Proměnné, operátory a výrazy

Jiří Zacpal

KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

KMI/ZP3CS – Základy programování 3 (C#)

Základní pojmy

- **Příkazy** – povel, který provede nějakou akci:

```
Console.WriteLine("Ahoj světe!" + i + 4);
```

- **Metody** – vznikají kombinací příkazů (+- funkce).
- **Identifikátory**
  - velká, malá písmena, čísla, podtržítko,
  - musí začínat písmenem nebo podtržítkem.
- **Proměnná**
  - pojmenování:
    - měl by začínat malým písmenem,
    - u víceslovných názvů by každé další slovo mělo začínat velkým písmenem (velbloudí zápis):

```
mojePromenna
```

  - deklarace proměnné:

```
int i = 12;
```

  - před použitím je nutné proměnné přiřadit hodnotu

Proměnné a datové typy

Typový systém

- Existují dva základní **typové systémy**:
  - **Dynamický typový systém** nás plně odstiňuje od toho, že proměnná má vůbec nějaký datový typ. Ona ho samozřejmě vnitřně má, ale jazyk to nedává najevo. Dynamické typování jde mnohdy tak daleko, že proměnné nemusíme ani deklarovat, jakmile do nějaké proměnné něco uložíme a jazyk zjistí, že nebyla nikdy deklarována, sám ji založí. Do té samé proměnné můžeme ukládat text, potom objekt uživatele a potom desetinné číslo.
  - **Statický typový systém** naopak striktně vyžaduje definovat typ proměnné a tento typ je dále neměnný. Jakmile proměnnou jednou deklarujeme, není možné její datový typ změnit.
- **C# je staticky typovaný jazyk => ** všechny proměnné musíme nejprve deklarovat s jejich datovým typem.
- Nevýhodou je, že díky deklaracím je zdrojový kód poněkud objemnější a vývoj pomalejší.
- Výhodou je, že nám kompiler před spuštěním zkontroluje, zda všechny datové typy sedí.

Hodnotové datové typy

Typ	Popis	Velikost	Rozsah	Příklad
`int`	celá čísla	32	-2^{31} až $2^{31}-1$	`int počet;` `počet=42;`
`long`	celá čísla	64	-2^{63} až $2^{63}-1$	`long dlouho;` `dlouho=42L;`
`float`	desetinná čísla	32	$+-1{,}5\times10^{45}$ až $+-3{,}4\times10^{38}$	`float des;` `des=0.42F;`
`double`	desetinná čísla	64	$+-5\times10^{324}$ až $+-1{,}7\times10^{308}$	`double des;` `des=0.42;`
`decimal`	peněžní hodnoty	128	28 významných číslic	`decimal mince;` `mince=0.42M;`
`char`	jeden znak	16	0 až $2^{16}-1$	`char znak;` `znak='a';`
`bool`	logická hodnota	8	true nebo false	`bool log;` `log=false;`

Referenční datový typ String

- slouží k uložení řetězce,
- vlastnost **Length**
  - vrací celé číslo, které představuje počet znaků v řetězci.
- metody:

StartsWith() EndsWith() a Contains()

- určí, zda řetězec začíná, končí nebo zda obsahuje určitý podřetězec (substring).
  - příklad:

```
string s = "Krokonosohroch";
Console.WriteLine(s.StartsWith("krok"));
Console.WriteLine(s.EndsWith("hroch"));
Console.WriteLine(s.Contains("nos"));
Console.WriteLine(s.Contains("roh"));
```

ToUpper() a ToLower()

- převedou string na velká (malá) písmena.

Referenční datový typ String

Trim(), TrimStart() a StrimEnd()

- odstraní bílé znaky (mezery) okolo, před, za řetězcem.

Replace()

- nahrazení části řetězce jiným.
  - příklad:

```
string s = "Java je nejlepší!";
s = s.Replace("Java", "C#");
Console.WriteLine(s);
```

Format()

- umožňuje vkládat do samotného textového řetězce zástupné značky.
  - příklad:

```
int a = 10;
int b = 20;
int c = a + b;
string s = string.Format("Když sečteme {0} a {1}, dostaneme {2}", a, b, c);
Console.WriteLine(s);
```

Aritmetické operátory

- + , - , * , / , % (modulo)
- lze je použít s číselnými typy
- pro typ string lze použít + pro spojení řetězců
- typy operátorů
- vlastnosti:
  - asociativita
  - priorita
  - arita

Konverze (parsování) mezi typy

Třída `Convert`

```
Convert.metoda(co);
```

- umožňuje konvertovat hodnoty mezi datovými typy
- příklad:

```
a = Convert.ToInt32(„82“);
```

