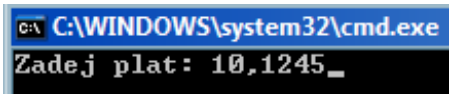
Zápis programu v jazyce C#

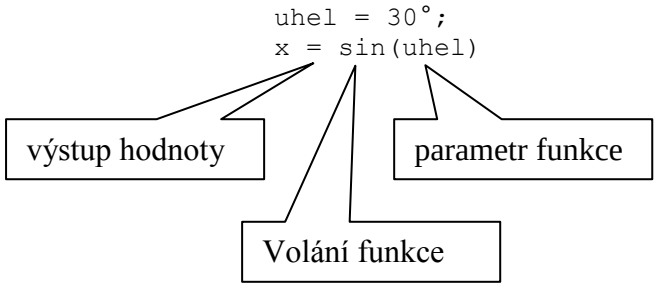
Základní syntaktická pravidla

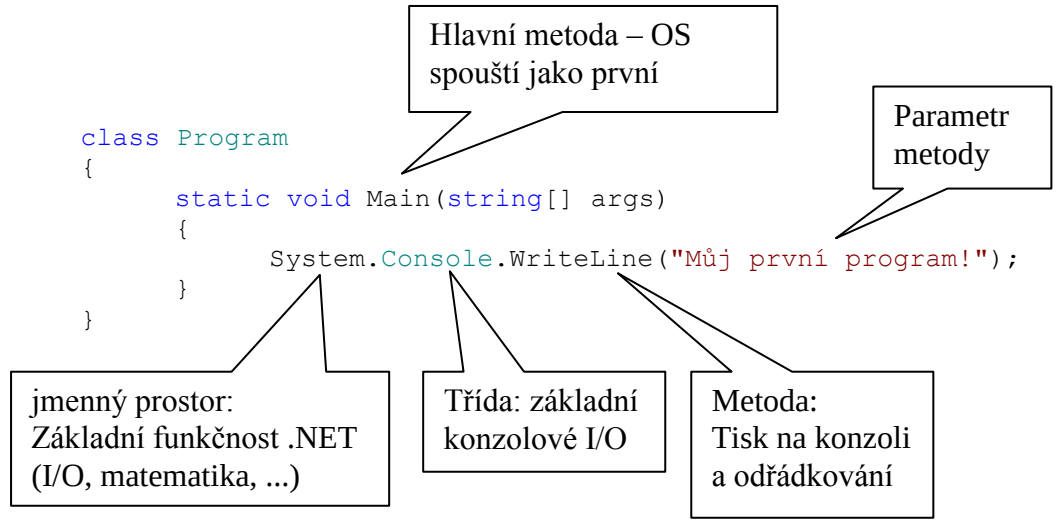
- C# = case sensitive jazyk → rozlišuje velikost písmen
- Tzv. bílé znaky (Enter, mezera, tab …) ve ZK překladač ignoruje
- každý příkaz končí `;`
- oddělovač v reálných číslech
  - zápis ZK: tečka
    ```
    plat = 10.1325;
    ```
  - spuštěný program: čárka (na CZ Windows, EN – tečka)



Rozbor Hello Word

- metoda = cca totéž co v matematice funkce





- `namespace` – jmenný prostor
- `class` – třída

- upravená varianta kódu výše

```
using System;

namespace MujJmennyProstor
{
	class Program
	{
		static void Main(string[] args)
		{
			Console.WriteLine("Můj první program!");
		}
	}
}
```

Klíčová slova

- (keywords)
- = slova se speciální významem pro překladač jazyka
- nelze využít jinak (např. k pojmenování proměnných)

abstract	event	new	struct
as	explicit	null	switch
base	extern	object	this
bool	false	operator	throw
break	finally	out	true
byte	fixed	override	try
case	float	params	typeof
catch	for	private	uint
char	foreach	protected	ulong
checked	goto	public	unchecked
class	if	readonly	unsafe
const	implicit	ref	ushort
continue	in	return	using
decimal	int	sbyte	virtual
default	interface	sealed	volatile
delegate	internal	short	void
do	is	sizeof	while
double	lock	stackalloc	else
long	static	enum	namespace
string	get	partial	set
value	where	yield	

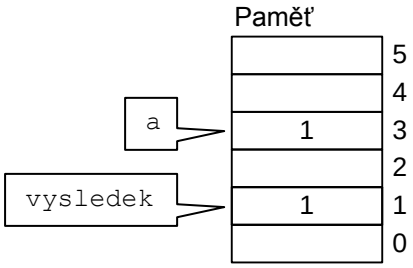
Proměnná

- (variable)
- = pojmenované místo v paměti pro uložení hodnoty (může se během práce programu měnit)

