

Jazyk C# 1

8. seminář

Radek Janoščík



**KATEDRA
INFORMATIKY**

UNIVERZITA PALACKÉHO V OLOMOUCI

- Enum a bitové masky
- Rozličnost řešení pro datum a čas
- Výjimky
- Vlastní výjimka a datové sloty

- Zatím jsme uměli jen výstup
- Samozřejmě je možný i vstup + manipulace se vzhledem
- Načtení řetězce od uživatele: `string inputA = Console.ReadLine();`
- Musíme počítat s tím, že uživatel nemusí zadat validní data (parsování čísel, ...)

```
1 static void readNumber(out int a, string text) {  
2     Console.Write(text);  
3     string inputA = Console.ReadLine();  
4     try {  
5         a = int.Parse(inputA);  
6     } catch (Exception e) {  
7         readNumber(out a, text);  
8     }  
9 }
```

- Můžeme měnit barvu pozadí i barvu písma

```
1 Console.BackgroundColor = ConsoleColor.Green;  
2 Console.ForegroundColor = ConsoleColor.Red;
```

- Resetování barev `Console.ResetColor()`;

- Vymazání: `Console.Clear()`;

- Nastavení velikosti (počet sloupců a řádků) `Console.SetWindowSize(20, 8)`;

- Změna titulku okna `Console.Title = "Moje uzasna aplikace"`;

- Reakce na stisk nějaké klávesy:

```
1 ConsoleKeyInfo info = Console.ReadKey();  
2 if (info.Key == System.ConsoleKey.Backspace) {  
3     Console.WriteLine("Zmackl backspace");  
4 }
```

- Uživatelsky definovaný datový typ, který obsahuje datové sloty a metody s nimi operujícími
- Jedná se o hodnotový datový typ ($\Rightarrow$ uloženy na zásobníku, kopie do parametrů metod, ...)
- „Odlehčené třídy“
- Definice podobná jako definice třídy:

```
1 public struct StorageItem {  
2     public int Length;  
3     public int Width;  
4     public int Height;  
5     public string Name;  
6  
7     public int Volume() { return Length * Width * Height; }  
8     public int Area() { return Length * Width; }  
9 }
```

■ Vytvoření struktury pomocí operátoru `new()` ;

```
1 StorageItem si = new StorageItem();  
2 si.Length = 32;
```

■ Nebo „rovnou“, musíme ale inicializovat všechny sloty (u tříd by nešlo)

```
1 StorageItem si;  
2 si.Length = 32;  
3 si.Name = "asdasd";  
4 si.Width = 12;  
5 si.Height = 30;
```

■ Jak je na tom přiřazení?

```
1 StorageItem dalsi = si;  
2 dalsi.Length = 5;  
3 Console.Write(si.Length); // co bude výsledkem?
```

- Mezi strukturami není dědičnost (implicitně dědí z `System.ValueType`)
- Implicitní bezparametrový konstruktor – nastavuje defaultní hodnoty
 - Lze však definovat své konstruktory (i s parametry)
- Struktury jsou hodnotový datový typ (uloženy na zásobníku)
- Implicitní metoda `Equals()` porovnává „po složkách“
- Můžeme definovat, jak se sloty naskládají do paměti
- Můžeme vynutit, aby daná struktura byla uložena na zásobníku (`ref`)

- Mechanismus umožňující používat operátor `[]` na naší třídě
- Indexovat nemusíme pouze pomocí čísel, ale pomocí libovolného typu
- Definice:

```
1 pristupnost navratovyTyp this [typIndexu identifikator] {  
2 pristupnost get { jak vybrat prvek }  
3 pristupnost set { jak uložit prvek }  
4 }
```

- Mohou být i více-dimenzionální

- Chtěl bych mít datovou třídu pro ukládání *key-value* párů

```
1 public class Dictionary<K, V> {  
2     private List<K> keys;  
3     private List<V> values;  
4     public Dictionary() {  
5         keys = new List<K>();  
6         values = new List<V>();  
7     }  
8     public V this[K key] {  
9         get { int index = keys.IndexOf(key);  
10            if (index > -1) return values[index];  
11            else return default(V); }  
12         set { if (keys.IndexOf(key) == -1)  
13             { keys.Add(key); values.Add(value); } }  
14     }  
15 }
```

- Chtěl bych mít datovou třídu pro ukládání *key-value* párů

```
1 public class Dictionary<K, V> {
2     private List<K> keys;
3     private List<V> values;
4     public Dictionary() {
5         keys = new List<K>();
6         values = new List<V>();
7     }
8     public V this[K key] {
9         get { int index = keys.IndexOf(key);
10             if (index > -1) return values[index];
11             else return default(V); }
12         set { if (keys.IndexOf(key) == -1)
13             { keys.Add(key); values.Add(value); }}
14     }
15 }
```

- Na naši třídu půjde použít operátor []

```
1 Dictionary<string, int> d = new Dictionary<string, int>();  
2 d["blabla"] = 33;  
3 d["ccc"] = 25;  
4 Console.WriteLine(d["blabla"]);
```

- Zkusme si třeba do vaší `IntMatrix` dodat 2D indexer

- Jak jsem na začátku semestru avizoval bude větší programátorský úkol
- Implementujte konzolovou deskovou hru „Česká dáma“
 - Varianta hráč proti hráči (bez počítačového hráče)
- Pravidla <http://www.deskovehry.info/pravidla/ceska-dama.htm>
 - Bez vícenásobných skoků
- Na úkol je „14 dnů“, bude za 10 bodů
 - Příští hodinu proběhne normálně výuka s malým úkolem (2 souběžně)
 - 5. 12. budou povinné konzultace (v kanceláři)
 - Udělám rozpis po 10 minutách
 - Další konzultace v konzultačních hodinách

- Jak jsem na začátku semestru avizoval bude větší programátorský úkol
- Implementujte konzolovou deskovou hru „Česká dáma“
 - Varianta hráč proti hráči (bez počítačového hráče)
- Pravidla <http://www.deskovehry.info/pravidla/ceska-dama.htm>
 - Bez vícenásobných skoků
- Na úkol je „14 dnů“, bude za 10 bodů
 - Příští hodinu proběhne normálně výuka s malým úkolem (2 souběžně)
 - 5. 12. budou povinné konzultace (v kanceláři)
 - Udělám rozpis po 10 minutách
 - Další konzultace v konzultačních hodinách



- Vizualizace herní desky a zobrazení možných tahů (2 body)
- Uživatelsky přívětivé ovládání (2 body)
- Korektní implementace pohybu „obyčejných kamenů“ (2 body)
- Korektní implementace pohybu dámy (2 body)
- Celková architektura řešení (2 body)

- Zkuste nemíchat logiku hry a zobrazení
- Zamyslete se nad datovou reprezentací šachovnice, figur, tahů
- Doporučuji si problém rozdělit na malé funkční celky, např.
 - Načtení tahů (jaký hráč, odkud kam)
 - Výpočet možných tahů daného pole, hráče
 - Provedení tahu (ověření soulad s pravidly)
 - Vykreslování desky
 - Ukončovací podmínky hry
 - Načtení dalšího tahu
 - ...
- Hratelnosti **velmi** pomůže způsob zadávání znaků a přehlednost šachovnice