Vyšší odborná škola obalové techniky
a střední škola, Štětí

Digitální učební materiály

Programování - Programování C#

Ivan Pomykacz

esf evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Licence

Disclaimer

Tento PDF dokument byl strojově vygenerován z HTML stránek

http://dumy.odbornaskola.cz/pomykacz/.

Je tedy možné, že sazba textu může obsahovat chyby. Jde převážně o vizuální a typografické chyby, které mohou narušit plynulou čitelnost textu. V některých případech může být ohrožena i funkčnost některých komponent (jako vnitřní odkazy).

Vzhledem k tomu, že vypracované materiály nebyly nikdy určeny pro výstupní formát PDF, autor se zříkává jakékoli odpovědnosti za nalezené chyby. Nesnažte se proto v této souvislosti autora kontaktovat.

Programování

Programování C#

Obsah

- Znaky a řetězce

Znaky a řetězce

Název školy	Vyšší odborná škola obalové techniky a Střední škola, Štětí, příspěvková organizace		
Adresa školky	Kostelní 134, 411 08 Štětí		
IČ	46773509		
Název operačního programu	OP Vzdělávání pro konkurenceschopnost		
Registrační číslo	CZ.1.07/1.5.00/34.1006		
Označení vzdělávacího materiálu	VY_32_INOVACE_23_PRG_457		
Název tématické oblasti (sady)	Programování		
Název materiálu	Znaky a řetězce		
Anotace	Text seznamuje zevrubněji s textovými datovými typy string a char. Vysvětluje použití typu char. Představuje metody, které lze provádět nad řetězci (např. převést písmena na malá). Ukazuje možnosti zřetězení metod.		
Autor	Ivan Pomykacz	**Jazyk**	čeština
Očekávaný výstup	Používá typ char. Používá metody objektu string. Rozumí principu zřetězení volání metod.		
Klíčová slova	string, char, metody objektu, zřetězení metod		
Druh výukového zdroje	Výklad	**Věková skupina žáků**	15+
Typ interakce	aktivita	**Ročník**	1.
Speciální vzdělávací potřeby	žádné		
Zhotoveno, (datum/období)	08.11.2013	**Celková velikost**	

Obsah

- Shrnutí
- Char
- String
  - Metody objektu `string`
    - Vše na malá
  - Zřetězení metod

Shrnutí

Již víme, že existuje typ `string`, který může obsahovat text. Textem rozumíme písmena abecedy, číslice a jiné znaky, jako např. interpunkce (tečka, čárka, otazník, vykřičník), ale i např. symboly z matematiky (závorky, aritmetické operátory), a tak podobně.

Řetězec může být prázdný, např.

```
string text = "";
```

Může obsahovat jeden znak

```
string text = "a";
```

ale i více znaků (proto řetězec)

```
string text = "Lorem ipsum dolor sit amet.";
```

Přičemž bereme v potaz, že **na velikosti písmen záleží**. Pro kompilátor jsou malé `p` a velké `P` natolik odlišné znaky, jako jako byste míchali hrušky s banány.

Char

Datový typ `char` je specifický ve dvou ohledech (oproti typu `string`). Může obsahovat pouze jeden znak. Hodnota se zapisuje do jednoduchých uvozovek, tedy:

```
char znak = 'a';
```

K čemu `char`? Předně typ `string` je poskládán právě z objektů typu `char`. Tudíž, proměnná `string slovo = "lorem";` se vlastně skládá (vnitřně) z pěti *charů*.

Tyto *chary* můžeme ze *stringu* vytáhnout třeba takto:

```
string slovo = "lorem";
char prvni;
char posledni;

prvni = slovo[0];
posledni = slovo[4];

Console.WriteLine("První znak: {0}, poslední znak {1}.", prvni, posledni);
```

Tato znalost nám pak pomůže v pokročilejší práci s typem `string`. Se samotným *charem* budeme pracovat málokdy, ale přeci, někdy na něj narazíme.

String

Nyní víme, jak se věci mají. String nám vlastně zjednodušuje práci s řetězci, které jsou poskládány z jednotlivých *charů*.