- proměnné v programu:

```
class Program
{
        static void Main(string[] args)
        {
                int a;
                int vysledek;

                a = 1;

                vysledek = a;
        }
}
```


- deklarace = zavedení proměnné
- inicializace proměnné = uložení (přiřazení) první hodnoty
- jméno proměnné = identifikátor
- proměnné v paměti počítače



Datový typ

- C# = typový jazyk, každá proměnná musí být určitého datového typu.
- datový typ specifikuje:
  - množinu hodnot, jakých může nabývat
  - množinu operací, které lze z datovým typem provést
    - aritmetické operace, např.:
      - čísla: `- +-*/` atd
      - texty: nelze násobit
    - relační operace – jak porovnávat
    - atd.
- Zatím pouze:
  - `int` celá čísla $\langle -2\times10^{9} ; 2\times10^{9} \rangle$
  - `double` reálná čísla $\langle 3{,}4\times10^{-308} ; 3{,}4\times10^{308} \rangle$ (v absolutních hodnotách)
  - `string` text (řetězec)

Inicializace proměnné

- (variable initilization)
- hodnota (neinicializované) proměnné dle typu:

```
int         0
double      0.0
string      ""
```

- **vždy předpokládáme**: hodnota proměnné není po deklaraci definována; programátor definuje ručně:
  - příklad: teplota, požadovaná počáteční hodnota 0

```
double teplota;
```

sice OK, ale...

```
double teplota = 0.0;
```

lepší

Deklarace proměnné

- (variable declaration)
- syntaxe:

```
DatovýTyp   identifikátor<, další identifikátory>;
```

- příklady

```
int mujPlat;
double stavUctu, i;
```

deklarace proměnné `mujPlat` typu `int`

je to příkaz!!!

- deklaraci s inicializací

```
int i = 0, Datum;
double pi = 3.14592;
```

- lze kdekoli v programu

C#

```
static void Main(string[] args)
{
      double a, b, c;
      // nactu a zprac. Koef.
      // ...
      double D;
      // vypoctu diskriminant
      // ...
      double koren1, koren2;
      // vypoctu koreny atd.
}
```

C, Pascal → jen na začátku

```
static void Main(string[] args)
{
      double a, b, c, D;
      double D;
      double koren1, koren2;
      // nactu a zprac. Koef.
      // vypoctu diskriminant
      // vypoctu koreny atd.
}
```

Identifikátor

- (identifier)
- = jakékoli uživatelsky vytvořené jméno v programu:
  - jména proměnných, konstant, metod, …

- syntaxe:
  - Délka neomezena
  - Libovolná kombinace písmen (malá i velká – case sensitive), číslic a podtržítek _
  - První znak vždy písmeno nebo podtržítko
- case sensitivita – odlišné identifikátory:

```
MujPlat     MUJPLAT     Mujplat
```

- Pravidla pro identifikátory proměnných:
  - (= dohoda mezi programátory, ne syntaxe!!!)
  - Používat významové identifikátory (vyjadřují účel).
    - složenina slov, první písmeno malé, další velké
    - Př.: označení mého platu

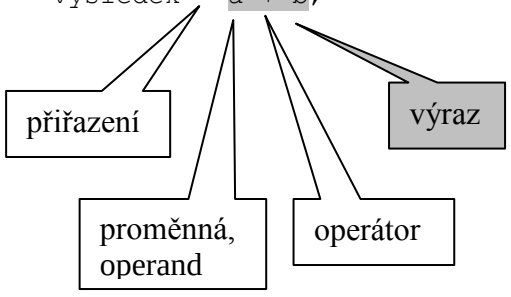
```
p           // špatně
muj_plat    // lepší, zastaralé
mujPlat     // nejlepší
```

  - Krátké identifikátory → jen v ustálených, naprosto jasných významech, např.:
    - počítadla v cyklech: `i j k`
    - Pomocné řetězce `s`
    - znaky `c`

Výpočty v programech

- součet dvou celých čísel, uložení výsledku

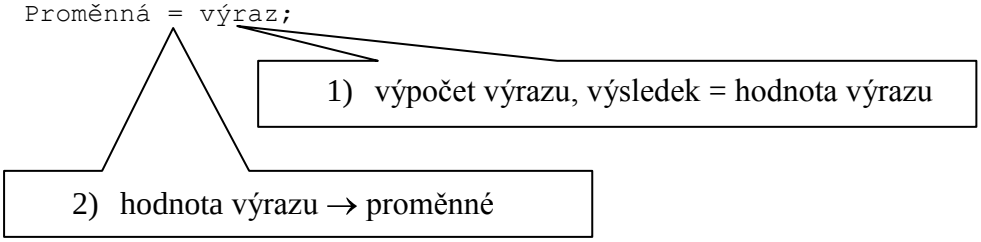
```
class Program
{
      static void Main(string[] args)
      {
            int a;
            int b, vysledek;

            vysledek = a + b;
      }
}
```



