EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

# Programování a algoritmizace

## Seminář

Strategický projekt UTB ve Zlíně, reg. č. CZ.02.2.69/0.0/0.0/16_015/0002204

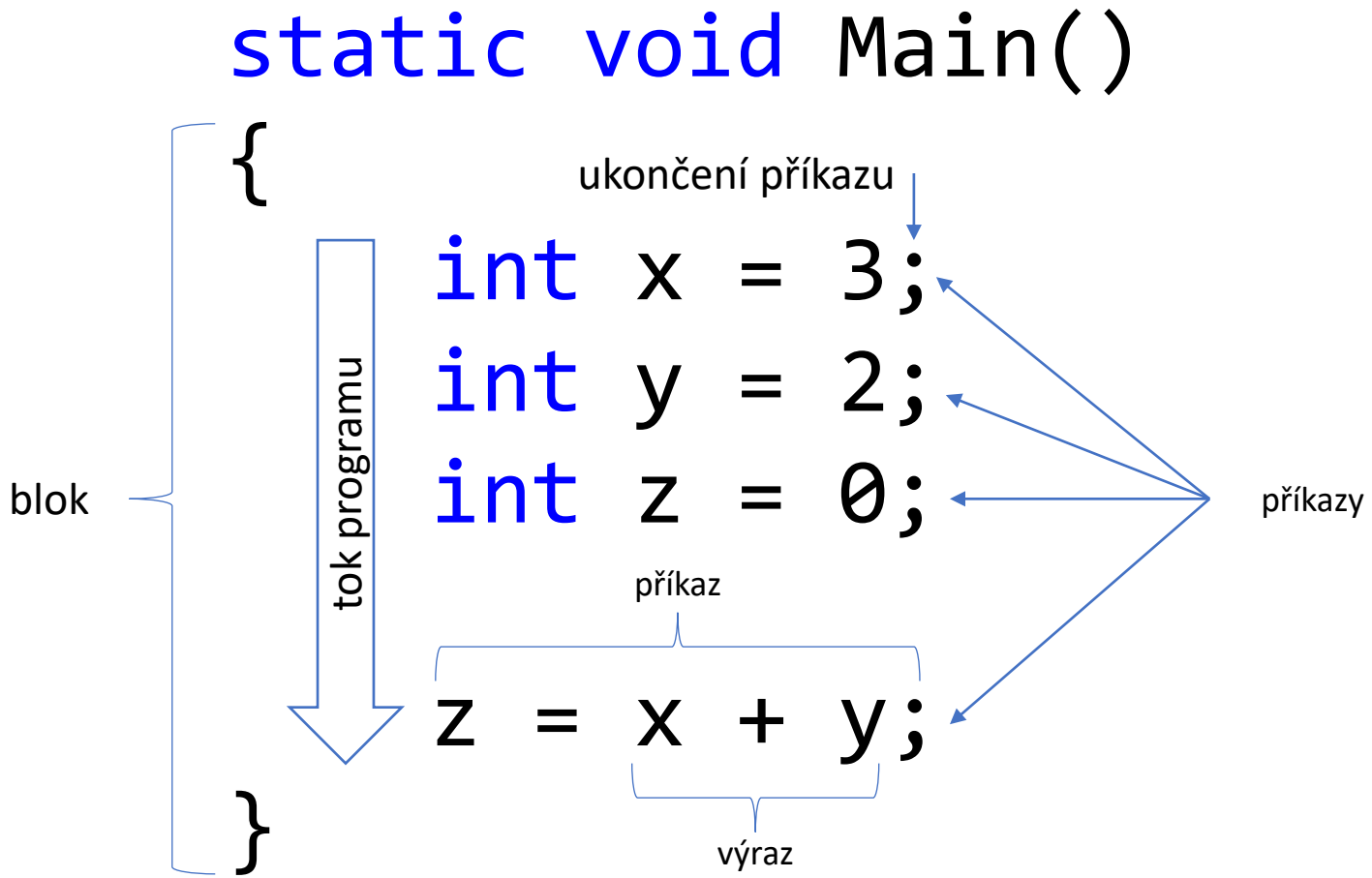
Ing. et Ing. Erik Král, Ph.D.
ÚPKS
Fakulta aplikované informatiky

# Obsah

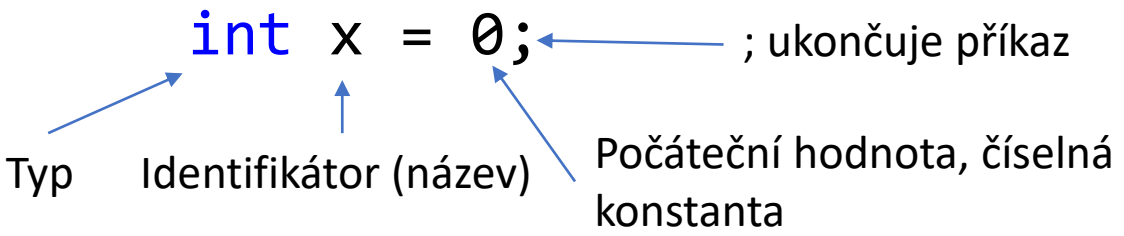
# Příkazy a výrazy

- V následujících snímcích probereme proměnné, příkazy a výrazy.
- Nejprve si ukažme jednoduchý kus kódu na kterém si popíšeme základní prvky programu.

# Příkazy a výrazy



# Definice proměnné




Každá proměnná musí být před použitím deklarována. Deklarace vytváří proměnné, uvádějí jejich typ, identifikátor a někdy i počáteční hodnoty [1]

# Nejpoužívanější typy

`int celeCislo;`

celé číslo se znaménkem, dostatečný rozsah pro většinu programů

`double desetinne;`

desetinné číslo se znaménkem, dvojitá přesnost, dostatečná přesnost pro většinu běžných programů

`bool logicka;`

Booleovská proměnná, nabývá dvou stavů `true` nebo `false`

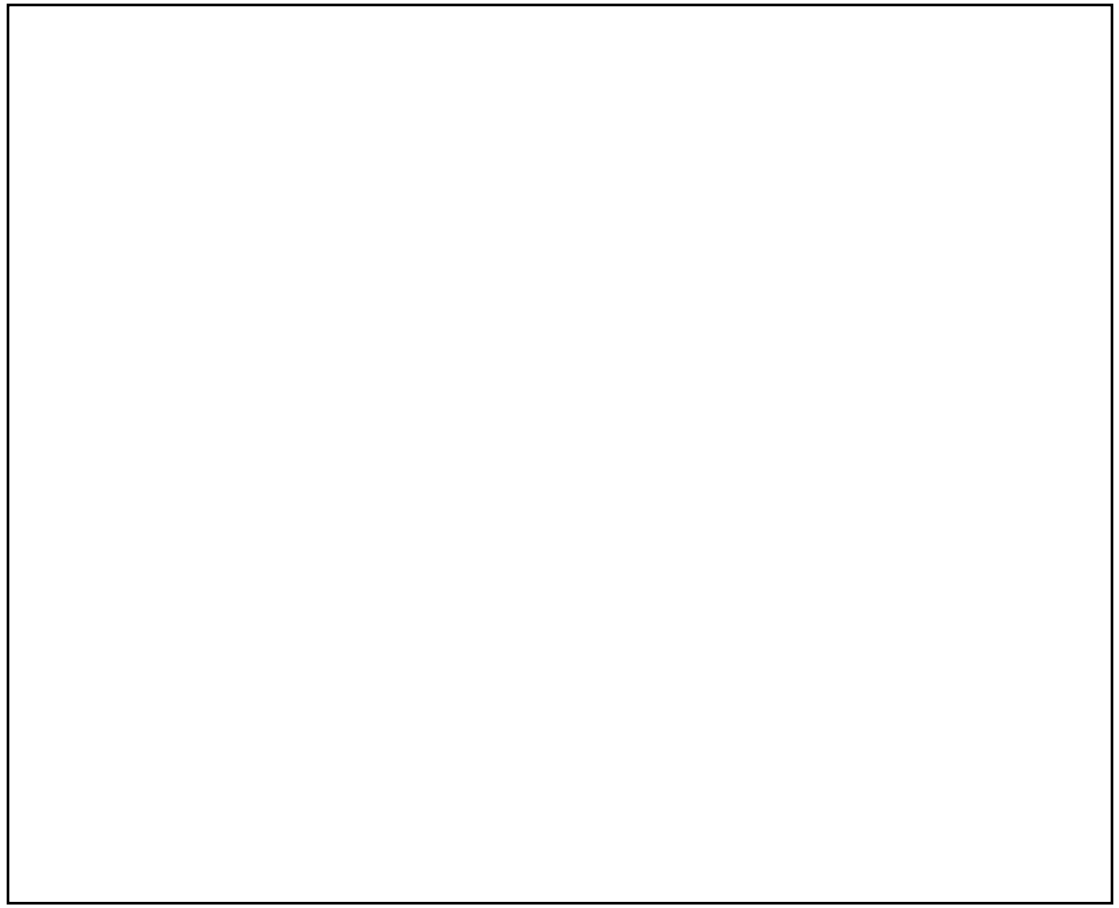
# Příklad na proměnnou a paměť

- V následujících snímcích si projdeme příklad na definici lokální proměnné z hlediska paměti RAM.
- Budeme definovat dvě proměnné, proměnnou *x* a *y* a potom pomocí operátoru přiřazení přiřadíme proměnné *y* hodnotou proměnné *x*.

# Proměnná a paměť

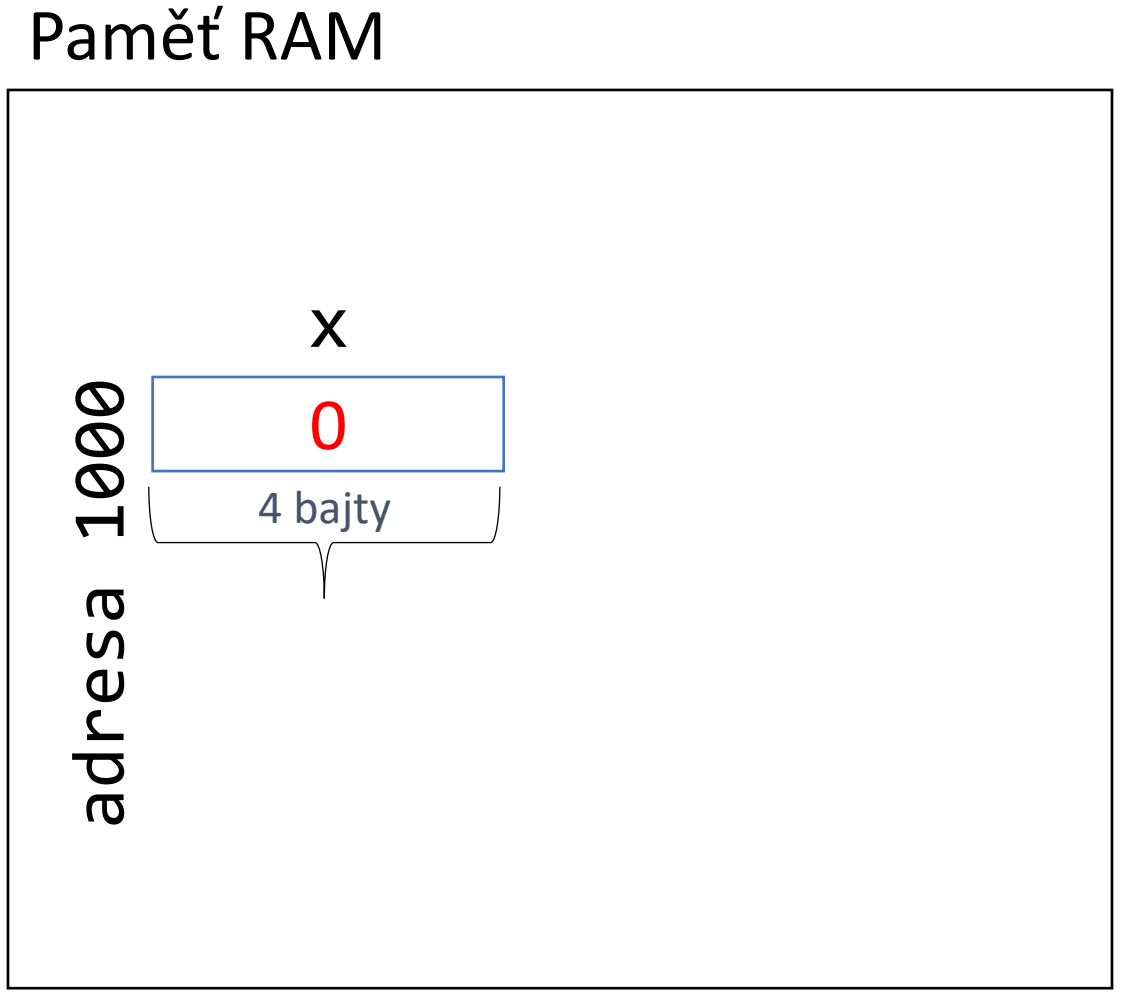
```
int x = 0;
x = 3;
int y = 0;
y = x;
```

Paměť RAM



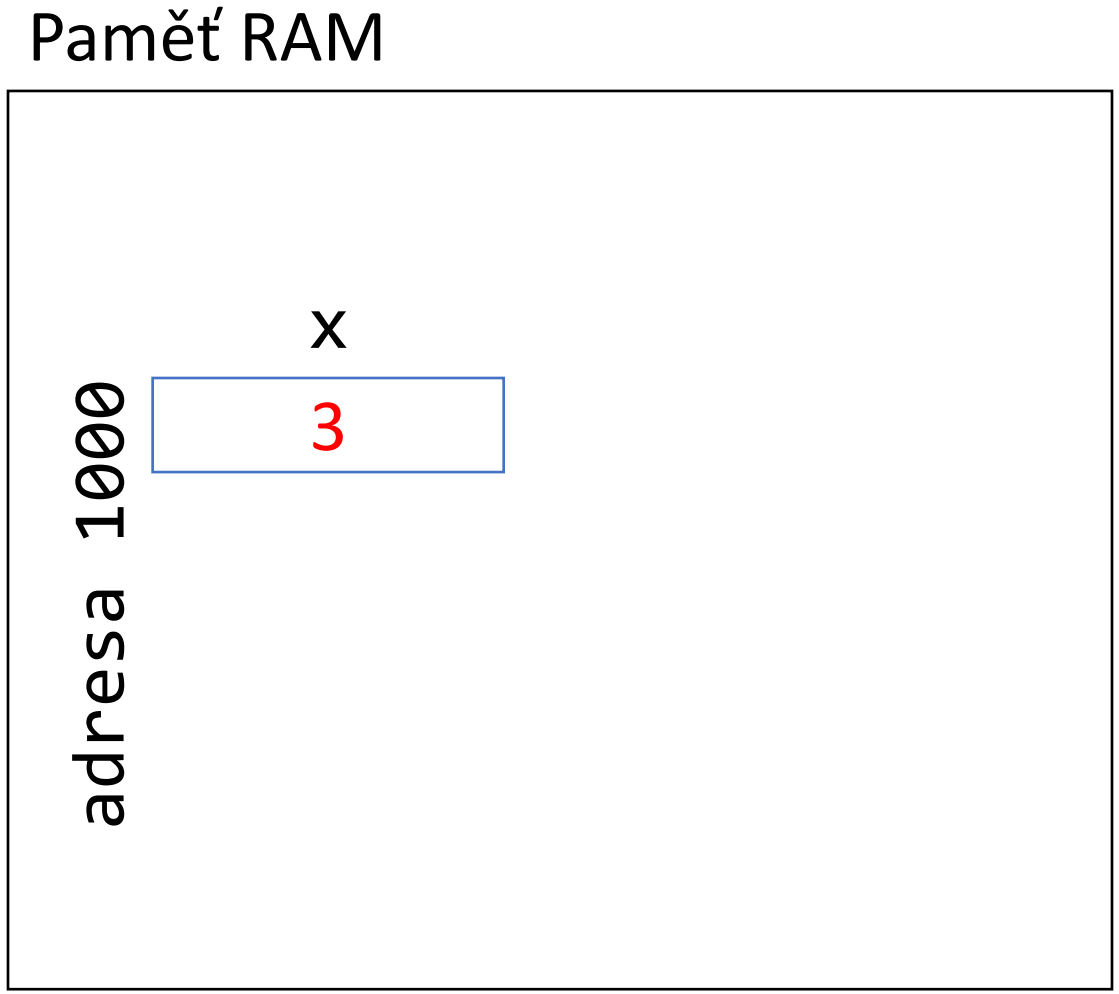
# Proměnná a paměť

```
int x = 0;
x = 3;
int y = 0;
y = x;
```



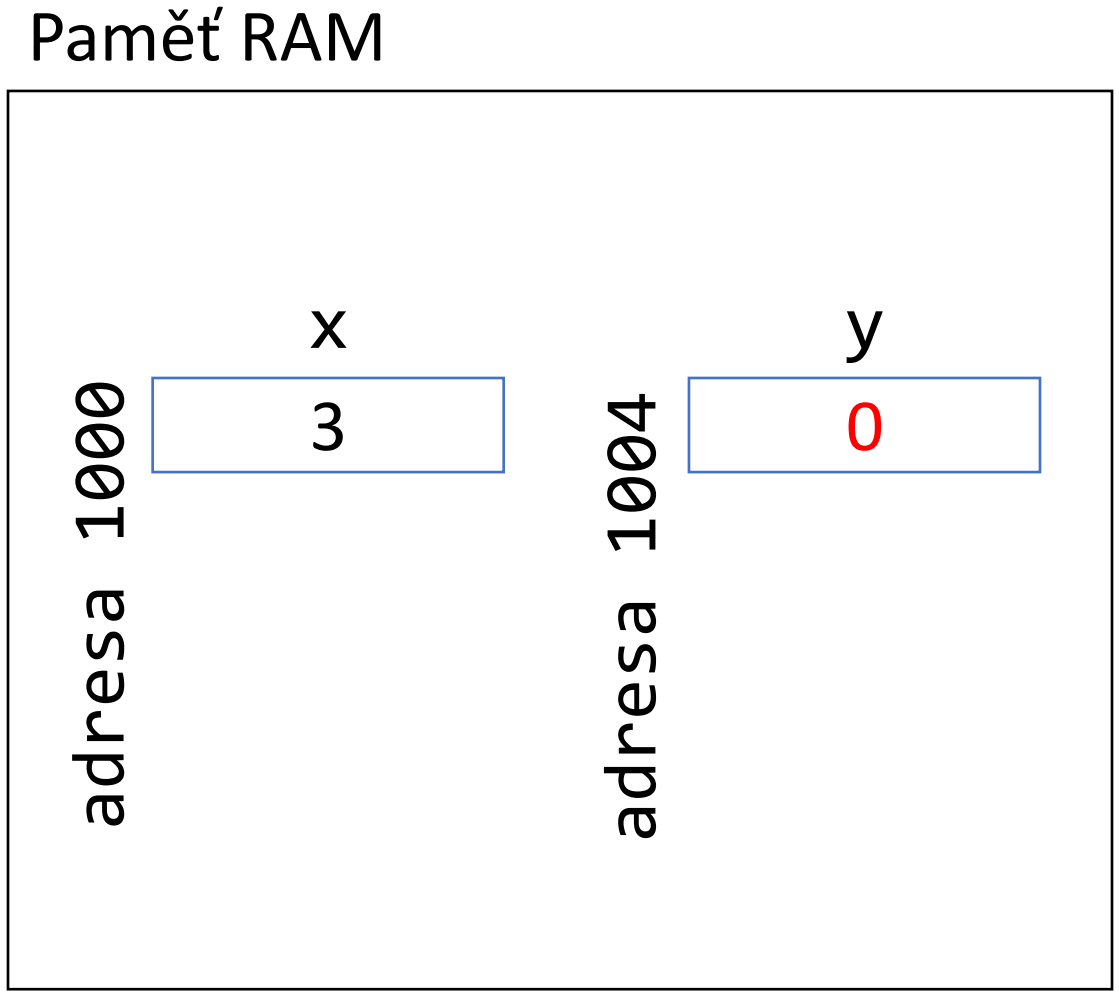
# Proměnná a paměť

```
int x = 0;
x = 3;
int y = 0;
y = x;
```



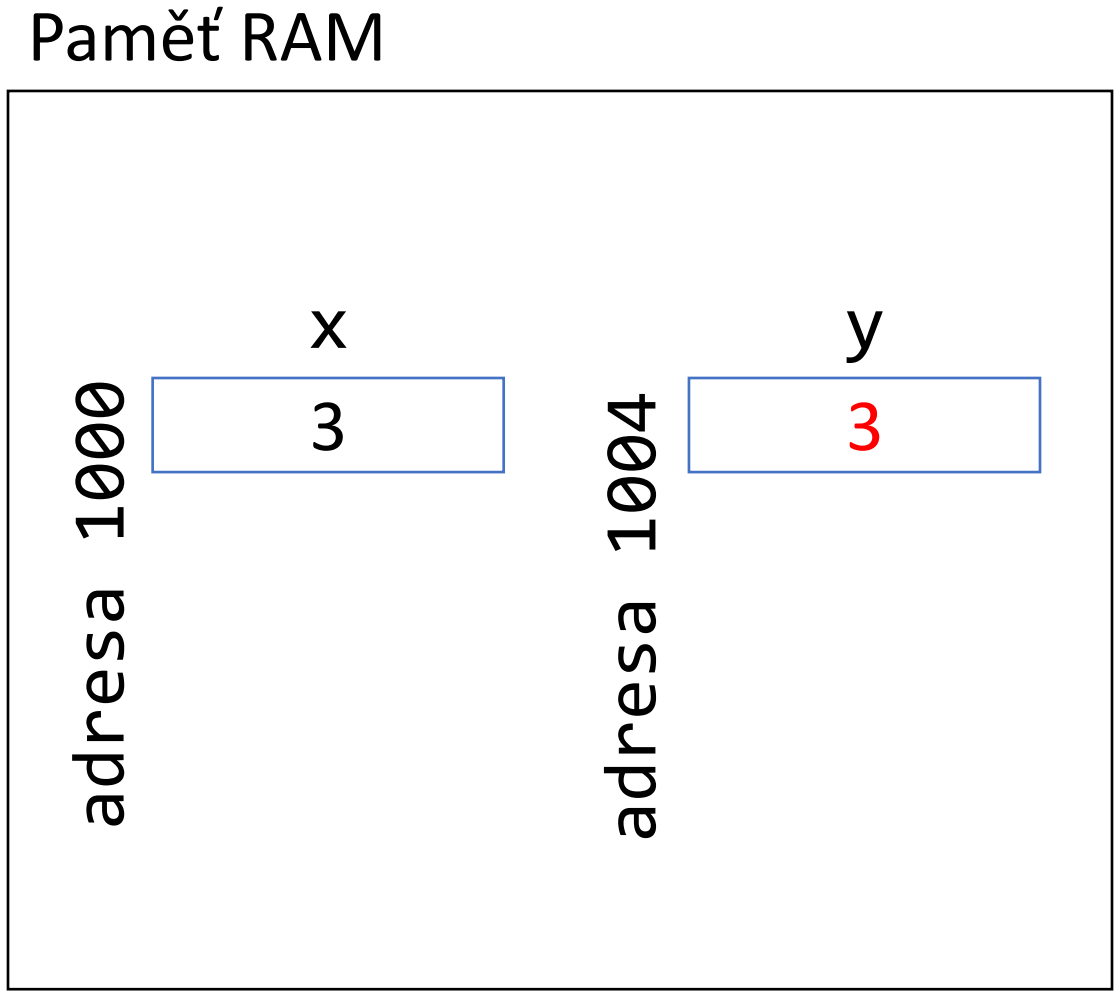
# Proměnná a paměť

```
int x = 0;
x = 3;
int y = 0;
y = x;
```



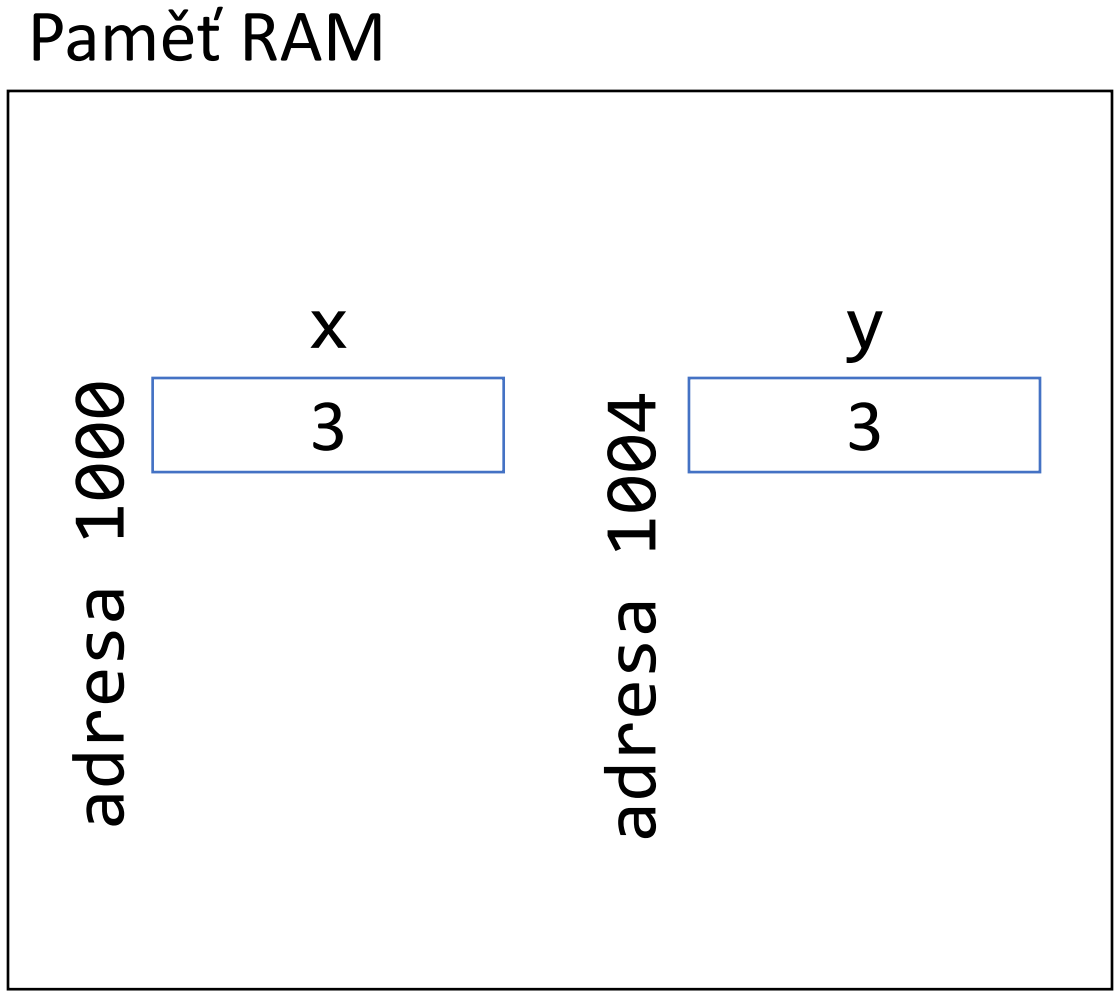
# Proměnná a paměť

```
int x = 0;
x = 3;
int y = 0;
y = x;
```



# Proměnná a paměť

```
int x = 0;
x = 3;
int y = 0;
y = x;
```



# Hodnotové a referenční typy

V jazyce C# rozlišujeme typy do dvou základních kategorií:

- hodnotové typy (například *int*) a
- referenční typy (například třída)

U hodnotových typů se kopíruje **hodnota**.

U referenční typů se kopíruje **reference**.

# Hodnotové typy

- Struktura
- Výčtový typ (Enum)
- Numerické typy (int, double atd.) a bool

# Referenční typy

- Třída (class) – i pole v .NET jsou třídy
- Rozhraní (interface)
- Delegate
- String – speciální případ, referenční typ, který se chová jako hodnotový

# Referenční typy

- Referenční typy jsou především třídy.
- Pokud definujeme instanci třídy, tak se na zásobníku alokuje místo pro referenci a na haldě se pak alokuje místo pro vlastní instanci.
- Při kopírování hodnoty se předává reference a můžeme mít více referencí na stejný objekt v paměti.
- Předávání referencí je rychlejší než předávání kopií celých objektů.

