



# Metódusok

Répásné Babucs Hajnalka



# Metódusok



- Utasítások logikailag összefüggő csoportja, melynek önálló neve és visszatérési értéke van
- Segítségével megváltoztathatjuk egy objektum állapotát, vagy kiolvashatjuk tulajdonságainak értékét, leírhatjuk viselkedését
- C#-ban metódusok csak osztályon belül definiálhatók (**tagfüggvény**)
- Ha nem adjuk meg, akkor az alapértelmezett hozzáférési szint a **private**.





# Metódusok felépítése



[módosító] visszatérési\_érték tagfüggvénynév([paraméterlista])

{ //Metódus törzse

lokális deklarációk;

utasítások;

[ return [kifejezés]; ]

}



```
class Program
{
    static void Main(string[] args)
    {
    }
}
```





# Paraméterek



- A metódusok rendelkezhetnek megadott vagy változó darabszámú paraméterrel (**params** kulcsszó)
  - void ParaméternélküliMetódus();
  - void EgyparaméteresMetódus(bool feltétel);
  - void TöbbparaméteresMetódus(int a, float b, string c);
  - void MindenbőlSokatElfogadóMetódus(params object[] paraméterTömb);





# Paraméterlista



➤ A **paraméterlista** arra szolgál, hogy a hívó programmodul értékeket adjon át a meghívott metódusnak, amellyel az el tudja végezni a feladatát.



➤ A metódus deklarációjában szereplő paraméterlistát szokás **formális paraméterek**nek nevezni



➤ A **metódus hívás**kor megadott argumentumok alkotják az **aktuális paraméterek**et









➤ Lehetnek változó, konstans értékek vagy kifejezések is



➤ Az aktuális paraméterlistának típusban, sorrendben és számban meg kell egyeznie a formális paraméterlistával



# Paraméter típusok

- 
- 
- 
- A paraméterek lehetnek érték, referencia, vagy out típusúak
    - Érték típusú: tipikusan bemenő paraméter. Nincs külön jelölése.
    - A referencia paraméter mind bemenő, mind visszatérő értéket tárolhat – a paraméter megváltozhat. A **ref** kulcsszó jelzi ezt a típust. A ref kulcsszónak a hívásnál is szerepelnie kell!
    - Az **out** kulcsszó jelzi az output paramétert. Az out kulcsszónak a hívásnál is szerepelnie kell!
- 
- 
- 





# Érték szerinti paraméterátadás



- A metódusban létrejövő lokális paraméterváltozóba átmásolódik a híváskor megadott adat vagy hivatkozás
- Az aktuális paraméterlista értékei inicializálják a formális paraméterlistában szereplő változókat
- A hívó oldalon szereplő aktuális paraméterek és a metódus formális paraméterlistáján szereplő változók más-más memóriaterületet foglalnak el
  - Ennek köszönhető, hogy a metódusok az érték szerinti paraméterátadásnál a hívó oldal változóinak értékét nem változtatják meg





# Cím szerinti (referencia) paraméterátadás



- A metódus hívásakor a megadott objektumra mutató hivatkozás adódik át a formális paraméternek
- A paraméteren keresztül magát az eredeti objektumot érjük el, legyen akár érték vagy referencia típusú
- A metódus formális paraméterlistáján és a híváskor az aktuális paraméterlistán is a **ref** kulcsszóval jelezni kell, hogy hivatkozást adunk át a metódusnak







# Kimenő (output) paraméterek



- Arra használjuk őket, hogy a metódusból adjunk vissza értékeket a hívó programkódba.
- Az aktuális paraméterlistán kifejezés vagy konstans nem feleltethető meg az out módosítóval rendelkező formális paraméternek, csak változó
- Mindenképpen értéket kell kapnia a metódusból való kilépés előtt
- Az **out** kulcsszóval jelzett formális és aktuális paraméter ugyanarra a memóriahelyre mutat, hasonlóan a referenciatípushoz
- A metódus hívása előtt az aktuális paramétert nem kell inicializálni





# Paramétertömbök



- A params módosító lehetővé teszi, hogy nulla vagy több aktuális paraméter tartozzon a formális paraméterhez
- Egy paraméterlistán csak egy paramétertömb lehet, és ennek kell az utolsónak lennie
- A paramétertömb csak azonos típusú elemeket tartalmazhat



```
private static int Osszegez(params int[] szamok)
{
    int osszeg = 0;
    foreach (var szam in szamok)
    {
        osszeg += szam;
    }
    return osszeg;
}
```

```
int t1 = Osszegez(1, 2); //3
int t2 = Osszegez(5, 10, 15, 30); //60
```



# Visszatérési érték



- A metódusok rendelkezhetnek visszatérési értékkel
  - Nem kötelező, ha nincs, ezt a **void** kulcsszóval kell jelölni
  - `void NincsVisszatérésiÉrtékem();`
  - `int EgészSzámotAdokVissza(float paraméter);`
- A visszatérési érték tetszőleges saját típus is lehet
- A paraméterek és a visszatérési érték határozzák meg azt a protokollt, amelyet a metódus használatához be kell tartani
  - Ez a metódus **szignatúrája**





# Metódus törzs



- Végrehajtandó utasítások, melyek használhatják a bemenő paramétereket
- A **függvény** visszatérő értéke a **return** kulcsszót követi
  - Ebből több is lehet a program különböző ágain
- Visszatérési érték nélküli (void) metódusnál a return utasítás nem kötelező
  - Ezeket a metódusokat hívjuk **eljárásnak**





# Alapértelmezett értékű, opcionális paraméterek

- Ha a metódusnak van olyan paramétere, amit az esetek többségében ugyanazzal az értékkel hívunk meg, a paraméternek adhatunk alapértelmezett értéket.
- Opcionális paraméterek csak a kötelező paraméterek után szerepelhetnek.

```
private static int Osszegez(int meddig=10)
{
    int osszeg = 0;
    for (int i=1; i<meddig; i++)
    {
        osszeg += i;
    }
    return osszeg;
}
```

```
int t1 = Osszegez(); //első 10 szám összege
int t2 = Osszegez(100); //első 100 szám összege
```







# Nevesített paraméterek



- Metódus hívásakor a paraméterekre a nevükkel is hivatkozhatunk.



```
void OpcionalisParameteres(int kotelezo, string opcionalisSzoveg = "tigris",  
                           int opcionalisSzam = 10)
```



- Helyes meghívások:

```
OpcionalisParameteres(6);
```

```
OpcionalisParameteres(6, "kistigris");
```

```
OpcionalisParameteres(6, opcionalisSzam: 12);
```

```
OpcionalisParameteres(opcionalisSzam: 12,  
                       kotelezo: 6 );
```



- Hibás meghívások:

```
OpcionalisParameteres();
```

```
OpcionalisParameteres(6, ,12);
```