Přiřazení

- = naplnění proměnné hodnotou (uložení hodnoty do proměnné)
- Syntaxe:

```
Proměnná = výraz;
```



- Příklad:

```
int x, y;
y = 10;           // výpočet výrazu, výsledek = 10
x = 10 + y;       // x = 20
x = 30 * x - 30;  // x = 570
```

- logická chyba:

```
int x, y;
x = 10 + y;       // hodnota y?
```

- datové typy a přiřazení → levá strana ≥ pravá:

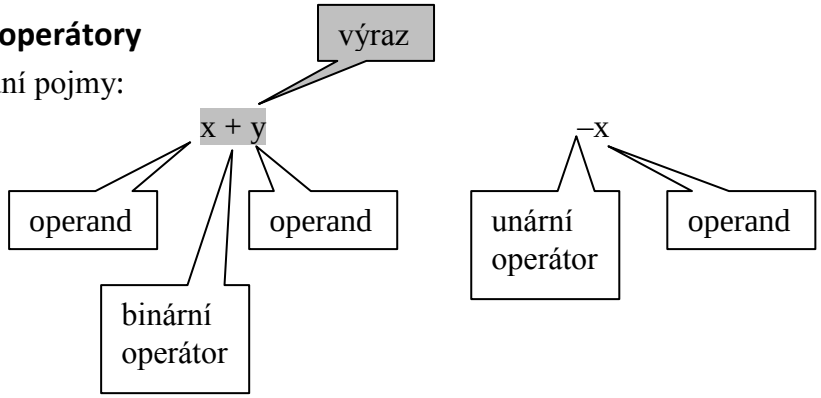
```
double x;
int y;

x = y;            // OK
y = x;            // špatně
```

- není to rovnice!!!
- složené přiřazení
  - `x = x + 4;` lze nahradit `x += 4;`
  - pro operátory

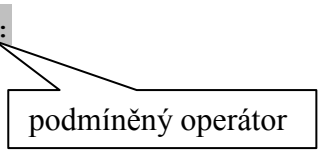
```
+      -      *      /      %      <<      >>      &      ^      |
+=     -=     *=     /=     %=     <<=     >>=     &=     ^=     |=
```

Aritmetické operátory

- základní pojmy:



- pozn.: ternární operátor → na 3 operandy,C#: `?:`



Binární

- sčítání `+`
- odčítání `-`
- násobení `*`
- dělení `/`
- dělení modulo `%`
- více o dělení:
  - reálné dělení: alespoň jeden z operandů = reálné číslo (`double`)
  - celočíselné dělení: oba operandy = celé číslo (`int`)
  - dělení modulo = zbytek po celočíselném dělení
  - příklad:

```
double d1 = 5, d2 = 2, d3;
int c1 = 5, c2 = 2, c3;

d3 = d1 / d2;           // 2.5
d3 = c1 / d2;           // 2.5, datový typ výrazu = double
c3 = c1 / c2;           // 2
c3 = c1 % c2;           // 1
```

Unární

- unární + –
  - příklad

```
int i, j = 10
i = -j;     // i = -10
```

- Inkrementační operátor `++`

```
int prom = 10, vysl;
```

1) `prom` do `vysl`, 2) `prom`+1

```
vysl = prom++;    // vysl = 10
vysl = ++prom;    // vysl = 11
```

1) `prom`+1, 2) `prom` do `vysl`

  - lze použít i samostatně

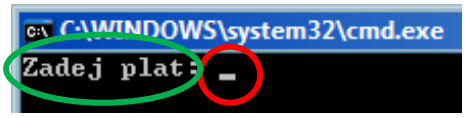
```
int prom = 10;
prom++;     // přednostně; = náhrada prom = prom + 1; prom = 11
++prom;     // prom = 12
```

  - Lze použít pouze na proměnné – `(a+b)++` nelze
  - ve složitějších výrazech

```
int prom = 10, vysl;

vysl = 3*prom++ + 10;// 3*prom+10; prom+1; vysl = 40; prom = 11
vysl = 3*++prom + 10;// prom+1; 3*prom+10; vysl = 46; prom = 12
```

Jednoduchý vstup/výstup

- (input, output)
- konzolové I/O:
  - **I**: klávesnice
  - **O**: textový výstup

