Code Academy

Dėstytojas

Vilmantas Neviera

Ref ir out

Data

Šiandien išmoksite

01 Ref

02 Out

Kas yra ref raktažodis?

ref raktažodis **C#** kalboje naudojamas, kad perduoti kintamąjį kaip nuorodą į metodą, o ne kaip reikšmę. Kai kintamasis perduodamas kaip nuoroda, tai reiškia, kad bet kokie pakeitimai, padaryti kintamajam metodo viduje, bus atspindėti ir išoriniame kintamajame. Tai yra, pakeitus kintamojo reikšmę metodo viduje, originalus kintamasis, išorėje metodo, taip pat bus pakeistas.

Kas yra ref raktažodis?

Paprasčiausiai tariant, **ref** leidžia mums keisti kintamųjų reikšmes metodo viduje.

Štai paprastas pavyzdys, kaip galime naudoti **ref**:

```
private static void Main(string[] args)
{
    int a = 100;
    Console.WriteLine(a);  // Spausdins: 100
    ChangeValue(ref a);
    Console.WriteLine(a);  // Spausdins: 200

}

1 reference
static void ChangeValue(ref int x)
{
    x = 200;
}
```

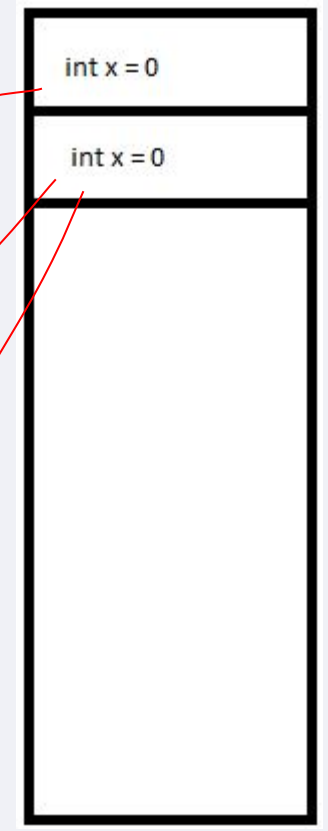
Čia, mes turime metodą **ChangeValue**,
kuris priima int tipo kintamąjį kaip nuorodą (per **ref** raktažodį).
Metode **ChangeValue** mes keičiame kintamojo **x** reikšmę į 200.
Kadangi **x** yra perduotas kaip nuoroda, šis pakeitimas atsispindi
ir kintamajame **a**, išorėje metodo **ChangeValue**.
Todėl, spausdindami **a** po metodo **ChangeValue**
iškvietimo, matome naują reikšmę - 200, o ne originalią reikšmę 100.
Tai labai svarbu suprasti, kad **ref** reikalauja, jog kintamasis
būtų iš anksto inicializuotas prieš perduodant jį kaip nuorodą į metodą.

Kodėl tai vyksta?

Programa kintamuosius saugo atmintyje, kiekvienam kintamajam išskirdama reikiama kiekį atminties, šie kintamieji “gyvens” atmintyje tol, kol neužsibaigs jų **scope**(šiuo metu, užtenka suprast, kad scope yra metodas kuriame kintamasis buvo sukurtas). Tad jeigu perduodame kintamąjį kaip parametrą į kitą metodą, naujas kintamasis yra sukuriamas ir išskiriama(**allocation**) nauja vieta atmintyje.

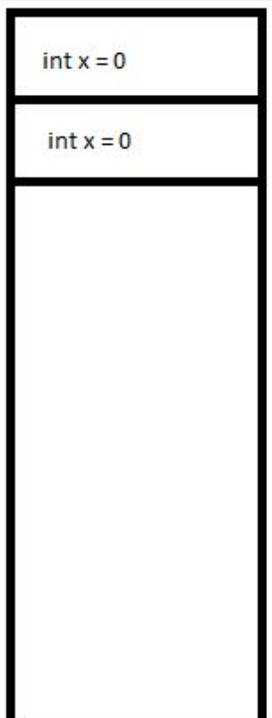
```
static void Main(string[] args)
{
    int x = 0;
    DoSomething(x);
    Console.WriteLine(x);
}

1 reference
public static void DoSomething(int x)
{
    x += 1;
}
```



Kodėl tai vyksta?

To pasekoje, jeigu mes metode **DoSomething()** pakeičiame kintamojo **x** reikšmę, originali reikšmė, kurią perdavėme **main** metode nepasikeičia, nes kaip matome, atmintyje yra sukurtas visai kitas kintamasis, kuris tiesiog turi tą pačią reikšmę.



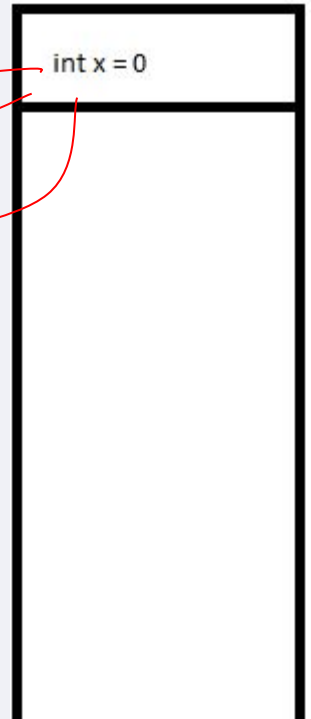
Jeigu galvojate, kaip programa atsirenka, kurį **x** kintamąjį naudoti, tai “po kapotu” kintamieji turi adresus, kaip kad turi butai ar namai.