Metoda `Parse`

```
typ.Parse(co);
```

- konvertuje (nejčastěji string) do jiného datového typu,
- v případě, že to nejde, vrátí výjimku:
  - ArgumentNullException - *co* je null,
  - FormatException – *co* není možné konvertovat (špatný formát),
  - OverflowException - *co* reprezentuje číslo menší než MinValue nebo větší než MaxValue.
- příklad:

```
a = int.Parse(„82“); (v a bude 82)
a = int.Parse(„abc“); (vrátí výjimku Format Exception)
a = int.Parse(„-2147483649“); (vrátí výjimku OverflowException)
```

Implicitně typované lokální proměnné

```
var identifikator = hodnota;
```

- typ proměnné je odvozen z typu inicializační hodnoty,
- proměnnou je nutné inicializovat,
- příklad:

```
var a = 8;
var b = „Jedna“;
var c; //Chyba
```

Metody

Metody

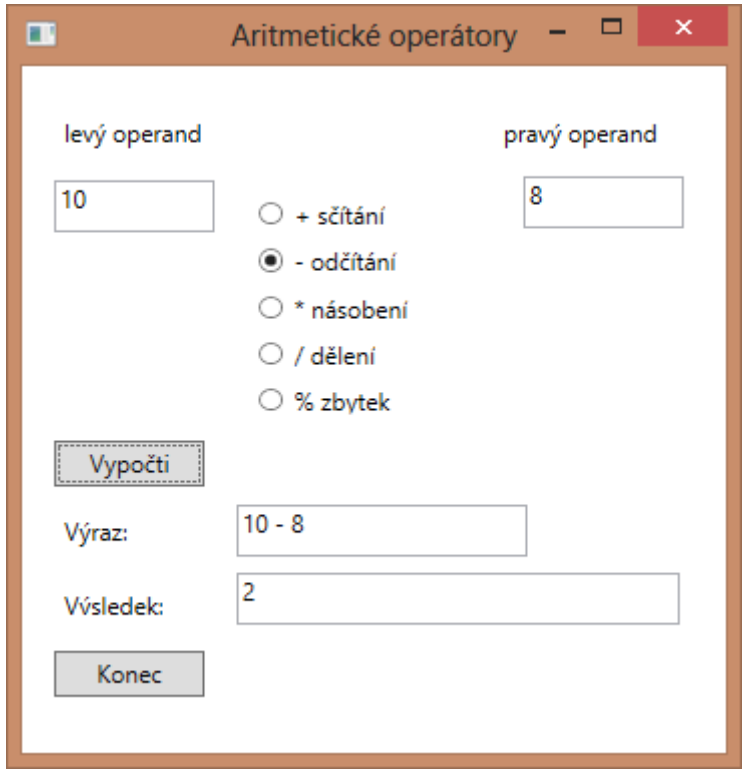
- metoda = pojmenovaná posloupnost příkazů (podobně jako funkce)
- deklarace:

```
typ JmenoMetody(parametry)
        {
            //tělo metody
            return NavratovaHodnota;
        }
```

- volání:

```
vysledek=JmenoMetody(hodnoty parametru)
```

Příklad 1


Aritmetické operátory
levý operand
pravý operand
10
8
+ sčítání
- odčítání
* násobení
/ dělení
% zbytek
Vypočti
Výraz:
10 - 8
Výsledek:
2
Konec

Příklad 1

- Otevřete projekt **zp3cs_2_datove_typy_1**.
- Do ovladače události pro klepnutí na tlačítko doplňte tento kód:

```
private void calculateClick(object sender, RoutedEventArgs e)
    {
        int lhs = int.Parse(lhsOperand.Text);
        int rhs = int.Parse(rhsOperand.Text);
        if ((bool)addition.IsChecked)
            addValues(lhs,rhs);
        else if ((bool)subtraction.IsChecked)
            subtractValues(lhs,rhs);
        else if ((bool)multiplication.IsChecked)
            multiplyValues(lhs,rhs);
        else if ((bool)division.IsChecked)
            divideValues(lhs,rhs);
        else if ((bool)remainder.IsChecked)
            remainderValues(lhs,rhs);
    }
```

Příklad 1

- Vytvořte metodu addValues:

```
private void addValues(int lhs, int rhs)
      {
          int outcome;
          outcome = lhs + rhs;
          expression.Text = lhsOperand.Text + " + " + rhsOperand.Text;
          result.Text = outcome.ToString();
      }
```

- Vytvořte metody pro další matematické operace.
- Vytvořte ovladač pro tlačítko Konec:

```
   private void quitClick(object sender, RoutedEventArgs e)
          {
              this.Close();
          }
```

Obor platnosti

= oblast programu, ve které je proměnná použitelná,

- lokální obor platnosti
  - proměnné definované v složeném příkazu:
    - tělo metody,
    - cyklus,
    - …
- třídní obor platnosti
  - proměnné definované v rámci těla třídy.
  - příklad:

```
class Trida
{
  int a;
  void Metoda()
  {
        a=8;
  }
}
```

Přetěžování metod

= jsou-li dva identifikátory stejné (ve stejném oboru platnosti)