# Příklady na referenční a hodnotový typ

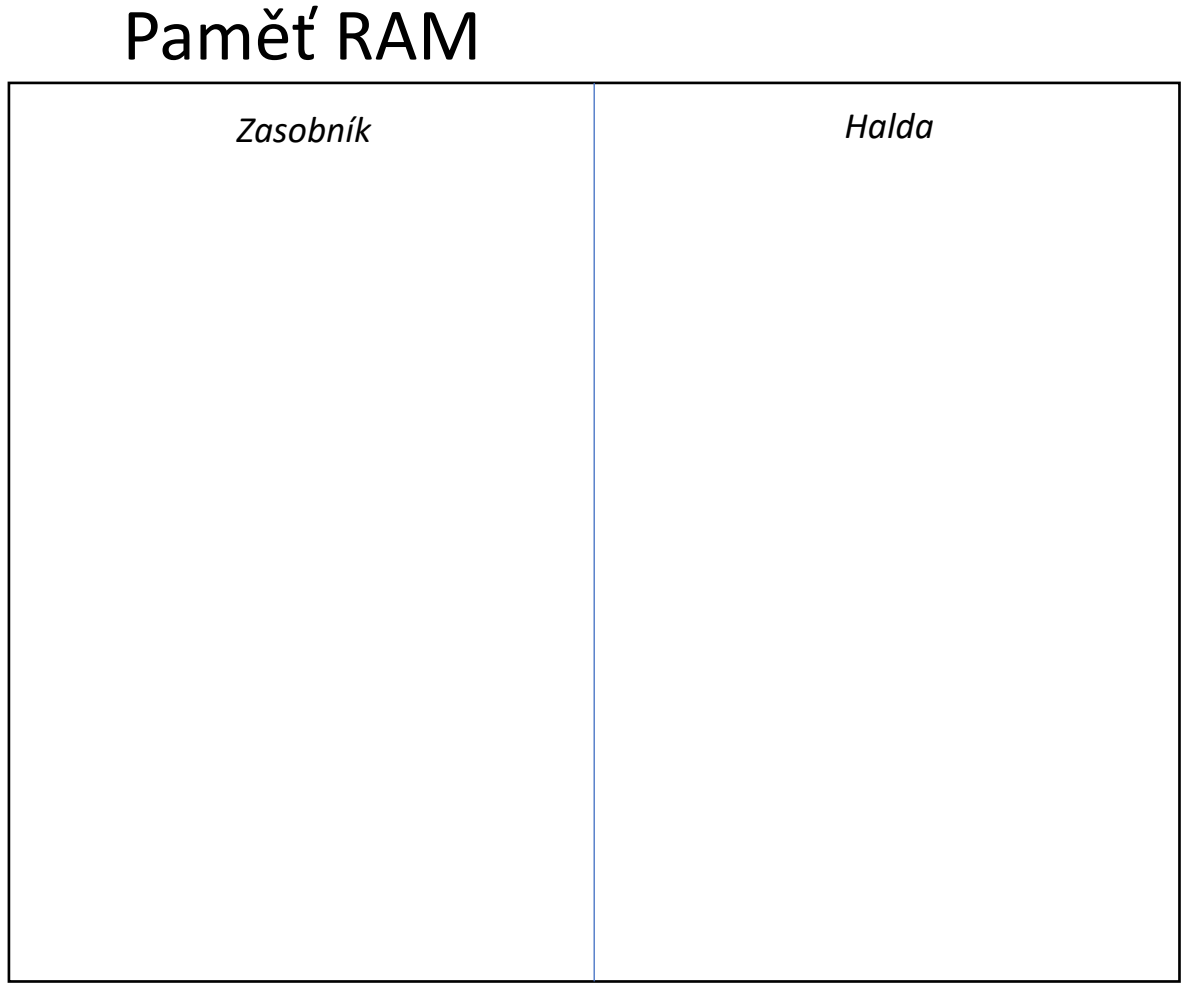
- Nejprve si ukážeme příklad na hodnotový typ, konkrétně strukturu.
- A poté si projdeme podobný příklad, ale na referenční typ, konkrétně třídu.
- V příkladu na hodnotový typ slouží konstruktor pouze pro změnu hodnoty už existující proměnné.

# Příklad  na hodnotový typ - definice struktury

```
struct Cislo
{
    public int x;
    public int y;

    public Cislo(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```

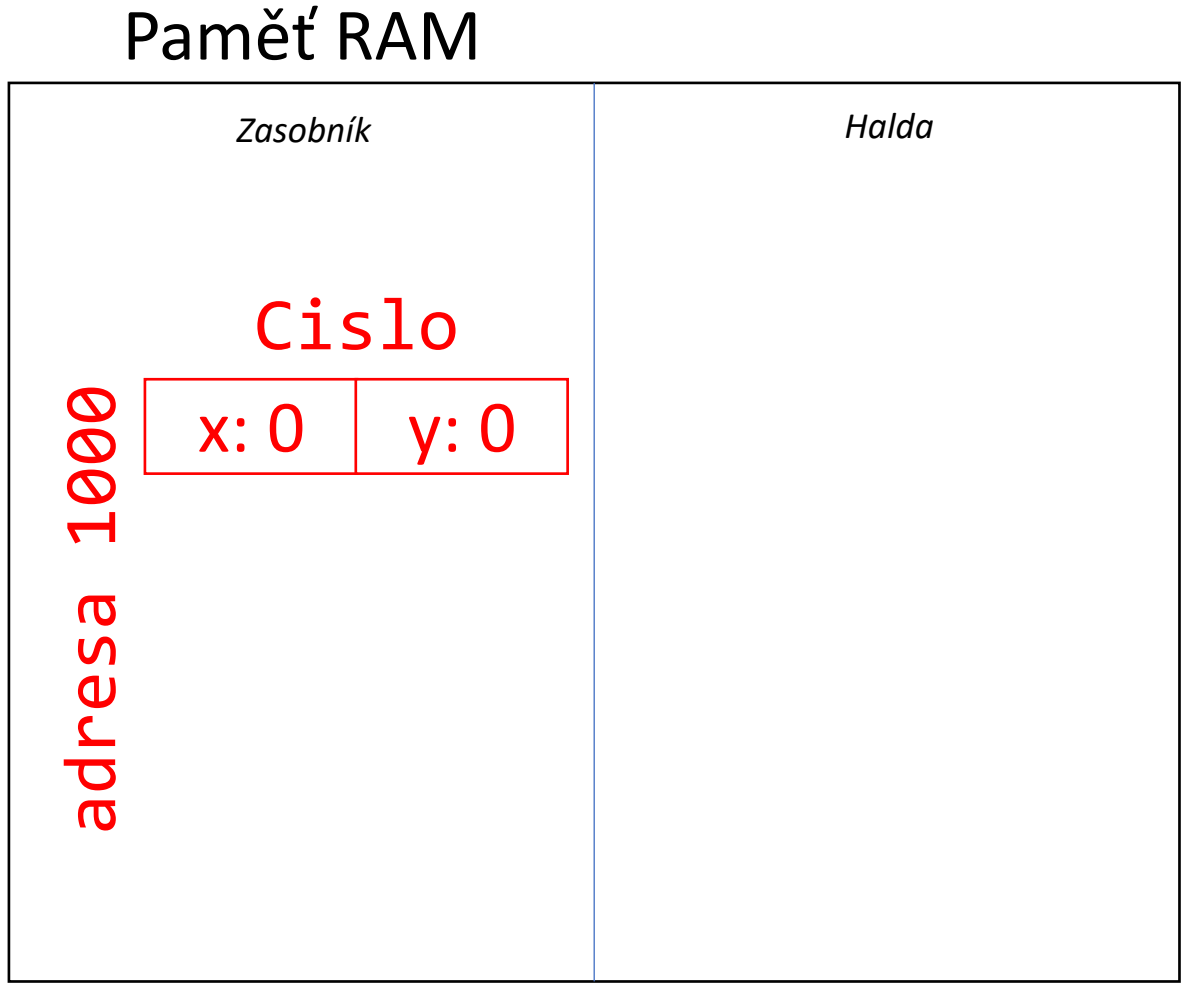
# Příklad  na hodnotový typ a paměť

```
static void Main(string[] args)
{
    Cislo c1;

    c1 = new Cislo(1, 2);
}
```



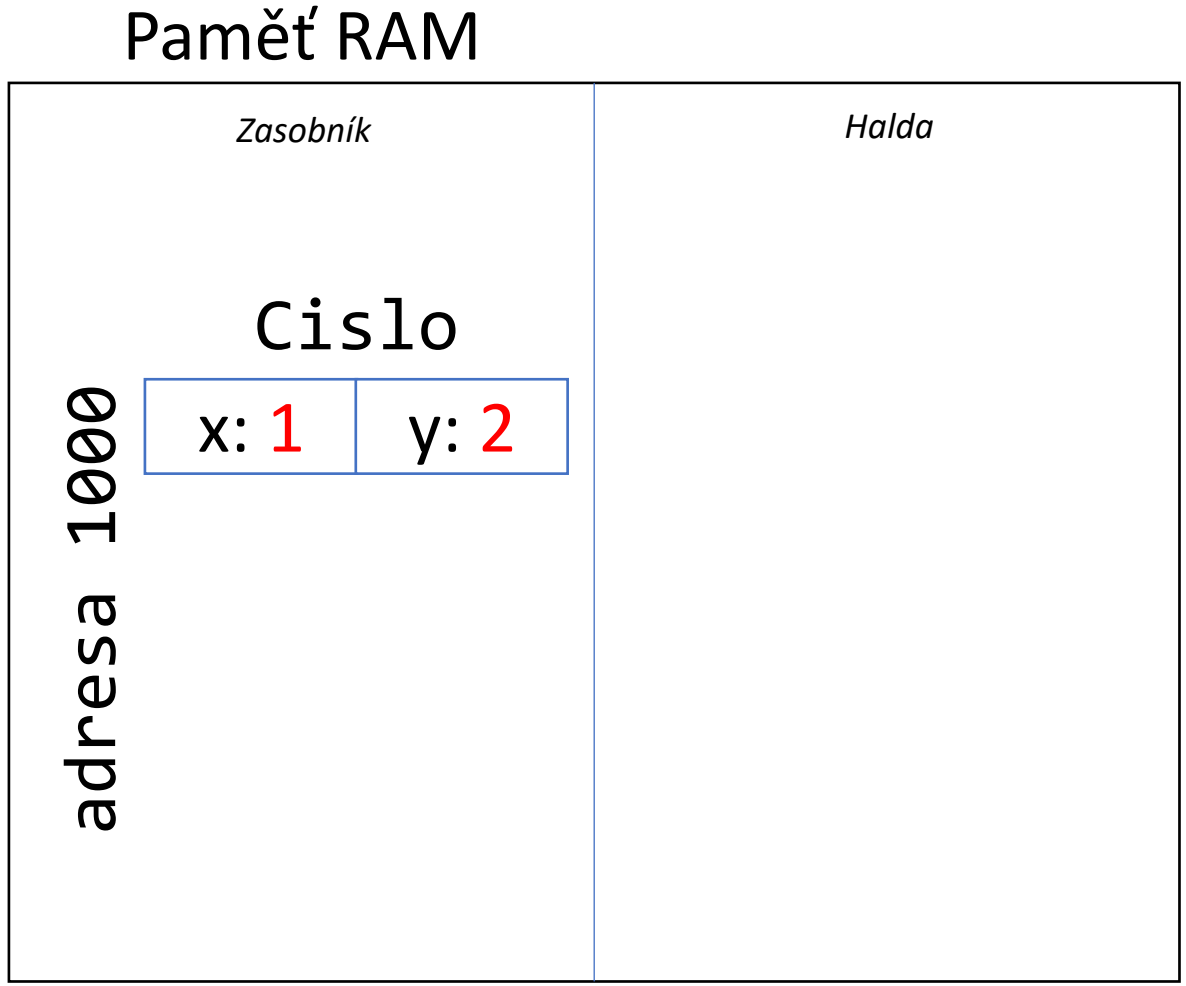
# Příklad  na hodnotový typ a paměť

```
static void Main(string[] args)
{
    Cislo c1;

    c1 = new Cislo(1, 2);
}
```



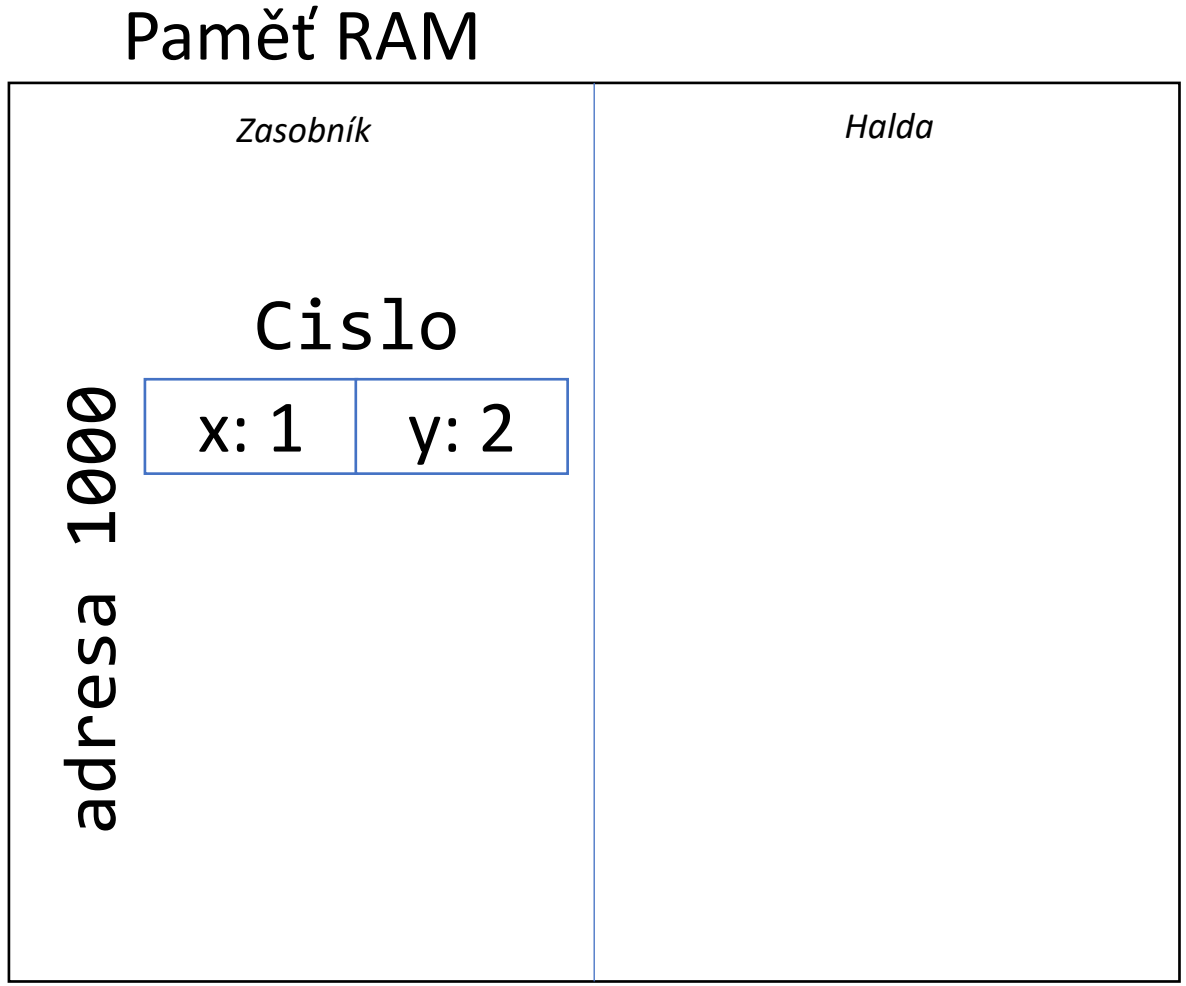
# Příklad  na hodnotový typ a paměť

```
static void Main(string[] args)
{
    Cislo c1;

    c1 = new Cislo(1, 2);
}
```



# Příklad  na hodnotový typ a paměť

```
static void Main(string[] args)
{
    Cislo c1;

    c1 = new Cislo(1, 2);
}
```

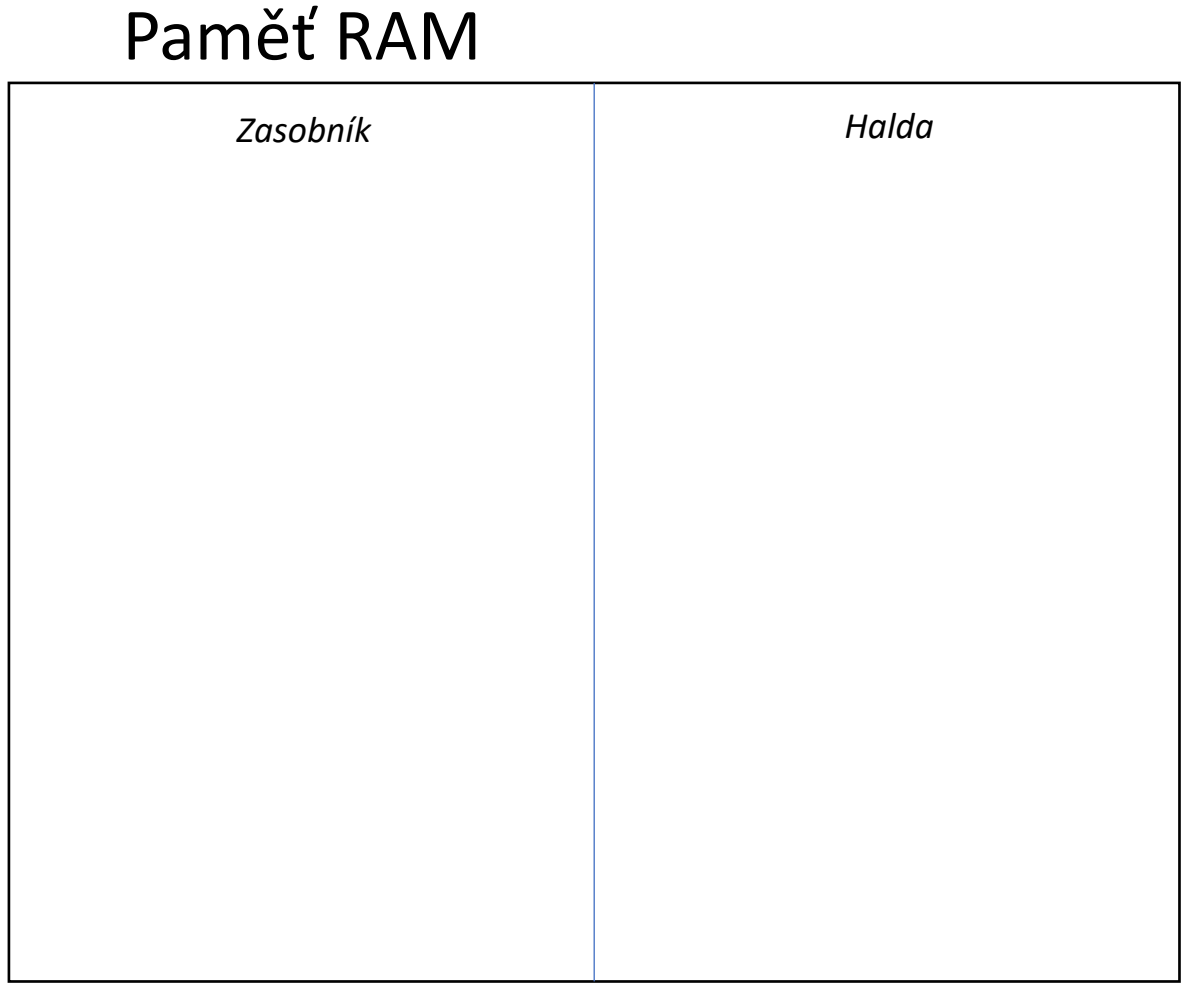


# Příklad  na referenční typ - definice třídy

```
class Cislo
{
    public int x;
    public int y;

    public Cislo(int x, int y)
    {
        this.x = x;
        this.y = y;
    }
}
```

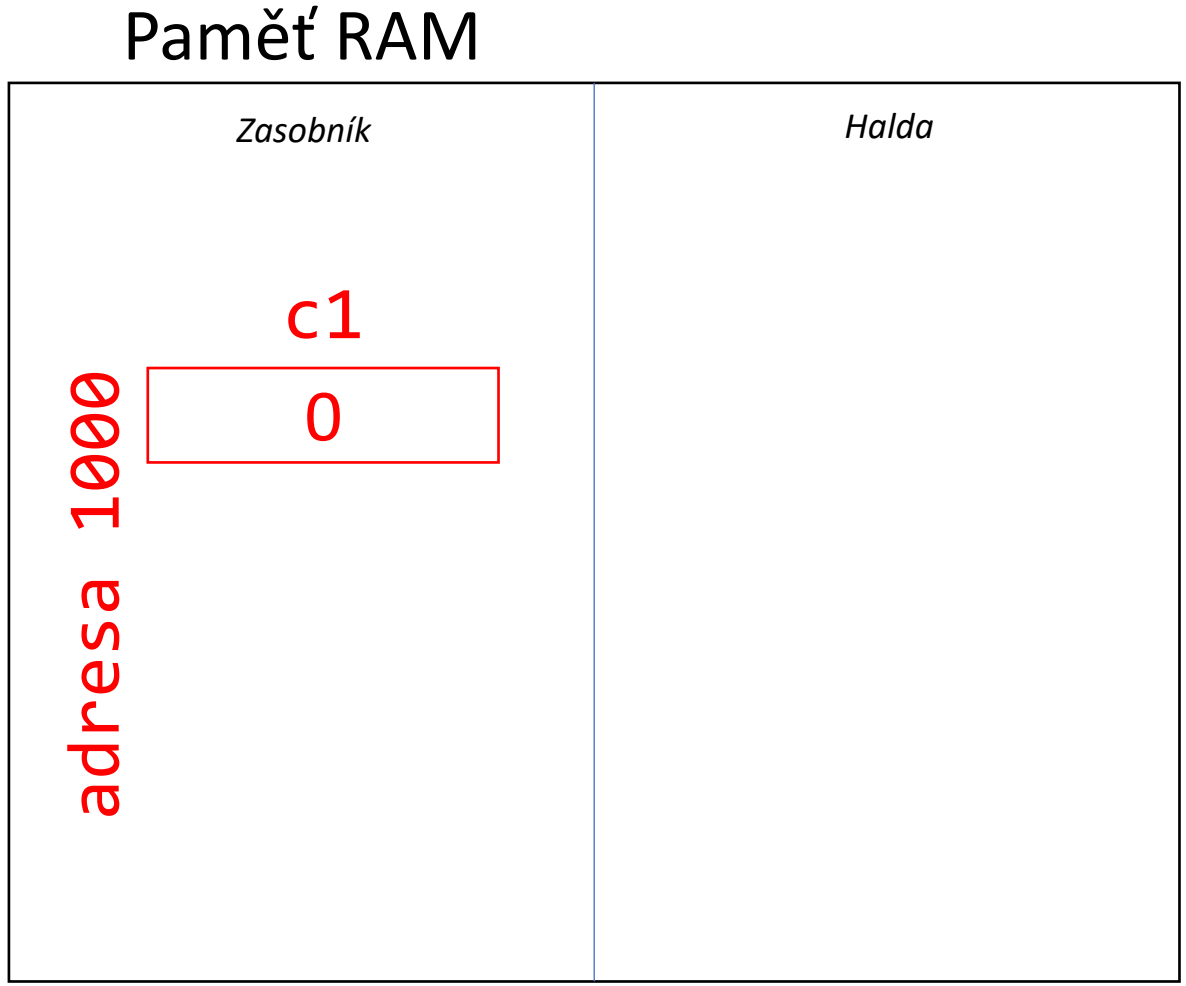
# Příklad  na referenční typ a paměť

```
static void Main(string[] args)
{
    Cislo c1 = null;

    c1 = new Cislo(1, 2);
}
```



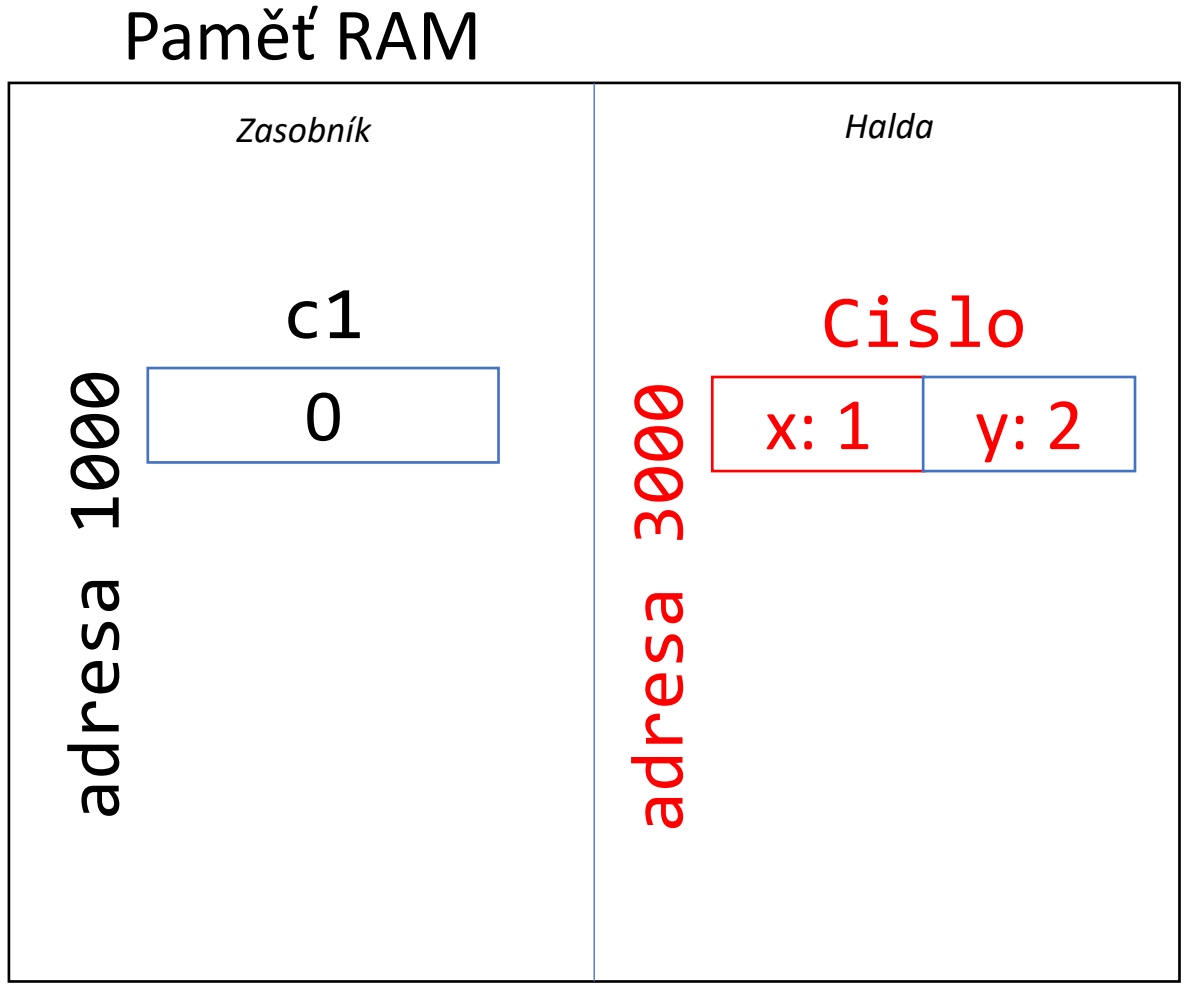
# Příklad  na referenční typ a paměť

