Seminář PRG
14. hodina - 6.12.2024

Gymnázium Voděradská 2024/2025
Jan Borecký

Dnešní téma

- String a všechno kolem něj
  - StringBuilder
  - Kódování znaků
  - Textové soubory

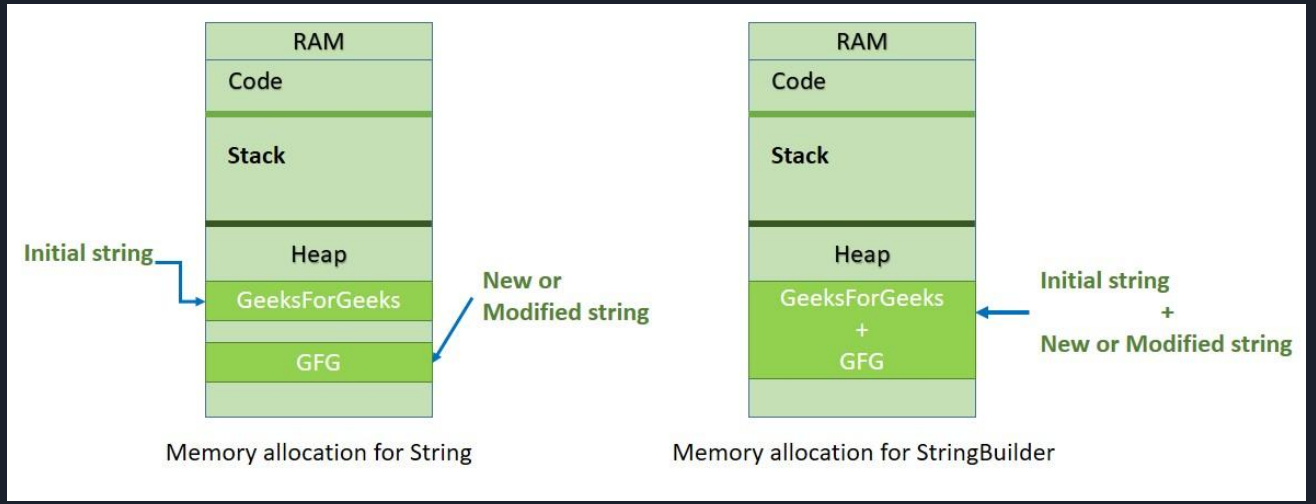
Co už známe

- Reprezentace 0 a více Unicode znaků
- Referenční datový typ (dosavadní byly hodnotové)
  - String je pod pokličkou pole charů
- Zápis ve dvojitých uvozovkách: *string text = “Hello!”;*
- Konverze jiných datových typů do stringu
  - var.ToString()
  - často nemusíme dělat nic
- Konverze stringu do jiných datových typů
  - int.Parse() (obdobně u dalších)
  - int.TryParse()
  - Convert.ToInt32()

StringBuilder

Rozdíly Stringu a StringBuilderu

- String nelze upravovat, po úpravě se vždy vytvoří úplně nový a starý se smaže
- StringBuilder nezabírá tolik paměti, protože se netvoří úplně nový po každé úpravě



Rozdíly Stringu a StringBuilderu

- Kvůli šetření paměti běží rychleji např. při spojování mnoha stringů v cyklu

- Pro využití potřebujeme přidat do souboru using System.Text

- StringBuilder se instancuje jako např. třídy
  - *StringBuilder myStringBuilder = new StringBuilder("Hello World!");*

- StringBuilderu můžeme specifikovat kapacitu (to neznamená délku)
  - Dokud tato kapacita není naplněna, StringBuilder nedělá nic i když se v něm upravuje. Jakmile se jí dosáhne, SB se v paměti realokuje a zdvojnásobí si kapacitu

Rozdíly Stringu a StringBuilderu

- Do StringBuilderu přidáváme další stringy pomocí funkce Append
  - *StringBuilder myStringBuilder = new StringBuilder("Hello World!");*
    myStringBuilder.Append(" What a beautiful day.");

- Nebo pomocí funkce Insert
  - *StringBuilder myStringBuilder = new StringBuilder("Hello World!");*
    myStringBuilder.Insert(6,"Beautiful "); //výsledek - Hello Beautiful World!

