# Zpravodaj moderníProgramování 1/2013: Datová struktura Dictionary

*Obtížnost: středně pokročilí*

## Kdy nestačí pole (seznam)

Chcete-li jednotným způsobem hromadně zpracovat větší množství dat, uložíte je do pole. Nebo do seznamu, když předem nelze určit počet prvků. Jak pole, tak seznam jsou indexovány souvislou řadou celých čísel začínající od nuly.

Rozeberme si předchozí větu:

- Zdůraznil jsem „souvislou“ – pokud by indexy netvořily souvislou řadu, pole (seznam) nebude vhodné řešení. Samozřejmě, pokud bude v řadě sem tam díra, tak se zas tolik neděje. Pokud by však rozestupy mezi využívanými indexy měly být v řádech tisíců, asi by bylo velké plýtvání vytvářet pole, ze kterého využijeme jedno promile nebo méně.
- Zdůraznil jsem „čísel“ – pole (seznamy) jsou indexovány čísly. Chcete-li indexovat něčím jiným, pole (seznam) vám nepomůže.
- Nezdůraznil jsem „od nuly“ – případ, kdy má být první index jiný než nula, s polem (seznamem) řešitelný je. Před každým přístupem do pole prostě k indexu něco přičtete nebo odečtete. Je tam sice riziko potenciální chyby programátora (to se dá omezit vytvořením třídy s vlastním indexerem), ale v zásadě proč ne.

## Dictionary neboli slovník

Pokud řada indexů není souvislá, nebo potřebujete indexovat např. řetězci, přichází na řadu typ `Dictionary`. V české lokalizaci Visual Studia 2012 se používá překlad ***slovník***, budu se tedy i já tohoto termínu držet. Na jiných platformách se můžete setkat s označením ***asociativní pole*** nebo ***hešovací tabulka*** – ve všech případech jde v zásadě o totéž.

Co si pod slovníkem představit, ukazuje následující obrázek:

| `["Marian"]` | `["Adrian"]` | `["Kristian"]` | `["Kilian"]` | `["Julian"]` |
|---|---|---|---|---|
| 5.2 | 4.1 | 8.8 | 1.0 | 4.8 |

Máte před sebou v podstatě pole, akorát že indexované řetězci. Obrázek může představovat např. slovník nazvaný `body`, který zachycuje body hráčů nějaké hry. Chcete-li zjistit/změnit Adrianovy body, nemusíte pracně hledat, který chlíveček patří Adrianovy, jednoduše napíšete `body["Adrian"]`. To je právě výhoda slovníku.

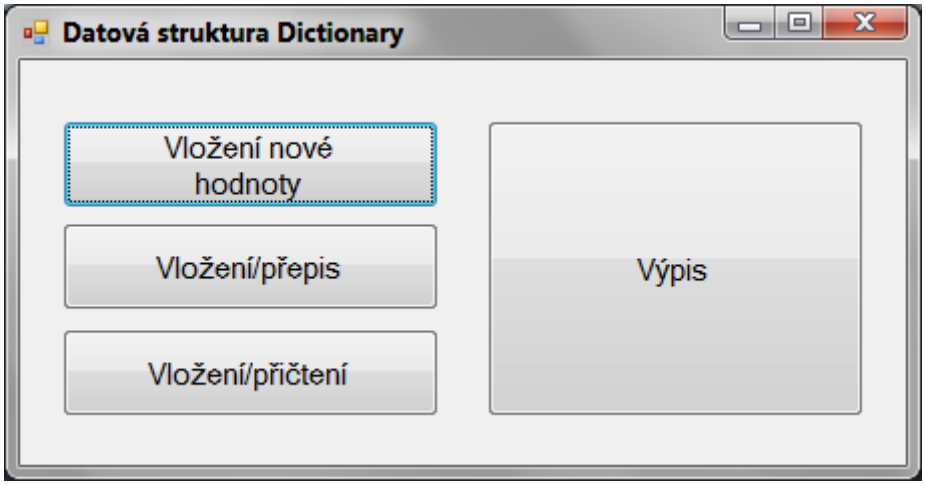
Alternativně si můžete slovník představovat jako seznam dvojic klíč-hodnota:

Marian – 5.2
Adrian – 4.1
Kristian – 8.8
Kilian – 1.0
Julian – 4.8

Poznamenám, že jde o neuspořádaný seznam, zapomeňte na to, že jste Kristiana viděli na třetí pozici. Přes tu trojku (nebo dvojku, pokud bychom začínali od nuly) se na jeho body nedostanete, slovník není pole.

## Technika práce se slovníkem

Ukažme si prakticky, jak se slovníkem pracovat. Uživatelské rozhraní našeho programu bude vypadat takto:




Pojďme se podívat na jednotlivé typické operace, které se se slovníky provádějí.