```
static void Main(string[] args)
{
    Cislo c1 = null;

    c1 = new Cislo(1, 2);
}
```



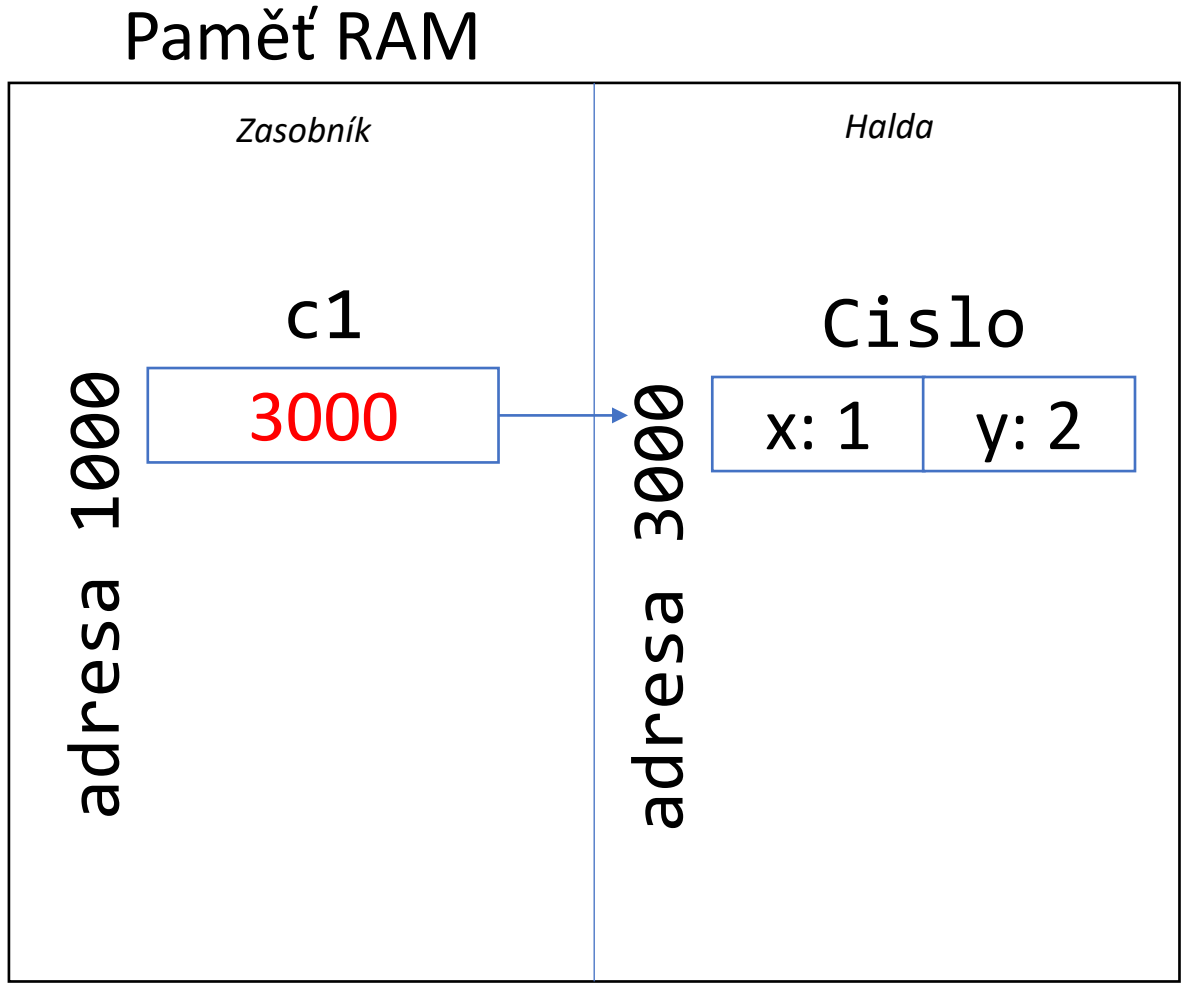
# Příklad  na referenční typ a paměť

```
static void Main(string[] args)
{
    Cislo c1 = null;

    c1 = new Cislo(1, 2);
}
```



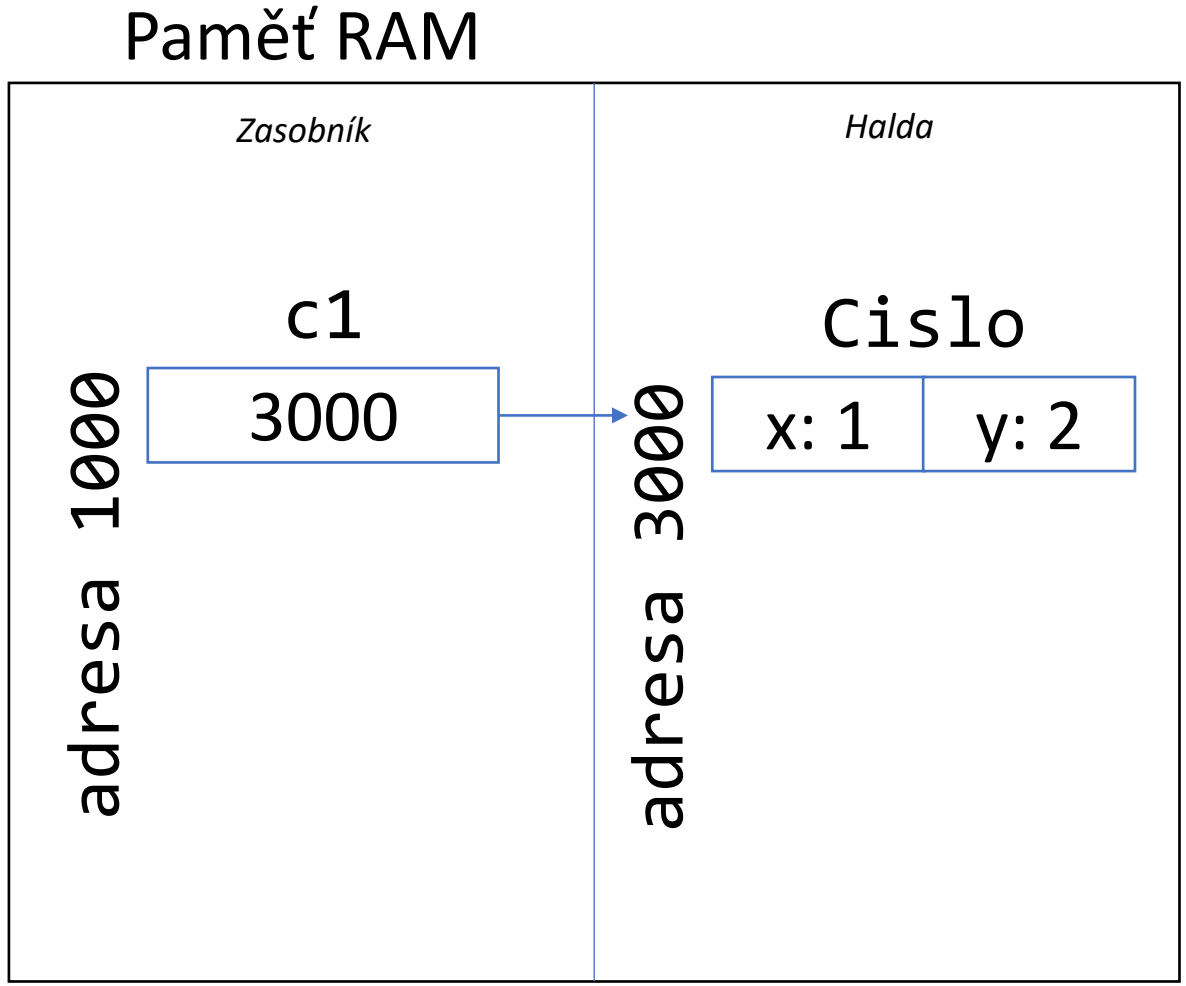
# Příklad  na referenční typ a paměť

```
static void Main(string[] args)
{
    Cislo c1 = null;

    c1 = new Cislo(1, 2);
}
```



# Příklad  na referenční typ a paměť

```
static void Main(string[] args)
{
    Cislo c1 = null;

    c1 = new Cislo(1, 2);
}
```



# Předávání argumentů metodám

- Argumenty předávají parametrům metod své hodnoty.
- Parametry jsou potom nezávisle proměnné.
- Jakákoli změna parametru se nijak neprojeví na původním volaném argumentu.

# Příklad
# Předávání argumentů bez klíčového slova ref

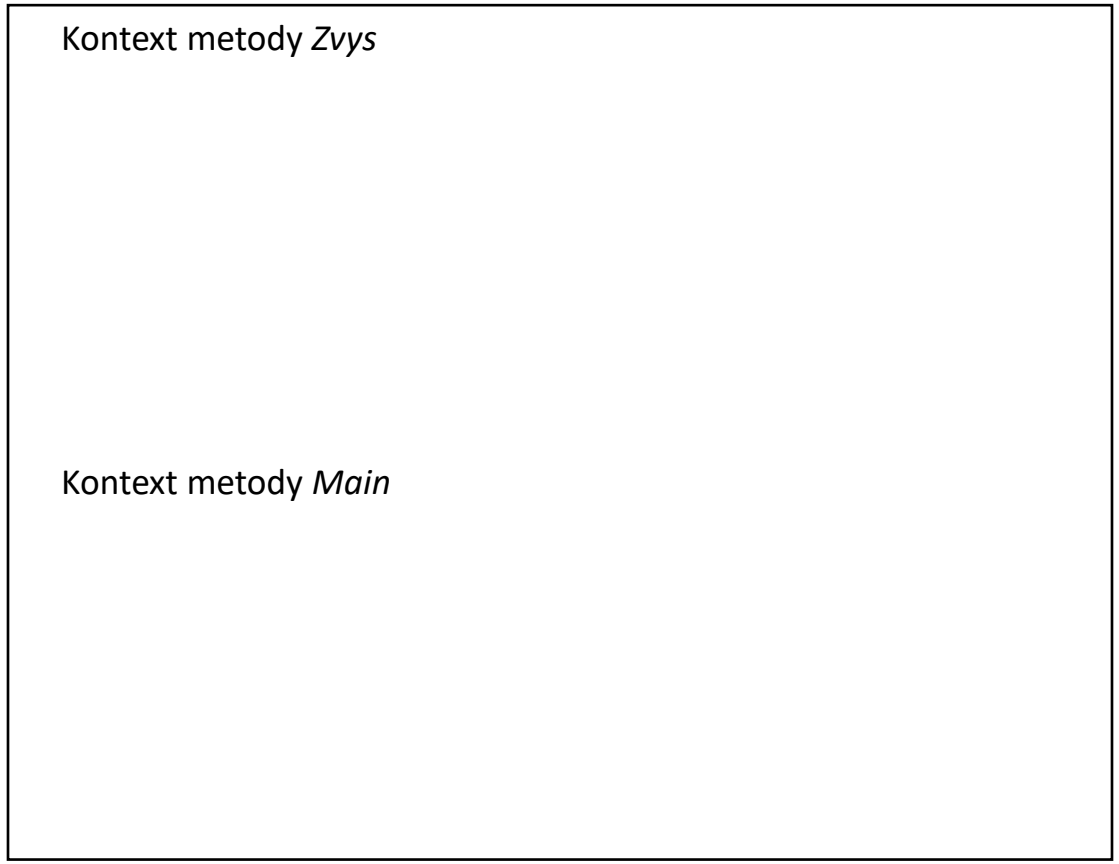
V následujícím příkladu si ukážeme předání argumentu hodnotového typu metodě bez klíčového slova *ref*.

# Příklad
# Předávání argumentů bez klíčového slova *ref*

```
static void Zvys(int x)
{
    ++x;
}
```

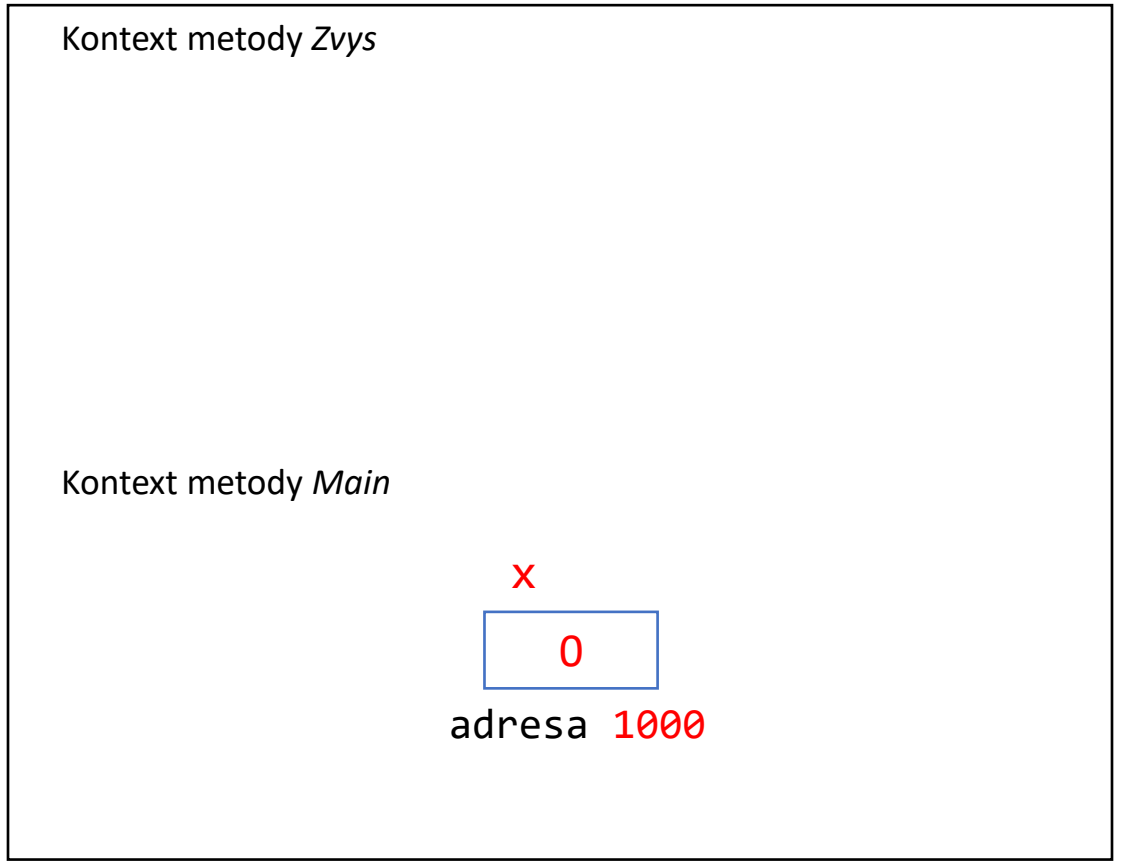
```
static void Main(string[] args)
{
    int x = 0;
    Zvys(x);
    Console.WriteLine(x);
}
```

Paměť RAM



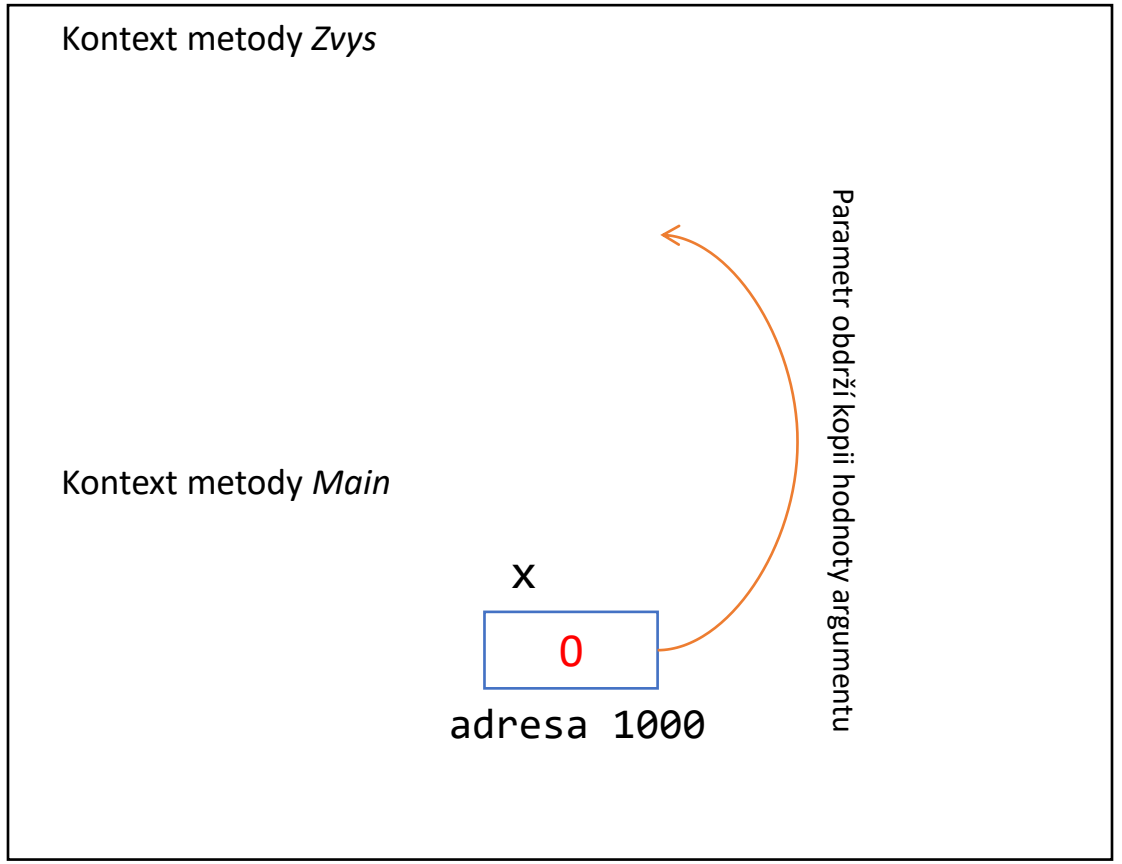
# Příklad
# Předávání argumentů bez klíčového slova *ref*

```
static void Main(string[] args)
{
    int x = 0;


}
```



# Příklad
# Předávání argumentů bez klíčového slova *ref*

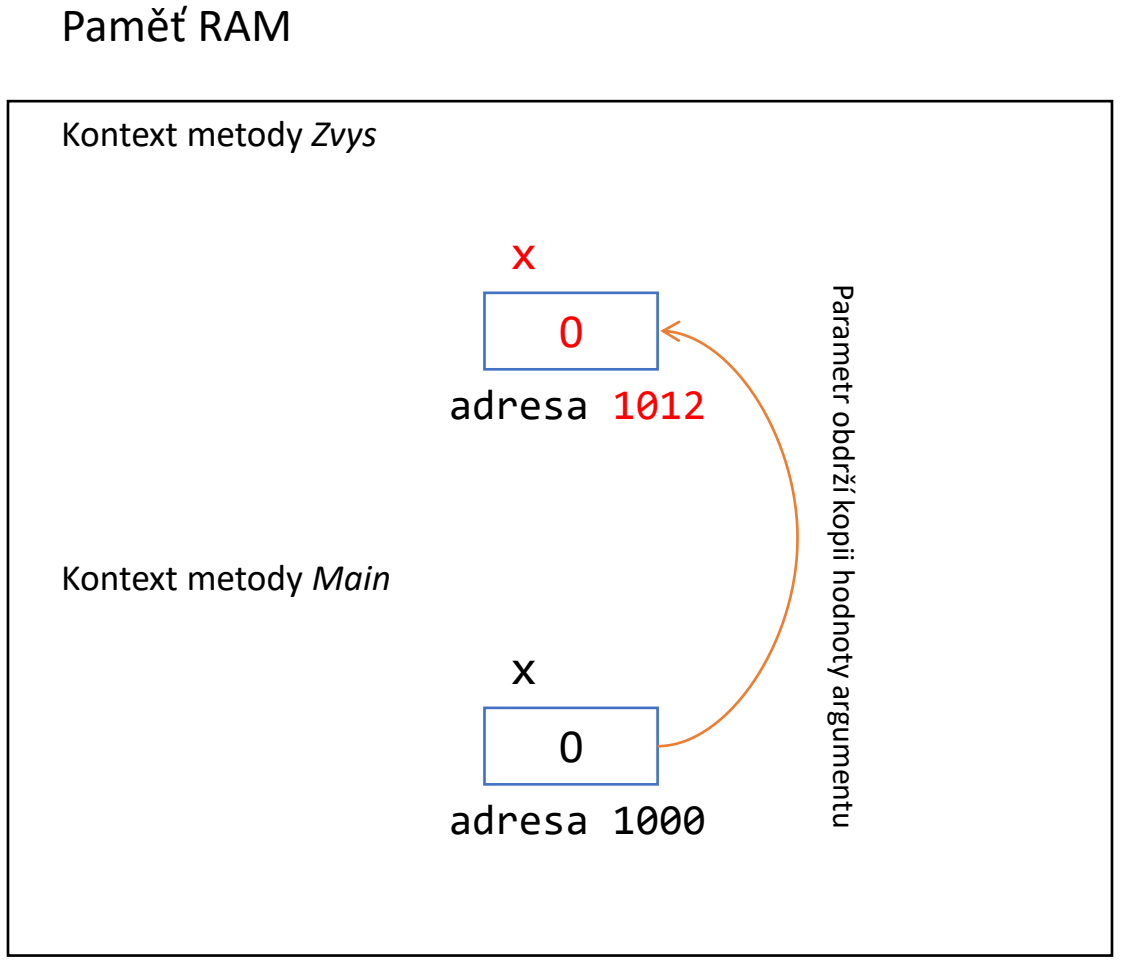
```
static void Main(string[] args)
{
    int x = 0;
    Zvys(x);

}
```



# Příklad
# Předávání argumentů bez klíčového slova *ref*

```
static void Zvys(int x)
{
    ++x;
}
```

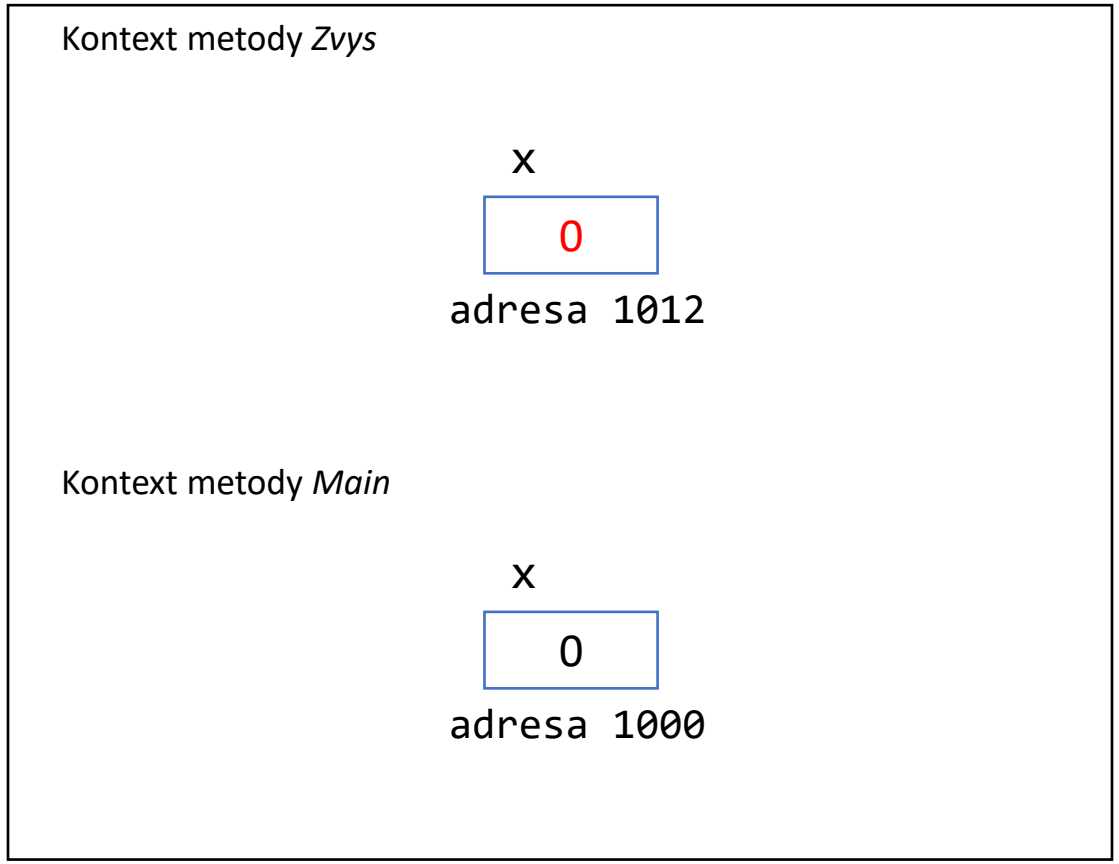
```
static void Main(string[] args)
{
    int x = 0;
    Zvys(x);

}
```



# Příklad
# Předávání argumentů bez klíčového slova *ref*

```
static void Zvys(int x)
{
    ++x;
}
```

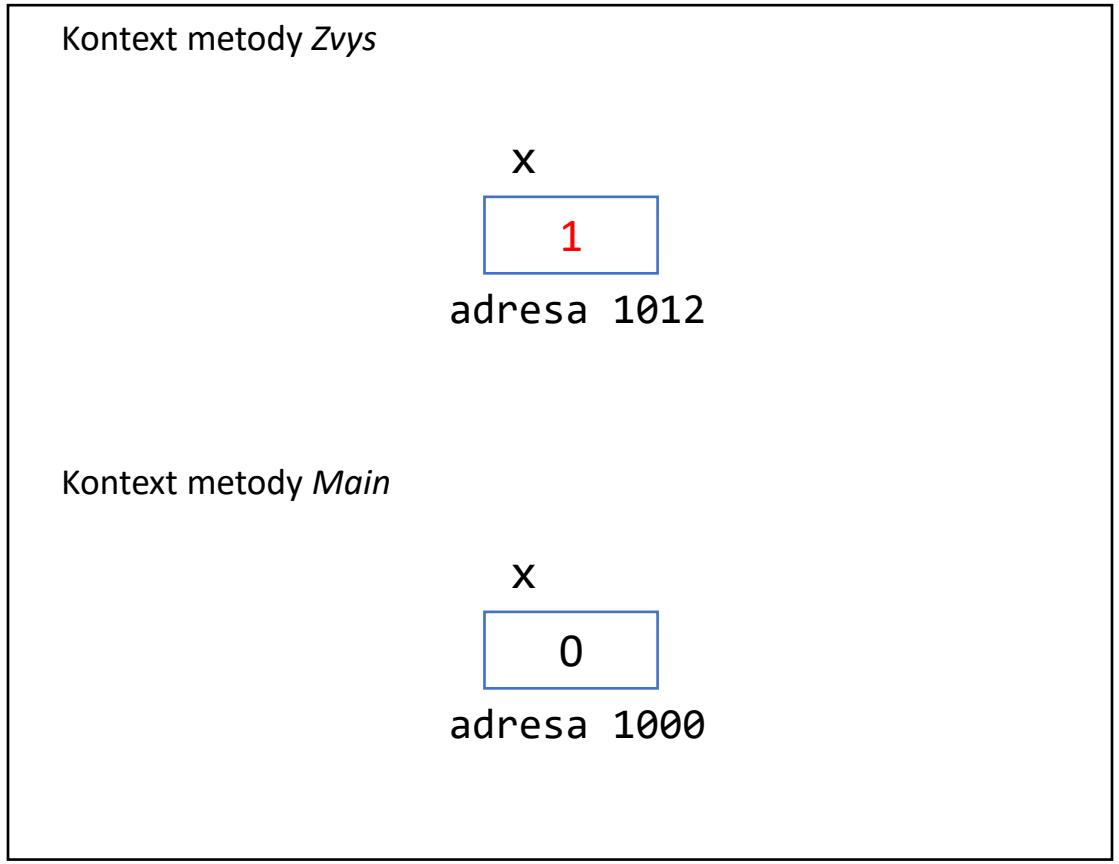
```
static void Main(string[] args)
{
    int x = 0;
    Zvys(x);

}
```



# Příklad
# Předávání argumentů bez klíčového slova ref

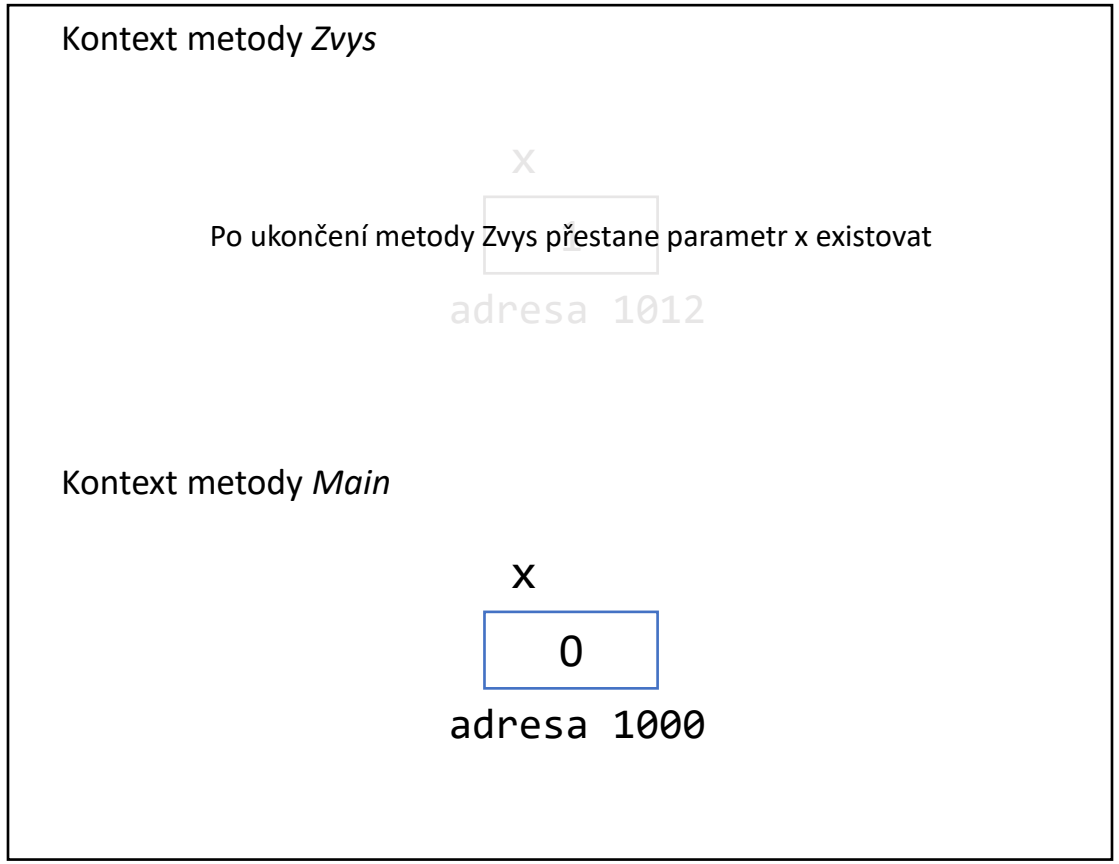
```
static void Zvys(int x)
{
    ++x;
}
```

