Budapesti Műszaki és Gazdaságtudományi Egyetem

Közlekedés- és Járműirányítási Tanszék

# Programozás
# BMEKOKAA146

Dr. Bécsi Tamás

5. előadás

# Tömbök átméretezése



- `public static void Resize<T>( ref T[] array, int newSize )`
- Példa:

```
int[] a=new int[20];
Array.Resize<int>(ref a, 22);
```

# Struktúrák



Hozzáférési szintek:

- **private** csak struktúrán belülről érhető el
- **public** bárki elérheti ezt a mezőt
- **internal** programon belülről (assembly) elérhető

```
struct pont
        {
            public int x;
            public int y;
        };
```

# Struktúrák



- A struktúra elemeihez hozzáférni a . struktúratag operátorral tudunk.
- A struktúra tag operátor a legmagasabb precedenciájú, csak a [],() zárja ki.
- A struktúrák esetében csak az értékadás operátort használhatjuk.
- A struktúrák érték típusok, értékadásnál másolás történik

```
Pont p, p1;
p = new Pont();
p.x = 1;
p.y = 2;
p1 = p;
```

# Függvények
# Bevezetés



- Egy idő után az egyetlen `Main()` függvénnyel megírt programunk túl nagy méretű lesz.
- Vannak programrészek, amelyekre több helyen is szükségünk lehet.
- A túl bonyolult algoritmusok a teljes program átláthatóságát csökkentik.
- A fentiek miatt célszerű a program tagolása, ahol az egyes alfeladatokat el tudjuk különíteni, így strukturálva a programot.

# Függvények, metódusok



- A függvények, illetve metódusok azt a célt szolgálják, hogy egy - adott esetben többször felhasználandó - kódrészletet összefoglaljuk, és azt a programunk több pontján is meghívjuk.
- A függvények rendelkezhetnek visszatérési értékkel.
- A függvények vehetnek át értékeket (paramétereket), az őket meghívó programrésztől.
- A függvények definíciója az alábbi:

```
<láthatóság> <visszatérési típus> név(<paraméterek>)
{
        <függvény kód>
}
```

# Függvények paraméterei



- A függvények paraméterei a paraméterlistában kerülnek átadásra.
- A paraméterlista vesszővel ellválasztott, `típus paraméternév` párosok, amelyet a függvény belül „változóként” kezel.
- Az objektumorientált programozás bevezetéséig a függvények és metódusok Console Application-ön belüli, program class-hoz tartozó kezelését tanuljuk. Ennek megfelelően a függvényeink `static` kulcsszóval kell bevezetnünk.
- A visszatérési érték nélküli függvények visszatérési típusa `void`.

```
static void szamkiir(int a)
        {
            Console.WriteLine(a);
        }
```

# 4. Függvények visszatérés



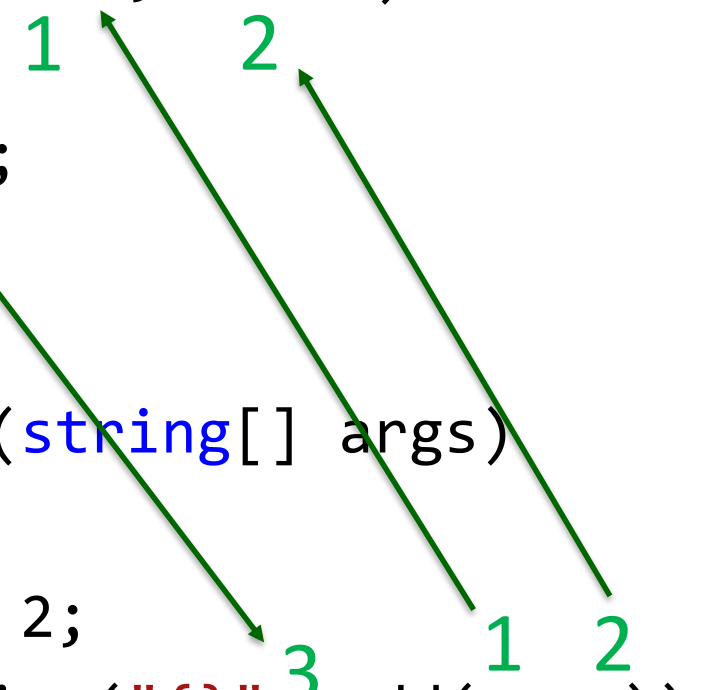
- A hívott függvény a hívó függvénynek a `return` utasítással adhat vissza értéket, melyet tetszőleges kifejezés követhet:
- Formája: `return kifejezés;`
- Ha szükséges, a kifejezés típusa a visszatérési típusra konvertálódik.
- A return utáni kifejezés opcionális. Adott esetben a return is elhagyható.
- A return utasítás a meghívásának pontján tér vissza a hívott függvényből, akkor is ha utána még más kódrészletek találhatóak.