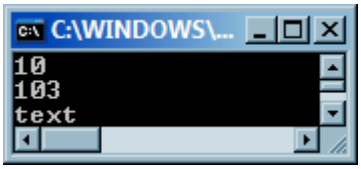


- Základní obsluha → přes třídu `Console`

Výstup

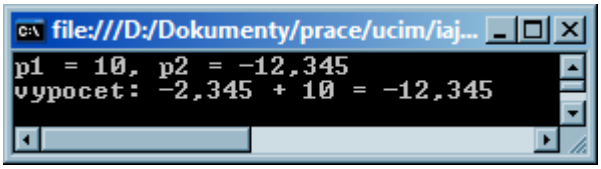
- `Console.Write(co)` → vytiskne `co`
- `Console.WriteLine(co)` → vytiskne `co` a odřádkuje
- „tisk čehokoli, co je převeditelné na text“

```
int prom = 10;
Console.WriteLine(prom);
Console.WriteLine(10 * prom + 3);
Console.WriteLine("text");
```



Formátovaný tisk

```
int p1 = 10;
double p2 = -12.345;
Console.WriteLine("p1 = {0}, p2 = {1}", p1, p2);

Console.WriteLine("vypocet: {2} + {0} = {1}", p1, p2, p1+p2);
```



Vstup

Řetězce

- princip:

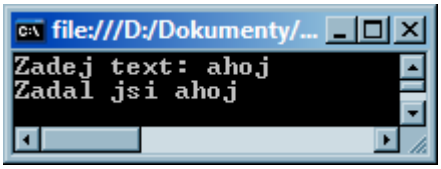
```
string text = Console.ReadLine();
```

metoda bez parametrů

vrací text na konzoli až do ENTER

- příklad:

```
string text;
Console.Write("Zadej text: ");
text = Console.ReadLine();
Console.WriteLine("Zadal jsi {0}", text);
```



čísla

- princip:
  1) načíst řetězec
  2) převedení řetězce na číslo

- příklad – celé číslo

```
Console.Write("Zadej číslo: ");
string text;
text = Console.ReadLine();
int cislo;
cislo = Int32.Parse(text);
```

Parametr typu `string`

Převede text na číslo typu `int`

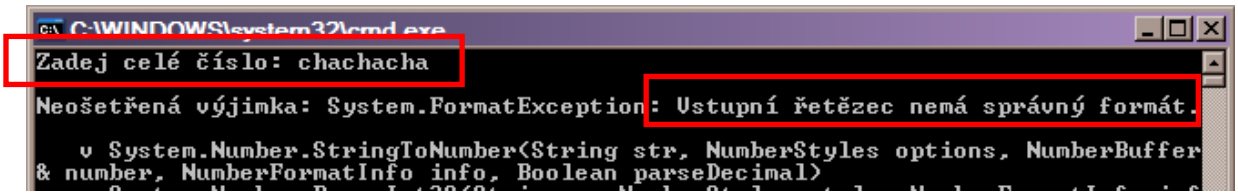
- lepší provedení I

```
int cislo;
cislo = Int32.Parse(Console.ReadLine());
```

- lepší provedení II → deklarace s inicializací z klávesnice

```
int cislo = Int32.Parse(Console.ReadLine());
double x = Double.Parse(Console.ReadLine());
```

- pokud se převod nepodaří → vyhození výjimky

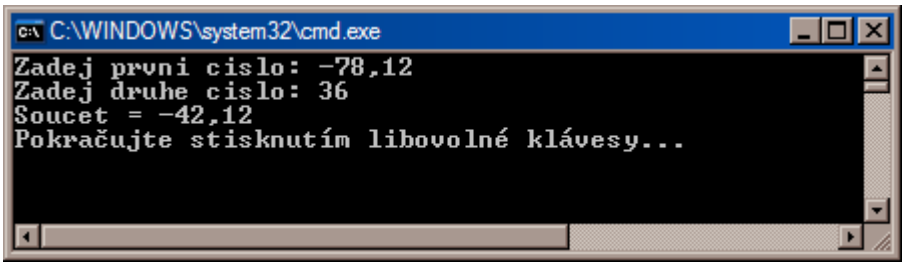


- řešení (později):
  1) ošetřit výjimku
  2) metoda `Int32.TryParse()`

Příklad kompletního programu

- Napište program, který sečte dvě čísla a jejich součet vypíše na obrazovku

```
static void Main(string[] args)
{
      Console.Write("Zadej prvni cislo: ");
      double cislo1 = Double.Parse(Console.ReadLine());
      Console.Write("Zadej druhe cislo: ");
      double cislo2 = Double.Parse(Console.ReadLine());
      // vypocet
      double soucet = cislo1 + cislo2;
      Console.WriteLine("Soucet = {0}", soucet);
}
```



Typový jazyk

- C# = typový jazyk – datový typ proměnné nelze změnit
- Příklady I

```
int i;
i = 10;          // OK
i = 0.325;       // nelze
i = "ahoj";      // nelze
```

- Příklady II

```
string s;
s = "10";        // OK
s = 10;          // nelze
```