```
static void Main(string[] args)
{
    int x = 0;
    Zvys(x);

}
```



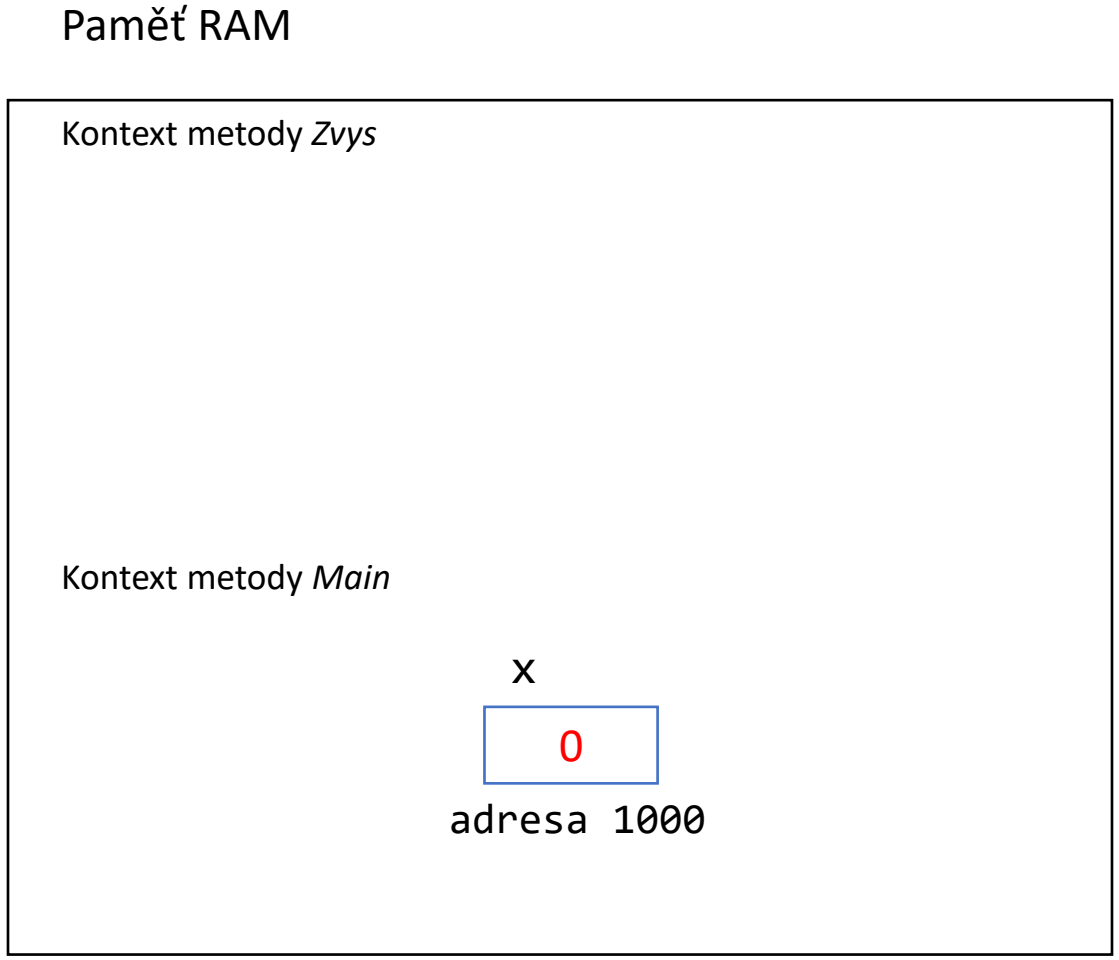
# Příklad
# Předávání argumentů bez klíčového slova *ref*

```
static void Main(string[] args)
{
    int x = 0;
    Zvys(x);

}
```



# Příklad
# Předávání argumentů bez klíčového slova *ref*

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



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

# Klíčové slovo *ref*

## Hodnotové typy

Ing. et Ing. Erik Král, Ph.D.
ÚPKS
Fakulta aplikované informatiky

# Modifikátor ref u hodnotových typů

- Modifikátor *`ref`* způsobí, že argument je předaný jako reference a ne jako hodnota.
- Jakákoli změna parametru se projeví na původním volaném argumentu
- Používá se především u hodnotových typů

# Příklad
# Předávání argumentů s klíčovým slovem ref

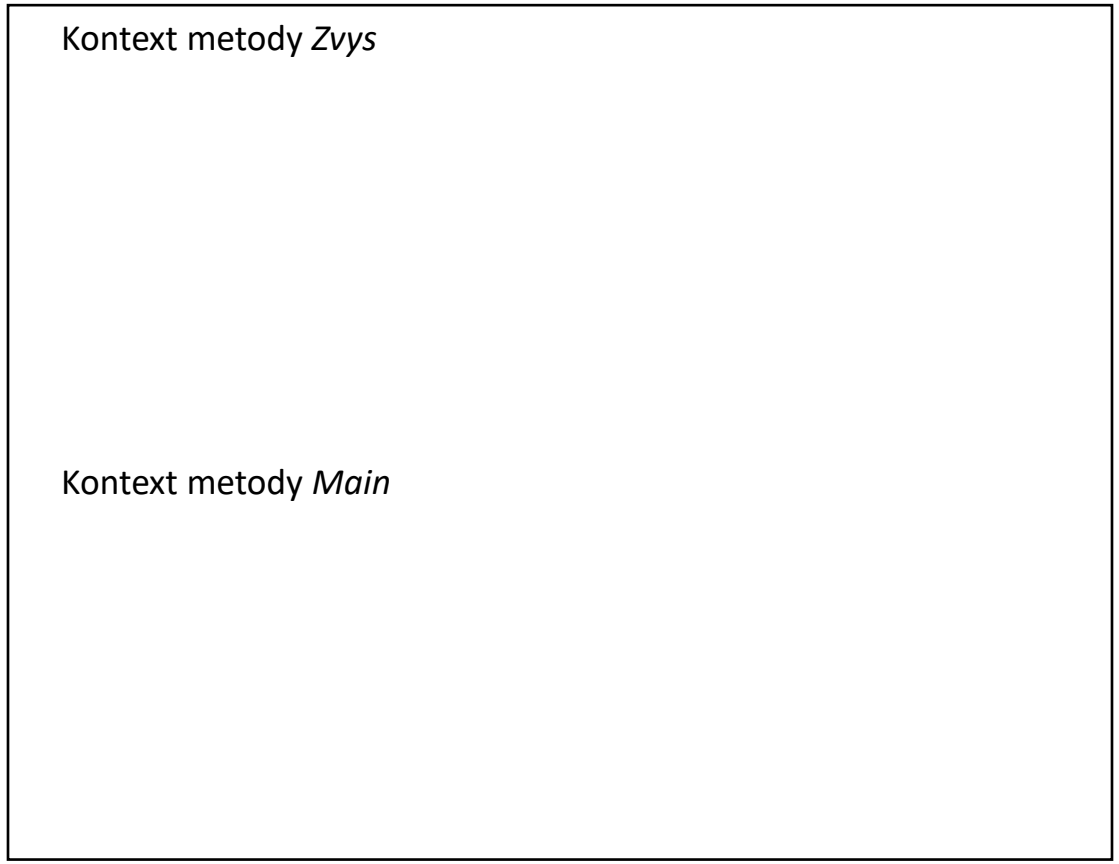
V následujícím příkladu si ukážeme předání argumentu hodnotového typu metodě s využitím klíčového slova *ref*.

# Příklad
# Předávání argumentů s klíčovým slovem *ref*

```
static void Zvys(ref int x)
{
    ++x;
}
```

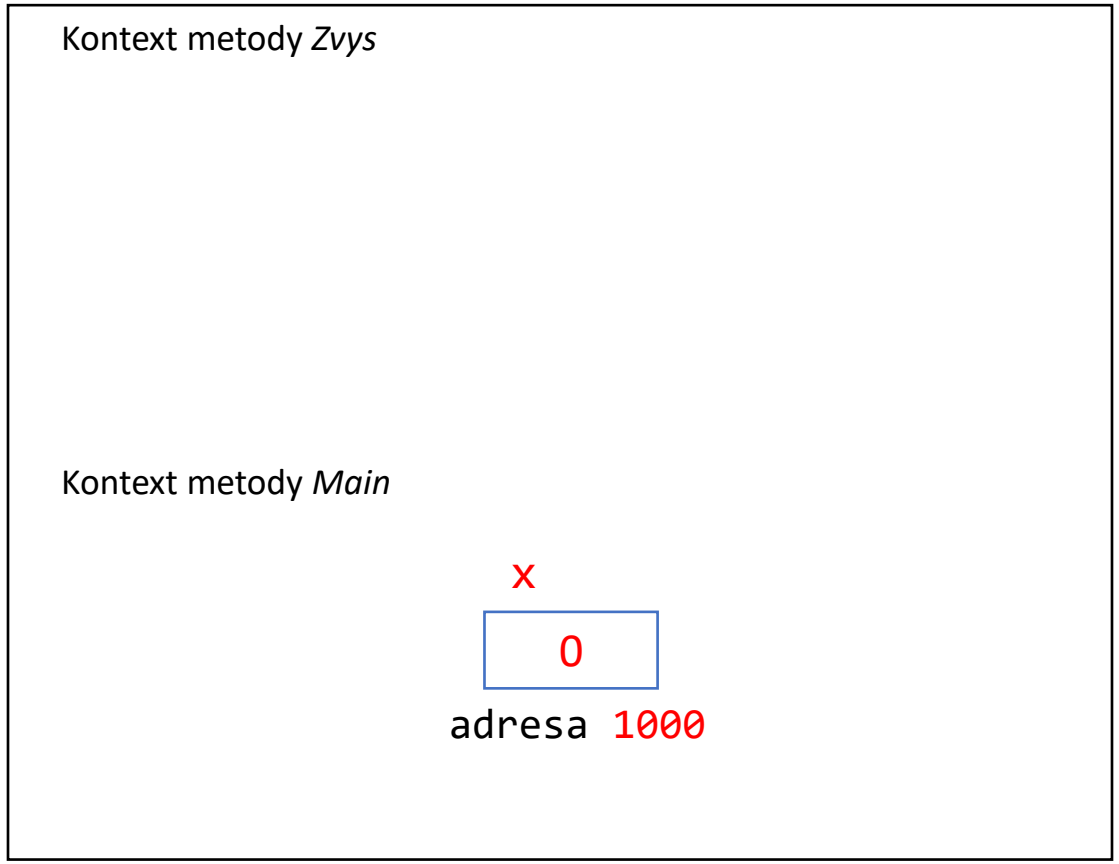
```
static void Main(string[] args)
{
    int x = 0;
    Zvys(ref x);
    Console.WriteLine(x);
}
```

Paměť RAM



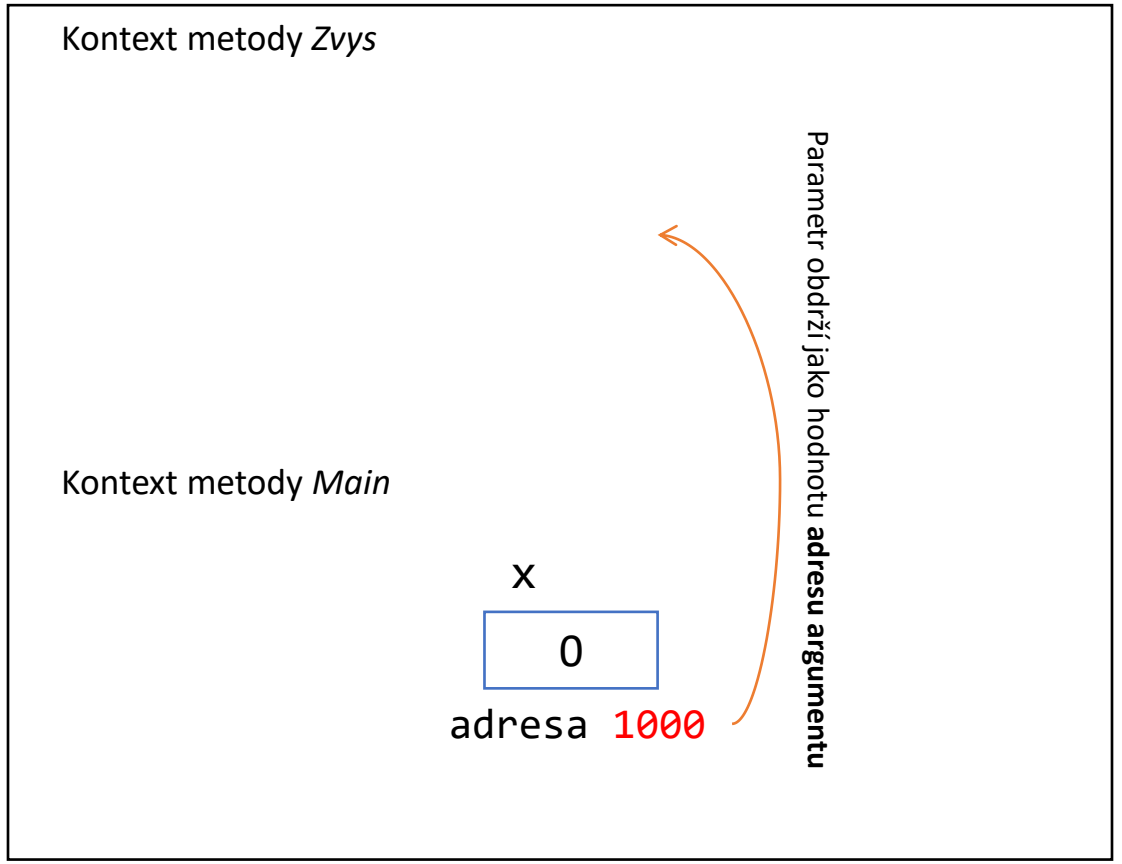
# Příklad
# Předávání argumentů s klíčovým slovem *ref*

```
static void Main(string[] args)
{
    int x = 0;


}
```



# Příklad
# Předávání argumentů s klíčovým slovem *ref*

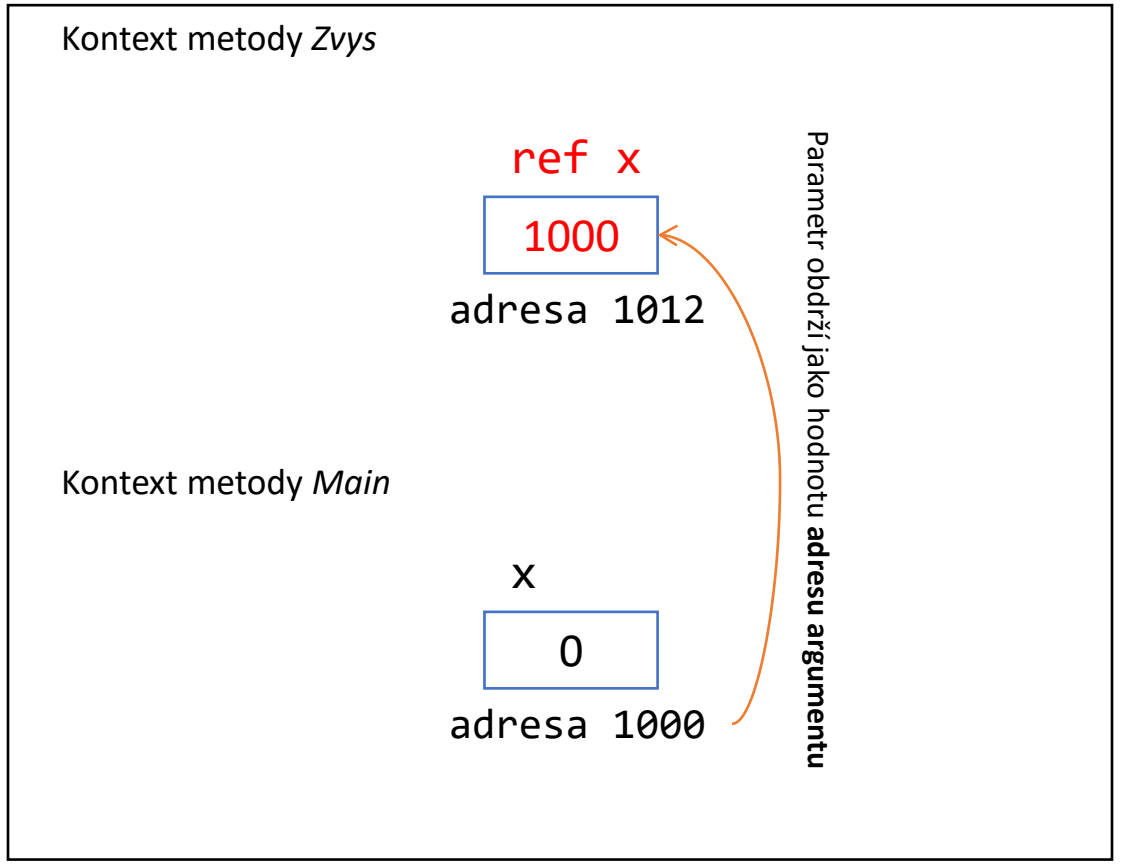
```
static void Main(string[] args)
{
    int x = 0;
    Zvys(ref x);

}
```



# Příklad
# Předávání argumentů s klíčovým slovem *ref*

```
static void Zvys(ref int x)
{
    ++x;
}
```

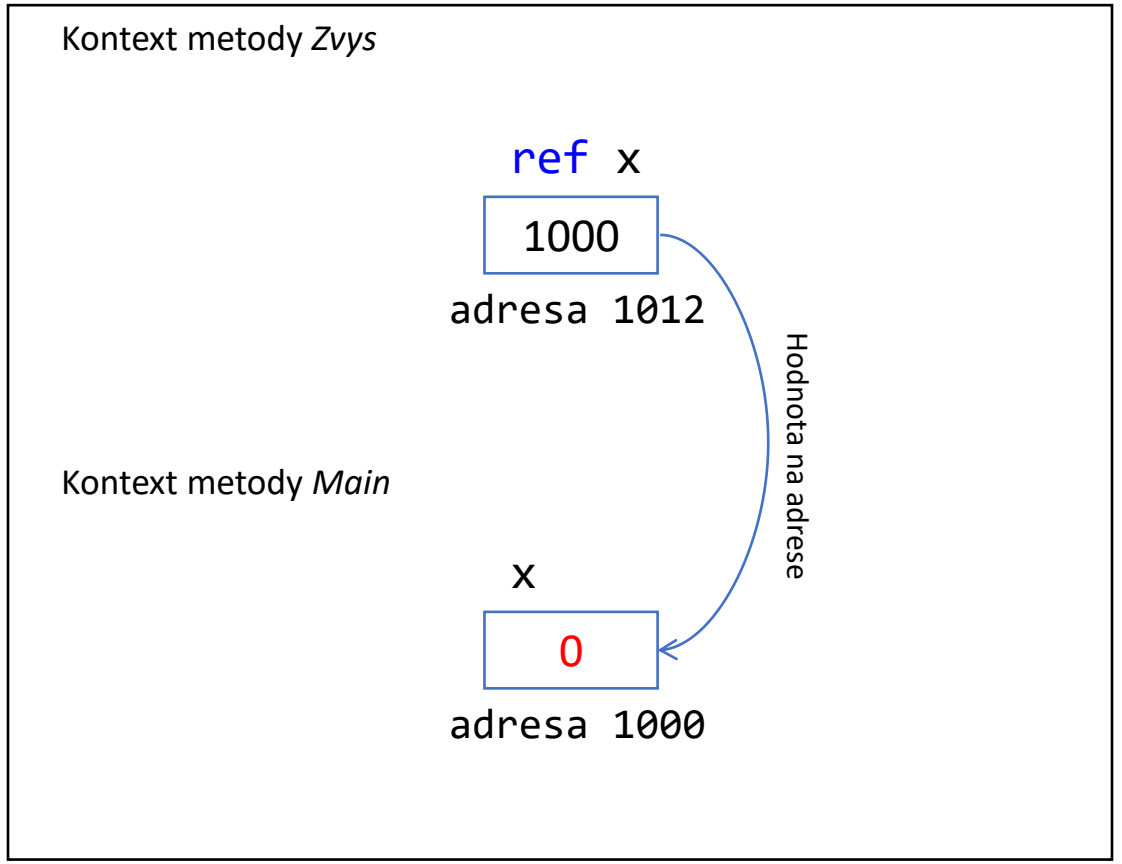
```
static void Main(string[] args)
{
    int x = 0;
    Zvys(ref x);
    Console.WriteLine(x);
}
```



# Příklad
# Předávání argumentů s klíčovým slovem *ref*

```
static void Zvys(ref int x)
{
    ++x;
}
```

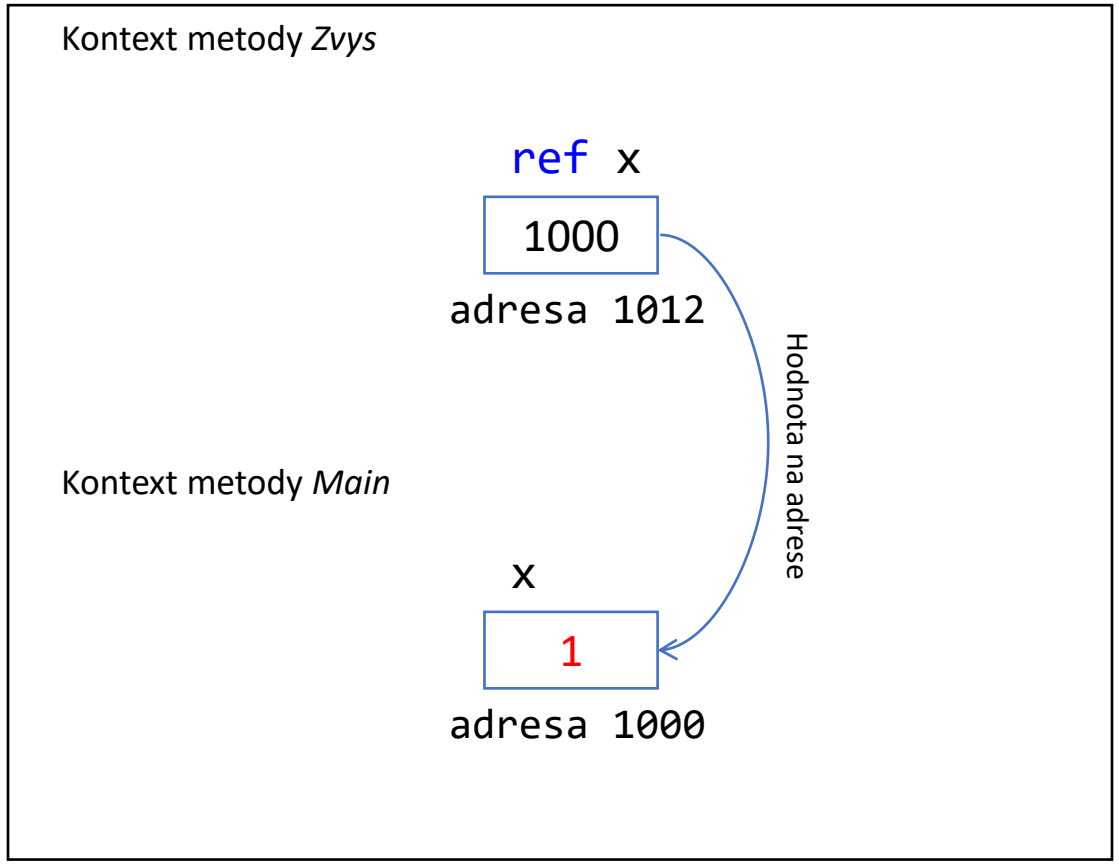
```
static void Main(string[] args)
{
    int x = 0;
    Zvys(ref x);
    Console.WriteLine(x);
}
```



# Příklad
# Předávání argumentů s klíčovým slovem *ref*

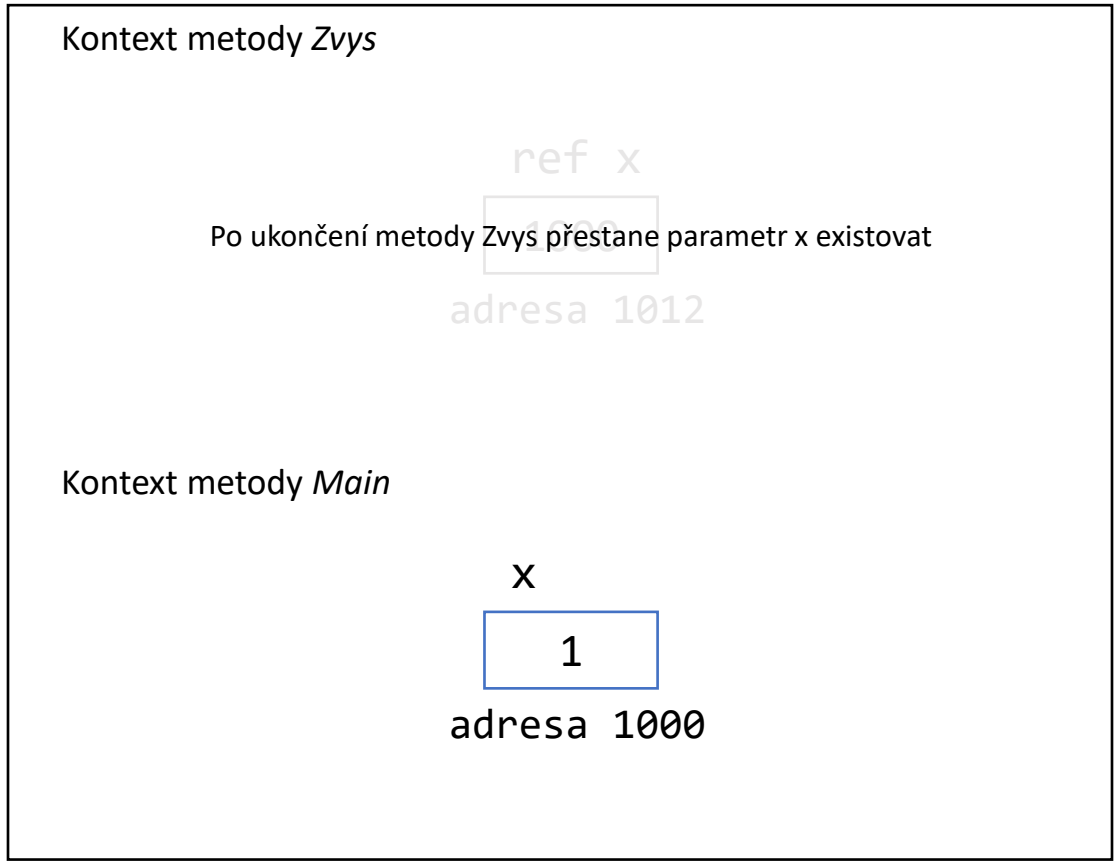
```
static void Zvys(ref int x)
{
    ++x;
}
```

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



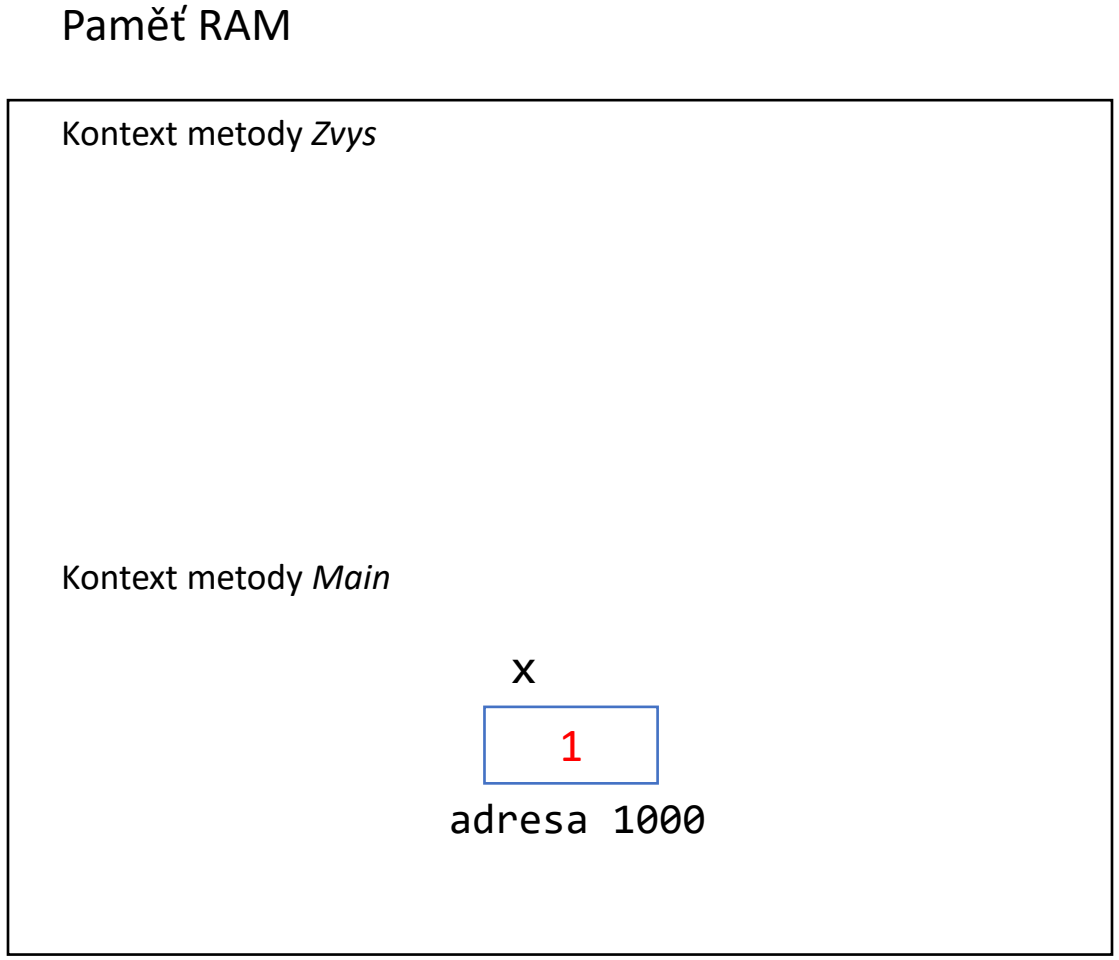
# Příklad
# Předávání argumentů s klíčovým slovem *ref*

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



# Příklad
# Předávání argumentů s klíčovým slovem *ref*

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



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

# Klíčové slovo *ref*

## Referenční typy

Ing. et Ing. Erik Král, Ph.D.
ÚPKS
Fakulta aplikované informatiky

# Modifikátor ref u referenčních typů

- Modifikátor *`ref`* způsobí, že argument je předaný jako reference a ne jako hodnota [2].
- Není jej nutné používat u referenčních typů (například instance tříd), protože ty se předávají jako reference automaticky
- Klíčové slovo *`ref`* se používá u referenčních typu výjimečně a to pouze pokud chceme měnit referenci samotnou, například vytvořit úplně novou instanci třídy *List*. Jde prakticky o referenci na referenci (nebo v jazyku C ukazatel na ukazatel).

# Příklad 1 argumentu referenčního typu s klíčovým slovem *ref*

- V prvním příkladu na referenční typy a jejich použití s klíčovým slovem `ref` si probereme příklad bez použití tohoto klíčového slova.
- Na příkladu si ukážeme si, že ve většině případů není nutné používat klíčové slovo `ref` u referenčních typů.
- Konkrétně budeme jako argument předávat referenci na instanci třídy s názvem *Trida*.

# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

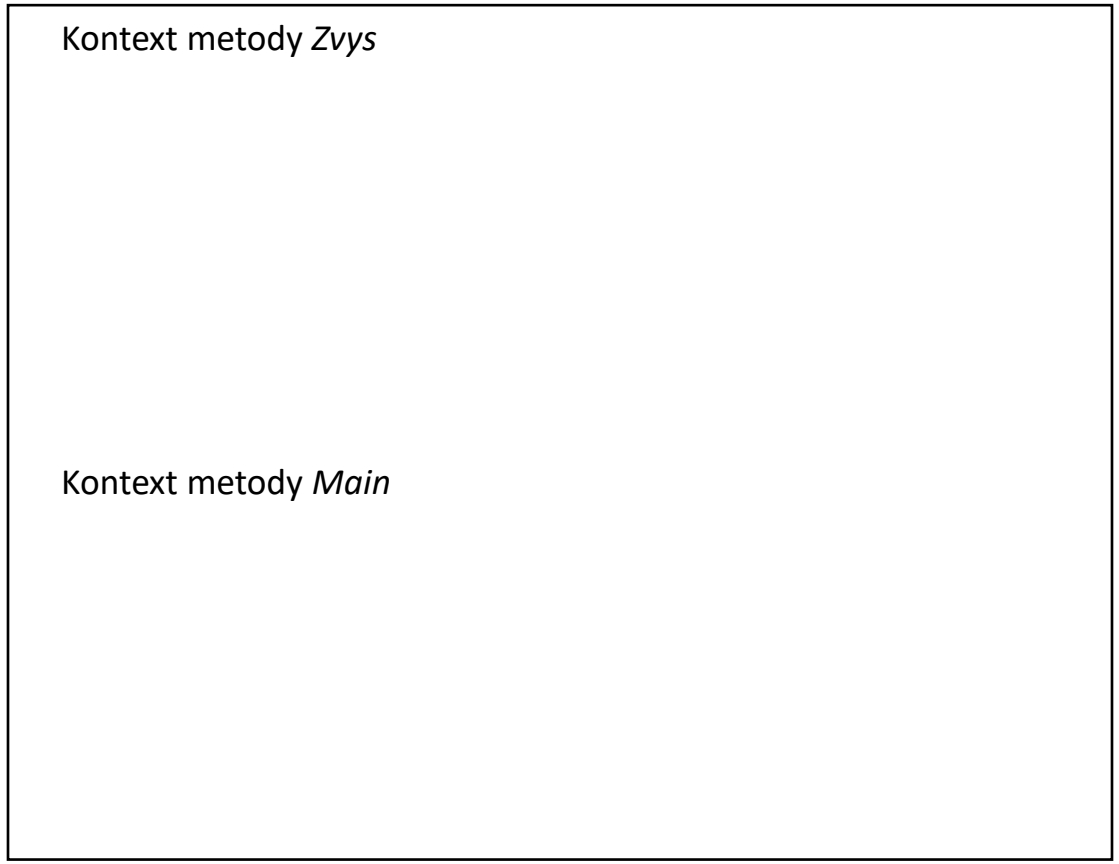
```
class Trida
{
    public int x;
}
```

# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