- Dále používáme funkci Remove pro odebrání charů z intervalu, který tomu dáme, třeba (5, 7)

- Můžeme specifický charakter nahradit jiným pomocí funkce Replace

- String samotný ze StringBuilderu dostaneme opět pomocí .ToString()

Kódování znaků

Reprezentace znaků

- Lidstvo používá mnoho znaků, nejenom latinku
- Chceme je všechny nějak reprezentovat
- Číselná reprezentace různých znaků:
  - 'H' - \u0048, 'e' - \u0065, 'l' - \u006c, 'l' - \u006c, 'o' - \u006f
  - '你' - \u4f60, '好' - \u597d
  - '𐐀' - dvě char instance - \ud801 a \udcbb, které tvoří jeden znak
    'α' - stejně tak - \ud801, \udcdf
  -  - také - \ud83d, \udc02
- Jak je repreznetovat nějak konzistentně nehledě na počet charů, kterými jsou reprezentovány a jiné další rozdíly?

Unicode

- Mezinárodní kódovací standard
- Má přes 1,1 milionu code pointů - písmen, symbolů, emoji...

Decimal	Hex	Example	Description
10	U+000A	N/A	LINE FEED
97	U+0061	a	LATIN SMALL LETTER A
562	U+0232	Ȳ	LATIN CAPITAL LETTER Y WITH MACRON
68,675	U+10C43	𐱃	OLD TURKIC LETTER ORKHON AT
127,801	U+1F339	🌹	ROSE emoji

UTF-16

- Unicode Transformation Format
- reprezentuje *code pointy* z Unicodu 16-bitovými *code unity*

Text „x∈𝕄“ je kódován takto:

Text	x	∈	𝕄
Znaky Unicode	U+0078	U+2208	U+1D544
UTF-16 kódování	`0078`	`2208`	`D835 DD44`
Bajty v UTF-16LE	`78 00`	`08 22`	`35 D8 44 DD`

Textové soubory

Práce se soubory

- Využíváme třídu File

- Pro práci se souborem potřebujeme cestu k němu uloženou jako string, např. “C:\Documents\textfile.txt”

- Před čímkoliv si nejdřív vždy ověříme, že soubor existuje pomocí funkce *File.Exists(adresa_souboru)*, která nám vrátí true/false

- Máme několik možností, jak číst:
  - File.ReadAllText(adresa_souboru)
  - File.ReadAllLines(adresa_souboru)
  - Využití StreamReaderu

StreamReader

- Objekt sloužící ke čtení textu z bytového streamu (nejenom soubory)

- Musíme ho instancovat pro daný stream
  - StreamReader reader = new StreamReader(adresa_souboru);

- Čteme několika způsoby (daná funkce vrací to, co přečetla):
  - reader.Read() - přečte jeden znak, případně tolik, kolik tomu v parametru řekneme
  - reader.ReadLine() - přečte jeden řádek
  - Další způsoby, např. ReadBlock, ReadToEnd, asynchronní varianty atd.

Zápis do souboru

- Obdobně, jako jsme při čtení měli metody ReadAllText() a ReadAllLines(), máme i jejich zapisovací varianty WriteAllText() a WriteAllLines()

- StreamReader má také svůj zapisovací protějšek - StreamWriter

- StreamWriterem zapisujeme obdobnými způsoby, jako jsme četli:
  - writer.Write() - zapíše textovou reprezentaci objektu, který tomu dáme
  - writer.WriteLine() - zapíše to, co tomu dáme, a ukončí to řádek (“odenteruje”)

Specifika StreamReaderu a StreamWriteru

- Abychom nenechali StreamReader zbytečně v paměti po tom, co s ním už nepracujeme, používáme klíčové slovo *using* se složenými závorkami

```
using (StreamWriter writer = new StreamWriter(“C:\Documents\textfile.txt”))
{
        writer.WriteLine(muj_string);
}
using (StreamReader reader = new StreamReader(“C:\Documents\textfile.txt”))
{
        string readLine = reader.ReadLine();
}
```

- Můžeme číst/psát po znacích/řádcích v cyklu a rovnou s textem pracovat

Děkuji za pozornost

Zpětná vazba:
https://forms.gle/nbsFUF1Se9A2h8bM6

Kontakt:
Mail - honza.borecky@seznam.cz
Discord - yeenya (Yeenya#6930)