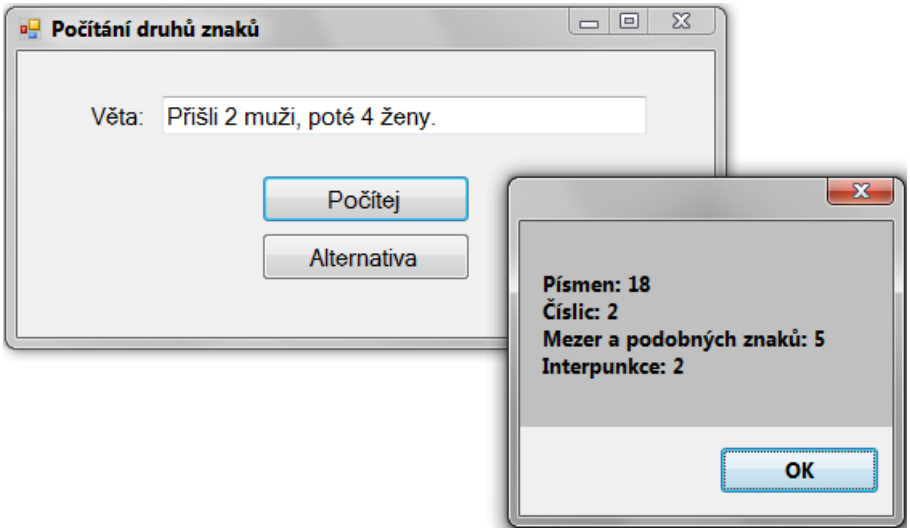
Zpravodaj moderníProgramování 04/2012: Práce se znaky

Předpoklad ke studiu tohoto čísla: šestá kapitola učebnice pro středně pokročilé (žlutá).

V tomto čísle Zpravodaje si ukážeme některé postupy pro práci se znakovými daty. Přestože význam celých textových řetězců typu `string` je mnohem větší, pro některé příležitosti se typ `char` (alternativně `Char` jako název třídy) hodit může.

Počítání druhů znaků

Začneme ukázkou knihovních statických metod ze třídy `Char` na rozlišování jednotlivých druhů znaků. Upravíme program na počítání áček ze třetí kapitoly žluté učebnice:




```
private void tlačítkoPočítej_Click(object sender, EventArgs e)
{
    // Příprava
    int početPísmen = 0;
    int početČíslic = 0;
    int početMezerApod = 0;
    int početInterpunkce = 0;

    // Projdi řetězec znak po znaku
    string zadanáVěta = poleVěta.Text;
    for (int index = 0; index < zadanáVěta.Length; index++)
    {
        if (Char.IsLetter(zadanáVěta, index))
            početPísmen++;

        if (Char.IsDigit(zadanáVěta, index))
            početČíslic++;
```

```
        if (Char.IsWhiteSpace(zadanáVěta, index))
            početMezerApod++;

        if (Char.IsPunctuation(zadanáVěta, index))
            početInterpunkce++;
    }

    // Zobraz výsledky
    MessageBox.Show(
        "Písmen: " + početPísmen.ToString() + Environment.NewLine +
        "Číslic: " + početČíslic.ToString() + Environment.NewLine +
        "Mezer a podobných znaků: " + početMezerApod.ToString()   +
                                                Environment.NewLine +
        "Interpunkce: " + početInterpunkce.ToString());
}
```

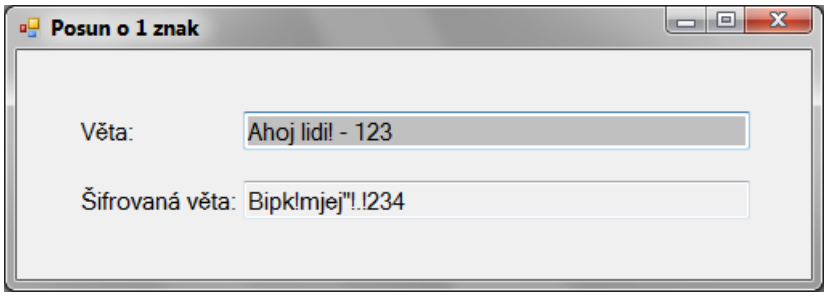
Metody `Char.IsNěco` přebírají textový řetězec a pozici analyzovaného znaku. Alternativně je možno předat pouze znak samotný jako hodnotu typu `char`. Viz dále včetně cyklu `foreach` (pro změnu):

```
private void tlačítkoAlternativa_Click(object sender, EventArgs e)
{
    // Příprava
    int početPísmen = 0;
    int početČíslic = 0;
    int početMezerApod = 0;
    int početInterpunkce = 0;

    // Projdi řetězec znak po znaku
    string zadanáVěta = poleVěta.Text;
    foreach (char znak in zadanáVěta)
    {
        if (Char.IsLetter(znak))
            početPísmen++;

        if (Char.IsDigit(znak))
            početČíslic++;

        if (Char.IsWhiteSpace(znak))
            početMezerApod++;

        if (Char.IsPunctuation(znak))
            početInterpunkce++;
    }

    // Zobraz výsledky
    MessageBox.Show(
        "Písmen: " + početPísmen.ToString() + Environment.NewLine +
        "Číslic: " + početČíslic.ToString() + Environment.NewLine +
        "Mezer a podobných znaků: " + početMezerApod.ToString()   +
                                                Environment.NewLine +
        "Interpunkce: " + početInterpunkce.ToString());
}
```