Vytvoření slovníku: V našem programu si slovník nazvaný `body` vytvoříme jako členskou proměnnou okna:

```
Dictionary<string, double> body = new Dictionary<string, double>();
```

Uvedeným zápisem vytváříme (zprvu prázdný) slovník, jehož klíče budou řetězce a jehož hodnoty desetinná čísla.

Vložení nové hodnoty: Pokud víme, že hodnotu s příslušným klíčem ve slovníku ještě nemáme, můžeme ji vložit voláním jeho metody `Add`:

```
private void tlačítkoVloženíNovéHodnoty_Click(object sender, EventArgs e)
{
    body.Add("Arnošt", 5.3);
    body.Add("Monika", 7.2);
}
```

V případě, že položka s daným klíčem již ve slovníku existuje, vznikne běhová chyba.

Jiné vložení nebo přepis: Položka se dá do slovníku přidat i jiným způsobem - jako byste přiřazovali do pole:

```
private void tlačítkoVloženíPřepis_Click(object sender, EventArgs e)
{
    body["Arnošt"] = 8.1;
}
```

Pokud hodnota s uvedeným klíčem ve slovníku již existuje, je přepsána stejně jako jakékoli jiné přiřazení do proměnné přepíše dosavadní hodnotu.

Pokud neexistuje, vytvoří se, jako kdybyste volali `Add`.

Vložení nebo přičtení: Dejme tomu, že chceme nějakému hráči přidat další body. Pokud už nějaké má, tak příslušnou položku ve slovníku navýšit, pokud je nemá, tak novou položku založit. To je možná nejtypičtější úkon:

```
private void tlačítkoVloženíPřičtení_Click(object sender, EventArgs e)
{
   VložNeboPřičti("Monika" , 1.9);
   VložNeboPřičti("Adriana", 6.6);
}

private void VložNeboPřičti(string jméno, double bodyČlověka)
{
   if (body.ContainsKey(jméno))
       body[jméno] += bodyČlověka;
   else
       body.Add(jméno, bodyČlověka);
}
```

Kdybychom existenci hodnoty s daným klíčem netestovali (volání `ContainsKey`), tak by nám to `+=` na Adrianě spadlo.

Zpracování všech položek: Jako u všech datových kolekcí potřebujeme vědět, jak zpracovat všechny položky. V našem programu to dělá tlačítko „Výpis“:

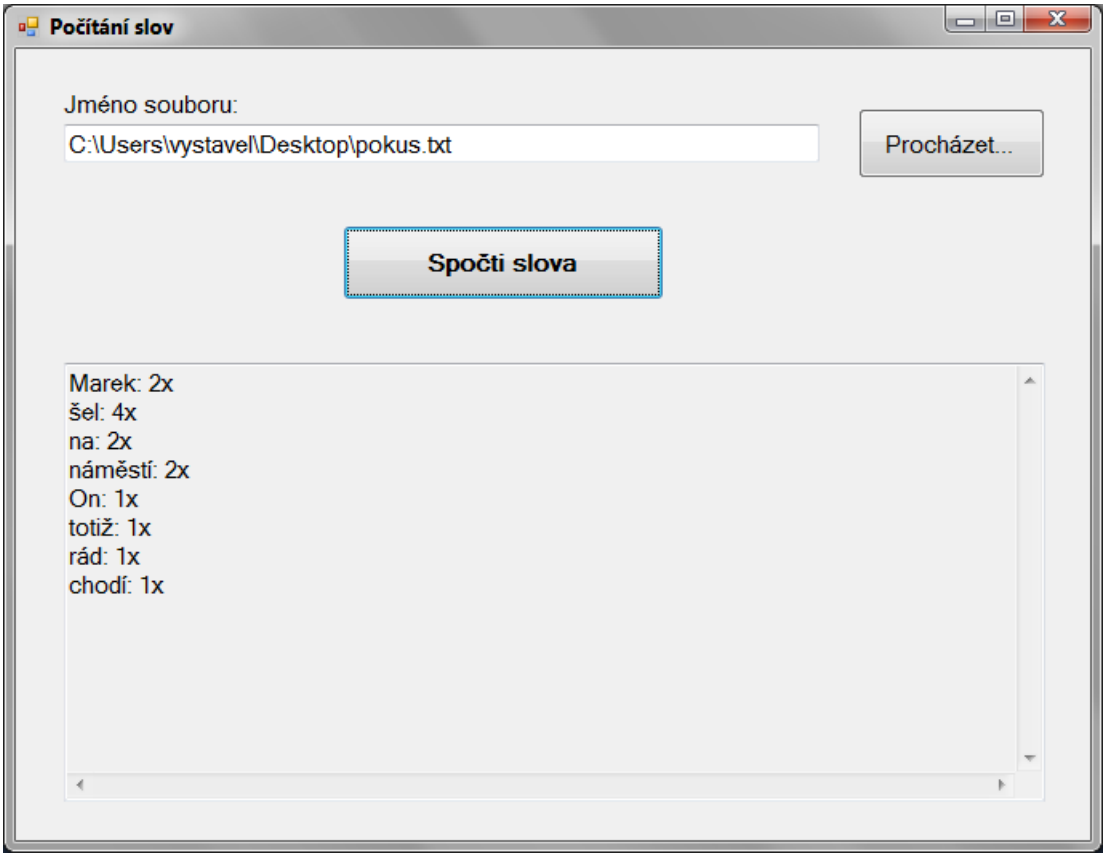
```
private void tlačítkoVýpis_Click(object sender, EventArgs e)
{
   string zpráva = "";
   foreach (var hodnota in body)
   {
       zpráva += hodnota.Key + ": " + hodnota.Value.ToString() +
                 Environment.NewLine;
   }
   MessageBox.Show(zpráva);
}
```

