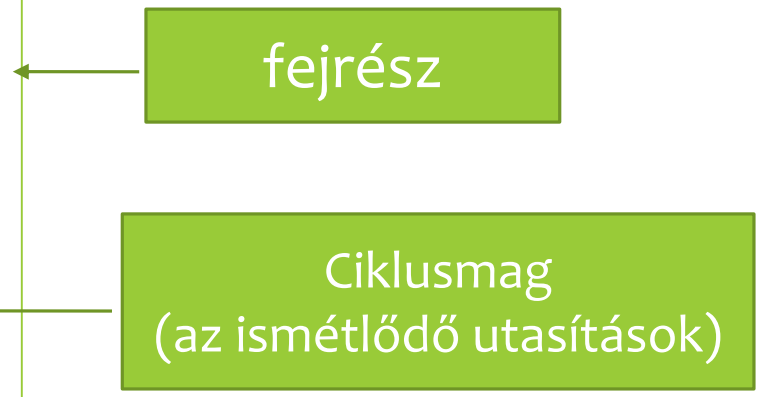
/*ha a fejrészből elhagyjuk a ciklusváltozó léptetését, akkor azt a ciklusmagban kell megtenni, különben ∞ ciklus lesz*/

```
int i=0;
for( ;i<100; i++)
{
   //utasítások
}
```

/*a ciklusváltozó definícióját kitehetjük a for ciklus elé, ha az értékére a cikluson kívül is szükség lesz – ilyenkor hatóköre nem csak a for ciklus blokkja*/

2. while ciklus (elöltesztelő feltételes ciklus)

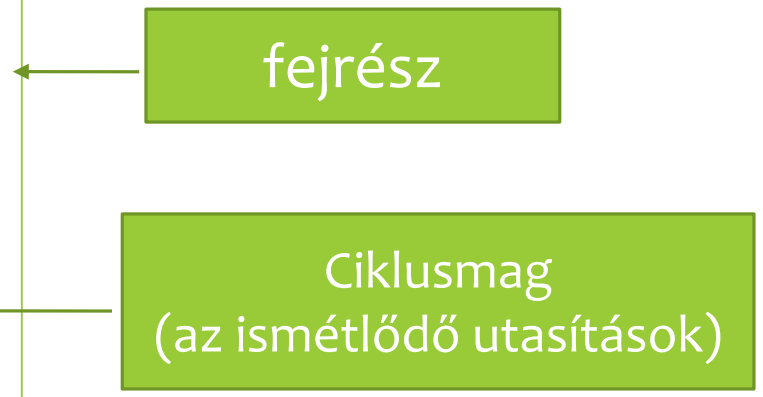
```
while (feltétel)
{
    //utasítások
}
```



1. A ciklus a vezérlő feltétel kiértékelésével kezdődik
2. A ciklusmag akkor fut le, ha a vezérlő feltétel igaz
3. Lehet, hogy a ciklusmag egyszer sem fut le

3. do-while ciklus (hátultesztelő feltételes ciklus)

```
do
{
    //utasítások
} while (feltétel);
```



1. A ciklus a ciklusmag végrehajtásával kezdődik
2. A ciklusmag akkor fut le, ha a vezérlő feltétel igaz
3. A ciklusmag egyszer biztosan lefut