```
static void Zvys(Trida t)
{
    ++t.x;
}
```

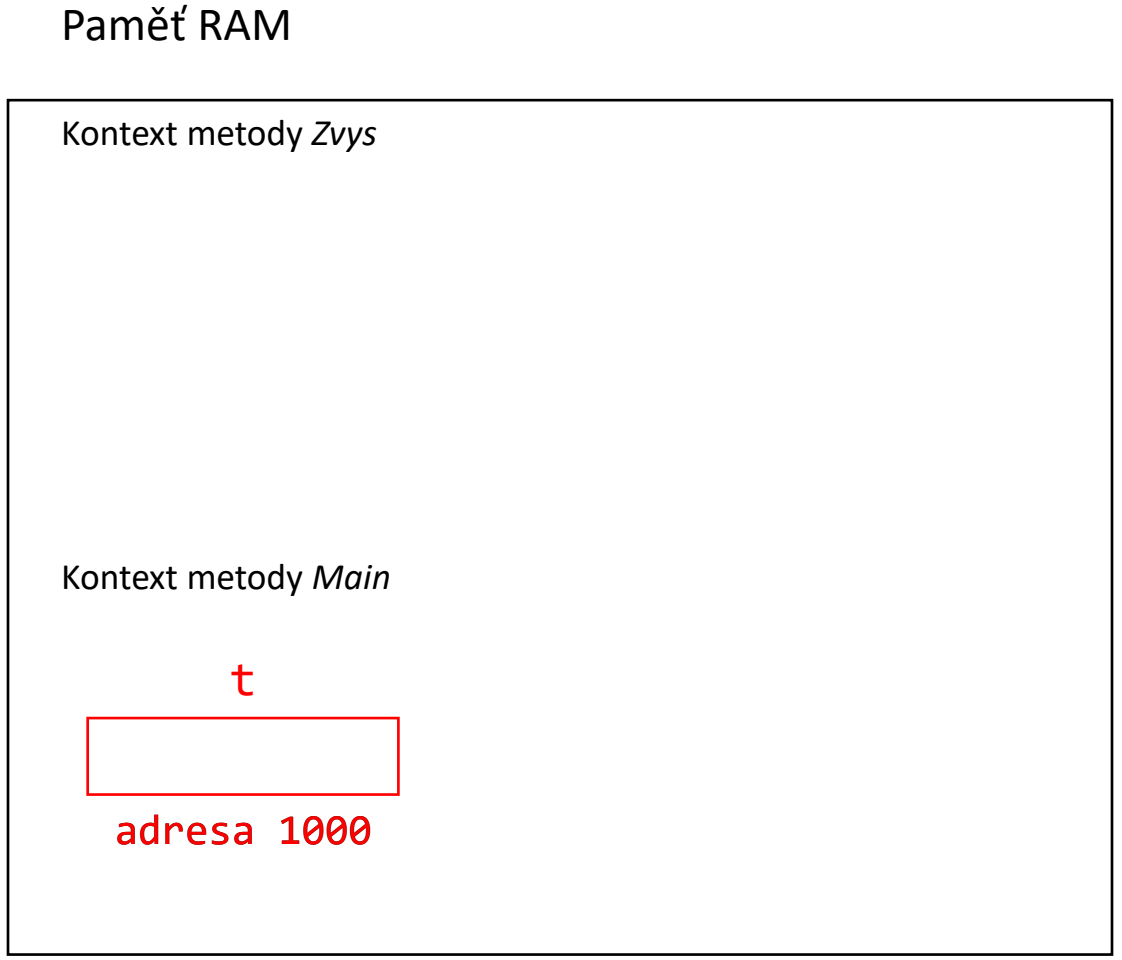
```
static void Main(string[] args)
{
    Trida t = new Trida();
    t.x = 1;
    Zvys(t);
    Console.WriteLine(t.x);
}
```

Paměť RAM



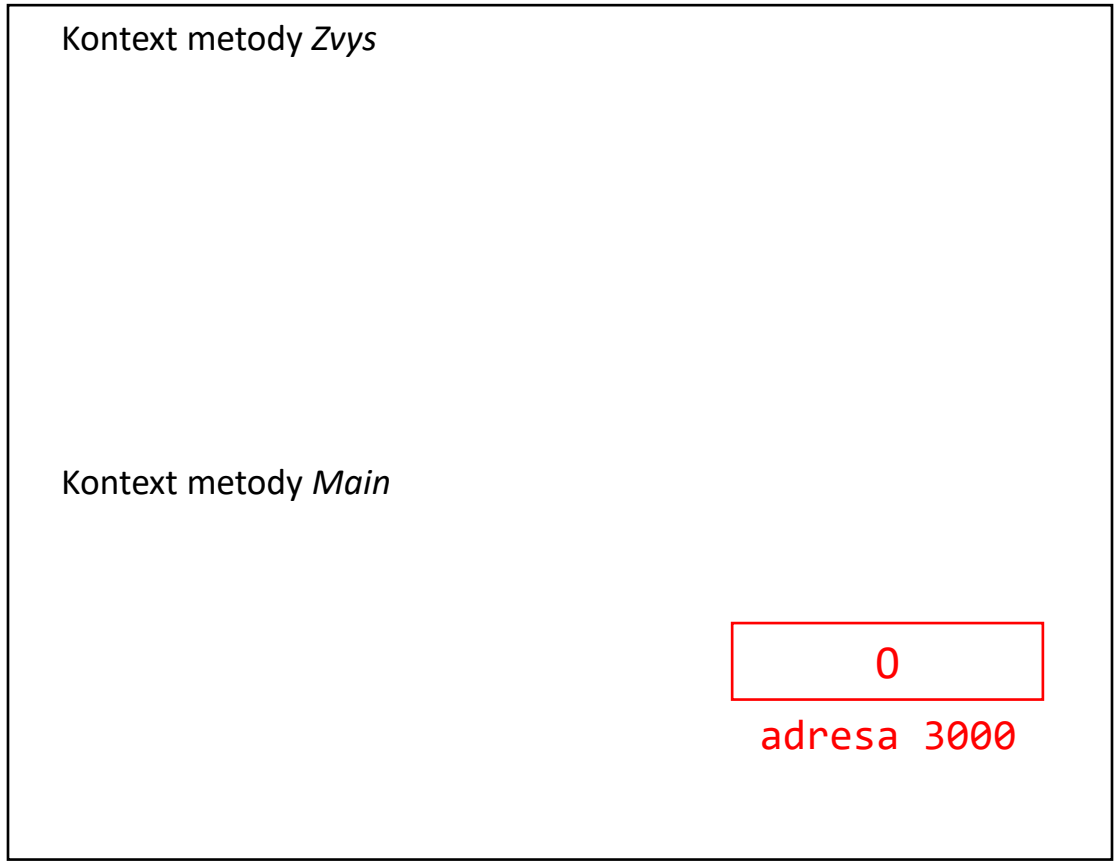
# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida();



}
```



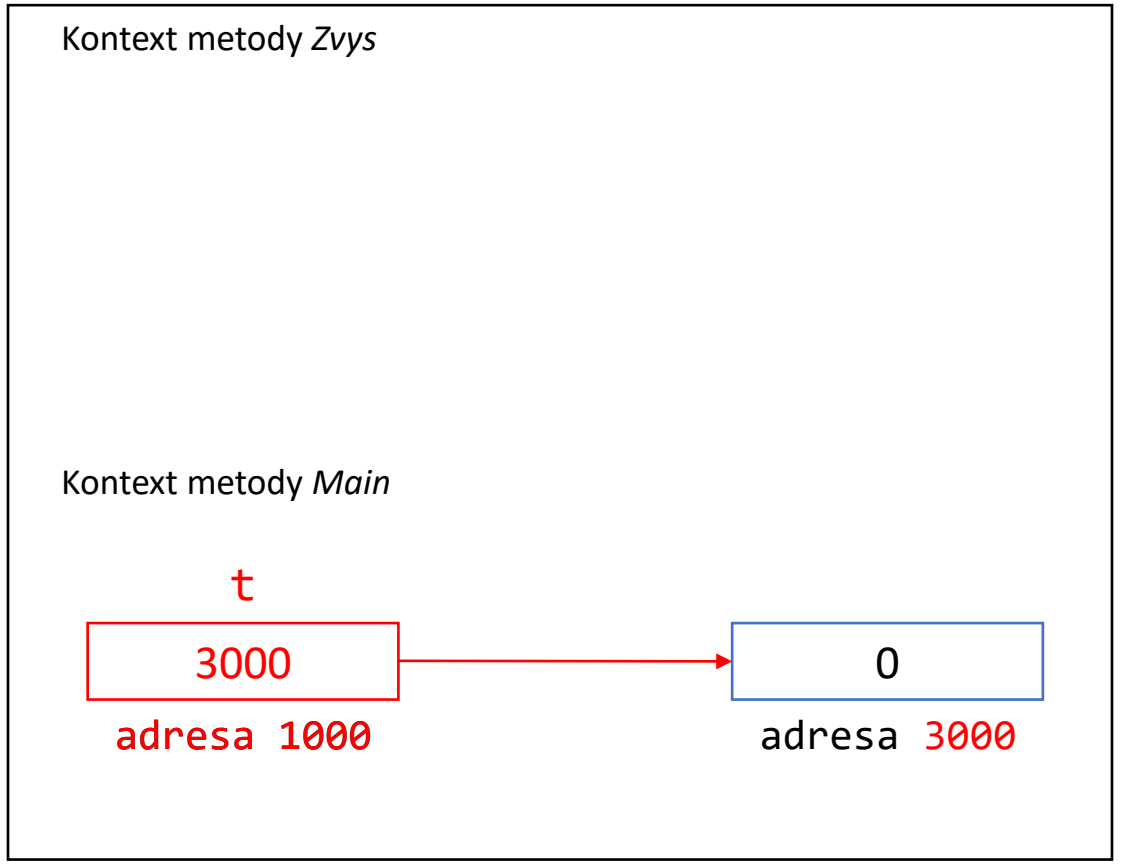
# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida();



}
```



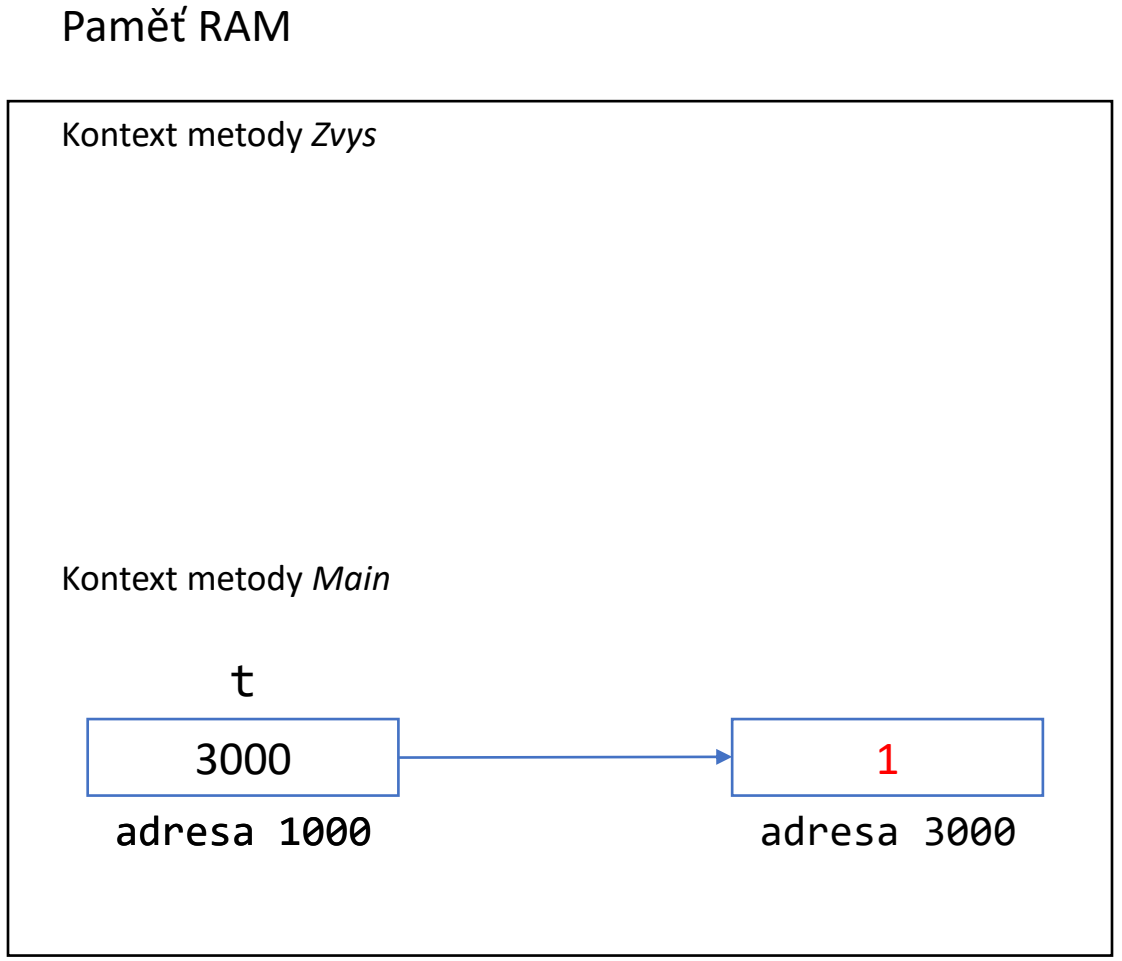
# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida();



}
```



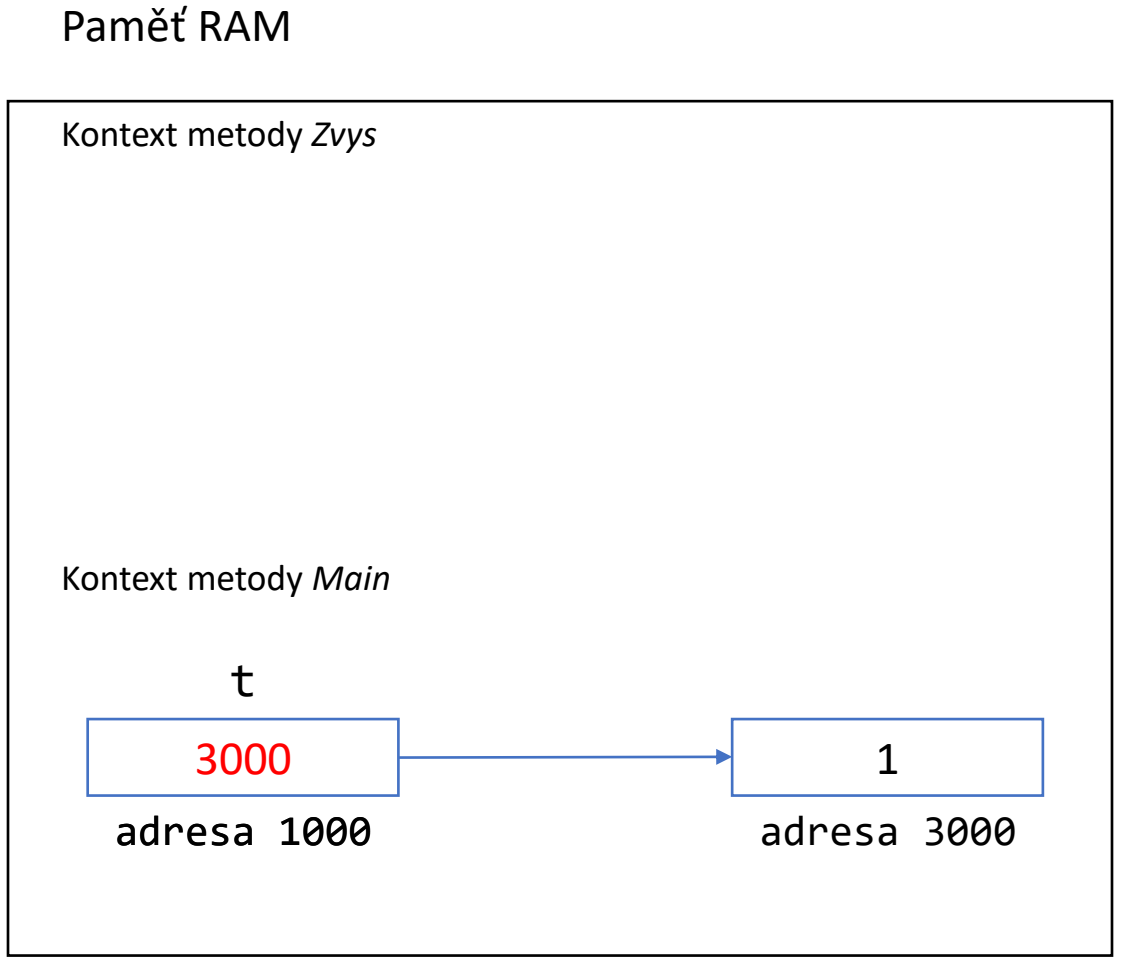
# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida();
    t.x = 1;


}
```



# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

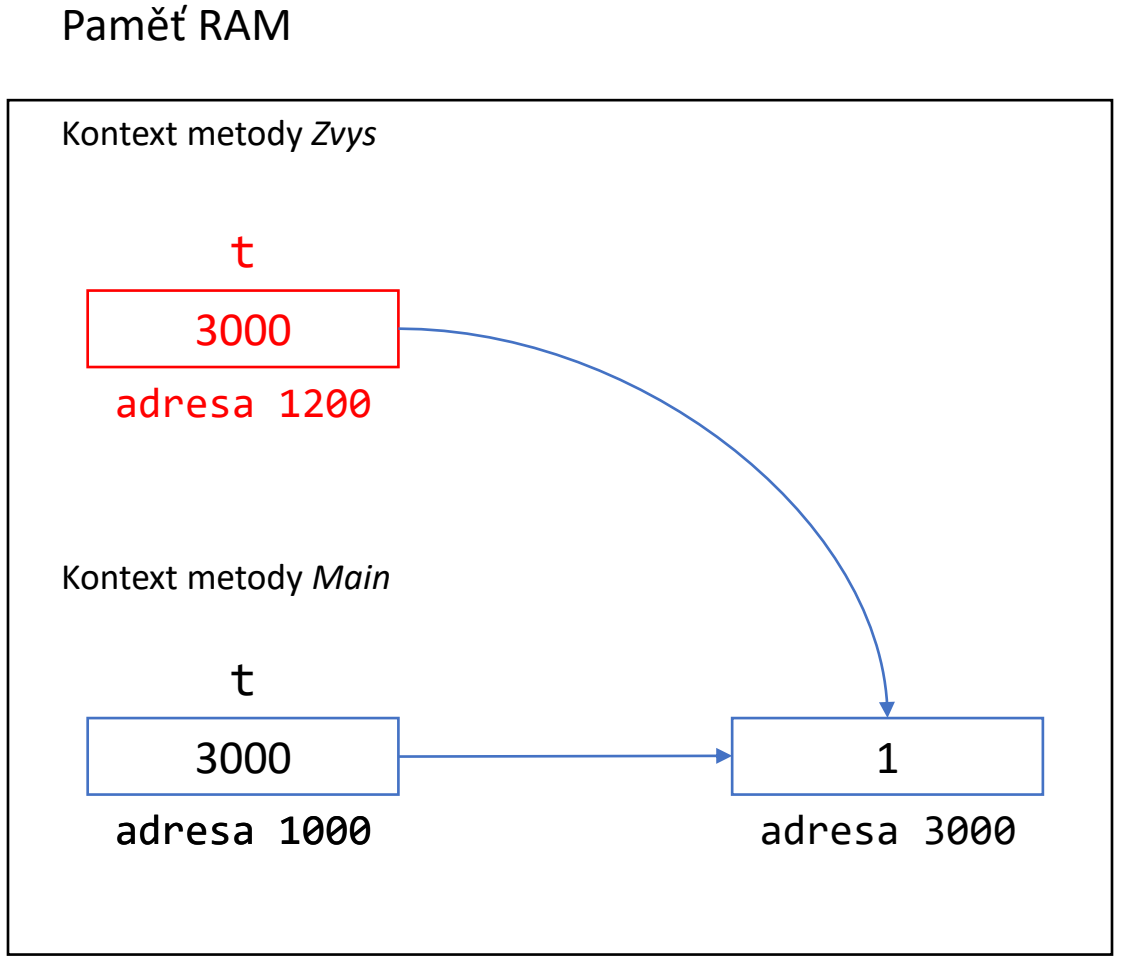
```
static void Main(string[] args)
{
    Trida t = new Trida();
    t.x = 1;
    Zvys(t);