Jaké operace můžeme s řetězci provádět? Pokaždé když napíšete proměnnou typu `string` a připojíte tečku, IDE vám nabídne metody, které se vážou k danému objektu (zde `string`).

```
public static void Main (string[] args)
{

    string text = "Lorem ipsum";
    text.
```

Kompletní přehled metod je samozřejmě v oficiální referenci jazyka.

Navíc, ne všechno jsou metody. Na obrázku je schválně zachycena vlastnost `Length`. Vlastnosti na rozdíl od metod neprovádí žádnou akci. Jsou to vlastně proměnné, které mají přímou vazbu k objektu.

Metody objektu `string`

Ukažme si to na příkladu. Do proměnné `answer` budeme ukládat odpověď, kterou uživatel napíše z klávesnice. Následně otestujeme, zda-li zadal "ano", resp. "ne".

```
string answer;
Console.Write("Je již rozhodnuto? ");

answer = Console.ReadLine();

if (answer == "ano") {
    Console.WriteLine("Výborně, můžeme pokračovat");
}
else if (answer == "ne") {
    Console.WriteLine("A kde to vázne?");
}
else {
    Console.WriteLine("Chyba! Odpověď nebyla rozpoznána.");
}
```

Určitě vás napadlo, co když uživatel zadá např. "**A**no" s velkým počátečním písmenem. To je ale přece úplně jiná hodnota a program tak vstup vyhodnotí jako chybu.

Ano, mohli bychom uživateli napovědět:

```
Console.Write("Je již rozhodnuto? Napiš pouze \"ano\" nebo \"ne\" a je třeba
psát pouze malými písmeny. ");
```

Ale mnohem lepší, tedy alespoň pro uživatele, bude, když se pokusíme jeho vstup přiměřeně opravit.

Vše na malá

Objekt `string` obsahuje mj. metodu `ToLower()`. Ta převádí text na malá písmena (tam kde to má smysl, čísla samozřejmě zůstávají pořád stejně veliká).

Funguje to tak, že po zavolání metody nad objektem string se vrátí upravená hodnota tohoto objektu. Tedy:

```
string text = "Lorem Ipsum";
string upraveno;
upraveno = text.ToLower();
```

Do proměnné `upraveno` se uloží modifikovaný text `lorem ipsum`. Původní obsah proměnné `text` zůstává nezměněn. Všimněte si, že návratová hodnota metody `ToLower()` je opět `string`. Co jiného by to také mohlo být, že?

Není vždy potřeba ukládat změny do proměnné, jelikož pokaždé se pracuje se stringem, stačí v podmínce zapsat:

```
if (answer.ToLower() == "ano") {
    Console.WriteLine("Výborně, můžeme pokračovat");
}
else if (answer.ToLower() == "ne") {
    Console.WriteLine("A kde to vázne?");
}
else {
    Console.WriteLine("Chyba! Odpověď nebyla rozpoznána.");
}
```

Zřetězení metod

Někdy potřebujeme provést na objektem více akcí. Např. chceme text převést na malá písmena a odstranit ze začátku a z konce řetězce bílé znaky (např. mezery, konce řádků, tabulátory).

Metodu `ToLower()` už známe. Ta, co nám umožní pročistit řetězec se nazývá `Trim()`.

S metodou `Trim()` to bude vypadat obdobně.

```
string text = "Lorem Ipsum";
string upraveno;
upraveno = text.ToLower();
upraveno = upraveno.Trim();
```

Celé to lze ovšem zapsat i kratším způsobem, tzv. zřetězením:

```
string text = "Lorem Ipsum";
string upraveno;
upraveno = text.ToLower().Trim();
```

Kvůli přehlednosti a i efektivnosti kódu by pak naše podmínka mohla vypadat:

```
string answer;
Console.Write("Je již rozhodnuto? ");
```

```
answer = Console.ReadLine().ToLower().Trim();

if (answer == "ano") {
    Console.WriteLine("Výborně, můžeme pokračovat");
}
else if (answer == "ne") {
    Console.WriteLine("A kde to vázne?");
}
else {
    Console.WriteLine("Chyba! Odpověď nebyla rozpoznána.");
}
```

Změna je pouze v úpravě řádku u volání metody `ReadLine()`.