Aritmetika v typu `char`

Typ `char` je takový zvláštní. Někdy se chová textově, někdy jako číslo odpovídající Unicode kódu znaku. To znamená, že se s ním dá dělat aritmetika.

Ukažme si to na příkladu zašifrování zadaného textu tak, že se každý zadaný znak posune o 1 dál, tzn. A na B, B na C atd. Zajisté to není žádná ohromující šifra, na jejíž prolomení jsou potřeba všechny superpočítače FBI, ale na procvičení je to pěkné.

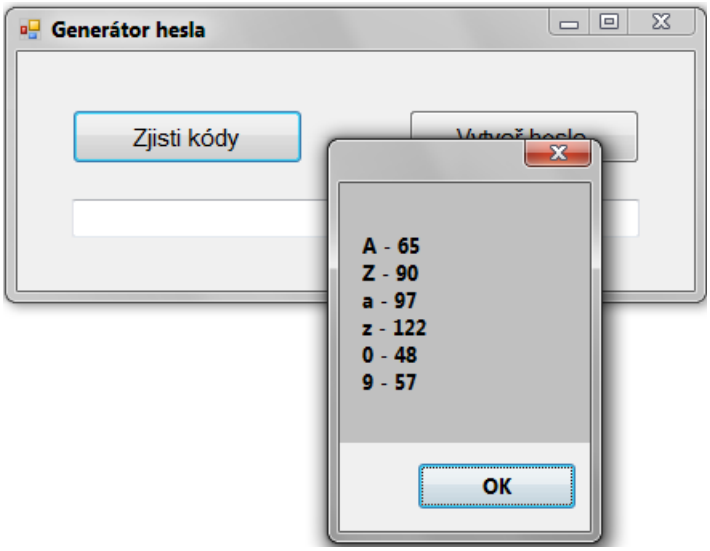



```
private void poleVěta_TextChanged(object sender, EventArgs e)
{
    string zadanáVěta = poleVěta.Text;
    string výsledek = "";
    foreach (char znak in zadanáVěta)
    {
        char novýZnak = (char)(znak + 1);
        výsledek += novýZnak.ToString();
    }
    poleŠifrovanáVěta.Text = výsledek;
}
```

Povšimněte si nutnosti konverze součtu na `char`. Prostě `char` + `int` je `int`.

Generování náhodného hesla

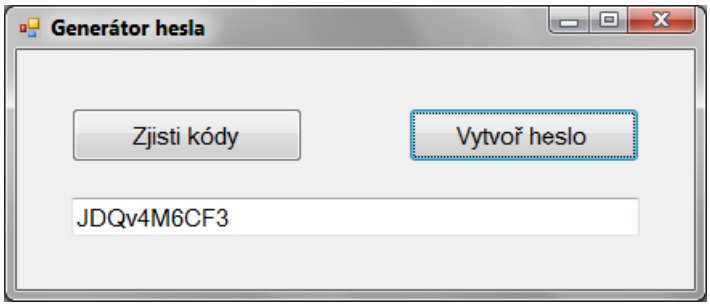
Řada systémů nastavuje uživatelům první heslo pomocí náhody. Na takovouto úlohu můžeme právě využít typ `char`. Dejme tomu, že budeme chtít hesla generovat z číslic a malých a velkých písmen anglické abecedy.