}
```



# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

```
static void Zvys(Trida t)
{
    ++t.x;
}
```

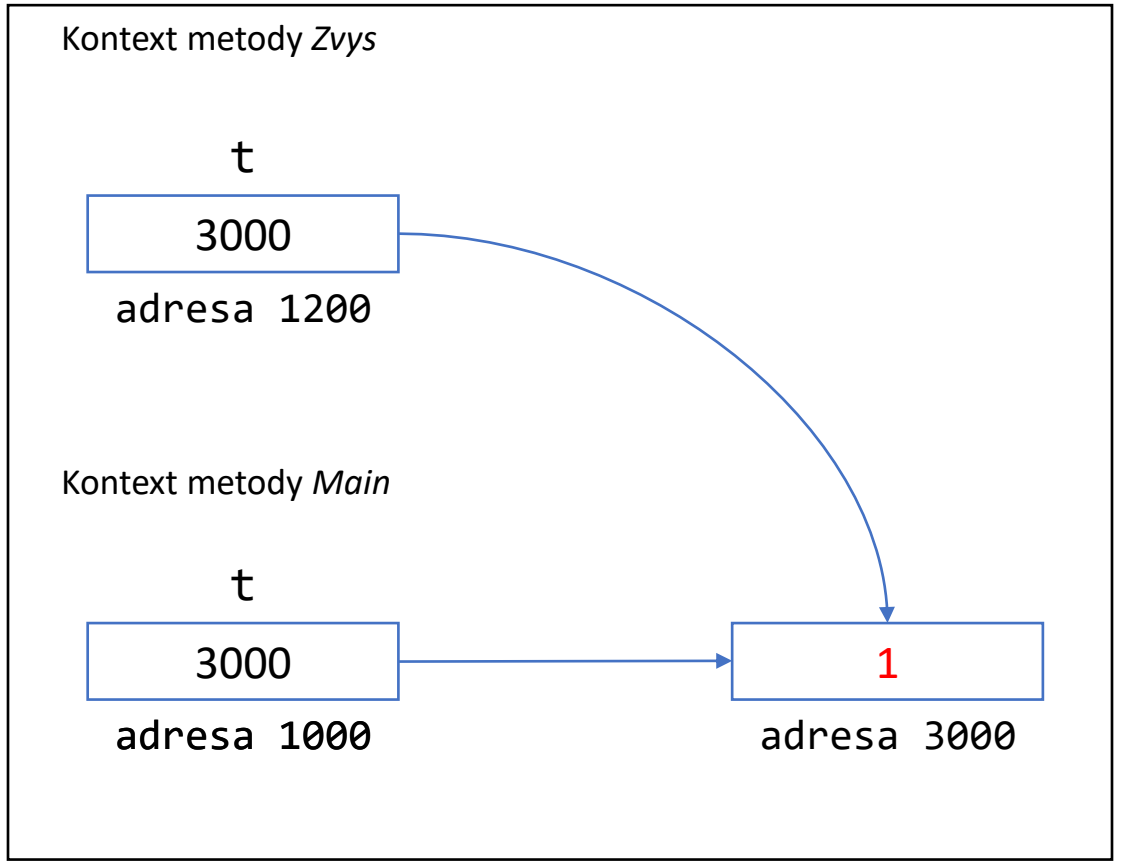
```
static void Main(string[] args)
{
    Trida t = new Trida();
    t.x = 1;
    Zvys(t);

}
```



# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

```
static void Zvys(Trida t)
{
    ++t.x;
}
```

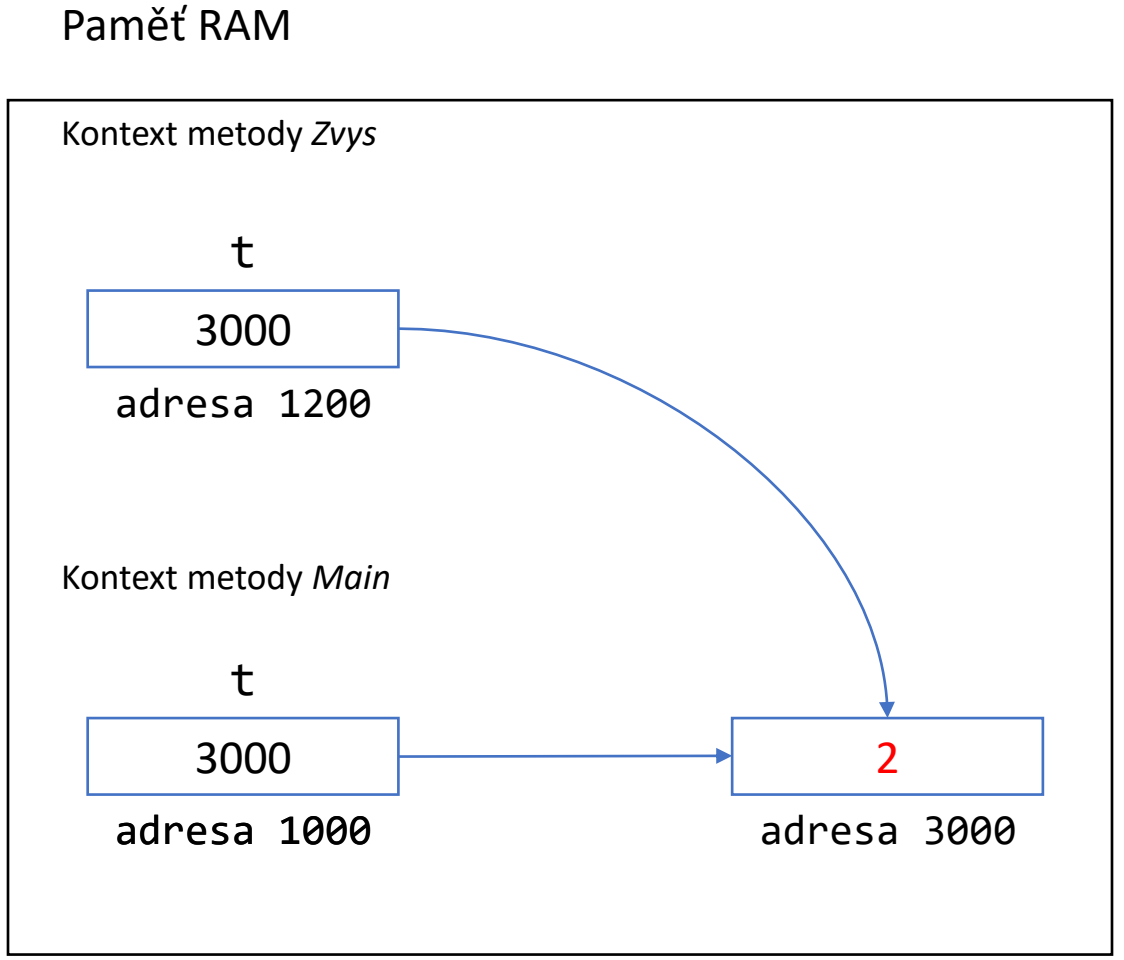
```
static void Main(string[] args)
{
    Trida t = new Trida();
    t.x = 1;
    Zvys(t);

}
```



# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

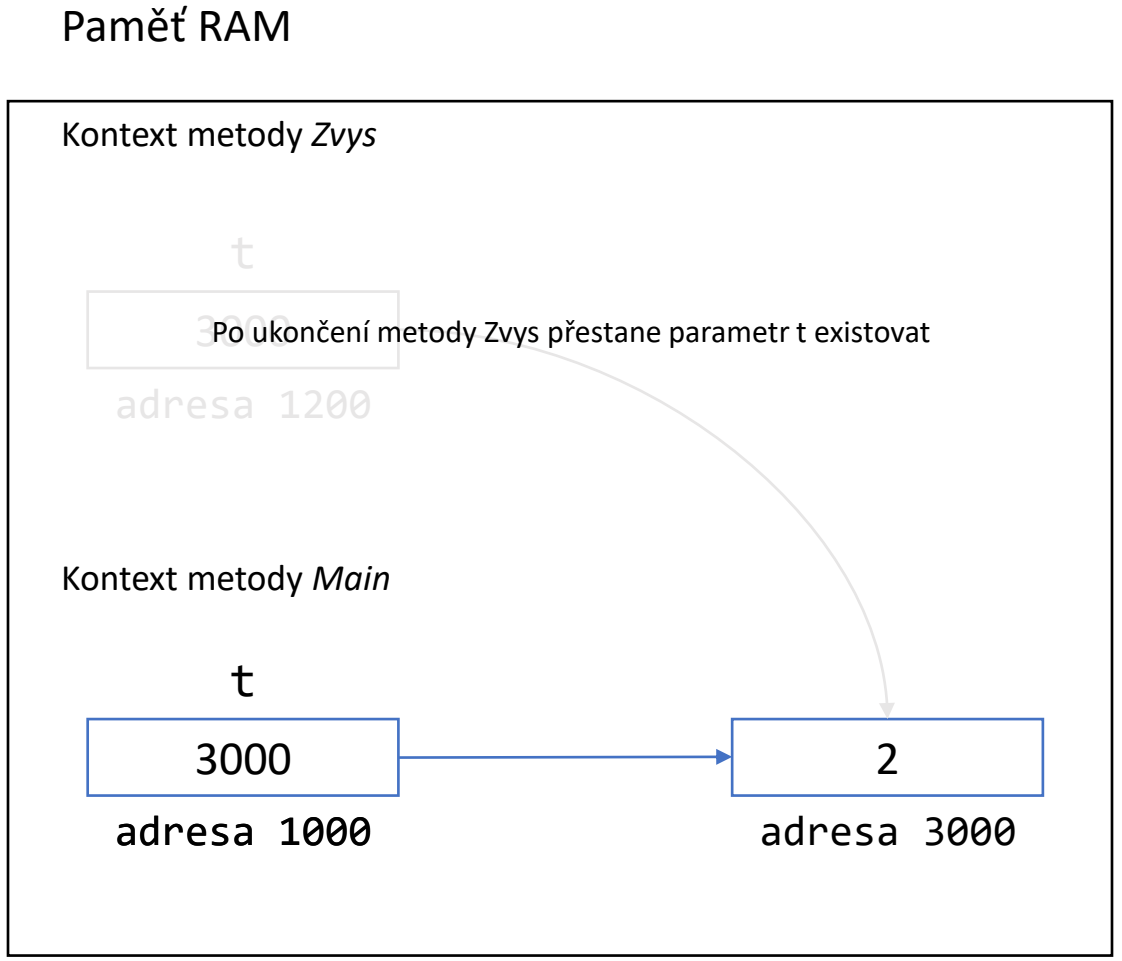
```
static void Zvys(Trida t)
{
    ++t.x;
}
```

```
static void Main(string[] args)
{
    Trida t = new Trida();
    t.x = 1;
    Zvys(t);

}
```



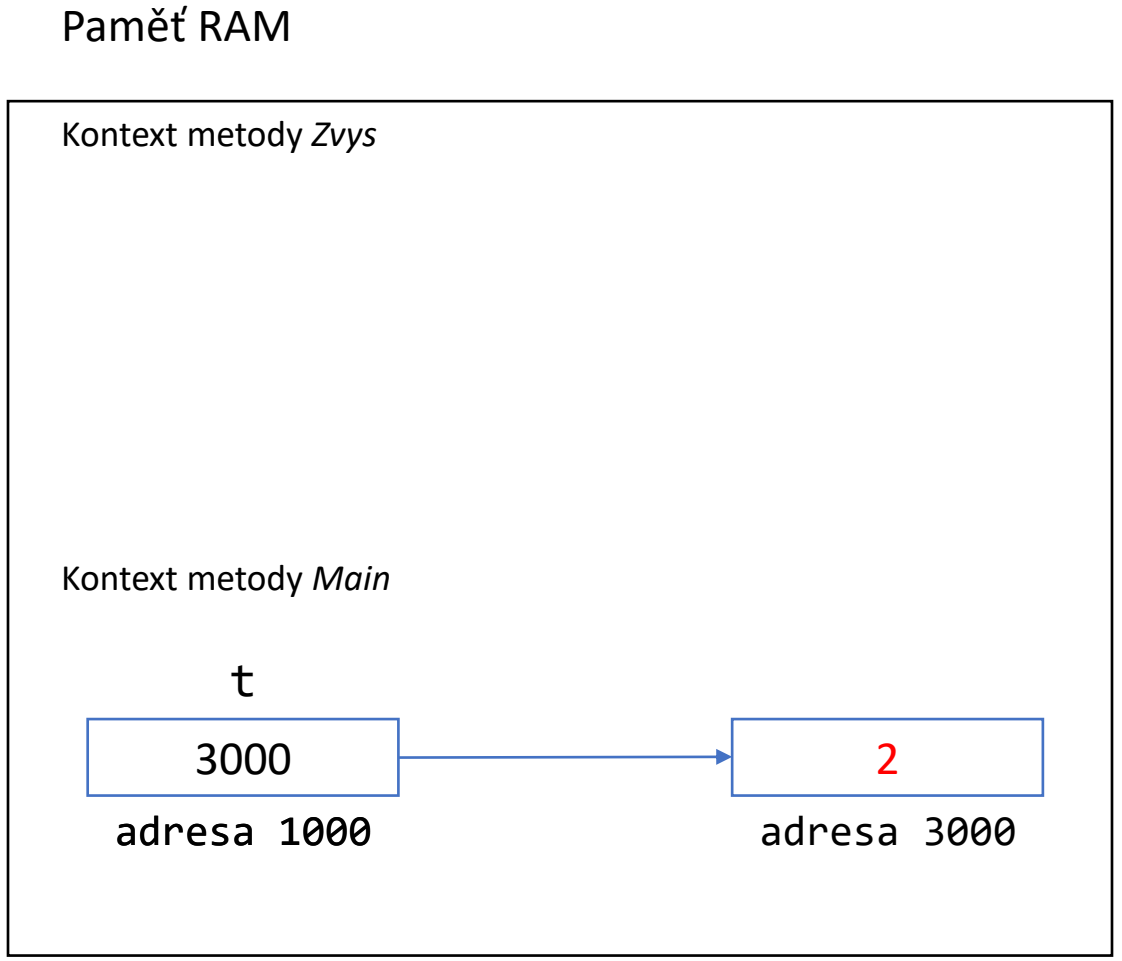
# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida();
    t.x = 1;
    Zvys(t);
    Console.WriteLine(t.x);
}
```



# Příklad 1 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida();
    t.x = 1;
    Zvys(t);
    Console.WriteLine(t.x);
}
```



# Příklad 2 argumentu referenčního typu s klíčovým slovem *ref*

- V následujícím příkladu si ukážeme příklad, kdy budeme chtít v metodě vytvořit novou instanci třídy a změnit referenci v parametru metody.
- Protože se jako argument metodě předává **kopie** reference, tak se tato změna reference neprojeví na originální referenci.

# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

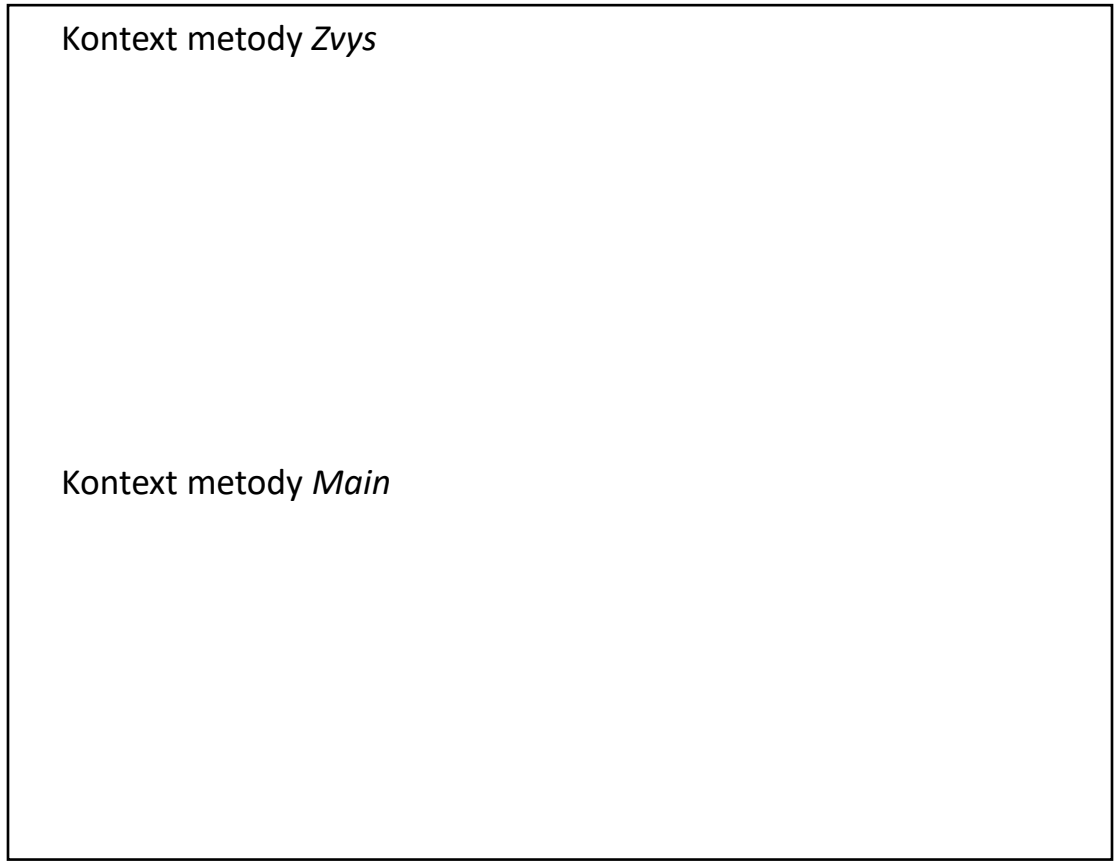
```
class Trida
{
    public int x;

    public Trida(int x)
    {
        this.x = x;
    }
}
```

# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

```
static void Zmen(Trida t)
{
    t = new Trida(1);
}
```

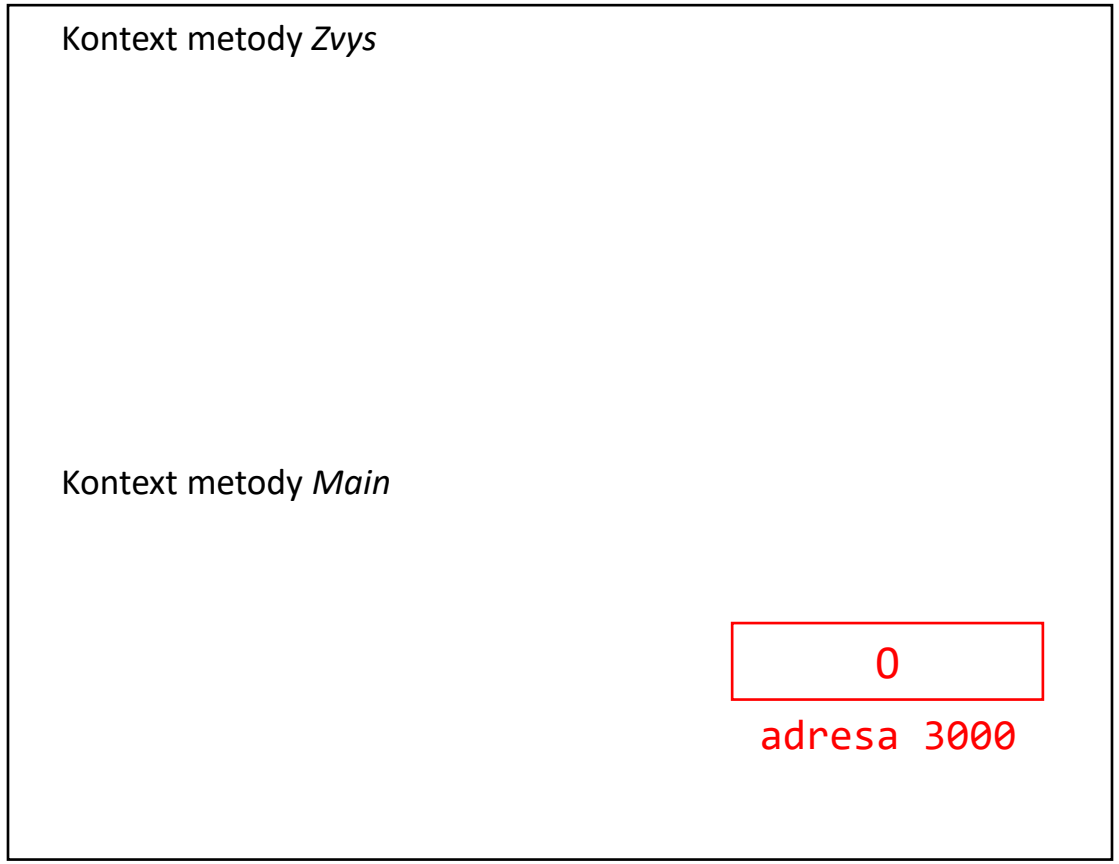
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(t);
    Console.WriteLine(t.x);
}
```

Paměť RAM



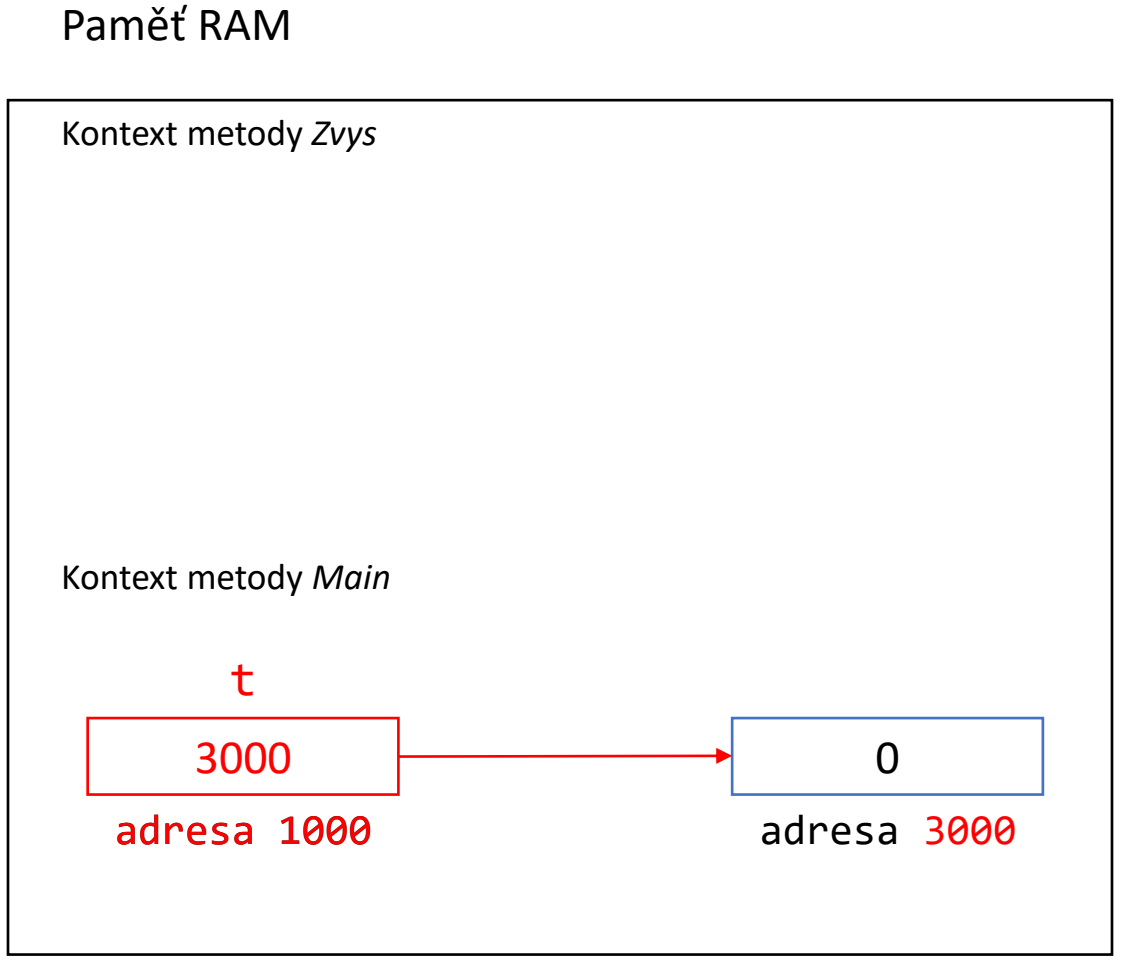
# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida(0);


}
```



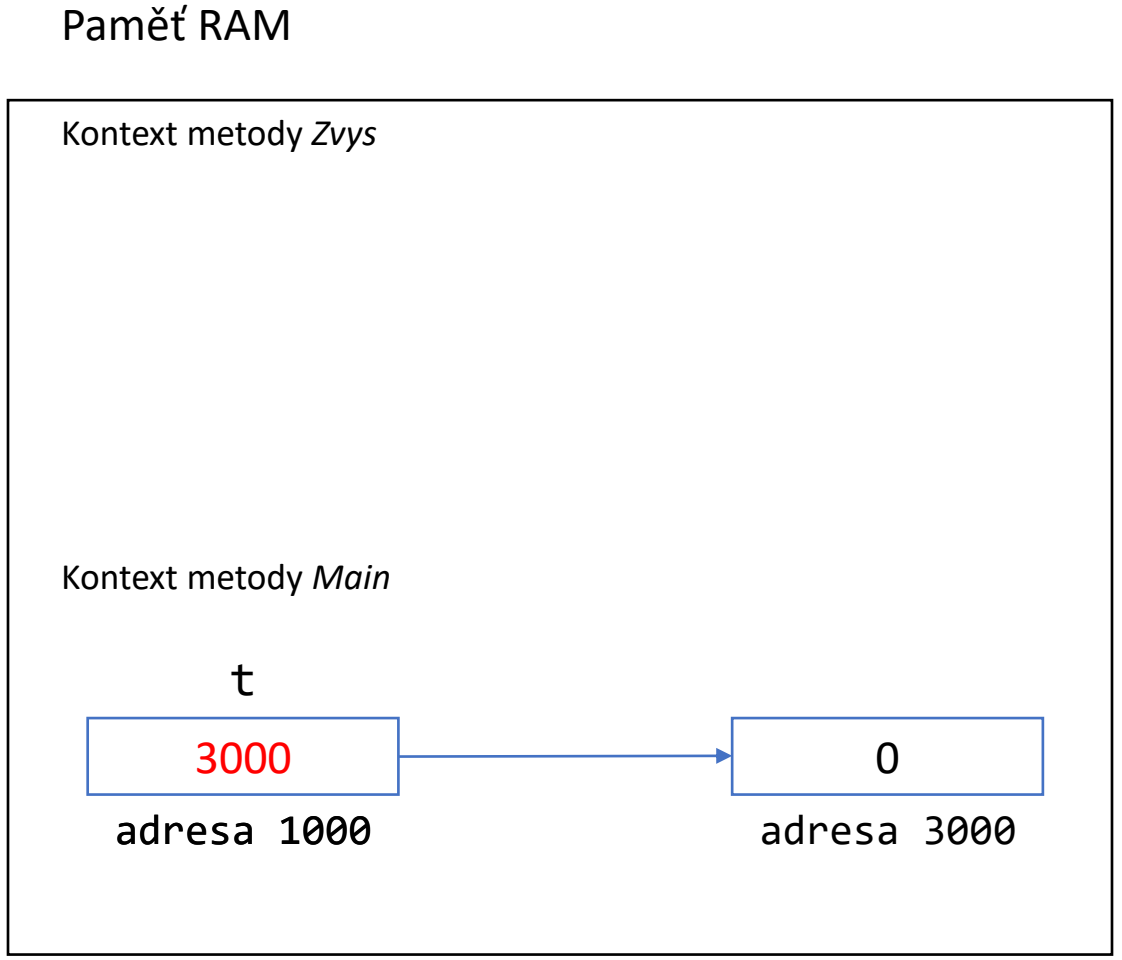
# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida(0);


}
```



# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

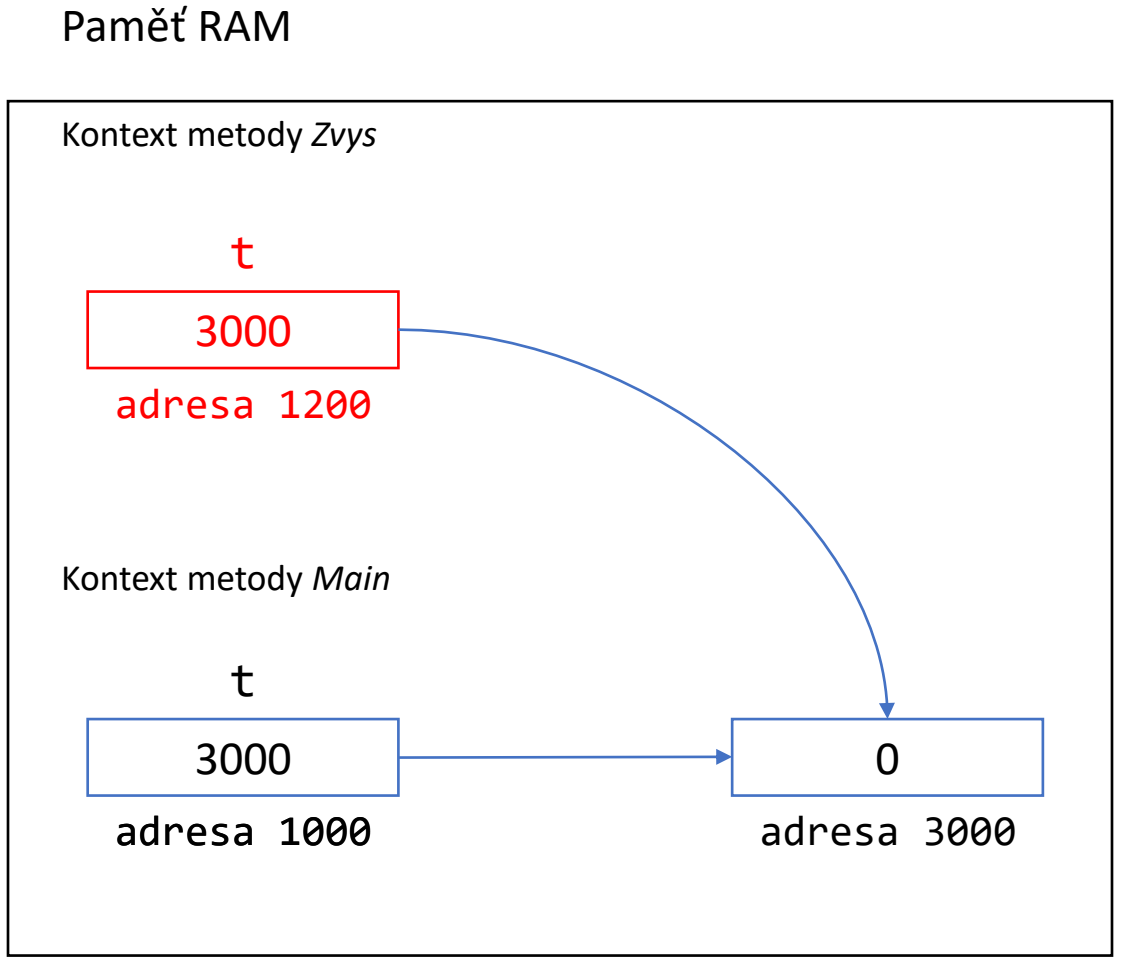
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(t);