Kodėl tai vyksta?

Tačiau, jeigu apsirašome metodą su **ref** raktažodžiu, mes pasakome, kad į šį metodą, perdavinėsime ne reikšmę, o originalų kintamąjį, kitaip sakant perduosime adresą atmintyje originalaus kintamojo.

```
static void Main(string[] args)
{
    int x = 0;
    DoSomething(ref x);
    Console.WriteLine(x);
}

1 reference
public static void DoSomething(ref int x)
{
    x += 1;
}
```



Užduotis nr. 1

- Sukurkite metodą **Swap**, kuris sukeičia dviejų int tipo kintamųjų reikšmes. Šie kintamieji turi būti perduodami kaip nuoroda (**ref**).
- Sukurkite metodą **IncrementByN**, kuris priima **int** tipo kintamąjį kaip nuorodą (**ref**) ir didina jo reikšmę nurodytu skaičiumi.
- Sukurkite metodą **TrimAndCapitalize**, kuris priima string tipo kintamąjį kaip nuorodą (**ref**). Metodas turi pašalinti iš eilutės pradžios ir pabaigos tarpus (jei tokių yra) ir padaryti pirmąją eilutės raidę didžiąja.

Kas yra out?

Out raktinis žodis yra speciali **C#** kalbos ypatybė, naudojama parametrams, kuriem atiduos reikšmę įgautą metode. Skirtingai nuo **ref** raktinio žodžio, **out** parametras nereikalauja pradinės reikšmės prieš perduodant jį į funkciją. Viskas, ką reikia padaryti, tai deklaruoti kintamąjį prieš jį perduodant kaip **out** parametrą.

Kas yra out?

Priklausomai nuo situacijos, **out** raktinis žodis gali būti naudingas skirtingose situacijose:

Kai funkcijos turi grąžinti daugiau nei vieną reikšmę.
Kai norite, kad funkcija priskirtų reikšmę kintamajam, kurio pradinė reikšmė yra nesvarbi.

Apsauga:
Taip pat reikėtų paminėti, kad funkcija, naudojanti out parametrą, privalo priskirti reikšmę šiam parametrui prieš išeinant iš funkcijos. Kitaip tariant, funkcijos, kurios naudoja **out** parametrus, privalo garantuoti, kad **out** parametras bus inicializuotas.

Kas yra out?

Ką verta pastebėti, kad width ir height kintamieji perduodami su out raktažodžiu yra tuo pačiu metu sukuriami ir perduodami.

Kaip matote aukščiau **GetDimensions()** metodo jie nėra deklaruojami, bet po **GetDimensions()** metodo jie yra naudojami spausdinimui.

```
static void Main(string[] args)
{
    GetDimensions(out int width, out int height);
    Console.WriteLine($"Plotis: {width}, ilgis: {height}"); // Spausdins "Plotis: 1024, ilgis: 768"
}

1 reference
public static void GetDimensions(out int width, out int height)
{
    width = 1024;
    height = 768;
}
```

Kas yra out?

```
public static int GetUserSelection()
{
    while (true)
    {
        Console.WriteLine("Įveskite pasirinkimą");
        if (!int.TryParse(Console.ReadLine(), out int input))
        {
            Console.WriteLine("Įvestas blogas pasirinkimas");
            continue;
        }
        return input;
    };
}
```

Čia matote du realius **out** pritaikymus.

Dviem būdais pamodifikuotas metodas iš vienos mūsų paskaitų.

Pirmasis metodas naudojant **while(true)** gali pasirodyti paprastesnis, bet geriau jo vengti, nes nėra tiesioginio kintamojo per kurį galima kontroliuoti ciklą.

Antrame pavyzdyje ciklą kontroliuojame kintamuoju **isValid**.

```
public static int GetUserSelection()
{
    bool isValid = false;
    int input = 0;
    while (!isValid)
    {
        Console.WriteLine("Įveskite pasirinkimą");
        if (!int.TryParse(Console.ReadLine(), out input))
        {
            Console.WriteLine("Įvestas blogas pasirinkimas");
            continue;
        }
        isValid = true;
    };
    return input;
}
```

Užduotis nr. 2

- Sukurkite metodą **GetUserData**, kuris priima du **string** tipo kintamuosius (**out**): vardą ir pavardę. Metodas turi nuskaityti iš konsolės vartotojo vardą ir pavardę, tuomet grąžinti juos per out parametrus.
- Sukurkite **while** ciklą, kuris tęsiasi tol, kol vartotojas neįves skaičiaus, didesnio už 100. Cikle naudokite metodą **TryParse**, kuris grąžina dvi reikšmes: ar pavyko konvertuoti įvestį į skaičių (**bool** tipo kintamasis) ir patį skaičių (jei konvertacija buvo sėkminga). Šis metodas naudoja **out** parametrą skaičiui grąžinti.
- Sukurkite metodą **Divide**, kuris priima du **double** tipo skaičius ir atlieka dalybą. Metodas turėtų grąžinti dalmenį kaip rezultatą ir liekaną per **out** parametrą. Šis metodas turi būti saugus, t.y., jeigu yra bandoma dalinti iš nulio, metodas grąžina false ir nesetina dalmenį bei liekaną.