Nejprve si zobrazme kódy hraničních znaků (se znaky pracujeme v typu `int`!):

```
private void tlačítkoZjistiKódy_Click(object sender, EventArgs e)
{
    int kódA = 'A';
    int kódZ = 'Z';
    int kóda = 'a';
    int kódz = 'z';
    int kód0 = '0';
    int kód9 = '9';

    MessageBox.Show(
        "A - " + kódA.ToString() + Environment.NewLine +
        "Z - " + kódZ.ToString() + Environment.NewLine +
        "a - " + kóda.ToString() + Environment.NewLine +
        "z - " + kódz.ToString() + Environment.NewLine +
        "0 - " + kód0.ToString() + Environment.NewLine +
        "9 - " + kód9.ToString()
        );
}
```

Vidíme, že Z-A = z-a = 25, což odpovídá 26 znakům anglické abecedy. Zřejmě tedy jde o souvislou řadu. Stejně tak cifry 9 - 0 = 9, neboli 10 cifer. Samozřejmě, všechno toto můžeme zjistit pomocí tabulky Unicode, ale proč si to nezkontrolovat? Člověk tak získá větší jistotu v použití prostředků, jež má k dispozici.



Jak nyní na heslo? Zvolíme si (např. pomocí náhody) jeho délku a poté pomocí cyklu patřičný počet náhodných znaků vygenerujeme.

Jak vygenerovat náhodný znak? Náhodná čísla generovaná metodou `Next` instance `Random` se vybírají z jednoho souvislého intervalu. My však v naší úloze musíme zohlednit skutečnost, že kódy všech dovolených znaků, jak jsme viděli, souvislou řadu netvoří.

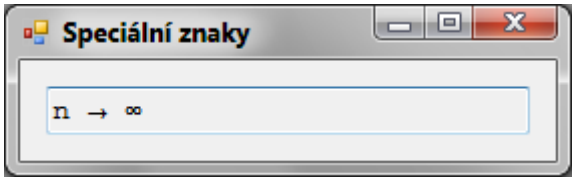
Dejme tedy tomu, že si všechny dovolené znaky soustředíme v jednom (souvislém) seznamu a poté budeme generovat náhodné číslo jako index do tohoto seznamu. Proč seznam a ne pole? Abychom nemuseli dopředu počítat, kolik těch znaků celkem bude.

Následuje kód, ve kterém jsem z dovolených znaků vyloučil ty, které vyvolávají nebezpečí záměny (vyvolávat nebezpečí záměny se totiž dle Obchodního zákoníku nesmí ;-, to je nekalá soutěž), to znamená nulu a ó, v Čechách navíc působí problémy Y a Z. Pokud byste chtěli být na uživatele obzvlášť hodní (a to byste asi měli!), můžete vynechat ještě velké I, které se plete s malým l. To už ale nechám na vás.

```
private void tlačítkoVytvořHeslo_Click(object sender, EventArgs e)
{
    // Definice intervalů
    char[] dolníMeze = new char[] { '1', 'A', 'P', 'a', 'p' };
    char[] horníMeze = new char[] { '9', 'N', 'X', 'n', 'x' };
    int početIntervalů = dolníMeze.Length;

    // Seznam možných znaků
    List<char> možnéZnaky = new List<char>();
    for (int index = 0; index < početIntervalů; index++)
    {
        // Všechny znaky z intervalu [index] přidej do seznamu
        for (char znak = dolníMeze[index];
             znak <= horníMeze[index];
             znak++)
        {
            možnéZnaky.Add(znak);
        }
    }

    // Náhodně heslo
    int délkaHesla = náhoda.Next(7, 12 + 1);
    int největšíČíslo = možnéZnaky.Count - 1;
    string heslo = "";
    for (int čísloZnaku = 0; čísloZnaku < délkaHesla; čísloZnaku++)
    {
        int náhodnýIndex = náhoda.Next(0, největšíČíslo + 1);
        char znakHesla = možnéZnaky[náhodnýIndex];
        heslo += znakHesla.ToString();
    }

    // Zobraz heslo
    poleHeslo.Text = heslo;
}
```

Práce se speciálními znaky

Na závěr si ještě ukážeme, jak pracovat se znaky, které nejde zadat normálně z klávesnice. Já jsem zkusil zobrazení textu „*n* → ∞“ (en jde k nekonečnu; schválně, jestli víte, kolik je limita řady 1 + 1/2+ 1/4 + 1/8 + … ?; taková řada je i v informatice docela užitečná):



Kód je krátký:

```
private void oknoProgramu_Load(object sender, EventArgs e)
{
    char nekonečno = '\u221E';
    poleZobrazení.Text = "n \u2192 " + nekonečno;
}
```

Speciální znak se uvede zpětné lomítko a u a následuje čtyřmístný šestnáctkový Unicode. Jak jej zjistit? Třeba ve Wordu se ukazuje, když pracujete s oknem *Vložit* > *Symbol*. Jak vidíte, se speciálním znakem lze pracovat jak v typu `char`, tak ve `stringu`.

A málem bych zapomněl. Pro správné zobrazení znaků bývá někdy potřeba změnit písmo. Já jsem zvolil namátkou Courier.

Radek Vystavěl, 16. dubna 2012

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