}
```



# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

```
static void Zmen(Trida t)
{

}
```

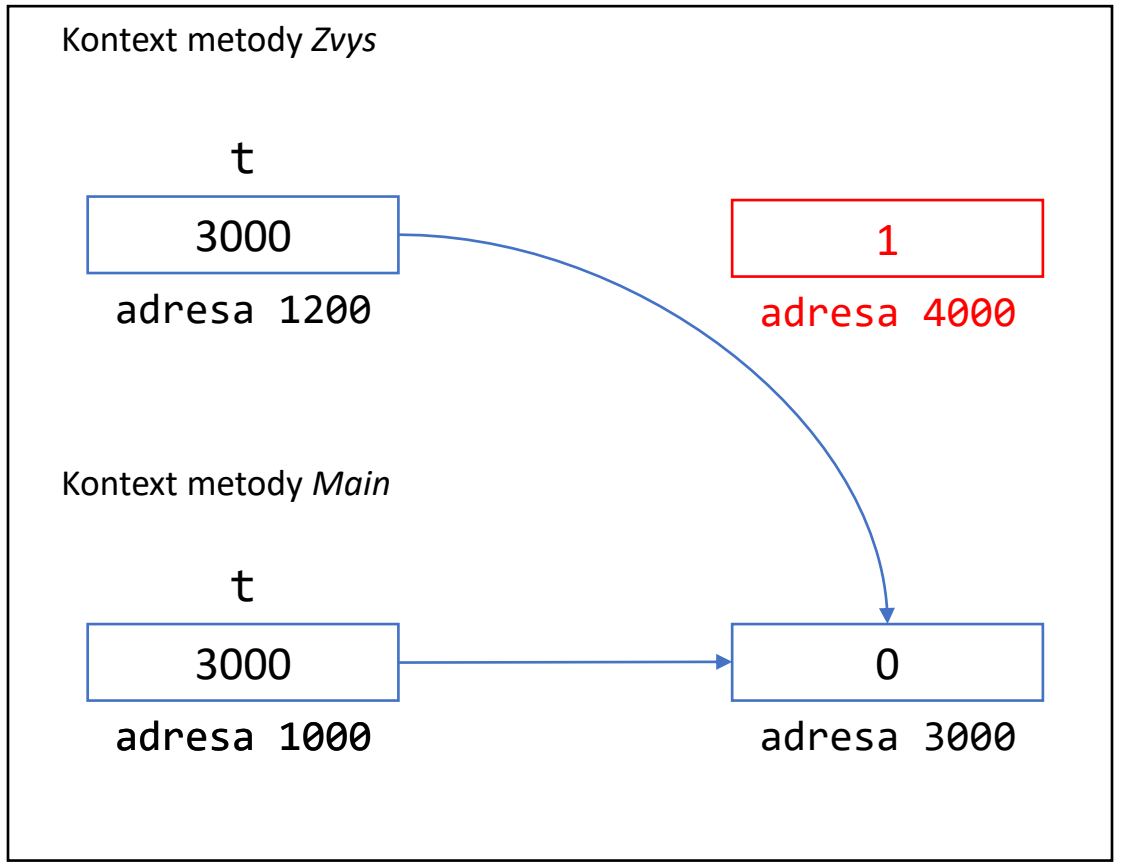
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(t);
    Console.WriteLine(t.x);
}
```



# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

```
static void Zmen(Trida t)
{
    t = new Trida(1);
}
```

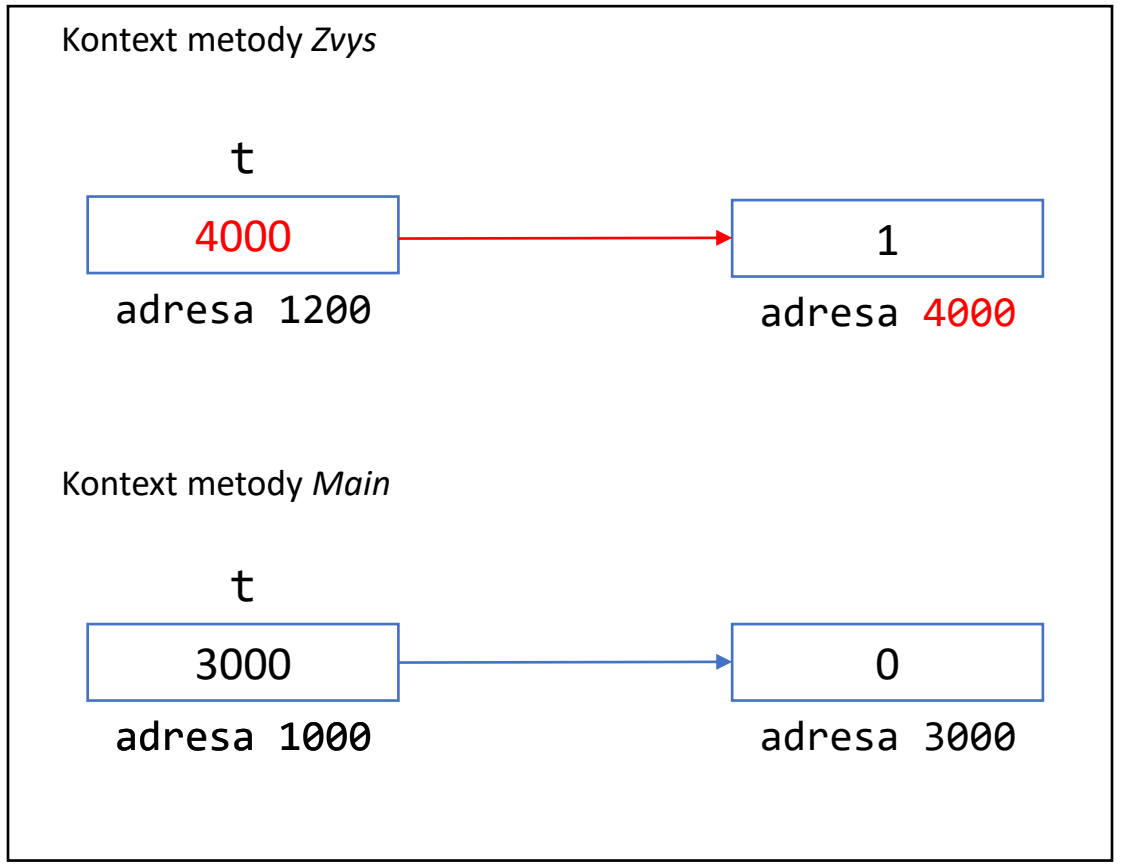
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(t);
    Console.WriteLine(t.x);
}
```



# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

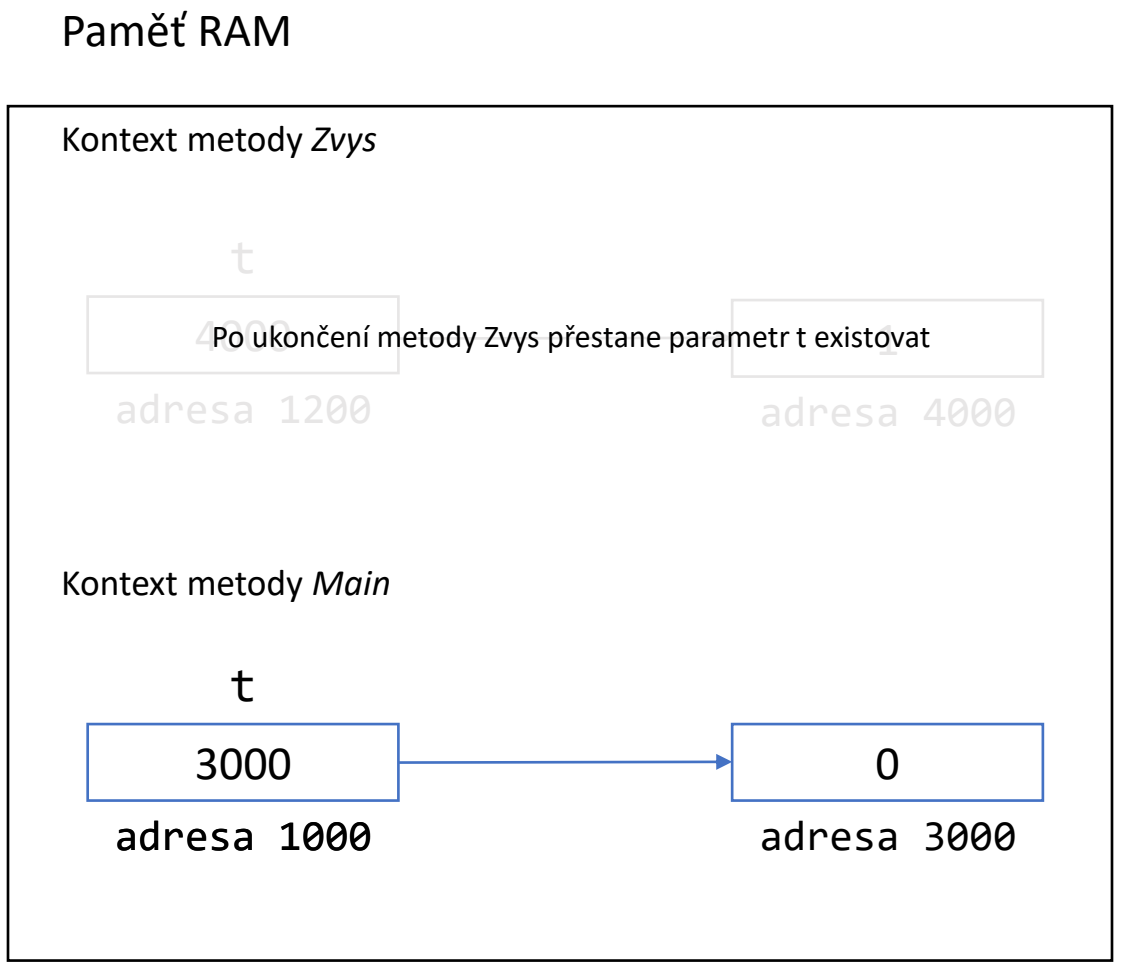
```
static void Zmen(Trida t)
{
    t = new Trida(1);
}
```

```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(t);
    Console.WriteLine(t.x);
}
```



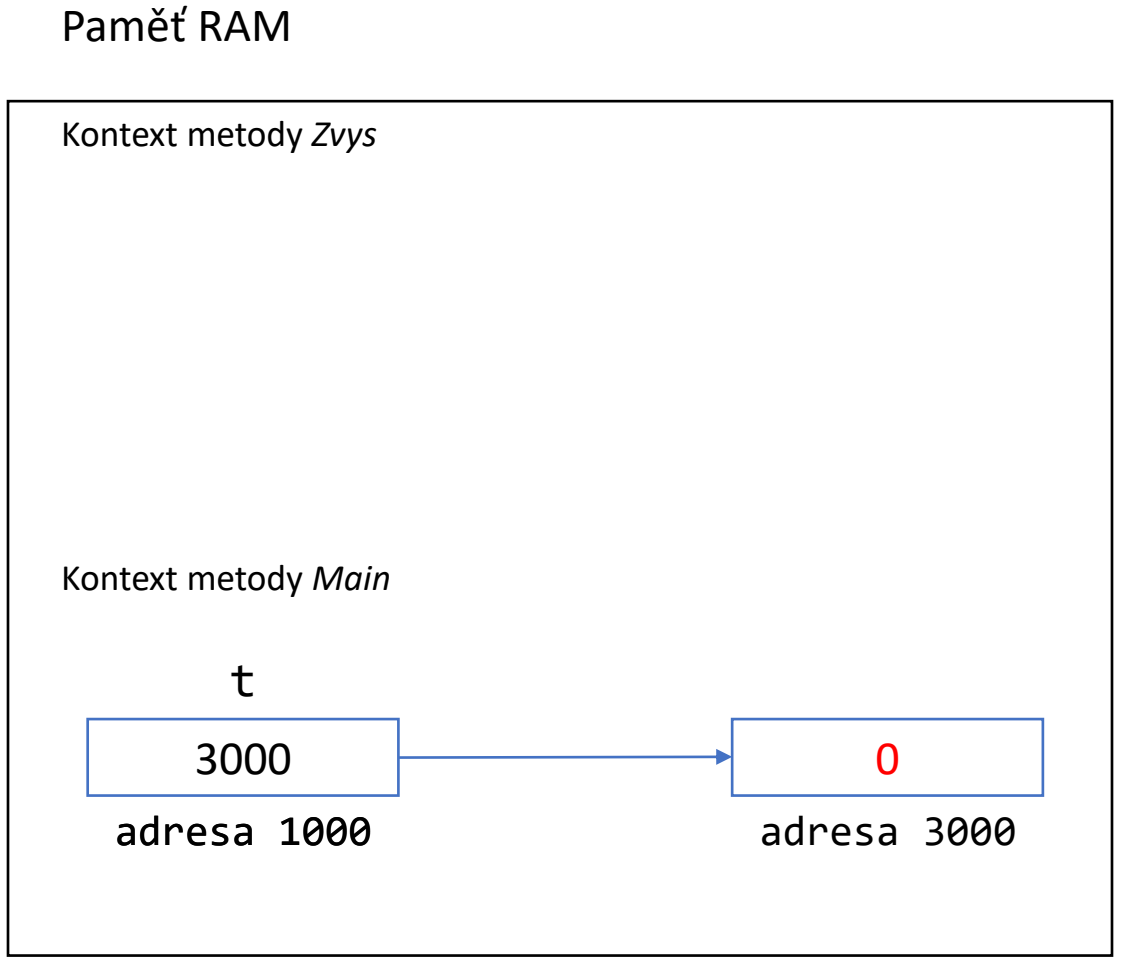
# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(t);
    Console.WriteLine(t.x);
}
```



# Příklad 2 argumentu referenčního typu bez klíčového slova *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(t);
    Console.WriteLine(t.x);
}
```



# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

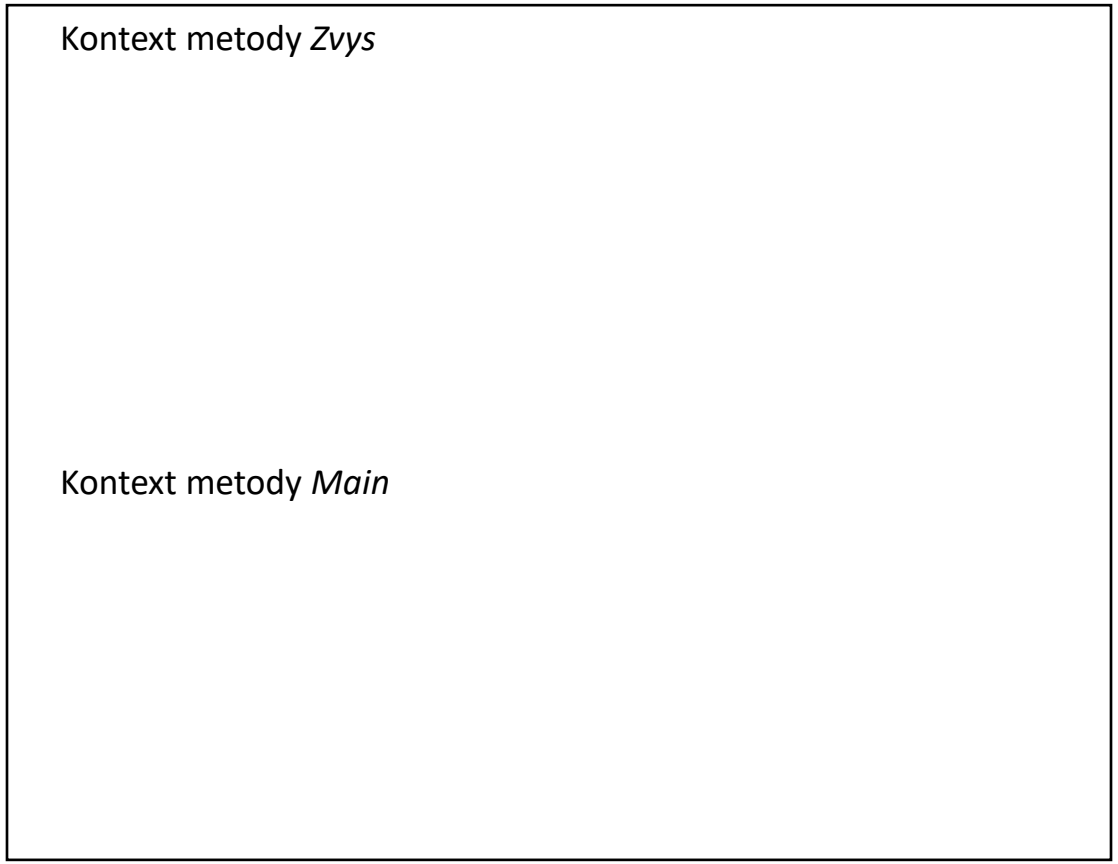
- V následujícím příkladu si ukážeme příklad, kdy budeme chtít v metodě vytvořit novou instanci třídy a změnit referenci v parametru metody.
- Protože se jako argument metodě předává **reference** reference, tak se tato změna reference **projeví**.

# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Zmen(ref Trida t)
{
    t = new Trida(1);
}
```

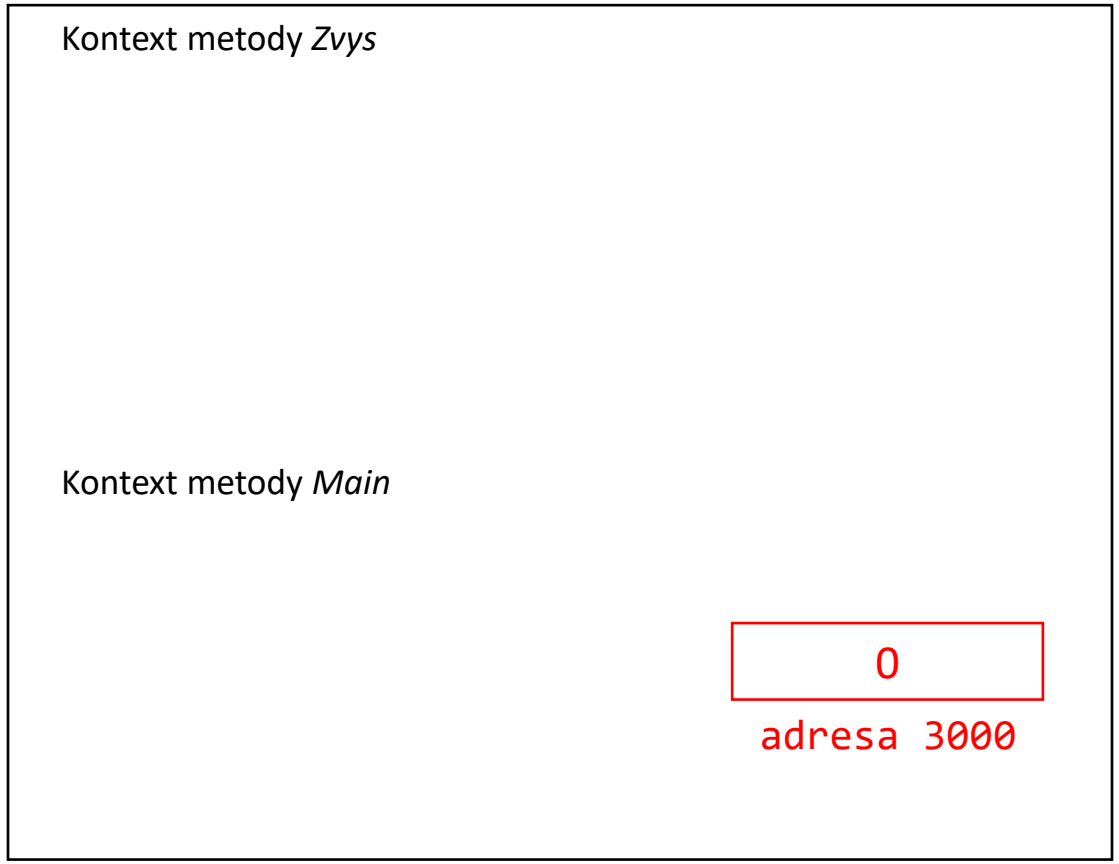
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);
    Console.WriteLine(t.x);
}
```

Paměť RAM



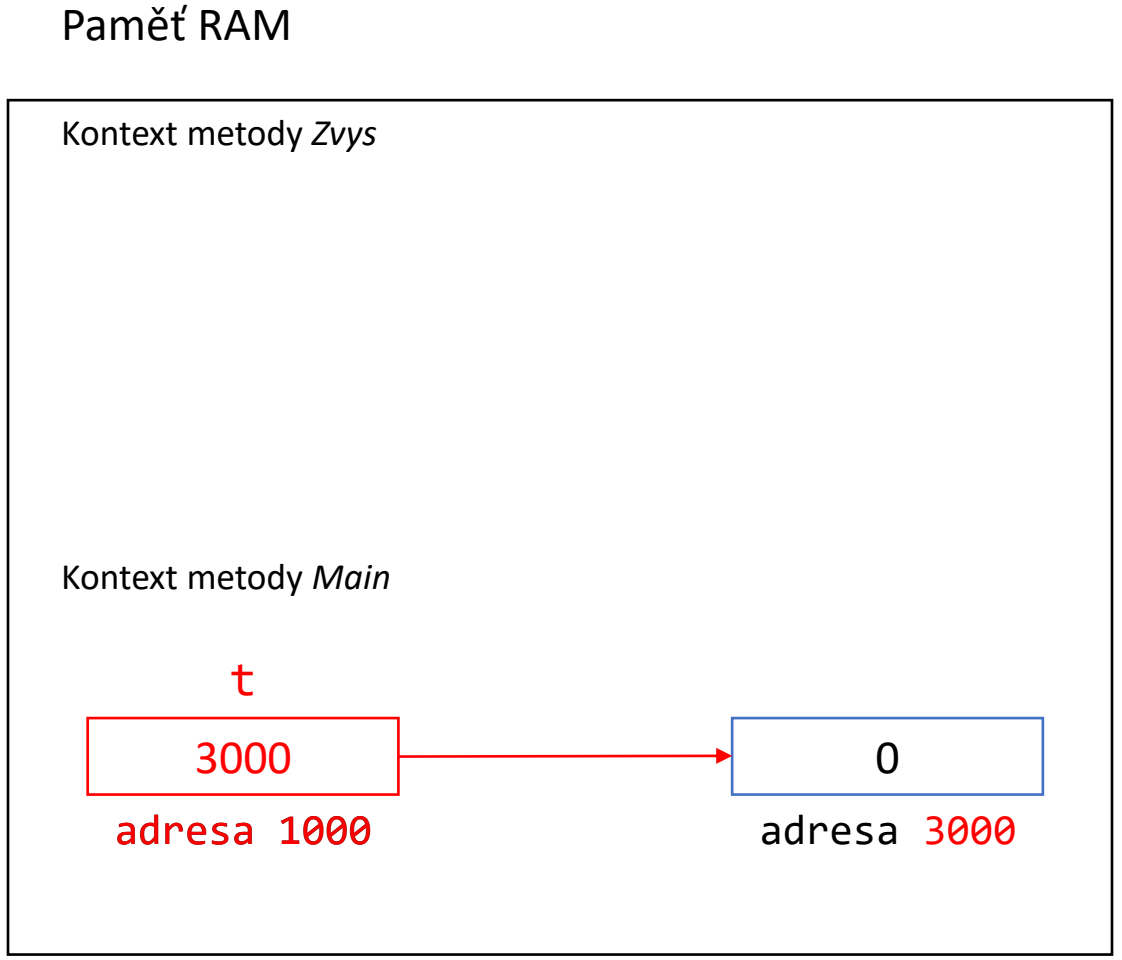
# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida(0);


}
```



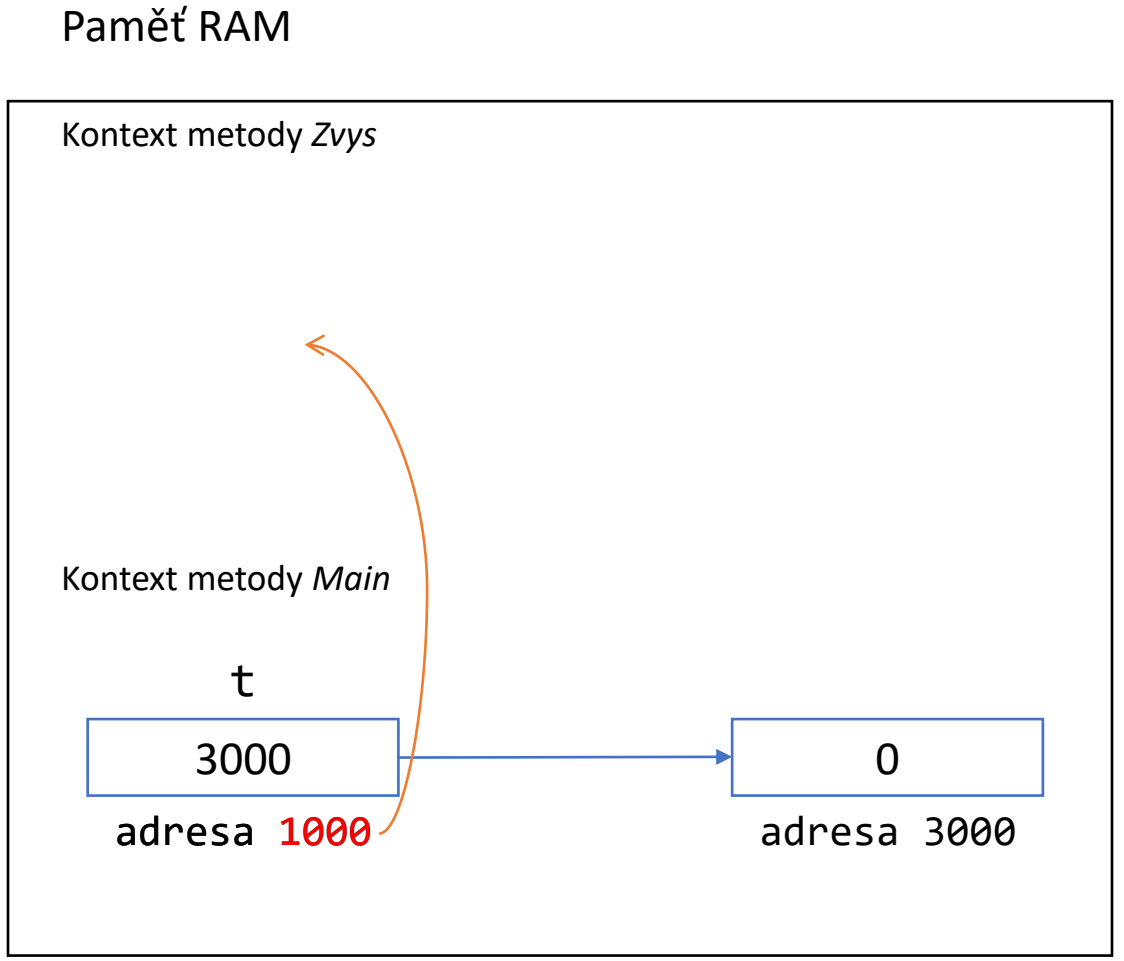
# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida(0);