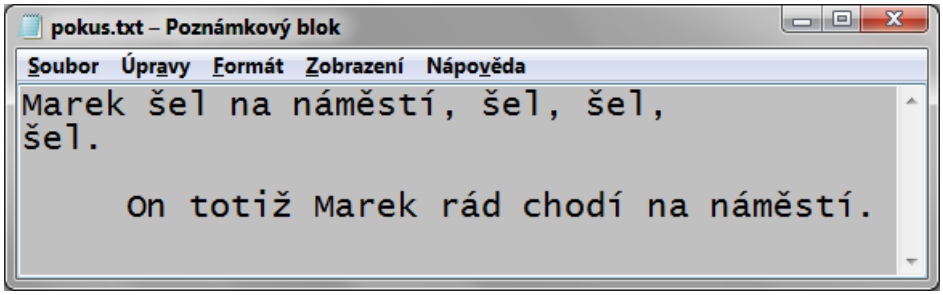
## Ukázka aplikace slovníku

V předchozí ukázce jsme si vysvětlili, jak technicky provést všechny hlavní operace se slovníkem. Nyní si ukážeme ještě jeden program. Ten na vstupu převezme od uživatele textový soubor a spočítá, kolikrát se v něm jaké slovo vyskytuje.

Principiálně jde o podobnou úlohu jako kontrola kostky z učebnice pro středně pokročilé, kde jsme sledovali, kolikrát padne jaké náhodné číslo. Prostě potřebujeme mnoho počitadel. U kostky jsme měli pole a indexovali jsme ho hozenými čísly. Nyní budeme mít slovník a budeme jej indexovat slovy, která v souboru nalezneme.

Obrázky ukazují uživatelské rozhraní (do něj patří ještě jeden `OpenFileDialog`) i testovací data:




No a samozřejmě kód:

```
public partial class Form1 : Form
{
 public Form1()
 {
   InitializeComponent();
 }

 private void tlačítkoProcházet_Click(object sender, EventArgs e)
 {
   if (oknoVýběruSouboru.ShowDialog() ==
       System.Windows.Forms.DialogResult.OK)
     poleJménoSouboru.Text = oknoVýběruSouboru.FileName;
 }
```

```
 private void tlačítkoSpočtiSlova_Click(object sender, EventArgs e)
 {
   // Připrav počitadla
   Dictionary<string, int> počty = new Dictionary<string, int>();

   // Čti soubor
   StreamReader soubor =
         new StreamReader(poleJménoSouboru.Text, Encoding.Default);
   string řádek;
   while ((řádek = soubor.ReadLine()) != null)
   {
     // Rozbij řádek na slova
     string[] slova = řádek.Split(
         new char[] { ' ', '.', ',', ';', '"' },
         StringSplitOptions.RemoveEmptyEntries);

     // Zpracuj všechna slova na řádku
     foreach (string slovo in slova)
     {
       if (počty.ContainsKey(slovo))
         počty[slovo]++;
       else
         počty.Add(slovo, 1);
     }
   }
   soubor.Close();

   // Zobraz výsledky
   string zpráva = "";
   foreach (var hodnota in počty)
   {
     zpráva += hodnota.Key + ": " + hodnota.Value.ToString() + "x" +
               Environment.NewLine;
   }
   poleVýsledky.Text = zpráva;
 }
}
```

## Závěrečné poznámky

- Ve webových aplikacích ASP.NET se setkáte s řadou kolekcí podobných slovníku - `ViewState`, `Session`, `Cache`, ...
- Pokud byste potřebovali výstup seřazený, bylo by možno použít třídu `SortedDictionary`, nebo (ještě efektivněji) výstup seřadit pomocí LINQ to Objects. Pro zajímavost, technologii LINQ to Objects se věnuje 11. kapitola učebnice pro pokročilé.

*Radek Vystavěl, 29. ledna 2013*

*Pokud Vám Zpravodaje moderníProgramování připadají užitečné, doporučte jejich odběr svým známým. Mohou se přihlásit na webu www.moderniProgramovani.cz.*