- metody se musí lišit počtem nebo typem parametrů,
- nelze napsat dvě metody, které se liší návratovou hodnotou.
- příklad:

```
private void zapisHonorar(double p)
 {
     Console.WriteLine("Plat konzultanta je: {0}", p * 1.1);
 }

private void zapisHonorar(int p)
 {
     Console.WriteLine("Plat konzultanta je: {0}", p * 2);
 }
```

Nepovinné parametry

- některé parametry jsou nepovinné
- definice:

```
typ Jmeno(PevnePar, typ vol1=h, typ vol2=h)
```

- příklad:

```
void volitelne(int prvni, double druhy=0.0, string treti=„Ahoj“) {...};

volitelne(99,123.45,“Světe“);
volitelne(100,54.321);
volitelne(treti:“Světe“,prvni:100);
```

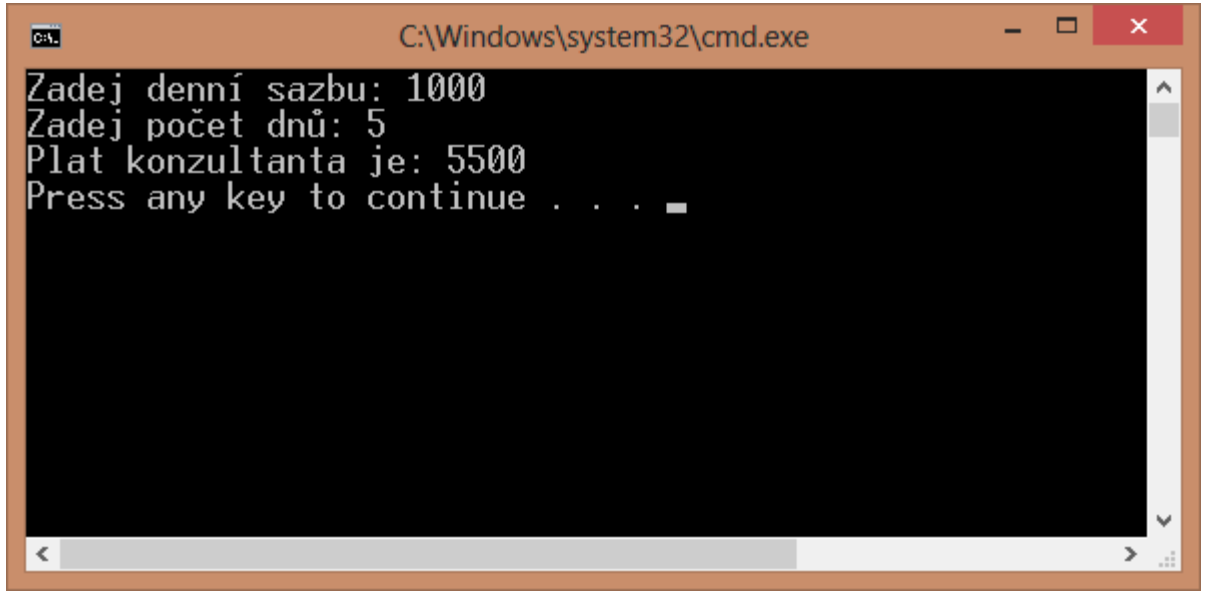
Přetížené metody s volitelnými parametry

- příklad:

```
void volitelne(int prvni, double druhy=0.0, string treti=„Ahoj“){...};

void volitelne(int prvni, double druhy=0.0, string treti=„Ahoj“, int ctvrty=100)
{...};



volitelne(99,123.45,“Světe“);
        1. metoda
volitelne(100,ctvrty:54);
        2. metoda
volitelne(1,2.5);
        nejednoznačné volání metody
```

Příklad 2

Napíšeme program pro výpočet platu konzultanta:



Příklad 2

- Vytvořte si novou konzolovou aplikaci.
- Do metody `main` doplňte příkazy:

```
double dailyRate = nactiDouble("Zadej denní sazbu: ");
int noOfDays = nactiInt("Zadej počet dnů: ");
zapisHonorar(spocitejHonorar(dailyRate, noOfDays));
```

- Vygenerujte metodu `nactiDouble` a doplňte kód:

```
Console.Write(p);
string line = Console.ReadLine();
return double.Parse(line);
```

- Vygenerujte metodu `nactiInt` a doplňte kód.
- Vygenerujte metodu `spocitejHonorar` a doplňte kód:

```
return dailyRate * noOfDays;
```

- Vygenerujte metodu `zapisHonorar` a doplňte kód:

```
Console.WriteLine("Plat konzultanta je: {0}", p * 1.1);
```

Příklad 2

- Do metody main doplňte tento kód:

```
zapisHonorar(spocitejHonorar());
zapisHonorar(spocitejHonorar(dailyRate: 100));
zapisHonorar((int)10);
```

- Doplňte přetíženou metodu `zapisHonorar` pro tisk celého čísla:
- Doplňte přetížené metody `spocitejHonorar` pro různý počet parametrů.
- Program vyzkoušejte.