}
```



# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

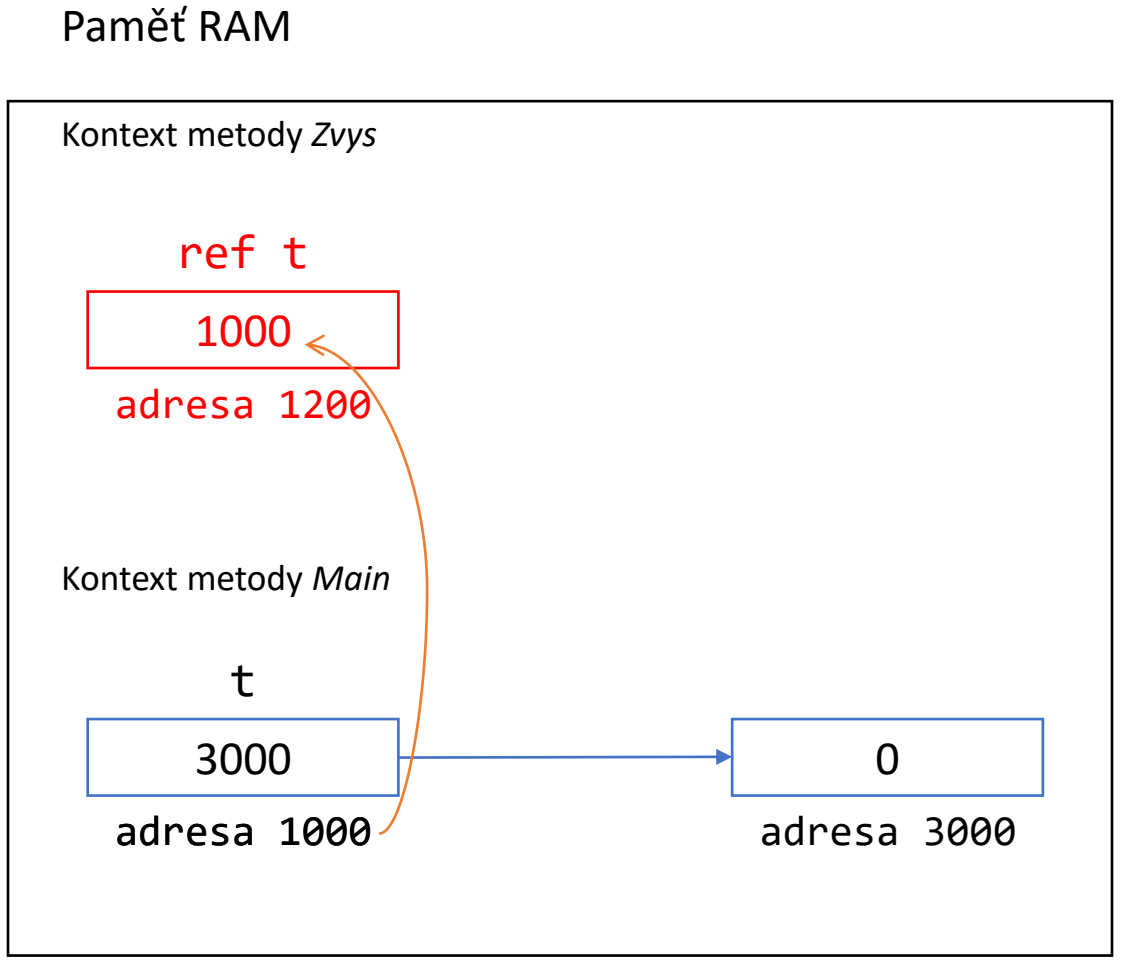
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);

}
```



# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Zmen(ref Trida t)
{

}
```

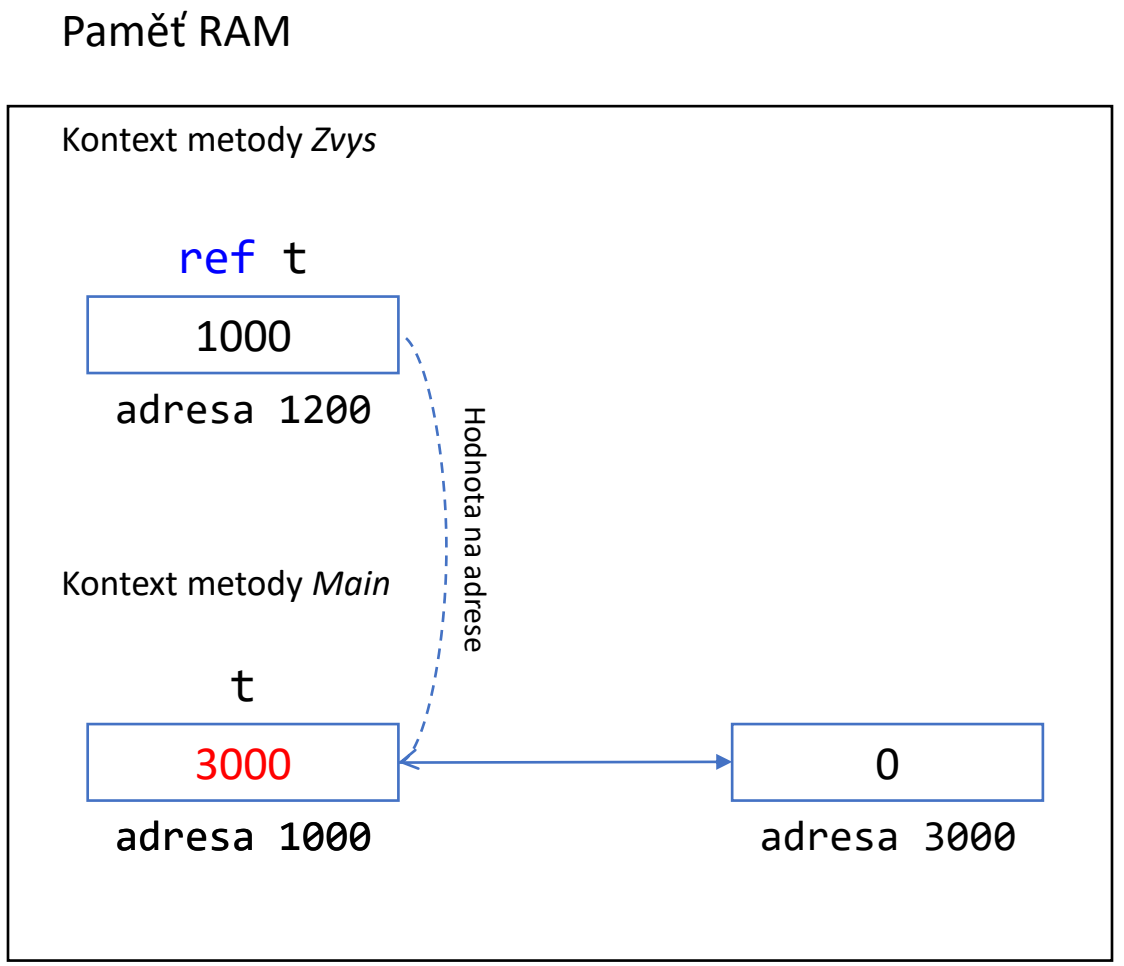
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);

}
```



# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Zmen(ref Trida t)
{
    t = new Trida(1);
}
```

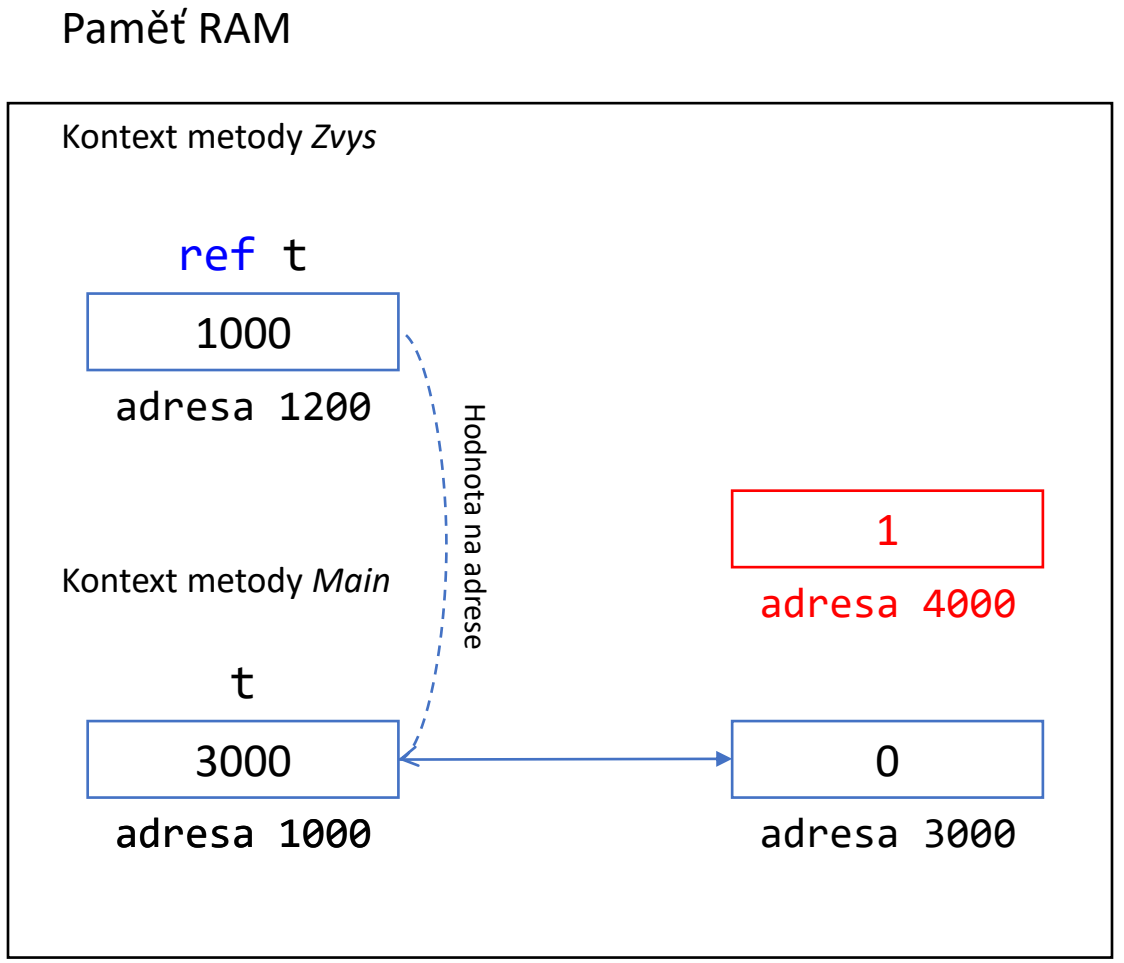
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);

}
```



# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Zmen(ref Trida t)
{
    t = new Trida(1);
}
```

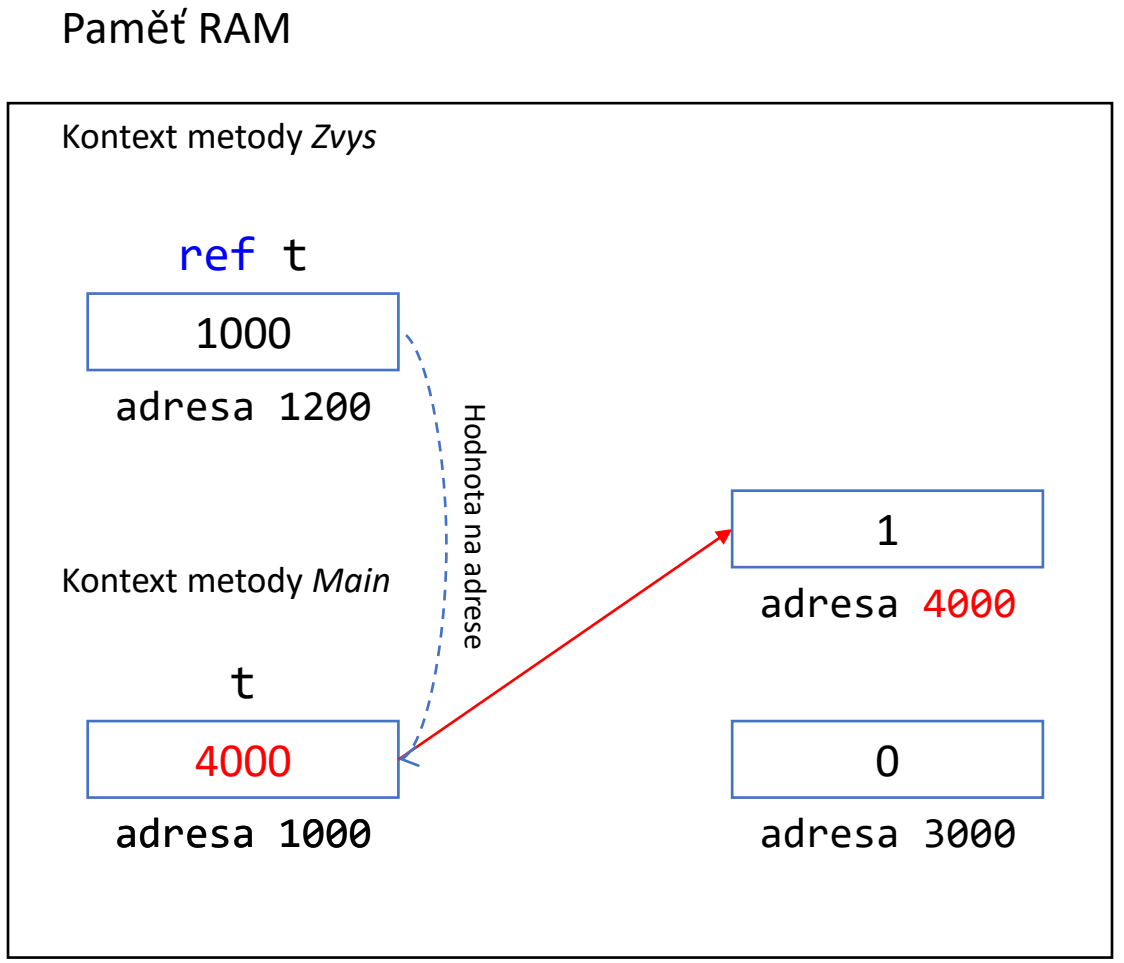
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);

}
```



# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Zmen(ref Trida t)
{
    t = new Trida(1);
}
```

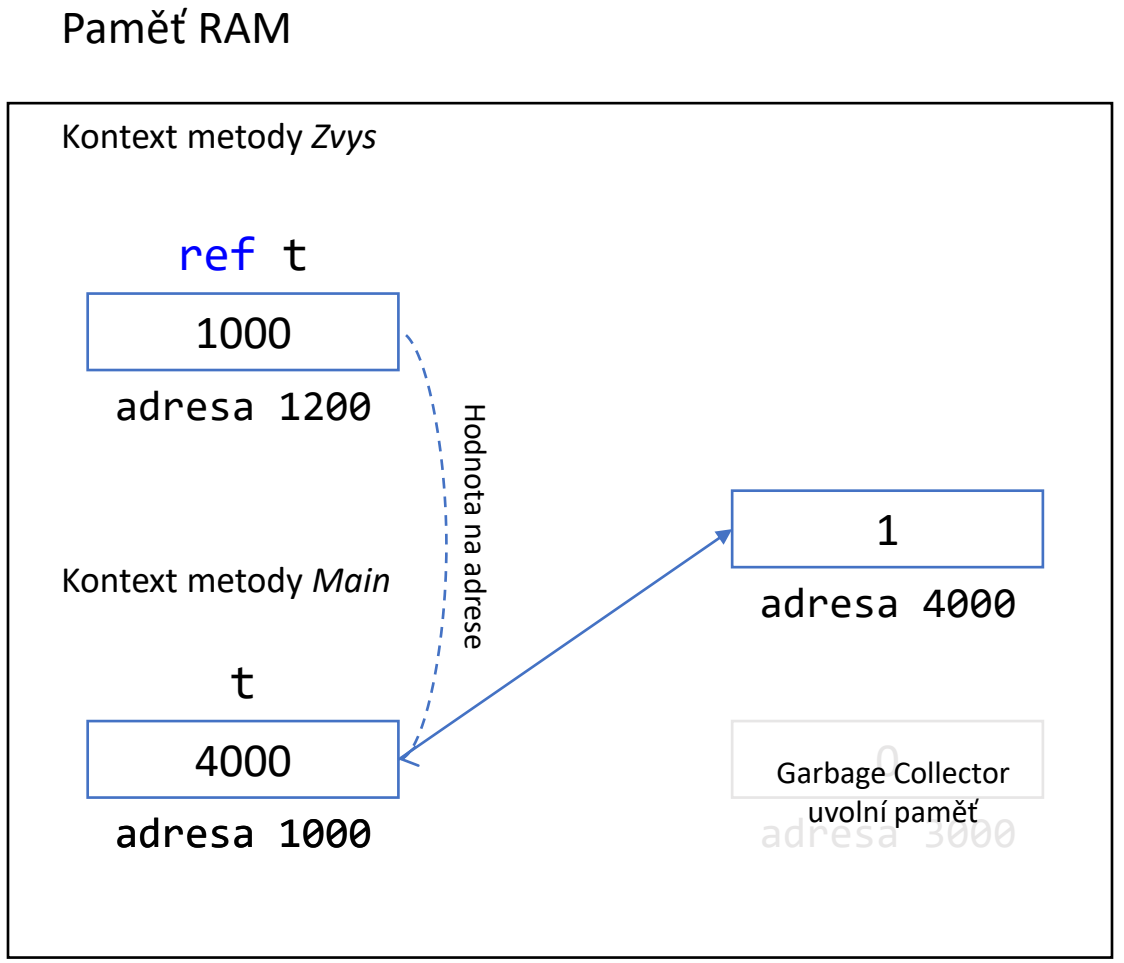
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);

}
```



# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Zmen(ref Trida t)
{
    t = new Trida(1);
}
```

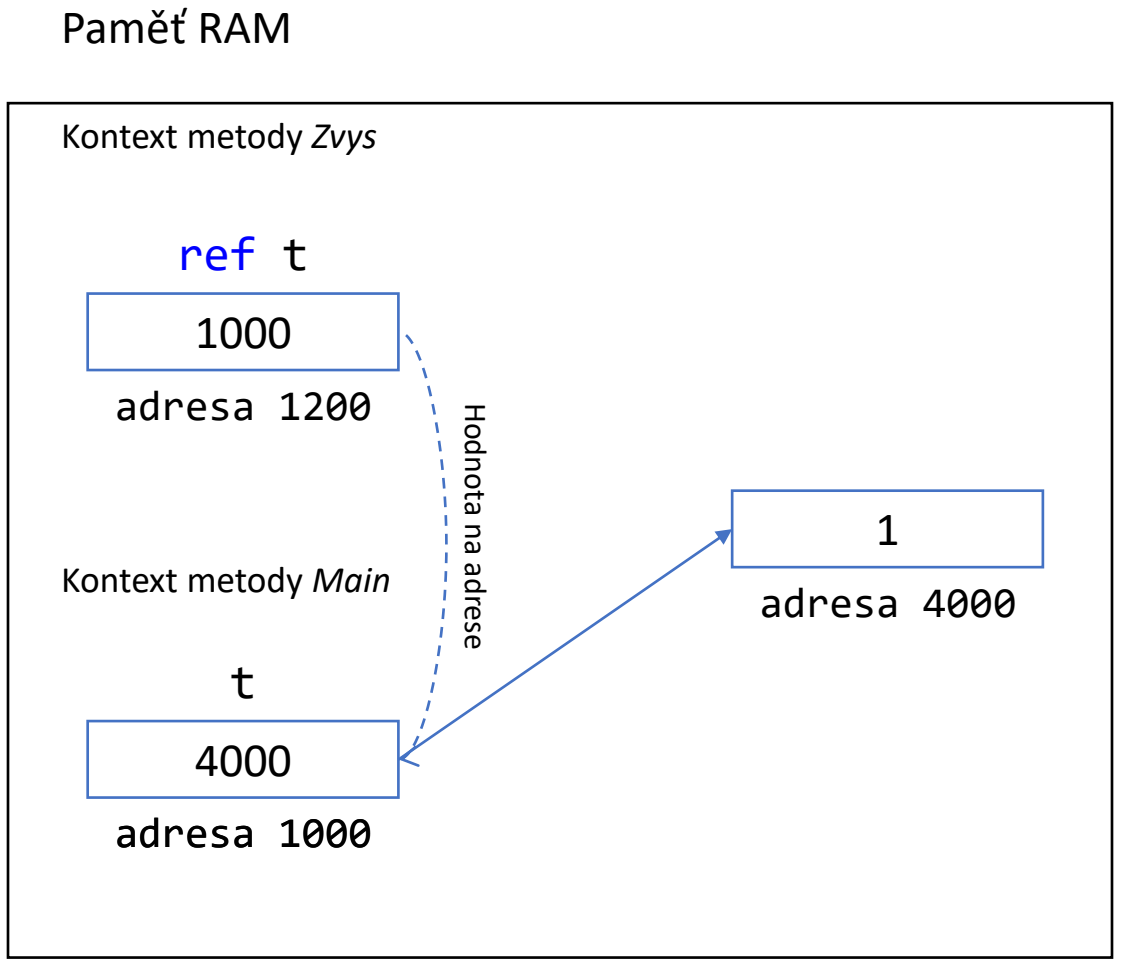
```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);

}
```



# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

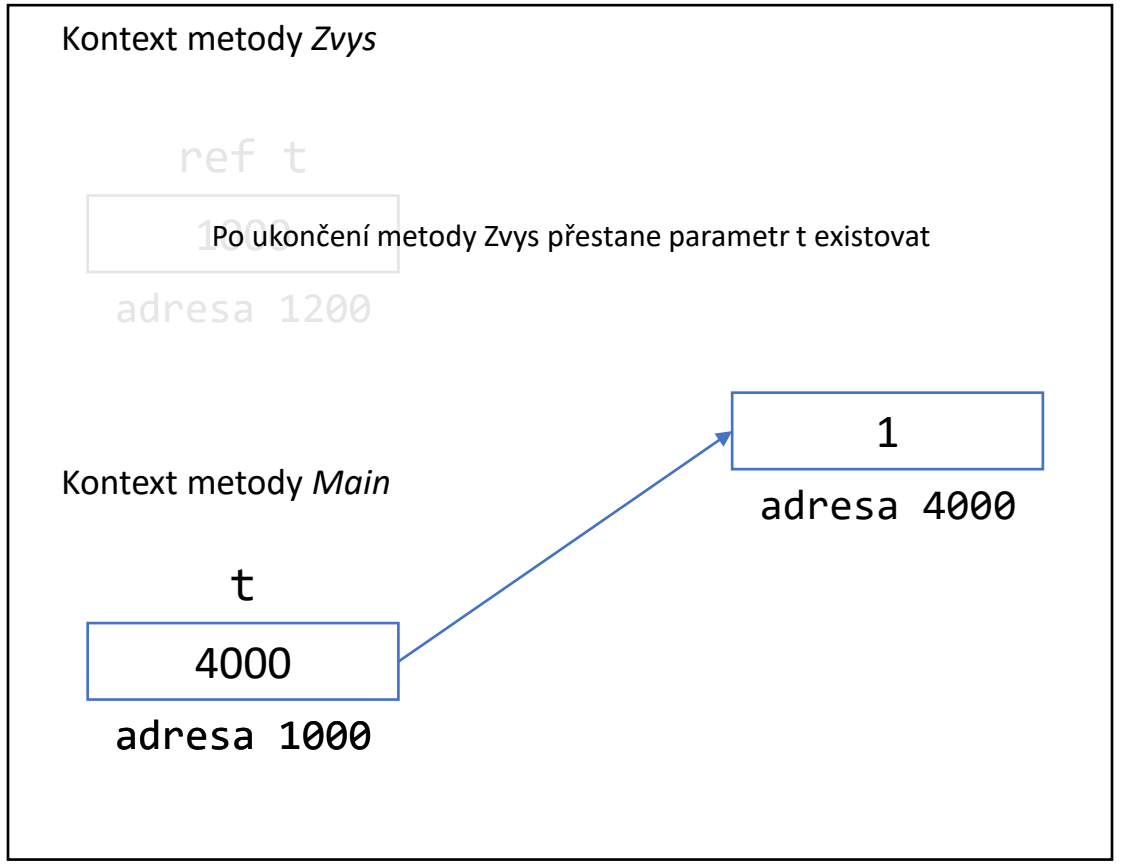
```
static void Zmen(ref Trida t)
{
    t = new Trida(1);
}
```

```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);

}
```



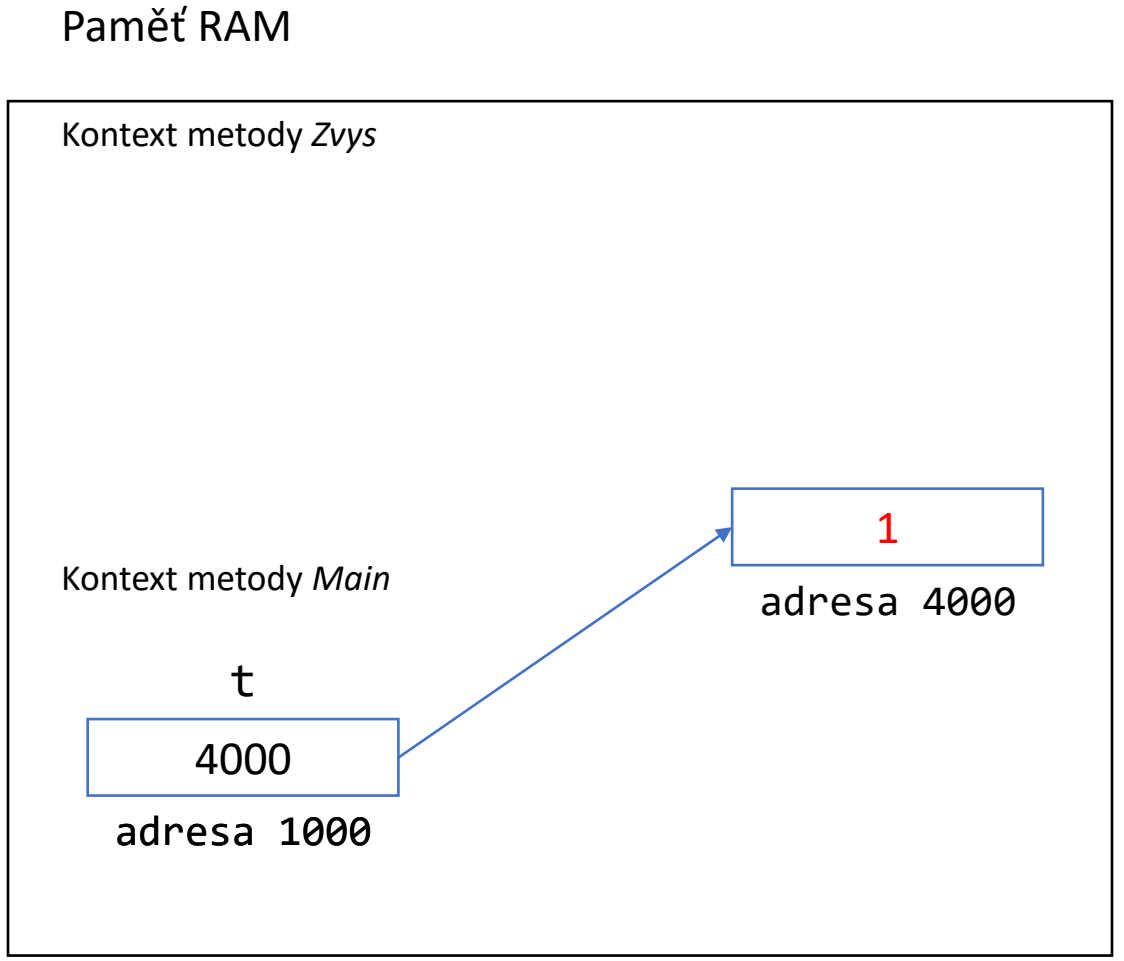
# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);

}
```



# Příklad 3 argumentu referenčního typu s klíčovým slovem *ref*

```
static void Main(string[] args)
{
    Trida t = new Trida(0);
    Zmen(ref t);
    Console.WriteLine(t.x);
}
```



# Klíčové slova out a in

- Klíčové slovo *out* [3] se chová podobně jako klíčové slovo *ref*, jen **musí mít v metodě povinně přiřazenou hodnotu** a argument předávaný s klíčovým slovem *out* nemusí být inicializovaný.
- Klíčové slovo *in* [4] se chová opět podobně jako klíčové slovo ref, ale parametr je jen pro čtení a **nemůžeme změnit jeho hodnotu**.

# Použité zdroje

[1] Variables - C# language specification | Microsoft Docs. [online]. Copyright © Microsoft 2021 [cit. 06.01.2021]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/variables

[2] ref keyword - C# Reference | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 03.12.2020]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/ref.

[3] out parameter modifier - C# Reference | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 03.12.2020]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/out-parameter-modifier

[4] in parameter modifier - C# Reference | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 03.12.2020]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/in-parameter-modifier

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

# Programování a algoritmizace

*Děkuji za pozornost*

Strategický projekt UTB ve Zlíně, reg. č. CZ.02.2.69/0.0/0.0/16_015/0002204

Ing. et Ing. Erik Král, Ph.D.
FAI, ÚPKS