# 4. Függvények példa



```
static int add(int a, int b)
{
    return a + b;
}

static void Main(string[] args)
{
  int x = 1, y = 2;
  Console.WriteLine("{}", add(x, y));
}
```



# 4. Függvények
# Paraméterátadás



- A függvények a paramétereket érték szerint veszik át, azaz amennyiben a paraméter értéke a függvényen belül változik, ez nincs hatással az átadott változó értékére.

# 4. Függvények
# Érték szerinti paraméterátadás példa



```
static void swap(int a, int b)
{
  int k = a;
      a = b;
      b = k;
}
 static void Main(string[] args)
{
  int x = 1, y = 2;
  Console.WriteLine("x:{0} y:{1}", x, y);
  swap(x, y);
  Console.WriteLine("x:{0} y:{1}", x, y);
  Console.ReadLine();
}
```

1 2

1 2

a=2, b=1, de csak a függvényen „belül”

„x:1 y:2” Tehát nem cserélődött fel A két változó értéke

# 4. Függvények
# Paraméterátadás (tömb típus)



- Az érték szerinti paraméterátadás „máshogy működőnek tűnik” tömb típusok esetén, hiszen a tömb elemeinek módosítása ténylegesen módosítja ezeket az értékeket.
- Ennek oka, hogy a tömb referencia típus.
- Lásd mintapélda

# Referencia szerinti paraméterátadás



- Amennyiben azt szeretnénk, hogy egy függvény változtathasson az átvett paraméteren, akkor referencia szerinti átadást alkalmazunk, ekkor a paraméterlistában a paramétert ref kulcsszóval látjuk el:

```
static void swap(ref int a, ref int b)
```

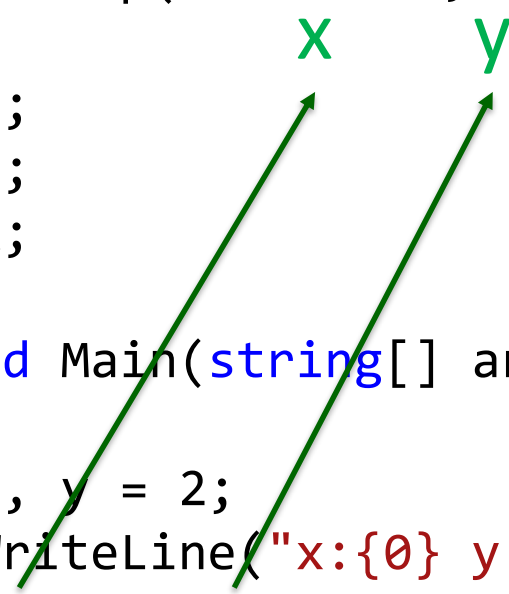
- Ugyanezt a ref kulcsszót kell használnunk, a függvény meghívásakor is:

```
swap(ref x,ref y);
```

# 4. Függvények
# Referencia szerinti paraméterátadás példa



```
static void swap(ref int a, ref int b)
{
  int k = a;
      a = b;
      b = k;
}
 static void Main(string[] args)
{
  int x = 1, y = 2;
  Console.WriteLine("x:{0} y:{1}", x, y);
  swap(ref x, ref y);
  Console.WriteLine("x:{0} y:{1}", x, y);
  Console.ReadLine();
}
```




a és b függvényen „belül” x és y szerepét viszi

„x:2 y:1” Tehát a két változó értéke felcserélődött

# Változók hatóköre



- Egy változó deklarálható osztályszinten, metódus szinten, vagy beágyazott szinten. Ezen változók mindig csak az adott szinten belül érvényesek.

```
class Program   {
        static int a = 1; //osztályszintű változó
        static void Main(string[] args)
        {
            int i=10;//függvény szinten deklarált változó
            for (int j = 0; j < i; j++)
            {
                int x = 3;
                a += x + j;
            }
            j = 3;//HIBÁS, nem elérhető
            x = 4;//HIBÁS, nem elérhető
        }
        static void add_a()
        {
            a++;//osztályszintű változó elérhető
        }
    }
```

j és x hatóköre

i hatóköre

a hatóköre